HP 64768

70433 Emulator Softkey Interface

User's Guide

HP Part No. 64768-97005
Printed in Japan
May 1995

Edition 3

Notice

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	64768-97001, March 1993
Edition 2	64768-97003, October 1993
Edition 3	64768-97005, May 1995

Using this Manual

This manual shows you how to use the following emulators with the Softkey Interface.

- HP 64768 70433 emulator

This manual:

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

This manual does not:

- Show you how to use every Softkey Interface command and option; the Softkey Interface is described in the *Softkey Interface Reference* manual.

For the most part, the HP 647680A and HP 64768B emulators all operate the same way. Differences of between the emulators are described where they exist. Both the HP 64768A and HP 64768B emulators will be referred to as the "HP 64768 emulator" or "70433 emulator".

Organization

Chapter 1 Introduction to the 64768 Emulator. This chapter briefly introduces you to the concept of emulation and lists the basic features of the 64768 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 program, run programs, set software breakpoins, search memory for data, and use the analyzer.

Chapter 3 "In-Circuit" Emulation. This chapter shows you how to install the emulator probe into a target system and how to use "in-circuit" emulation features.

Chapter 4 Configuring the Emulator. This chapter shows you how to: restrict the emulator to real-time execution, select a target system clock source, allow the target system to insert wait states, and select foreground or background monitor.

Chapter 5 Using the Emulator. This chapter describes emulation topics which are not covered in the "Getting Started" chapter.

Appendix A Using the Foreground. This appendix describes the advantages and disadvantages of foreground and background monitors and how to use foreground monitor.

Appendix B Using the Format Converter. This appendix describes the usage of the file format converter.

Conventions

Example commands throughout the manual use the following conventions:

bold	Commands, options, and parts of command syntax.
bold italic	Commands, options, and parts of command syntax which may be entered by pressing softkey.
normal	User specified parts of a command.
$	Represents the HP-UX prompt. Commands which follow the "$" are entered at the HP-UX prompt.
<RETURN>	The carriage return key.

Notes

Contents

Illustrations

Notes

Introduction to the 64768 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 64768 emulator is designed to replace the 70433 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.

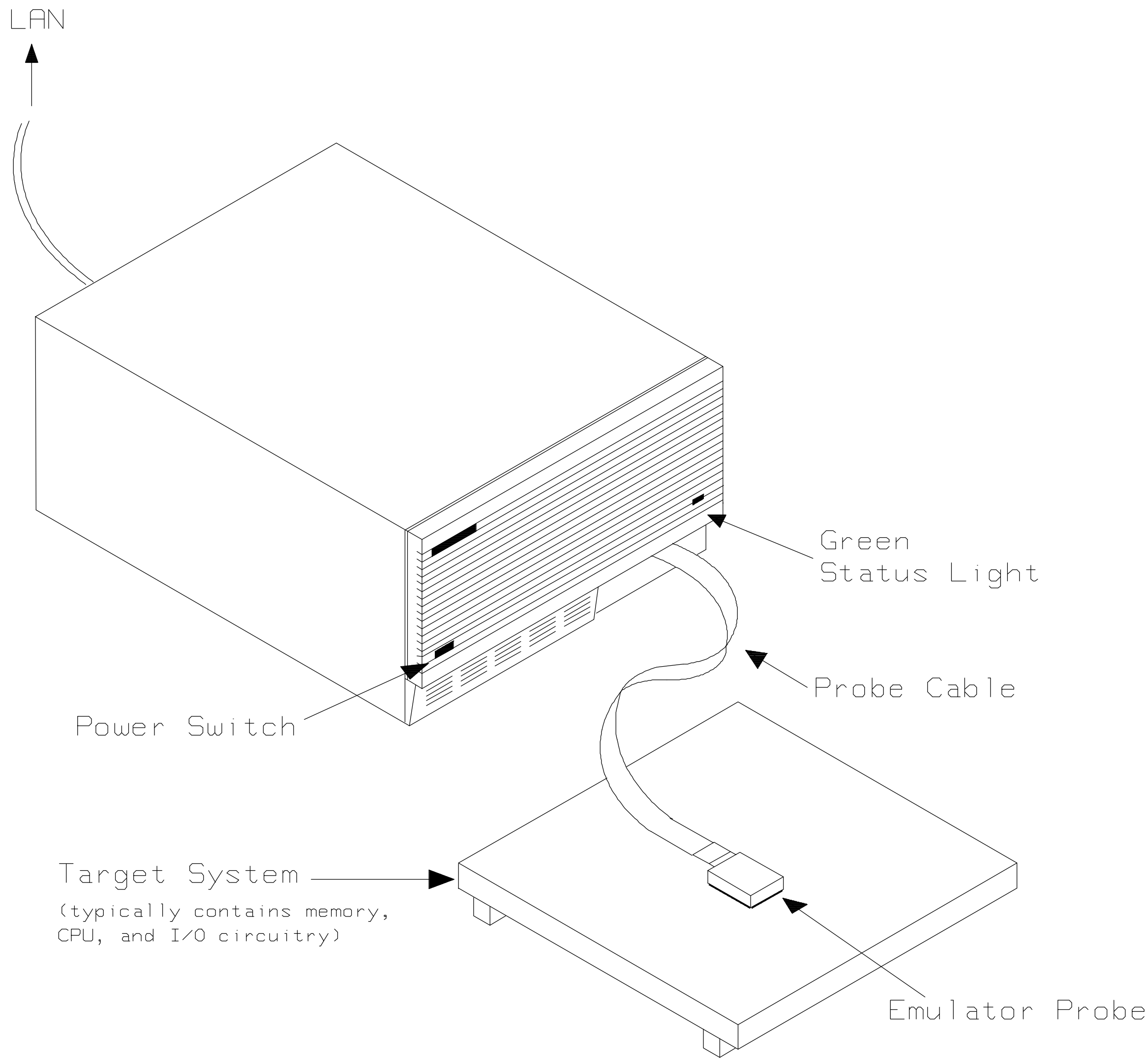


Figure 1-1 HP 64768 Emulator for uPD70433

Features of the HP 64768 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 132-pin PGA type of 70433 microprocessor is supported.The HP 64768 emulator probe has a 132-pin PGA connector.When you use 120-pin QFP type microprocessor, you must use with PGA to QFP adapter; refer to the "In-Circuit Emulation Topics "chapter in this manual.

Clock Speeds

The HP 64768A emulator runs with an internal clock speed of 12.5MHz (system clock), or with target system clocks from 4 to 25 MHz.

The HP 64768B emulator runs with an internal clock speed of 12.5MHz (system clock), or with target system clocks from 4 to 32 MHz.

Emulation memory

The HP 64768 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

You can define up to 16 memory ranges (at 256 byte boundaries and at least 256 byte in length). The monitor occupies 2K bytes leaving 126K,510K,1022K 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 64768 emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

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

The 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 70433 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 emulation monitor.

You can also define software breakpoints in your program. The emulator uses the 70433 BRK 3 instruction to provide software breakpoint. When you define a software breakpoint, the emulator places a BRK 3 instruction at the specified address; after the BRK 3 instruction causes emulator execution to break out of your program, the emulator replaces BRK 3 with 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.

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 70433 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. User program execution 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 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

Reset, Hold Request While in Background Monitor

If you use background monitor, RESET and HLDRQ from target system are ignored while in monitor.

User Interrupts While in Background Monitor

If you use the background monitor, NMI and INTP0-5 from target system are suspended until the emulator goes into foreground operation. Other interrupts are ignored.

Interrupts While Executing Step Command

While stepping user program with the foreground monitor used, interrupts are accepted if they are enabled in the foreground monitor program.

While stepping user program with the background monitor used, interrupts are ignored.

Note

You should not use step command in case the interrupt handler’s punctuality is critical.

Unbreaking into the Monitor

The emulator can not break into the monitor if the microprocessor is in hold state. The emulator will break into the monitor after hold state because break request is suspended.

The emulator can not break into the monitor if the microprocessor is in reset state by RESET signal from target system.

CLKOUT enable bit

CLKOUT signal can be enabled/disabled by ENCLK bit, which is bit 5 of PRC register. You must not clear ENCLK bit(ENCLK bit is "1" in reset). The emulator will not work properly if CLKOUT signal is disabled

DMA Support

Direct memory access to emulation memory by external DMA controller is not permitted.

Accessing SFR

When you access SFR(Special Function Registers), you must use reg commands. If you access SFR with m commands, you will access to the actual memory you mapped(as target system ROM or RAM, emulation ROM or RAM).

Accessing Reserved Area of I/O Space

When you access reserved area of I/O space(0FF80h-0FFFFh) with "io" commands in the background monitor, the emulator operates exceptionally. When you display reserved area of I/O space, theemulator displays "FFh" . When you modify reserved area of I/Ospace, the emulator does not modify value.

Evaluation Chip

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

Notes

2

Getting Started

Introduction

This chapter will lead you through a basic, step by step tutorial that shows how to use the HP 64768 emulator (for the 70433 microprocessor) with the Softkey Interface.

This chapter will:

- Tell you what must be done before you can use the emulator as shown in the tutorial examples.
- Describe the demo program used for this chapter's examples.

This chapter will show you how to:

- Start up the Softkey Interface.
- Load programs into emulation and target system memory.
- Enter emulation commands to view execution of the demo program.

Before You Begin

Prerequisites

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

1. Connected the emulator to your computer. The HP 64700 Series Installation/Service manual show you how to do this.

2. Installed the Softkey Interface software on your computer. Refer to the HP 64700 Series Installation/Service manual for instructions on installing software.

3. In addition, you should read and understand the concepts of emulation presented in the Concepts of Emulation and Analysis manual. The Installation/Service manual also covers HP 64700 system architecture. A brief understanding of these concepts may help avoid questions later.

   You should read the Softkey Interface Reference manual to learn how to use the Softkey Interface in general. For the most part, this manual contains information specific to the 70433 emulator.

A Look at the Demo Program

The demo program is listed in "C" and assembly in Figure 2-1. The demo program is skdemo consisting of source program skdemo.c.

Where is the skdemo Software?

The demo program is shipped with the Softkey Interface and may be copied from the following directory.

/usr/hp64000/demo/emul/hp64768

What Does the Program Do?

The program is designed to go into a continuous loop checking for a new command (NEW_CMD). The while loop skips the "if" statement because semaphore is assigned the value of NO_CMD ('f') and is never equal to NEW_CMD ('t'). Also, the switch statement's expression never matches the cases CMD_A, CMD_B, or CMD_C because the switch statement evaluates the value (cmd_code) which is initially assigned the value of NO_CMD ('f').

As you progress through this tutorial, you will make the appropriate changes to the data in memory to make this program step through each branch of the program. When semaphore signals a new command (NEW_CMD):

1. Semaphore is assigned the value of CMD_STARTED ('s'),
2. The status "Command received ", is displayed.
3. The switch statement is executed. If a command has NOT been specified in cmd_code, the default command result "Invalid command entered" is displayed.

Symbolic Constants

Symbolic constants can be used in different parts of the program to make the program run, step, break, trace, or define memory ranges. The symbols are truncated to 15 characters. Symbolic names (local and global) are labeled:

Static Local Symbols

```
cmd_loop          main
await_comd        cmd_code
cmd_result        command_A
command_B         command_C
command_I         command_R
no_command        semaphore
status
```

Static Global Symbols

```
_cmd_loop         _main
_strcpy
```

cmd_code

The switch statement evaluates the value of cmd_code with the cases within it. You will change the cmd_code (to ’A’, ’B’, or ’C’) to match each of the cases as you progress through the steps in this manual. As you enter into each branch of the switch statement:

- If case CMD_A is satisfied, the cmd_result (Command ’A’ entered) is displayed.
- If case CMD_B is satisfied, the cmd_result (Command ’B’ entered) is displayed.
- If case CMD_C is satisfied, the cmd_result (Command ’C’ entered) is displayed.
- When the case statement is completed, semaphore equals the value of CMD_FINISHED (’f’).

semaphore

Semaphore is assigned the value of NO_CMD (0H). You will change the value of "semaphore" to NEW_CMD (’t’) to satisfy the "if" statement. This variable is used to synchronize commands.

```
/* "NEC V55PI DEMONSTRATION PROGRAM" */
/* "skdemo.c" */
#include <rcopy.h>

/* DEFINES */
#define TRUE                     1
#define FALSE                    0
#define NO_CMD                   'f'
#define NEW_CMD                  't'
#define CMD_STARTED              's'
#define CMD_FINISHED             'f'
#define CMD_A                    'A'
#define CMD_B                    'B'
#define CMD_C                    'C'

extern void my_rompseg();

static unsigned char status     [] = "                           ";
static unsigned char cmd_result [] = "                           ";

static unsigned char semaphore = NO_CMD;
static unsigned char cmd_code = NO_CMD;
static unsigned char await_comd [] = "Awaiting command       ";
static unsigned char no_command [] = "No command entered     ";
static unsigned char command_R  [] = "Command received       ";
static unsigned char command_A  [] = "Command 'A' entered    ";
static unsigned char command_B  [] = "Command 'B' entered    ";
static unsigned char command_C  [] = "Command 'C' entered    ";
static unsigned char command_I  [] = "Invalid Command entered";


/*****************************************************************/

main()
{
    /* To display symbolically:                                  */
    /* display local_symbol_in skdemo.c:                         */
    /* display memory status thru cmd_code repeatively bytes     */

    rcopy(my_rompseg);

    strcpy (status,await_comd);

    /* initialize control variables */

    semaphore = NO_CMD;
    cmd_code = NO_CMD;
    strcpy (cmd_result,no_command);

    /* Call command loop function */
```

Figure 2-1 C Source skdemo.c (Cont’d)

```
    cmd_loop ();

}   /* main_prog */

/****************************************************************/
cmd_loop (t)
{
    /* Stay in endless loop, checking for a new command         */
    /* When a new command is detected, it is executed           */
    /* New command can be entered using the following:          */

    /* Place the value 'B' in cmd_code memory location          */
    /* modify memory cmd_code bytes to 'B '                     */
    /* Place the value 't' in semaphore to indecate NEW_CMD     */
    /* modify memory semaphore bytes to 't'                     */
    /* Display the result in cmd_result                         */
    /* display memory cmd_result blocked bytes                  */

    while (TRUE){
        if (semaphore == NEW_CMD){
            semaphore = CMD_STARTED;
            strcpy (status, command_R);
            switch (cmd_code){
                case CMD_A :
                    strcpy (cmd_result, command_A);
                    break;

                case CMD_B :
                    strcpy (cmd_result, command_B);
                    break;

                case CMD_C :
                    strcpy (cmd_result, command_C);
                    break;

                default :
                    strcpy (cmd_result, command_I);
            }   /* switch */

            semaphore = CMD_FINISHED;
        }   /* if (semaphore == NEW_CMD) body */
    }   /* while (TRUE) loop */
}   /* FUNCTION cmd_loop */
```

Figure 2-1 C Source skdemo.c (Cont'd)

Compiling the Demo Program

The demo program is written for and compiled/linked with the InterTools V20 Family C Compiler and Linking Locator. The demo program was compiled with the following command.

```
$ cv55 skdemo.c
-S /usr/local/itools/rtlibs/inc -d -do -sd
<RETURN>
```

Linking the Demo Program

The following command was used to generate the absolute file. The "skdemo.lc" linker command file is shown in figure 2-2.

```
$ llink skdemo.ol -o skdemo.ab -c skdemo.lc
-b _my_rompseg -rc stsep -L
/usr/local/itools/rtlibs/cv55s/lm/libcv55s
<RETURN>
```

```
LOCATE ( {code}     : #80000 );
LOCATE ( {constant} : #80500 );
LOCATE ( {stsep}    : #10000 );
LOCATE ( {data}     : #15000 );
```

Figure 2-2 Linker Command File for "skdemo.lc"

Generate HP Absolute file

To generate HP absolute file for the Softkey Interface, you need to use "**v55cnv**" absolute file format converter. The v55cnv converter is provided with the Softkey Interface. To generate HP absolute file, enter the following command:

```
$ v55cnv skdemo <RETURN>
```

You will see that skdemo.X, skdemo.L, and skdemo.A are generated. The file skdemo.X contains the absolute code of the program. The file skdemo.L contains the list of global symbols. The files *skdemo.A* contains the list of local symbols for the respective files.

Note

The required executable files are included in the following directory. **/usr/hp64000/demo/emul/hp64768**

Entering the Softkey Interface

If you have installed your emulator and Softkey Interface software as directed in the HP 64700 Series Emulators Softkey Interface Installation Notice, you are ready to enter the interface. The Softkey Interface can be entered through the **pmon** User Interface Software or from the HP-UX shell.

From the "pmon" User Interface

If **/usr/hp64000/bin** is specified in your PATH environment variable, you can enter the **pmon** User Interface with the following command.

```
$ pmon <RETURN>
```

If you have not already created a measurement system for the 70433 emulator, you can do so with the following commands. First you must initialize the measurement system with the following command.

```
MEAS_SYS msinit <RETURN>
```

After the measurement system has been initialized, enter the configuration interface with the following command.

```
msconfig <RETURN>
```

To define a measurement system for the 70433 emulator, enter:

```
make_sys emv55 <RETURN>
```

Now, to add the emulator to the measurement system, enter:

```
add <module_number> naming_it n70433 <RETURN>
```

Enter the following command to exit the measurement system configuration interface.

```
end <RETURN>
```

If the measurement system and emulation module are named "emv55" and "n70433" as shown above, you can enter the emulation session with the following command:

```
emv55 default n70433 <RETURN>
```

If this command is successful, you will see a display similar to figure 2-3. The status message shows that the default configuration file has been loaded. If the command is not successful, you will be given an error message and returned to the **pmon** User Interface. Error messages are described in the *Softkey Interface Reference* manual.

For more information on creating measurements systems, refer to the *Softkey Interface Reference* manual.

```
                  HPB3067-19301 A.05.00 11Nov92 Unreleased
                         V55 SOFTKEY USER INTERFACE

                      A Hewlett-Packard Software Product
                      Copyright Hewlett-Packard Co. 1992

  All Rights Reserved. Reproduction, adaptation, or translation without prior
  written  permission  is prohibited, except as allowed under copyright laws.

                           RESTRICTED RIGHTS LEGEND

     Use , duplication , or disclosure  by the  Government is  subject to
     restrictions as set forth in subparagraph (c) (1) (II) of the Rights
     in Technical Data and Computer Software clause at  DFARS 52.227-7013.
     HEWLETT-PACKARD Company , 3000 Hanover St. , Palo Alto, CA 94304-1181


STATUS:   Starting new session__________________________________________...R....



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

Figure 2-3 Softkey Interface Display

From the HP-UX Shell

If **/usr/hp64000/bin** is specified in your PATH environment variable, you can also enter the Softkey Interface with the following command.

`$` ***`emul700 -uskemul`*** `<emul_name> <RETURN>`

The "emul_name" in the command above is the logical emulator name given in the HP 64700 emulator device table (/usr/hp64000/etc/64700tab).

```
#-------------+---------+--------------------+----+------+------+----+----+----
#             |         |                    |xpar| baud |parity|flow|stop|char
# logical name|processor|      physical      |mode| rate |      |    |bits|size
#  (14 chars) |  type   |       device       |    |      |      |XON |    |
#             |         |                    |OFF |      | NONE |RTS | 2  | 8
#-------------+---------+--------------------+----+------+------+----+----+----
#
v55PI          n70433    /dev/emulcom23      OFF    9600  NONE  XON   2    8
```

If this command is successful, you will see a display similar to figure 2-3. The status message shows that the default configuration file has been loaded. If the command is not successful, you will be given an error message and returned to the HP-UX prompt. Error messages are described in the Softkey Interface Reference manual.

Configure the Emulator for Examples

To do operations described in this chapter (loading absolute program into emulation memory, displaying memory contents, etc), you need to configure the emulator as below. For detailed description of each configuration option (question), refer to the "Configuring the Emulator" chapter.

To get into the configuration session of the emulator, enter the following command.

`modify configuration` <RETURN>

Answer to the series of questions as below.

Micro-processor clock source? ***internal*** <RETURN>
Enter monitor after configuration? ***yes*** <RETURN>
Restrict to real-time runs? ***no*** <RETURN>
Modify memory configuration? ***yes*** <RETURN>
Monitor type? ***background*** <RETURN>

Now you should be facing memory mapping screen. The address ranges 0h thru 1f6ffh and fff00h thru fffffh are mapped as emulation RAM by default. Three mapper terms must be specified for the demo program. Enter the following lines to map the program code area as emulation ROM, data area as emulation RAM.

delete all <RETURN>
0h ***thru*** 0ffh **emulation ram** <RETURN>
10000h ***thru*** 1ffffh ***emulation ram*** <RETURN>
80000h ***thru*** 80fffh ***emulation rom*** <RETURN>
0fff00h **thru** 0fffff ***emulation rom*** <RETURN>
end <RETURN>

Modify emulator pod configuration? ***no***

Modify debug/trace options? ***no*** <RETURN>
Modify simulated I/O configuration? ***no*** <RETURN>
Modify interactive measurement specification? ***no*** <RETURN>

If you wish to save the configuration specified above, answer this question as shown.

Configuration file name? skdemo <RETURN>

Now you are ready to go ahead. Above configuration is used through out this chapter.

On-Line Help

There are two ways to access on-line help in the Softkey Interface. The first is by using the Softkey Interface help facility. The second method allows you to access the firmware resident Terminal Interface on-line help information.

Softkey Driven Help

To access the Softkey Interface on-line help information, type either "help" or "?" on the command line; you will notice a new set of softkeys. By pressing one of these softkeys and <RETURN>, you can cause information on that topic to be displayed on your screen. For example, you can enter the following command to access "system command" help information.

```
? system_commands <RETURN>
```

```
---SYSTEM COMMANDS & COMMAND FILES---

?                          displays the possible help files
help                       displays the possible help files

!                          fork a shell (specified by  shell variable SH)
! <hell command>           fork a shell and execute a shell command

pwd                        print the working directory
cd <directory>             change the working directory

pws                        print the default symbol scope
cws <YMB>                  change the working symbol - the working symbol also
                           gets updated when displaying local symbols and
                           displaying memory mnemonic


forward <UI> "command"     send the command in the quoted string from this user
                           interface to another one.  Replace  with the name
                           of the other user interface as shown on the softkeys:

-More--(15%)
```

The help information is scrolled on to the screen. If there is more than a screenful of information, you will have to press the space bar to see the next screenful, or the <RETURN> key to see the next line, just as you do with the HP-UX **more** command. After all the information on the particular topic has been displayed (or after you press "q" to quit scrolling through information), you are prompted to press <RETURN> to return to the Softkey Interface.

Pod Command Help

To access the emulator's firmware resident Terminal Interface help information, you can use the following commands.

```
display pod_command <RETURN>
pod_command 'help cf' <RETURN>
```

The command enclosed in string delimiters (", ', or ^) is any Terminal Interface command, and the output of that command is seen in the pod_command display. The Terminal Interface help (or ?) command may be used to provide information on any Terminal Interface command or any of the emulator configuration options (as the example command above shows).

Note

If you want to use the Terminal Interface command by entering from keyboard directly, you can do it after entering the following command.

```
pod_command keyboard
```

```
Pod Commands
  Time                Command
14:42:14 help cf

  cf - display or set emulation configuration
    cf                - display current settings for all config items
    cf <item>         - display current setting for specified <item>
    cf <item>=<value> - set new <value> for specified <item>
    cf <item> <item>=<value> <item> - set and display can be combined
  help cf <item>    - display long help for specified
  --- VALID CONFIGURATION <item> NAMES ---
    clk     - select clock source
    dsize   - select data bus width
    hold    - en/dis HLDRQ input from the target system
    mne     - select mnemonic for inverse assembly
    mon     - select foreground or background monitor
    nmi     - en/dis NMI from the target system
    rad     - select run address translation method

STATUS:   n70433--Running in monitor____________________________________...R....
pod_command 'help cf'


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

Loading Absolute Files

The "load" command allows you to load absolute files into emulation or target system memory. If you wish to load only that portion of the absolute file that resides in memory mapped as emulation RAM or ROM, use the "load emul_mem" syntax. If you wish to load only the portion of the absolute file that resides in memory mapped as target RAM, use the "load user_mem" syntax. If you want both emulation and target memory to be loaded, do not specify "emul_mem" or "user_mem". For example:

```
load skdemo <RETURN>
```

Note

When loading a program if the status line shows

```
"ERROR:      No absolute file, No database:
skdemo
```

, you may NOT be in the directory that your program is in. To find out what directory you are in, enter:

```
! pwd <RETURN>
```

The "!" allows you to use an HP-UX shell command. To move into the correct directory, enter:

```
cd <directory path>  <RETURN>
```

You can also specify the pathname where your program resides. For example, you could enter:

```
load /usr/hp64000/demo/emul/hp64768/skdemo
<RETURN>
```

Displaying Symbols

When you load an absolute file into memory (unless you use the "nosymbols" syntax), symbol information is also loaded. Both global symbols and symbols that are local to a source file can be displayed.

Global

To display global symbols, enter the following command.

```
display global_symbols <RETURN>
```

Listed are address ranges associated with a symbol, the segment that the symbol is associated with, and the offset of that symbol within the segment.

```
Global symbols in skdemo.X
Static symbols
Symbol name ____________________ Address range __ Segment _____________ Offset
__DSX                         1500:002C                                    0000
__SSX                         1500:002E                                    0000
__iob                         1500:0040                                    0000
__main                        8000:00C2                                    0000
__term_putc                   8000:015B                                    0000
__xgetc                       8000:0110                                    0000
_cmd_loop                     8020:0081                                    0000
_errno                        1500:008E                                    0000
_getc                         8000:0072                                    0000
_main                         8020:0029                                    0000
_rcopy                        8010:00CB                                    0000
_stderr                       1500:008A                                    0000
_stdin                        1500:0082                                    0000
_stdout                       1500:0086                                    0000
_strcpy                       8010:00FA                                    0000

STATUS:   n70433--Running in monitor____________________________________...R....
display global_symbols


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

Local

When displaying local symbols, you must include the name of the source file in which the symbols are defined. For example,

```
display local_symbols_in skdemo.c: <RETURN>
```

As you can see, the procedure symbols and static symbols in "skdemo.c" are displayed.
To list the next symbols, press the <PGDN> or <Next> key. the source reference symbols in "skdemo.c" will be displayed.

Listed are: address ranges associated with a symbol, the segment that the symbol is associated with, and the offset of that symbol within the segment.

```
Symbols in skdemo.c:
Static symbols
Symbol name ____________________ Address range __ Segment _____________ Offset
await_comd                      1000:0090                                    0090
cmd_code                        1000:00A8                                    00A8
cmd_loop                        8020:0081                                    0058
cmd_result                      1000:00AC                                    00AC
command_A                       1000:0048                                    0048
command_B                       1000:0030                                    0030
command_C                       1000:0018                                    0018
command_I                       1000:0000                                    0000
command_R                       1000:0060                                    0060
main                            8020:0029                                    0000
no_command                      1000:0078                                    0078
semaphore                       1000:00AA                                    00AA
status                          1000:00C8                                    00C8

Source reference symbols

STATUS:   cws: skdemo.c:________________________________________________...R....
display  local_symbols_in skdemo.c:


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

Source Lines

To display the address ranges associated with the program's source file, you must display the local symbols in the file. For example:

```
display local_symbols_in skdemo.c: <RETURN>
```

And scroll the information down on the display with up arrow,or <Next> key.

```
Symbols in skdemo.c:
Source reference symbols
Line range _____________________ Address range __ Segment _____________ Offset
#1-#34                        8020:0029 - 002C                              0000
#35-#40                       8020:002D - 0037                              0004
#41-#42                       8020:0038 - 004F                              000F
#43-#46                       8020:0050 - 0058                              0027
#47-#47                       8020:0059 - 0061                              0030
#48-#48                       8020:0062 - 0079                              0039
#49-#51                       8020:007A - 0080                              0051
#52-#56                       8020:0081 - 0084                              0058
#57-#70                       8020:0147                                     011E
#71-#71                       8020:0085 - 008F                              005C
#72-#72                       8020:0090 - 0098                              0067
#73-#73                       8020:0099 - 00B0                              0070
#74-#74                       8020:0123 - 013B                              00FA
#75-#76                       8020:00B4 - 00CB                              008B
#77-#77                       8020:00CC - 00CE                              00A3

STATUS:   n70433--Running in monitor____________________________________...R....
display local_symbols_in skdemo.c:


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

Displaying Memory in Mnemonic Format

You can display, in mnemonic format, the absolute code in memory. For example to display the memory of the demo program,

```
display memory main mnemonic <RETURN>
```

```
 Memory  :mnemonic :file = skdemo.c:
   address    data
  8020 0029  C8000000    PREPARE 0000,00
  8020 002D  680508      PUSH 0805
  8020 0030  680000      PUSH 0000
  8020 0033  9ACB001080  CALL FAR PTR 801CB
  8020 0038  0F363E1000  MOV DS3,IY,DWORD PTR 0010
  8020 003D  0F76        PUSH DS3/VPC
  8020 003F  57          PUSH IY
  8020 0040  0F3E160000  MOV DS2,DW,DWORD PTR 0000
  8020 0045  0F7E        PUSH DS2
  8020 0047  52          PUSH DW
  8020 0048  9AFA001080  CALL FAR PTR 801FA
  8020 004D  83C404      ADD SP,04
  8020 0050  0F363E0800  MOV DS3,IY,DWORD PTR 0008
  8020 0055  D6C60566    MOV DS3:BYTE PTR [IY],66
  8020 0059  0F3E1E0C00  MOV DS2,BW,DWORD PTR 000C
  8020 005E  63C60766    MOV DS2:BYTE PTR [BW],66

STATUS:   n70433--Running in monitor____________________________________...R....
display memory main mnemonic


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

Notice that you can use symbols when specifying expressions. The global symbol **main** is used in the command above to specify the starting address of the memory to be displayed.

Display Memory with Symbols

If you want to see symbol information with displaying memory in mnemonic format, the emulator Softkey Interface provides "set symbols" command. To see symbol information, enter the following command.

```
set symbols on <RETURN>
```

```
 Memory  :mnemonic :file = skdemo.c:
   address   label          data
  8020 0029     :_main     C8000000    PREPARE 0000,00R  8020 002D
680508      PUSH 0805
  8020 0030                680000      PUSH 0000
  8020 0033                9ACB001080  CALL FAR PTR :_rcopy
  8020 0038                0F363E1000  MOV DS3,IY,DWORD PTR 0010
  8020 003D                0F76        PUSH DS3/VPC
  8020 003F                57          PUSH IY
  8020 0040                0F3E160000  MOV DS2,DW,DWORD PTR 0000
  8020 0045                0F7E        PUSH DS2
  8020 0047                52          PUSH DW
  8020 0048                9AFA001080  CALL FAR PTR :_strcpy
  8020 004D                83C404      ADD SP,04
  8020 0050                0F363E0800  MOV DS3,IY,DWORD PTR 0008
  8020 0055                D6C60566    MOV DS3:BYTE PTR [IY],66
  8020 0059                0F3E1E0C00  MOV DS2,BW,DWORD PTR 000C
  8020 005E                63C60766    MOV DS2:BYTE PTR [BW],66

STATUS:   n70433--Running in monitor____________________________________...R....
set symbols  on


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

As you can see, the memory display shows symbol information.

Display Memory with Source Code

If you want to reference the source line information with displaying memory in mnemonic format, the emulator Softkey Interface provides "set source" command. To reference the source line information in inverse video, enter the following command:

```
set source on inverse_video on <RETURN>
```

```
 Memory  :mnemonic :file = skdemo.c:
   address   label          data
       1     /* "NEC V55PI DEMONSTRATION PROGRAM" */
       2     /* "skdemo.c" */
       3     #include <rcopy.h>
       4
       5     /* DEFINES */
       6     #define TRUE            1
       7     #define FALSE           0
       8     #define NO_CMD          'f'
       9     #define NEW_CMD         't'
      10     #define CMD_STARTED     's'
      11     #define CMD_FINISHED    'f'
      12     #define CMD_A           'A'
      13     #define CMD_B           'B'
      14     #define CMD_C           'C'
      15
      16     extern void_rompseg();

STATUS:   n70433--Running in monitor____________________________________...R....
set source on inverse_video on


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

To see the memory without source line referencing, enter the following command:

```
set source off <RETURN>
```

Running the Program

The "run" command lets you execute a program in memory. Entering the "run" command by itself causes the emulator to begin executing at the current program counter address. The "run from" command allows you to specify an address at which execution is to start.

From Transfer Address

The "run from transfer_address" command specifies that the emulator start executing at a previously defined "start address". Transfer addresses are defined in assembly language source files with the END assembler directive (i.e., pseudo instruction). Enter:

```
run from transfer_address <RETURN>
```

From Reset

The "run from reset" command specifies that the emulator begin executing from reset vector as actual microprocessor does.

(See "Running From Reset" section in the "In-Circuit Emulation" chapter).

Displaying Memory

The demo program "skdemo.c" alters memory upon user commands. User commands are entered by modifying the memory locations **cmd_code** and **semaphore**.

Using Symbolic Addresses

In the following display, the memory range is displayed using symbolic addresses status and cmd_code.

The memory display window is periodically updated. For example, enter the following command:

```
display memory cmd_code thru status+1fh
blocked bytes <RETURN>
```

This command string is used to specify the range of memory from **cmd_code** to **status+1fh**.

```
 Memory  :bytes :blocked :update
   address       data       :hex                                       :ascii
  1000 00A8-AF   66   00   66   00   4E   6F   20   63     f . f .  N o   c
  1000 00B0-B7   6F   6D   6D   61   6E   64   20   65     o m m a  n d   e
  1000 00B8-BF   6E   74   65   72   65   64   20   20     n t e r  e d
  1000 00C0-C7   20   20   20   00   20   20   20   00           .        .
  1000 00C8-CF   41   77   61   69   74   69   6E   67     A w a i  t i n g
  1000 00D0-D7   20   63   6F   6D   6D   61   6E   64       c o m  m a n d
  1000 00D8-DF   20   20   20   20   20   20   20   00                      .
  1000 00E0-E7   20   20   20   00   00   00   00   00           .  . . . .







STATUS:   n70433--Running user program__________________________________...R....
display memory cmd_code thru status+1fh  blocked  bytes


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

Modifying Memory

This demo program simulates a primitive command interpreter. In the program, the command code memory location is set to a desired value (initially 66H, ascii character "f"). Memory locations **semaphore** and **cmd_code** correspond to memory address 1009:0044 hex and 1009:0045 hex respectivity.

You can use the modify memory command to send commands to the sample program. For example, to enter the command ’t’ at address 1009:0044 and enter command ’A’ at address 1009:0045: use the following commands.

```
modify memory cmd_code string to ’A’ <RETURN>
modify memory semaphore string to ’t’
<RETURN>
```

After the memory location are modified, the memory display shows the following

```
 Memory  :bytes :blocked :update
   address       data       :hex                                  :ascii
  1000 00A8-AF   41   00   66   00   43   6F   6D   6D     A . f .  C o m m
  1000 00B0-B7   61   6E   64   20   27   41   27   20     a n d    ' A '
  1000 00B8-BF   65   6E   74   65   72   65   64   20     e n t e  r e d
  1000 00C0-C7   20   20   20   00   20   20   20   00           .        .
  1000 00C8-CF   43   6F   6D   6D   61   6E   64   20     C o m m  a n d
  1000 00D0-D7   72   65   63   65   69   76   65   64     r e c e  i v e d
  1000 00D8-DF   20   20   20   20   20   20   20   00                     .
  1000 00E0-E7   20   20   20   00   00   00   00   00           .  . . . .









STATUS:   n70433--Running user program__________________________________...R....
modify memory semaphore string  to 't'


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

Breaking into the Monitor

The "break" command allows you to divert emulator execution from the user program to the monitor. You can continue user program execution with the "run" command. To break emulator execution from the demo program to the monitor, enter the following command.

```
break <RETURN>
```

Notice that the current address is pointed out with inverse video in displaying memory when the execution breaks to the monitor.

Using Software Breakpoints

Software breakpoints are handled by the 70433 BRK 3 instruction. When you define or enable a software breakpoint, the emulator will replace the opcode at the software breakpoint address with a breakpoint interrupt instruction(BRK 3).

Caution

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

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. Further, your program won’t work correctly.

Note

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

Note

Because software breakpoints are implemented by replacing opcodes with the breakpoint interrupt instruction, you cannot define software breakpoints in target ROM. Them you can use software breakpoints.

When software breakpoints are enabled and the emulator detects the BRK 3 interrupt instruction, it generates a break into the monitor. Since the system controller knows the locations of defined software breakpoints, it can determine whether the BRK 3 instruction in your target program.

If the BRK 3 interrupt was generated by a software breakpoint, execution breaks to the monitor, and the breakpoint interrupt instruction(BRK 3) is replaced by original opcode. A subsequent run or step command will execute from this address.

If the BRK 3 interrupt was generated by a BRK 3 interrupt instruction in the target program, execution still breaks to the monitor, and an "undefined breakpoint" status message is displayed. To continue program execution, you must run or step from the target program's breakpoint interrupt vector address.

Enabling/Disabling Software Breakpoints

When you initially enter the Softkey Interface, software breakpoints are disabled. To enable the software breakpoints feature, enter the following command.

```
modify software_breakpoints enable <RETURN>
```

When software breakpoints are enabled and you set a software breakpoint, the 70433 breakpoint interrupt instruction (BRK 3) will be placed at the address specified. When the breakpoint interrupt instruction is executed, program execution will break into the monitor.

Setting a Software Breakpoint

To set a software breakpoint at line 68 of "skdemo.c", enter the following command.

```
modify software_breakpoints set line 70
<RETURN>
```

To see the address where the software breakpoint has been set, enter the following command:

```
display memory line 70 mnemonic <RETURN>
set source on inverse_video off <RETURN>
```

```
 Memory  :mnemonic :file = skdemo.c:
   address   label          data
      70              if (semaphore == NEW_CMD){
* 8020 0085                CC          BRK 3
  8020 0086                363E0800    OR DS0:BYTE PTR [BW][IX],AL
  8020 008A                D6803D74    CMP DS3:BYTE PTR [IY],74
  8020 008E                7575        BNE/Z :itoo/skdemo.c:+000DC
      71                  semaphore = CMD_STARTED;
  8020 0090                0F3E1E0800  MOV DS2,BW,DWORD PTR 0008
  8020 0095                63C60773    MOV DS2:BYTE PTR [BW],73
      72                  strcpy (status, command_R);
  8020 0099                0F36161800  MOV DS3,DW,DWORD PTR 0018
  8020 009E                0F76        PUSH DS3/VPC
  8020 00A0                52          PUSH DW
  8020 00A1                0F363E0000  MOV DS3,IY,DWORD PTR 0000
  8020 00A6                0F76        PUSH DS3/VPC
  8020 00A8                57          PUSH IY
  8020 00A9                9AFA001080  CALL FAR PTR :_strcpy

STATUS:   n70433--Running in monitor____________________________________...R....
set  source  on  inverse_video  off


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

The asterisk (*) in left side of the address lists points out that the software breakpoint has been set. The opcode at the software breakpoint address was replaced to the software breakpoint instruction (BRK 3).

Displaying Software Breakpoints

To display software breakpoints, enter the following command.

```
display software_breakpoints <RETURN>
```

```
Software breakpoints  :enabled
   address          label                                              status
  8020 0085         skdemo.c:                         line   70  pending








STATUS:   n70433--Running in monitor____________________________________...R....
display software_breakpoints


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

The software breakpoints display shows that the breakpoint is pending. When breakpoints are hit they become inactivated. To reactivate the breakpoint so that is "pending", you must reenter the "modify software_breakpoints set" command.

After the software breakpoint has been set, enter the following command to cause the emulator to continue executing the demo program.

`run <RETURN>`

A message on the status line shows that the software breakpoint has been hit. The status line also shows that the emulator is now executing in the monitor.

The software breakpoint address is pointed out with inverse video in displaying memory in mnemonic format. To see the software breakpoint with memory, enter the following command.

`display memory line 70` ***`mnemonic`*** `<RETURN>`

Notice that the original opcode was replaced at the address that the software breakpoint has been set.

Clearing a Software Breakpoint

To remove software breakpoint defined above, enter the following command.

```
modify software_breakpoints clear line 70
<RETURN>
```

The breakpoint is removed from the list, and the original opcode is restored if the breakpoint was pending.

To clear all software breakpoints, you can enter the following command.

```
modify software_breakpoints clear <RETURN>
```

Displaying Registers

Enter the following command to display registers. You can display the basic registers, or an individual register.

```
display registers <RETURN>
```

```
Registers

Next_PC 8020:0085H
 PSW F083   [    s    c]
  PC 0085   SP 0FF4   IX 0000   IY 00AA   BP 0FF4
  PS 8020   SS 1500  DS0 1500  DS1 0000  DS2 0100   DS3 0100
  AW 0041   BW 00AC   CW 0000   DW 0000













STATUS:   n70433--Running in monitor      Software break: 08020:00085___...R....
display registers


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

Note

To display or modify SFR(Special Function Register), you must use "display/modify register" commands.

Refer to "REGISTER CLASS and NAME" section in "Using the Emulator" chapter .

Stepping Through the Program

The step command allows you to step through program execution an instruction or a number of instructions at a time. Also, you can step from the current program counter or from a specific address. To step through the example program from the address of the software breakpoint set earlier, enter the following command.

```
step <RETURN>, <RETURN>, <RETURN>, ...
```

You will see the inverse-video moves according to the step execution. You can continue to step through the program just by pressing the <RETURN> key; when a command appears on the command line, it may be entered by pressing <RETURN>.

```
Registers


Step_PC 8020:0085H  MOV DS3,IY,DWORD PTR 0008
Next_PC 8020:008AH
 PSW F083   [    s    c]
  PC 008A   SP 0FF4   IX 0000   IY 00AA   BP 0FF4
  PS 8020   SS 1500  DS0 1500  DS1 0000  DS2 0100   DS3 0100
  AW 0041   BW 00AC   CW 0000   DW 0000


Step_PC 8020:008AH  CMP DS3:BYTE PTR [IY],74
Next_PC 8020:008EH
 PSW F083   [    s    c]
  PC 008E   SP 0FF4   IX 0000   IY 00AA   BP 0FF4
  PS 8020   SS 1500  DS0 1500  DS1 0000  DS2 0100   DS3 0100
  AW 0041   BW 00AC   CW 0000   DW 0000


STATUS:   n70433--Stepping complete_____________________________________...R....
step
```

You can step program execution by source lines, enter:

```
step source <RETURN>
```

Source line stepping is implemented by single stepping assembly instructions until the next PC is outside of the address range of the current source line. When source line stepping is attempted on assembly code, stepping will complete when a source line is found. To terminate stepping type <Ctrl>-C.

Note

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

1) Instructions that manipulate sreg or xsreg: MOV sreg,reg16, MOV sreg,mem16, POP sreg, MOV xsreg,reg, MOV xsreg, mem16, POP xsreg.
2) Prefix instructions: PS:, SS:, DS0:, DS1:, DS2:, DS3:, IRAM, REPC, REPNC, REP, REPE, REPZ, REPNE, REPNZ, BUSLOCK.
3) EI, DI,RETI, RETRBI.
4) POP PSW.
5) FINT.
6) BRKCS, BRK 3, BRK imm8, BRKV(V=1), CHKIND(mem32>reg16 or (mem32+2)<reg16), FPO1 fp-op, FPO1 fp-op,mem;, FPO2 fp-op, FPO2 fp-op,mem;, TSKSW, MOVSPB, MOVSPA.
7) RSTWDT.
8) The first instruction of interrupt processing routine.

Enter the following command to cause sample program execution to continue from the current program counter.

```
run <RETURN>
```

Using the Analyzer

HP 64700 emulators contain an emulation analyzer. The emulation analyzer monitors the internal emulation lines (address, data, and status). Optionally, you may have an additional 16 trace signals which monitor external input lines. The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

Source Line Referencing

A trace may be taken and displayed using source line referencing. Also, lines of the source program can be displayed with the trace list where the trace occurred.

To display the trace with source code in inverse video, enter the following command:

`set` ***`source on inverse_video on`*** `<RETURN>`

Specifying a Simple Trigger

Suppose you want you trace program execution after the point at which the sample program read the byte value 74H('t') from the address semaphore. The following command make this trace specification.

`trace after semaphore` ***`data`*** `0xx74h` ***`status read`*** `<RETURN>`

Note that the analyzer is to search for a lower byte read of 74H because the address is even.

The STATUS message shows "Emulation trace started."

When the memory location (semaphore) is modified, the trace is completed, and the STATUS message shows "Emulation trace complete." The program acts upon the case statement where cmd_code was changed in memory to 'A'. Enter the following command:

`modify memory semaphore` ***`string to`*** `'t' <RETURN>`

Display the Trace

The trace listings which following are of program execution on the 70433 emulator. To see the trace list, enter the following command:

```
display trace <RETURN>
```

```
Trace List                  Offset=0
Label:      Address       Data    Opcode or Status w/ Source Lines   time count
Base:       symbols       hex            mnemonic w/symbols           relative
after  skdemo:semaphore     FF74    xx74  read mem                      2.6   uS
+001   :skdemo.:+000065     FF74  BNE/Z :itoo/skdemo.c:+000DC         320     nS
+002   :skdemo.:+000067     3E0F    3E0F  fetch                         1.6   uS
+003   :skdemo.:+000069     081E    081E  fetch                         1.9   uS
      ##########skdemo.c - line    71 ##########################################
                  semaphore = CMD_STARTED;
+004   :skdemo.:+000067     081E  MOV DS2,BW,DWORD PTR 0008           320     nS
+005   :skdemo.:+00006B     6300    6300  fetch                         3.5   uS
+006   :skdemo.:+00006D     07C6    07C6  fetch                         1.9   uS
+007          015008        00AA    00AA  read mem                      1.9   uS
+008   :skdemo.:+00006F     0F73    0F73  fetch                         1.9   uS
+009          01500A        0100    0100  read mem                      1.9   uS
+010   :skdemo.:+00006C     0100  MOV DS2:BYTE PTR [BW],73            320     nS
+011   skdemo:semaphore     0073    xx73  write mem                     3.5   uS
+012   :skdemo.:+000071     1636    1636  fetch                         1.9   uS

STATUS:   n70433--Running user program    Emulation trace complete______...R....
display trace


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

The trace list shows the trace after line (semaphore = CMD_STARTED).

To list the next lines of the trace, press the <PGDN> or <NEXT> key.

Displaying Trace with No Symbol

The trace listing shown above has symbol information because of the "**set symbols on**" setting before in this chapter. To see the trace listing with no symbol information, enter the following command.

```
set symbols off <RETURN>
```

```
Trace List                    Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                    relative
after    0100AA     FF74    xx74  read mem                               2.6   uS
+001     08028E     FF74  BNE/Z 80305                                  320     nS
+002     080290     3E0F    3E0F  fetch                                  1.6   uS
+003     080292     081E    081E  fetch                                  1.9   uS
      ##########skdemo.c - line    71 ##########################################
                  semaphore = CMD_STARTED;
+004     080290     081E  MOV DS2,BW,DWORD PTR 0008                    320     nS
+005     080294     6300    6300  fetch                                  3.5   uS
+006     080296     07C6    07C6  fetch                                  1.9   uS
+007     015008     00AA    00AA  read mem                               1.9   uS
+008     080298     0F73    0F73  fetch                                  1.9   uS
+009     01500A     0100    0100  read mem                               1.9   uS
+010     080295     0100  MOV DS2:BYTE PTR [BW],73                     320     nS
+011     0100AA     0073    xx73  write mem                              3.5   uS
+012     08029A     1636    1636  fetch                                  1.9   uS

STATUS:   n70433--Running user program    Emulation trace complete______...R....
set symbols  off


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

As you can see, the analysis trace display shows the trace list without symbol information.

Displaying Trace with Time Count Absolute

Enter the following command to display count information relative to the trigger state.

```
display trace count absolute <RETURN>
```

```
Trace List                  Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                    absolute
after    0100AA     FF74    xx74  read mem                          ------------
+001     08028E     FF74  BNE/Z 80305                               + 320     nS
+002     080290     3E0F    3E0F  fetch                             +   1.9   uS
+003     080292     081E    081E  fetch                             +   3.8   uS
      ##########skdemo.c - line    71 ##########################################
                  semaphore = CMD_STARTED;
+004     080290     081E  MOV DS2,BW,DWORD PTR 0008                 +   4.16  uS
+005     080294     6300    6300  fetch                             +   7.68  uS
+006     080296     07C6    07C6  fetch                             +   9.60  uS
+007     015008     00AA    00AA  read mem                          +  11.5   uS
+008     080298     0F73    0F73  fetch                             +  13.4   uS
+009     01500A     0100    0100  read mem                          +  15.4   uS
+010     080295     0100  MOV DS2:BYTE PTR [BW],73                  +  15.7   uS
+011     0100AA     0073    xx73  write mem                         +  19.2   uS
+012     08029A     1636    1636  fetch                             +  21.1   uS

STATUS:   n70433--Running user program    Emulation trace complete______...R....
display trace count  absolute


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

If you want to see the relative time of the each states, enter the following command.

```
display trace count relative <RETURN>
```

Displaying Trace with Compress Mode

If you want to see more executed instructions on a display, the 70433 emulator Softkey Interface provides compress mode for analysis display. To see trace display with compress mode, enter the following command:

```
display trace compress on <RETURN>
```

```
Trace List                   Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                    relative
after    0100AA     FF74    xx74  read mem                              2.6   uS
+001     08028E     FF74  BNE/Z 80305                                 320     nS
      ##########skdemo.c - line    71 ##########################################
                  semaphore = CMD_STARTED;
+004     080290     081E  MOV DS2,BW,DWORD PTR 0008                     3.8   uS
+007     015008     00AA    00AA  read mem                              7.36  uS
+009     01500A     0100    0100  read mem                              3.8   uS
+010     080295     0100  MOV DS2:BYTE PTR [BW],73                     320     nS
+011     0100AA     0073    xx73  write mem                             3.5   uS
      ##########skdemo.c - line    72 ##########################################
                  strcpy (status, command_R);
+013     080299     1636  MOV DS3,DW,DWORD PTR 0018                     2.6   uS
+015     015018     0060    0060  read mem                              5.12  uS
+017     01501A     0100    0100  read mem                              3.8   uS
+019     08029E     0F52  PUSH DS3/VPC                                  2.2   uS

STATUS:   n70433--Running user program    Emulation trace complete______...R....
display trace compress  on


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

As you can see, the analysis trace display shows the analysis trace lists without fetch cycles. With this command you can examine program execution easily.

If you want to see all of cycles including fetch cycles, enter following command:

```
display trace compress off <RETURN>
```

The trace display shows you all of the cycles the emulation analyzer have captured.

Reducing the Trace Depth

The default states displayed in the trace list is 256 states. To increase the number of states, use the "display trace depth" command.

```
display trace depth 512 <RETURN>
```

Using the Storage Qualifier

You can use storage qualifier to trace only states with specific conditions. Suppose that you would like to trace only states which write the message to the cmd_result area. To accomplish this, you can use the "**trace only**" command like following.

> trace after cmd_result ***only range*** cmd_result ***thru*** +1fh ***status write*** <RETURN>

Only write accesses to address ***cmd_result*** through ***cmd_result+1fh*** will be stored in the trace buffer.

Modify the command input byte with the following command.

> modify memory semaphore ***string to*** 't' <RETURN>

```
Trace List                    Offset=0
Label:  Address   Data          Opcode or Status w/ Source Lines       time count
Base:     hex     hex                       mnemonic                     relative
after    0100AC     0043    xx43  write mem                           ------------
+001     0100AD     6F00    6Fxx  write mem                              57.60  uS
+002     0100AE     006D    xx6D  write mem                              57.60  uS
+003     0100AF     6D00    6Dxx  write mem                              55.68  uS
+004     0100B0     0061    xx61  write mem                              57.60  uS
+005     0100B1     6E00    6Exx  write mem                              55.68  uS
+006     0100B2     0064    xx64  write mem                              55.68  uS
+007     0100B3     2000    20xx  write mem                              58.88  uS
+008     0100B4     0027    xx27  write mem                              57.60  uS
+009     0100B5     4100    41xx  write mem                              55.04  uS
+010     0100B6     0027    xx27  write mem                              57.60  uS
+011     0100B7     2000    20xx  write mem                              55.04  uS
+012     0100B8     0065    xx65  write mem                              57.60  uS
+013     0100B9     6E00    6Exx  write mem                              55.04  uS
+014     0100BA     0074    xx74  write mem                              57.60  uS

STATUS:   n70433--Running user program    Emulation trace started_______...R....
modify memory semaphore string  to 't'


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

The display shows that the message bytes are written to the location **cmd_result**. You will find the status line still shows "Emulation trace started" because the analyzer trace buffer is not filled up. As the length of resulting message consists of 24 bytes, only 24 states are stored in the trace buffer. If you want to stop the trace, enter the following command.

> stop_trace <RETURN>

The status line will shows "Emulation trace halted".

Trigger the Analyzer at an Instruction Execution State

The emulator analyzer can capture states of instruction execution. If you want to trigger the analyzer when an instruction at a desired address is executed, you should not set up the analyzer trigger condition to detect only the address. If you do so, the analyzer will be also triggered in case that the address is accessed to fetch the instruction, or read the data from address. You should use the "**exec**" status qualifier.Suppose that you want to trace the states of the execution after the instruction at *line 83* of the *skdemo.c* file, enter the following command. The *line 83* of the file *skdemo.c* is executed when the memory location "cmd_code" is set ’C’ and "semaphore" is set ’t’.

```
trace after line 83 status exec <RETURN>
```

The message "Emulation trace started" will appear on the status line. To trigger the analyzer, enter the following commands.

```
modify memory cmd_code string to ’C’ <RETURN>
modify memory semaphore string to ’t’
<RETURN>
```

The status line now shows "Emulation trace complete".

```
Trace List                  Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                   mnemonic                      relative
after    0802EA     243E  INSTRUCTION--opcode unavailable           ------------
+001     0802EE     0F00    0F00  fetch                                 1.6   uS
+002     0802F0     5776    5776  fetch                                 1.9   uS
+003     015024     0018    0018  read mem                              1.9   uS
+004     0802F2     3E0F    3E0F  fetch                                 1.9   uS
+005     015026     0100    0100  read mem                              3.8   uS
+006     0802EF     0100  PUSH DS3/VPC                                320     nS
+007     0802F4     0416    0416  fetch                                 1.6   uS
+008     0802F1     0416  PUSH IY                                     320     nS
+009     015FF2     0100    0100  write mem                             1.6   uS
+010     0802F2     0100  MOV DS2,DW,DWORD PTR 0004                   640     nS
+011     015FF0     0018    0018  write mem                             1.3   uS
+012     0802F6     0F00    0F00  fetch                                 3.8   uS
+013     0802F8     527E    527E  fetch                                 1.9   uS
+014     015004     00AC    00AC  read mem                              1.9   uS

STATUS:   n70433--Running user program    Emulation trace complete______...R....
modify memory semaphore string  to ’t’


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

The emulator has disassemble capability in trace listing. When the emulator disassembles instructions in stored trace information, the fetch cycles of each instruction are required. When you displayed the results of analyzer trace, some lines which include "INSTRUCTION--opcode unavailable" message were displayed. Each line was instruction execution cycle at the address in the left side of the displayed because the fetch states for the instructions were not stored by the analyzer.

To display complete disassembles in the trace listing, you should modify location of trigger state in trace list, referred to as the "trigger position", to "**about**" instead of "**after**".

Emulator Analysis Status Qualifiers

The following analysis status qualifiers may also be used with the 70433 emulator.

For a Complete Description

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

Qualifier	Status bits	Description
bg	0xxxxxxxxxxxxxxxx0xy	background
cpu	0x0000xx01xxx1xxxxy	cpu cycle
dma	0xx0000xx11xxx1xxxxy	DMA memory access
exec	0xxxx01xxxxxxxxxxxxy	execute instruction
fetch	0xx0000xx10xxx1xxxxy	program fetch
fg	0xxxxxxxxxxxxxxxx1xy	foreground
grd	0xxxx00xxxxxxx1x0xxy	guarded memory access
halt	0xx11xxxxxxxxx1xxxxy	halt
hold	0xxxxxxxxxxxxx0xxxxy	hold acknowledge
int	0xxxx10xxxxxxxxxxxxy	interrupt acknowledge
io	0xx0000xx01x001xxxxy	I/O access
mem	0x10000xxxxxxx1xxxxy	memory access
memio	0xx00000111xxx1xxxxy	memory to io
memsfr	0xx00000011xxx1xxxxy	memory to sfr
ms	01x0000xx01xxx1xxxxy	macro service
read	0xx0000xxx10xx1xxxxy	read
refresh	0xx0000xx001xx1xxxxy	refresh cycle
sfr	0xx0000xx01x111xxxxy	sfr access
stop	0xx10xxxxxxxxx1xxxxy	stop
write	0xx0000xxx11xx1xxxxy	write
wrrom	0xxxx00xxxxxxx10xxxy	write to rom

Resetting the Emulator

To reset the emulator, enter the following command.

```
reset <RETURN>
```

Exiting the Softkey Interface

There are several options available when exiting the Softkey Interface: exiting and releasing the emulation system, exiting with the intent of reentering (continuing), exiting locked from multiple emulation windows, and exiting (locked) and selecting the measurement system display or another module.

End Release System

To exit the Softkey Interface, releasing the emulator so that other users may use the emulator, enter the following command.

```
end release_system <RETURN>
```

Ending to Continue Later

You may also exit the Softkey Interface without specifying any options; this causes the emulator to be locked. When the emulator is locked, other users are prevented from using it and the emulator configuration is saved so that it can be restored the next time you enter (continue) the Softkey Interface.

```
end <RETURN>
```

Ending Locked from All Windows

When using the Softkey Interface from within window systems, the "end" command with no options causes an exit only in that window. To end locked from all windows, enter the following command.

```
end locked <RETURN>
```

This option only appears when you enter the Softkey Interface via the **emul700** command. When you enter the Softkey Interface via ***pmon*** and **MEAS_SYS**, only one window is permitted.

Refer to the *Softkey Interface Reference* manual for more information on using the Softkey Interface with window systems.

Selecting the Measurement System Display or Another Module

When you enter the Softkey Interface via **pmon** and **MEAS_SYS**, you have the option to select the measurement system display or another module in the measurement system when exiting the Softkey Interface. This type of exit is also "locked"; that is, you can continue the emulation session later. For example, to exit and select the measurement system display, enter the following command.

```
end select measurement_system <RETURN>
```

This option is not available if you have entered the Softkey Interface via the **emul700** command.

Notes

3

In-Circuit Emulation Topics

Introduction

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

This chapter will:

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

Prerequisites

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

Installing the Emulator Probe into a Target System

The 70433 emulator probe has a 132-pin PGA connector;
The emulator probe is also provided with a conductive pin protector to protect the delicate gold-plated pins of the probe connector from damage due to impact. Since the protector is non-conductive, you may run performance verification with no adverse effects when the emulator is out-of-circuit.

Caution

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

Caution

Make sure target system power is OFF. Do not install the emulator probe into the target system microprocessor socket with power applied to the target system. The emulator may be damaged if target system power is not removed before probe installation.

Caution

Make sure pin 1 of probe connector is aligned with pin 1 of the socket. When installing the emulation probe, be sure that probe is inserted into the processor socket so that pin 1 of the connector aligns with pin 1 of the socket. Damage to the emulator probe will result if the probe is incorrectly installed.

Caution

Protect your target system CMOS components. If you target system contains any CMOS components, turn ON the target system first, then turn ON the emulator. Likewise, turn OFF your emulator first, then turn OFF the target system.

Pin Protector

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

Conductive Pin Guard

HP emulators are shipped with a conductive plastic or conductive foam pin guard over the target system probe pins. This guard is designed to prevent impact damage to the pins and should be left in place while you are not using the emulator. However, when you do use the emulator, either for normal emulation tasks, or to run performance verification on the emulator, you must remove this conductive pin guard to avoid intermittent failures due to the target system probe lines being shorted together.

Caution

Always use the pin protectors and guards as described above. Failure to use these devices may result in damage to the target system probe pins. Replacing the target system probe is expensive; the entire probe and cable assembly must be replaced because of the wiring technology employed.

Installing into a PGA Type Socket

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

- Remove the 70433 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.

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 QFP Type Socket

To connect the 70433 emulator microprocessor connector to the NEC EV-9200GD-120 socket on the target system, use the NEC EV-95001GD-120 adapter.

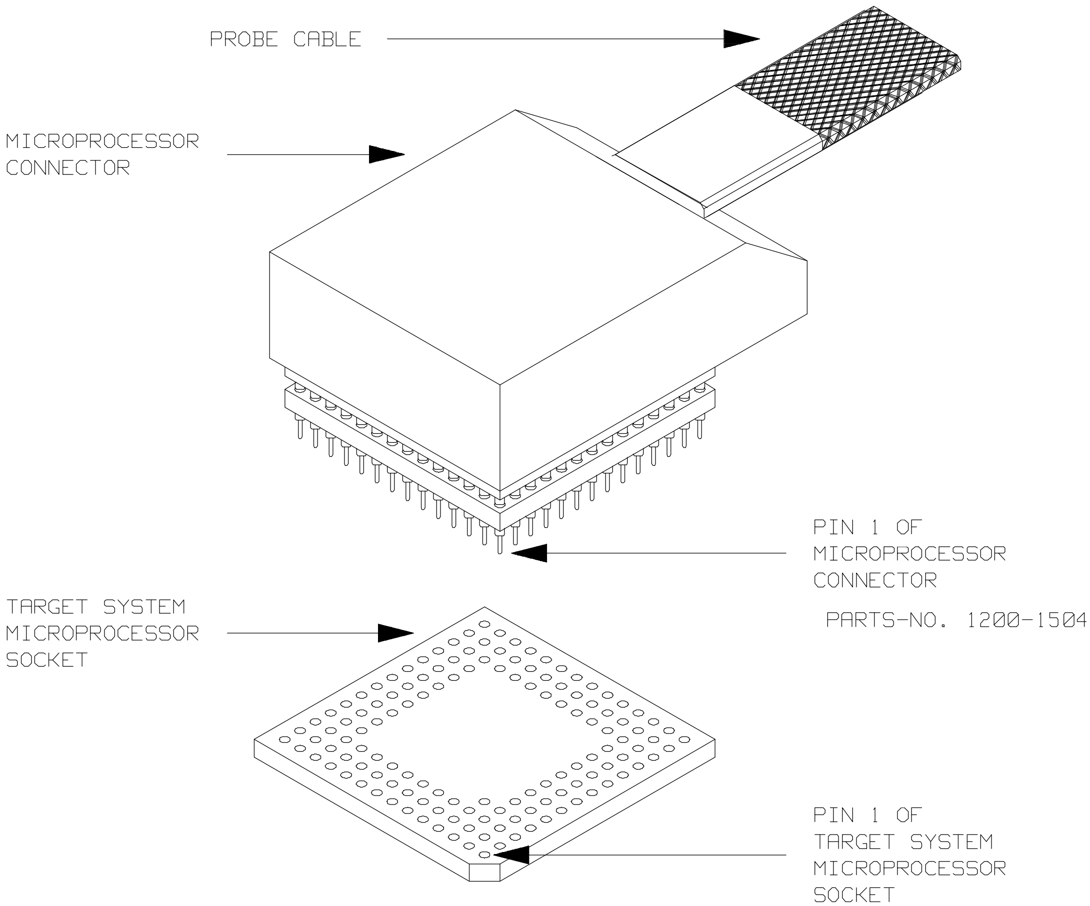


Figure 3-1 Installing into a 70433 PGA type socket

In-Circuit Configuration Options

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

Using the Target System Clock Source

The default emulator configuration selects the internal 12.5 MHz (system clock speed) clock as the emulator clock source. You should configure the emulator to select an external target system clock source for the "in-circuit" emulation.

Allowing the Target System to Insert Wait States

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

Enabling NMI, HLDRQ and RESET Input from the Target System

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

Running the Emulator from Target Reset

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

At First, you must specify the emulator responds to RESET signal by the target system (see the "Enable RESET inputs from target system?" configuration in "Configuring the Emulator" chapter of this manual).

To specify a run from target system reset, enter the follwing command:

```
run from reset <RESET>
```

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

Note

In the "Awaiting target reset" status(T>), you can not break into the monitor. If you enter "r rst" in out-of-circuit or in the configuration that emulator does not accepted target system reset(cf rst=dis), you must reset the emulator.

The 70433 emulator supports power on reset. If you want program to be executed by power on reset, execute the following process.

1) Enter "reset"

2) Turn OFF your target system

3) Enter "run from reset"

4) Turn ON your target system

Note

When you enter "r from reset", you will see c> system prompt if you use external clock. This status is the same as "Awaiting target reset" status.

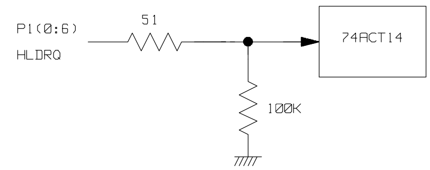
Pin State in Background

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

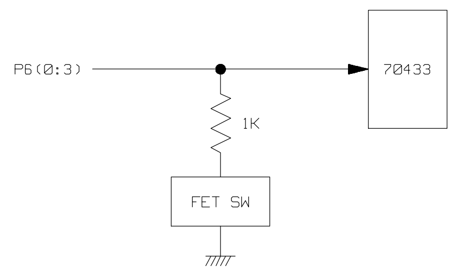
Address Bus	Same as foreground
Data Bus	Always high impedance otherwise you direct the emulator to access target memory. When accessing target memory, I/O by background monitor, same as foreground.
ASTB	Same as foreground.
DEX	Same as foreground
WRL,$\overline{\text{WRH}}$	Always high level. Except when accessing target memory, I/O by background monitor, same as foreground.
RD	Same as foreground except for emulation memory write. When accessing emulation memory, low.
Other	Same as foreground

Target System Interface

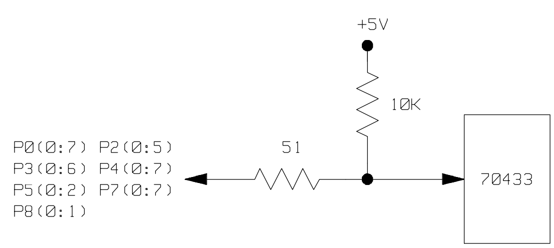
P1(0:6) HLDRQ — These signals are connected to 74ACT14 through 51 ohm series register and 100K ohm pull-down register.



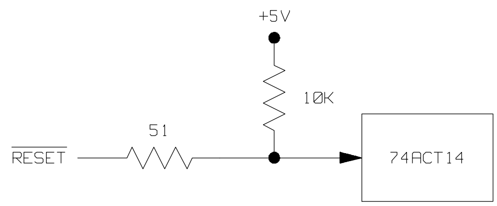
P6(0:3) — These signals are connected to 70433 emulation processor and FET Switch through 1K ohm register.



P0(0:7) P2(0:5) P3(0:6) P4(0:7) P5(0:2) P7(0:7) P8(0:1) — These signals are connected to 70433 emulation processor through 51 ohm register and 10K ohm pull-up register.



RESET — This signal is connected to 74ACT14 through 51 ohm register and 10K ohm pull-up register.



Other signals — These signals are connected to 74FCT245 or 74FCT244 through 51 ohm register and 10K ohm pull-up register.

4

Configuring the Emulator

Introduction

The 64768 emulator can be used in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing target system software, or you can use the emulator in-circuit when integrating software with target system hardware. Emulation memory can be used in place of, or along with, target system memory. You can use the emulator's internal clock or the target system clock. You can execute target programs in real-time or allow emulator execution to be diverted into the monitor when commands request access of target system resources (target system memory, register contents, etc.)

The emulator is a flexible instrument and it may be configured to suit your needs at any stage of the development process. This chapter describes the options available when configuring the 7330 emulator.

The configuration options are accessed with the following command.

```
modify configuration <RETURN>
```

After entering the command above, you will be asked questions regarding the emulator configuration. The configuration questions are listed below and grouped into the following classes.

General Emulator Configuration:

- Specifying the emulator clock source. (Internal/external.)
- Selecting monitor entry after configuration.
- Restricting to real-time execution.

Memory Configuration:

- Selecting the emulation monitor type.
- Specifying the monitor location.
- Mapping memory.

Emulator Pod Configuration:

- Selecting Date bus size.
- Selecting mnemonic type for memory display.
- Selecting algorithm for physical run addresses.
- Specifying Reset value for the stack segment.
- Specifying Reset value for the stack pointer.
- Enabling RESET inputs from target system.
- Enabling NMI inputs from target system.
- Enabling READY inputs from target system.
- Enabling HLDRQ inputs from target system.
- Selecting target memory access size.

Debug/Trace Configuration:

- Enabling breaks on writes to ROM.
- Specifying tracing of foreground/background cycles.
- Specifying tracing of internal DMA cycles.

– Specifying tracing of refresh cycles.

Simulated I/O Configuration: Simulated I/O is described in the *Simulated I/O reference manual*.

External Analyzer Configuration: See the *Analyzer Softkey Interface User's Guide.*

Interactive Measurement Configuration: See the chapter on coordinated measurements in the *Softkey Interface Reference* manual.

General Emulator Configuration

The configuration questions described in this section involve general emulator operation.

Micro-processor Clock Source?

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

internal	Selects the internal clock oscillator as the emulator clock source. The emulators' internal clock speed is 12.5 MHz (system clock).
external	Selects an external target system clock source. In the case of HP 64768A, the emulator runs with target system clock from 4 to 25 MHz. And, in the case of HP 64768A, the emulator runs with target system clock from 4 to 25 MHz. And,

Note

Changing the clock source drives the emulator into the reset state. If you answer "yes" to the "Enter monitor after configuration?" question that follows, the emulator resets (due to the clock source change) then breaks into the monitor when the configuration is saved.

Enter Monitor After Configuration?

This question allows you to select whether the emulator will be running in the monitor or held in the reset state upon completion of the emulator configuration.

How you answer this configuration question is important in some situations. For example, when the external clock has been selected and the target system is turned off, reset to monitor should not be selected; otherwise, configuration will fail. When an external clock source is specified, this question becomes "Enter monitor after configuration (using external clock)?" and the default answer becomes "no".

yes — When reset to monitor is selected, the emulator will be running in the monitor after configuration is completed. If the reset to monitor fails, the previous configuration will be restored.

no — After the configuration is complete, the emulator will be held in the reset state.

Restrict to Real-Time Runs?

This configuration allows to you 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.

no — All commands, regardless of whether or not they require a break to the emulation monitor, are accepted by the emulator.

yes — When runs are restricted to real-time and the emulator is running the user program, all commands that cause a break (except "reset", "break", "run", and "step") are refused. For example, the following commands are not allowed when runs are restricted to real-time:

- Display/modify registers.
- Display/modify target system memory.
- Display/modify I/O.

Caution

If your target system circuitry is dependent on constant execution of program code, you should restrict the emulator to real-time runs. This will help insure that target system damage does not occur. However, remember that you can still execute the "reset", "break", and "step" commands; you should use caution in executing these commands.

Memory Configuration

The memory configuration questions allows you to select the monitor type, to select the location of the monitor, and to map memory. To access the memory configuration questions, you must answer "yes" to the following question.

Modify memory configuration?

Monitor Type?

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 user program. Monitor program execution can take place in the "background" or "foreground" emulator modes.

In the *foreground* emulator mode, the emulator operates as would the target system processor.
In the *background* emulator mode, foreground execution is suspended so that the emulation processor may be used for communication with the system controller, typically to perform tasks which access target system resources.

A *background monitor* program operates entirely in the background emulator mode; that is, the monitor program does not execute as if it were part of the target program. The background monitor 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.

A *foreground monitor* program performs its tasks in the foreground emulator mode; that is, the monitor program executes as if it were part of the target program. Breaks into the monitor always put the emulator in the background mode; however, foreground monitors switch back to the foreground mode before performing monitor functions.

Note

All memory mapper terms are deleted when the monitor type is changed!

background	The default emulator configuration selects the background monitor. A memory overlay is created and the background monitor is loaded into that area.

Note

While running in background monitor, the 64768 emulator ignores target system reset.

When the background monitor is selected, the execution of the monitor is hidden from the target system (except for background cycles). When you select the background monitor and the current monitor type is "foreground", you are asked the next question.

1. Reset map (change of monitor type requires map reset)?

This question will be asked if you change the monitor type (in this case, you have changed the monitor type from "foreground" to "background"). This question reminds you that the map will be reset and allows you to confirm your decision.

no	The memory map is not reset, and the monitor type is not changed.
yes	This memory map is reset due to the change in monitor type.

foreground	When you select the foreground monitor, processor address space is taken up. The foreground monitor takes up 2K bytes of memory. When the foreground monitor is selected, breaking into the monitor still occurs in a brief background state, but the rest of the monitor program, the saving of registers and the dispatching of emulation commands, is executed in foreground.

Note

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

When you select the foreground monitor and the current monitor type is "background", you are asked the next question.

1. Reset map (change of monitor type requires map reset)?

This question will be asked if you change the monitor type (in this case, you have changed the monitor type from "background" to "foreground"). This question reminds you that the map will be reset and allows you to confirm your decision.

no	The memory map is not reset, and the monitor type is not changed.
yes	This memory map is reset due to the change in monitor type.

2. Foreground monitor location?

You can relocate the monitor to any 2K byte boundary. The location of a foreground monitor is important because it will occupy part of the processor address space. Foreground monitor locations must not overlap the locations of target system programs. When entering monitor block addresses, you must only specify addresses on 2K byte boundaries; otherwise, the configuration will be invalid, and the previous configuration will be restored.

Note

You should not load the foreground monitor provided with the 70433 emulator at the base address 0 or 0ff800 hex; because the 70433 microprocessor's vector table and SFR are located respectively.

3. Monitor filename?

This question allows you to specify the name of the foreground monitor program absolute file. Remember that you must assemble and link your foreground monitor starting at the 2K byte boundary specified for the previous "Foreground monitor location?" question.

The monitor program will loaded after you have answered all the configuration questions.

Only the 2k bytes of memory reserved for the monitor are loaded at the end of configuration; therefore, you should not link the foreground monitor to the user program. If it is important that the symbol database contain both monitor and user program symbols, you can create a different absolute file in which the monitor and user program are linked. Then, you can load this file after configuration.

Using the Foreground Monitor. When using the foreground monitor, your program should set up a stack. The foreground monitor assumes that there is a stack in the foreground program, and this stack is used to save PS, PC, and PSW upon entry into the monitor.

Mapping Memory

The emulation memory consists of 128k, 512k or 1M bytes, mappable in 256 byte blocks. However, you may use 126k,510k or 1022k bytes of emulation memory for your target system, because 2k bytes of emulation memory is occupied by the monitor. The emulation memory system does not introduce wait states.

Note

You can insert wait states on accessing emulation memory. Refer to the "Enable READY input from the target system?" section in this chapter.

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.

When a foreground monitor selected, a 2k byte block is automatically mapped at the address specified by the "Foreground monitor location?" question.

Note

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

Blocks of memory can also be characterized as guarded memory. Guarded memory accesses will generate "break to monitor" requests. Writes to ROM will generate "break to monitor" requests if the "Enable breaks on writes to ROM?" configuration item is enabled (see the "Debug/Trace Configuration" section which follows).

Determining the Locations to be Mapped

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. The linker load map listing will show what locations your program will occupy in memory.

Emulator Pod Configuration

To access the emulator pod configuration questions, you must answer "yes" to the following question.

Modify emulator pod configuration?

Date bus size?

This configuration specifies which the data bus size microprocessor operates with 8bit or 16bit.

16	Selecting 16bit data bus size specifies that the microprocessor operates with 16bit data bus size.
8	Selecting 8bit data bus size specifies that the microprocessor operates with 8bit data bus size.

Note

The 64768 emulator operates in accordance with this configuration instead of D8/16 signal from target system. D/8/16 signal from target system is ignored.

Note

Changing the data bus size drives the emulator into the reset state. If you answer "yes" to the "Enter monitor after configuration?" question, the emulator resets (due to the data bus size change) then breaks into the monitor when the configuration is saved.

Memory display mnemonic?

This configuration specifies the type of mnemonic that are used by the monitor program to display memory. When a command requests the monitor to display memory, the monitor program will look at the mnemonic type setting to determine whether uPD70433(V55PI) or iAPX86/10(8086) mnemonic should be used.

70433	Selecting the 70433 mnemonic type specifies that the emulator will display memory in uPD70433(V55PI) mnemonic.

8086 — Selecting the 8086 mnemonic type specifies that the emulator will display memory in iAPX86(8086) mnemonic.

The default emulator configuration selects the **70433** mnemonic type at power up initialization.

Note

The instruction that is not include iAPX86/10 mnemonic is displayed with uPD70433 mnemonic even if you specify this item is **8086**.

Segment Algorithm?

The run and step commands allow you to enter addresses in either logical form (segment:offset, e.g., 0F000H:0000H) or physical form (e.g., 0F0000H). When a physical address (non-segmented) is entered with either a run or step command, the emulator must convert it to a logical (segment:offset) address.

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

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

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

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

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

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

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

Reset value for the stack segment?

This question allows you to specify the stack segment(SS) after the emulation reset. This configuration is useful only if foreground monitor is used.

Reset value for the stack pointer?

This question allows you to specify the stack pointer(SP) after the emulation reset. This configuration is useful only if foreground monitor is used.

Note

When you are using the foreground monitor, the stack address should be defined in an emulation memory or target system RAM area which is not used by target program.

Respond RESET from target system?

The 64768 emulator can respond or ignore target system reset while running in user program or waiting for target system reset (refer to "run from reset" command in the *Softkey Interface Reference* manual). While running in background monitor, the 64768 emulator ignores target system reset completely independent on this setting.

yes	Specify that, this is a default configuration, make the emulator to respond to reset from target system. In this configuration, emulator will accept reset and execute from reset vector (0FFFF0 hex) as same manner as actual microprocessor after reset is inactivated.
no	The emulator ignores reset signal from target system completely, even while in foreground (executing user program).

Respond NMI from target system?

This question allows you to specify whether or not the emulation processor accepts NMI signal generated by the target system.

yes	The emulator accepts NMI signal generated by the target system. When the NMI is accepted, the emulator calls the NMI procedure as actual microprocessor. Therefore, you need to set up the

	NMI vector table, if you want to use the NMI interrupt.
no	The emulator ignores NMI signal from target system completely.

Note

When target NMI signal is enabled, it is in effect while the emulator is running the target program. While the emulator is running background monitor, NMI will be suspended until the emulator goes into foreground operation.

Respond READY from target system?

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

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

Respond to HLDRQ from target system

This configuration allows you to specify whether or not the emulator accepts HLDRQ(Bus Hold Request) signal generated by the target system.

yes	The emulator accepts HLDRQ signal. When the HLDRQ is accepted, the emulator will respond as actual microprocessor.
no	The emulator ignore HLDRQ signal from target system completely.

Target memory access size

This configuration 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.

Words	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. At an odd address, the emulator will access target memory using byte cycles.
Bytes	Selecting the byte access mode specifies that the emulator will access target memory using upper and lower byte cycles (one byte at a time).

The default emulator configuration selects the **byte** access size 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.

Debug/Trace Configuration

The debug/trace configuration questions allows you to specify breaks on writes to ROM, enable/disable the software breakpoints feature, and specify that the analyzer trace foreground/background execution. To access the debug/trace configuration questions, you must answer "yes" to the following question.

Modify debug/trace options?

Break Processor on Write to ROM?

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

yes	Causes the emulator to break into the emulation monitor whenever the user program attempts to write to a memory region mapped as ROM.
no	The emulator will not break to the monitor upon a write to ROM. The emulator will not modify the memory location if it is in emulation ROM.

Note

The **wrrom** trace command status option allows you to use "write to ROM" cycles as trigger and storage qualifiers. For example, you could use the following command to trace about a write to ROM:

```
trace about status wrrom <RETURN>
```

Trace Background or Foreground Operation?

This question allows you to specify whether the analyzer trace only foreground emulation processor cycles, only background cycles, or both foreground or background cycles.

foreground	Specifies that the analyzer trace only foreground cycles. This option is specified by the default emulator configuration.
background	Specifies that the analyzer trace only background cycles. (This is rarely a useful setting.)
both	Specifies that the analyzer trace both foreground and background cycles. You may wish to specify this option so that all emulation processor cycles may be viewed in the trace display.

Trace Internal DMA cycles?

This question allows you to specify whether or not the analyzer trace the 70433 emulation processor's internal DMA cycles.

yes	Specifies that the analyzer will trace the 70433 internal DMA cycles.
no	Specifies that the analyzer will not trace the 70433 internal DMA cycles.

Trace refresh cycles?

This question allows you to specify whether or not the analyzer trace the 64768 emulation processor's refresh cycles.

no	Specifies that the analyzer will not trace the 70433 refresh cycles.
yes	Specifies that the analyzer will trace the 70433 refresh cycles.

Simulated I/O Configuration

The simulated I/O feature and configuration options are described in the *Simulated I/O reference* manual.

Interactive Measurement Configuration

The interactive measurement configuration questions are described in the chapter on coordinated measurements in the *Softkey Interface Reference* manual. Examples of coordinated measurements that can be performed between the emulator and the emulation analyzer are found in the "Using the Emulator" chapter.

Saving a Configuration

The last configuration question allows you to save the previous configuration specifications in a file which can be loaded back into the emulator at a later time.

Configuration file name? <FILE>

The name of the last configuration file is shown, or no filename is shown if you are modifying the default emulator configuration.

If you press <RETURN> without specifying a filename, the configuration is saved to a temporary file. This file is deleted when you exit the Softkey Interface with the "end release_system" command.

When you specify a filename, the configuration will be saved to a file; the filename specified with extensions of ".EA". . The file with the ".EA" extension is the "source" copy of the file.

Ending out of emulation (with the "end" command) saves the current configuration, including the name of the most recently loaded configuration file, into a "continue" file. The continue file is not normally accessed.

Loading a Configuration

Configuration files which have been previously saved may be loaded with the following Softkey Interface command.

```
load configuration <FILE> <RETURN>
```

This feature is especially useful after you have exited the Softkey Interface with the "end release_system" command; it saves you from having to modify the default configuration and answer all the questions again. To reload the current configuration, you can enter the following command.

```
load configuration <RETURN>
```

Notes

5

Using the Emulator

Introduction

The "Getting Started" chapter shows you how to use the basic

This chapter discuss:

- Register names and classes.
- Hardware breakpoint
- Features available via "pod_command".

This chapter shows you how to:

- Access internal RAM/SFR.
- Store the contents of memory into absolute files.
- Make coordinated measurements.

REGISTER CLASS and NAME

Summary 70433 register designator. All available register class names and register names are listed below.

<REG_CLASS>

<REG_NAME>	Description
*(All basic registers)	
AW, BW	BASIC registers.
CW, DW	
BP, IX, IY	
DS0, DS1,	
DS2, DS3	
SS, SP	
PC, PS, PSW	

PORT(Port registers)

P0	Port 0	
P1	Port 1	(Read Only)
P2	Port 2	
P3	Port 3	
P4	Port 4	
P5	Port 5	
P6	Port 6	(Read Only)
P7	Port 7	
P8	Port 8	
PM0	Port 0 mode	
PM2	Port 2 mode	
PM3	Port 3 mode	
PM4	Port 4 mode	
PM5	Port 5 mode	
PM7	Port 7 mode	
PM8	Port 8 mode	
PMC2	Port 2 mode control	
PMC3	Port 3 mode control	
PMC4	Port 4 mode control	
PMC5	Port 5 mode control	
PMC7	Port 7 mode control	
PMC8	Port 8 mode control	
PRDC	Port read control	

ROP(Real-time Output port registers)

RTPC	Real-time output port control
RTPD	Real-time output port display
P7L	Port 7 buffer(Low)
P7H	Port 7 buffer(high)
RTP	Real-time output port

TIME(Timer registers)

TM0	Timer 0	
TM1	Timer 1	
TM2	Timer 2	
TM3	Timer 3	
CT00	Timer capture 00	
CT01	Timer capture 01	
CT00	Timer capture 10	
CT10	Timer capture 11	
CM00	Timer compare 00	
CM01	Timer compare 01	
CM10	Timer compare 10	
CM11	Timer compare 11	
CM20	Timer compare 20	
CM21	Timer compare 21	
CM22	Timer compare 22	
CM23	Timer compare 23	
CM30	Timer compare 30	
CM31	Timer compare 31	
TMC	Timer control	
TOC	Timer output control	
STC	Software timer counter	(Read Only)
STMC	Software timer counter compare	

PWMU(PWM uint registers)

PWM	PWM
PMWC	PWM control

DMA(DMA registers)

DMAM0	DMA mode 0
DMAM1	DMA mode 1
DMAC0	DMA control 0
DMAC1	DMA control 1
TC0	Terminal counter 0
TC1	Terminal counter 1
TCM0	Terminal counter modulo 0
TCM1	Terminal counter modulo 1
MAR0	DMA memory address 0
MAR1	DMA memory address 1
UDC0	DMA up/down counter 0
UDC1	DMA up/down counter 1
DCM0	DMA compare 0
DCM1	DMA compare 1
DPTC0	DMA read/write pointer 0
DPTC1	DMA read/write pointer 1
DMAS	DMA status

PI(Parallel I/F registers)

PAB	Parallel interface buffer
PAC0	Parallel interface control 1
PAC1	Parallel interface control 2
PAS	Parallel interface status
PAI0	Parallel interface acknowledge interval 0 (Write Only)
PAI1	Parallel interface acknowledge interval 1 (Write Only)

AD(Analog-Digital conversion registers)

ADM	A/D convertor mode	
ADCR0	A/D conversion result 0	(Read Only)
ADCR1	A/D conversion result 1	(Read Only)
ADCR2	A/D conversion result 2	(Read Only)
ADCR3	A/D conversion result 3	(Read Only)

UART(UART registers)

ASP	Protocol select	
UARTM0	UART mode 0	
UARTM1	UART mode 1	
UARTS0	UART status 0	
UARTS1	UART status 1	
RXB0	Receive buffer 0	(Read Only)
RXB1	Receive buffer 1	(Read Only)
TXB0	UART Transfer buffer 0	(Write Only)
TXB1	UART Transfer buffer 1	(Write Only)
PRS0	Prescaler 0	
PRS1	Prescaler 1	
RXBRG0	Receive baud rate generator 0	
RXBRG1	Receive baud rate generator 1	
TXBRG0	Transfer baud rate generator 0	
TXBRG1	Transfer baud rate generator 1	

CSI(Clocked serial I/F registers)

ASP	Protocol select	
CSIM0	Clocked serial interface mode 0	
CSIM1	Clocked serial interface mode 1	
SBIC0	SBI control 0	
SBIC1	SBI control 1	
RXB0	Receive buffer 0	
RXB1	Receive buffer 1	
SIO0	Clocked serial I/O shift 0	(Write Only)
SIO1	Clocked serial I/O shift 1	(Write Only)
PRS0	Receive baud rate generator 0	
PRS1	Receive baud rate generator 1	
TXBRG0	Transfer baud rate generator 0	
TXBRG1	Transfer baud rate generator 1	

PROC(Processor status registers)

STBC	Standby control
PRC	Processor control
PWC0	Programmable wait control 0
PWC1	Programmable wait control 1
RFM	Refresh mode
MBC	Memory block control
WDM	Watchdog timer mode

INTC(Interrupt control registers)

IMC	Interrupt mode control	
MK0	Interrupt mask flag 0	
MK1	Interrupt mask flag 1	
IC09	Interrupt demand control 09	
IC10	Interrupt demand control 10	
IC11	Interrupt demand control 11	
IC12	Interrupt demand control 12	
IC13	Interrupt demand control 13	
IC14	Interrupt demand control 14	
IC16	Interrupt demand control 16	
IC17	Interrupt demand control 17	
IC18	Interrupt demand control 18	
IC19	Interrupt demand control 19	
IC20	Interrupt demand control 20	
IC21	Interrupt demand control 21	
IC22	Interrupt demand control 22	
IC23	Interrupt demand control 23	
IC24	Interrupt demand control 24	
IC25	Interrupt demand control 25	
IC26	Interrupt demand control 26	
IC27	Interrupt demand control 27	
IC28	Interrupt demand control 28	
IC29	Interrupt demand control 29	
IC30	Interrupt demand control 30	
IC31	Interrupt demand control 31	
IC32	Interrupt demand control 32	
IC36	Interrupt demand control 36	
IC37	Interrupt demand control 37	
ISPR	In-service priority	(Read Only)
INTM	External interrupt mode	

BANK (register bank)

PS_<N>	ps of register bank <N>
PC_<N>	pc of register bank <N>
PSW_<N>	psw of register bank <N>
AW_<N>	aw of register bank <N>
BW_<N>	bw of register bank <N>
CW_<N>	cw of register bank <N>
DW_<N>	dw of register bank <N>
SP_<N>	sp of register bank <N>
BP_<N>	bp of register bank <N>
IX_<N>	ix of register bank <N>
IY_<N>	iy of register bank <N>
DS0_<N>	ds0 of register bank <N>
DS1_<N>	ds1 of register bank <N>
DS2_<N>	ds2 of register bank <N>
VPC_<N>	vpc of register bank <N>
SS_<N>	ss of register bank <N>

Hardware Breakpoints

The analyzer may generate a break request to the emulation processor. To break when the analyzer trigger condition is satisfied, use the "break_on_trigger" trace option.

Additionally, you can see the program states before the breakpoint in trace listing. Specify the trigger position at the end of trace listing by using "before" option.

When the trigger condition is found. the emulator execution will break into the emulation monitor. Then you can also see the trace listing mentioned above, enter the following commands.

```
trace before <QUALIFIER> break_on_trigger
<RETURN>
```

Without the trigger condition, the trigger will never occur and will never break.

Loading Program Option

You can load program any memory space with "offset_by" option. When using this option, you must specify "nosymbols" option at same time. This option is effective, when you use V Series AxLS Assembler/Linker and V Series AxLS C Compiler, and load the file at extended memory space.

To load program any memory space, enter following command:

```
load <file_name> nosymbols offset_by
<offset_addr>
```

<file_name> is the HP Absolute file(with .X suffix). <offset_addr> is the value of offset address. You can load program at address that is added offset address. For example, when you generate program to be loaded at address 5000h and specify 10000h as offset address, the program is loaded at address 15000h.

But you can not treat symbols because you must specify "nosymbols" option.

Displaying Memory Option

You can refer symbols in operand by using "with_data_segment" option. This option is available when direct addressing mode is used in the program and is effective when data segment register(DS0 or DS1) does not be often changed in the program.

Suppose you generate the following program. In the following program, direct addressing mode is used and data segment register DS0 does not be changed.

```
               COMN    SEGMENT PARA COMMON 'COMN'
3000                           DW      6FH DUP (?)
               Stk             LABEL   WORD
               COMN    ENDS

               DATA    SEGMENT PARA PUBLIC 'DATA'
2000 4141      SOU_A           DW      'AA'
2002 4242      SOU_B           DW      'BB'
2004 4343      SOU_C           DW      'CC'
2006 ????      DES_A           DW      ?
2008 ????      DES_B           DW      ?
200A ????      DES_C           DW      ?
               DATA    ENDS

               CODE    SEGMENT PARA PUBLIC 'CODE'
               ASSUME  PS:CODE,DS0:DATA,SS:COMN

1000 B80002   Init:            MOV     AW,DATA
1003 8ED8                      MOV     DS0,AW
1005 8ED0                      MOV     SS,AW
1007 BCDE00                    MOV     SP,OFFSET Stk
100A 8B1E0000  Loop:           MOV     BW,[0000H]
100E 891E0600                  MOV     [0006],BW
1012 8B1E0200                  MOV     BW,[0002H]
1016 891E0800                  MOV     [0008],BW
101A 8B1E0400                  MOV     BW,[0004H]
101E 891E0A00                  MOV     [000AH],BW
1022 33C0      Clear:          XOR     AW,AW
1024 A30600                    MOV     [0006],AW
1027 A30800                    MOV     [0008],AW
102A A30A00                    MOV     [000AH],AW
102D EBDB                      BR      Loop
               CODE    ENDS
               END     Init
```

To display memory in mnemonic format, enter following commands.

`display memory Init` ***`mnemonic`*** `<RETURN>`
`set symbols on <RETURN>`

```
 Memory  :mnemonic :file = sample.asm:
   address   label          data
  0100 0000     :Init      B80002      MOV AW,0200
  0100 0003                8ED8        MOV DS0,AW
  0100 0005                8ED0        MOV SS,AW
  0100 0007                BCDE00      MOV SP,00DE
  0100 000A  sample.:Loop  8B1E0000    MOV BW,WORD PTR 0000
  0100 000E                891E0600    MOV WORD PTR 0006,BW
  0100 0012                8B1E0200    MOV BW,WORD PTR 0002
  0100 0016                891E0800    MOV WORD PTR 0008,BW
  0100 001A                8B1E0400    MOV BW,WORD PTR 0004
  0100 001E                891E0A00    MOV WORD PTR 000A,BW
  0100 0022  sample:Clear  33C0        XOR AW,AW
  0100 0024                A30600      MOV WORD PTR 0006,AW
  0100 0027                A30800      MOV WORD PTR 0008,AW
  0100 002A                A30A00      MOV WORD PTR 000A,AW
  0100 002D                EBDB        BR SHORT /sample.asm:Loop
  0100 002F                00D4        ADD AH,DL

STATUS:   n70433--Running in monitor____________________________________...R....
set symbols on


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

As you can see, you can not see symbols in operand because direct addressing mode is used.

In this case, you can see symbols in operand by specifying data segment value by "with_data_segment" option. Enter following command.

`display memory Init` ***`mnemonic`***
`with_data_segment` `200h <RETURN>`

```
 Memory  :mnemonic :file = sample.asm:
   address   label          data   :data segment = 0200
  0100 0000     :Init      B80002      MOV AW,0200
  0100 0003                8ED8        MOV DS0,AW
  0100 0005                8ED0        MOV SS,AW
  0100 0007                BCDE00      MOV SP,00DE
  0100 000A  sample.:Loop  8B1E0000    MOV BW,WORD PTR :SOU_A
  0100 000E                891E0600    MOV WORD PTR :DES_A,BW
  0100 0012                8B1E0200    MOV BW,WORD PTR :SOU_B
  0100 0016                891E0800    MOV WORD PTR :DES_B,BW
  0100 001A                8B1E0400    MOV BW,WORD PTR :SOU_C
  0100 001E                891E0A00    MOV WORD PTR :DES_C,BW
  0100 0022  sample:Clear  33C0        XOR AW,AW
  0100 0024                A30600      MOV WORD PTR :DES_A,AW
  0100 0027                A30800      MOV WORD PTR :DES_B,AW
  0100 002A                A30A00      MOV WORD PTR :DES_C,AW
  0100 002D                EBDB        BR SHORT /sample.asm:Loop
  0100 002F                00D4        ADD AH,DL

STATUS:   n70433--Running in monitor____________________________________...R....
display memory Init mnemonic with_data_segment 200h


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

As you can see, the symbols in operand are displayed. This data segment value is available until you specify another data segment value or "with_dada_segment none" option.

Analyzer Topic

The analyzer captures the data bus of the 70433 microprocessor. When you specify a data in the analyzer trigger condition or store condition, the ways of the analyzer data specification differ according to the data size.

To trigger the analyzer when the 70744 microprocessor accesses the word data 1234H at address 1000H in 8bit data bus size. the data bus activity of the cycles will be as follows.

```
Sequencer level Address bus  Data bus
                 1   1000H       xx34
             2   1001H       xx12
```

In this case, you need to use the analyzer sequential trigger capabilities. We do not describe the detail about the sequential trigger feature. Only how to trigger the analyzer at this example is described. To specify the condition, enter:

```
trace find_sequence 1000h data 0xx34h
restart status exec trigger after 1001h data
0xx12h <RETURN>
```

The "restart" condition is specified to restart sequencer when any states except for "exec" state are generated between sequencer level 1 and 2.

Features Available via Pod Commands

Several emulation features available in the Terminal Interface but not in the Softkey Interface may be accessed via the following emulation commands.

```
display pod_command <RETURN>
pod_command '<Terminal Interface command>'
<RETURN>
```

Some of the most notable Terminal Interface features not available in the Softkey Interface are:

- Copying memory
- Searching memory for strings or numeric expressions.
- Sequencing in the analyzer.
- Performing coverage analysis.

Refer to your Terminal Interface documentation for information on how to perform these tasks.

Note

Be careful when using the "pod_command". The Softkey Interface, and the configuration files in particular, assume that the configuration of the HP 64700 pod is NOT changed except by the Softkey Interface. Be aware that what you see in "modify configuration" will NOT reflect the HP 64700 pod's configuration if you change the pod's configuration with this command. Also, commands which affect the communications channel should NOT be used at all. Other commands may confuse the protocol depending upon how they are used. The following commands are not recommended for use with "pod_command":

stty, po, xp - Do not use, will change channel operation and hang.
echo, mac - Usage may confuse the protocol in use on the channel.
wait - Do not use, will tie up the pod, blocking access.
init, pv - Will reset pod and force end release_system.
t - Do not use, will confuse trace status polling and unload.

Accessing Internal RAM/SFR

If you access to the 70433 microprocessor's internal RAM, you can use the "display or modify memory" commands and the "display or modify register(s)" commands. When you designate address, you must use the "**fcode iram**" option. To specify an address, add this option just before an address expression. Enter the following commands:

```
display memory fcode iram <ADDRESS> blocked
words <RETURN>
```

or

```
display register BANK<N> <RETURN>
```

If you wish to access register in the current register bank, you must use the "display or modify register(s)" commands. Otherwise you will destroy the monitor program.

After you use the "**fcode iram**" option, you can access to the 70433 microprocessor's internal RAM without using "**fcode iram**" option until you use the "**fcode none**" option.

When you access SFR(Special Function Registers) of the 70433 microprocessor, you must use the "display or modify register(s)" commands. You can access SFR regardless of memory mapping.

Storing Memory Contents to an Absolute File

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

```
store memory 800h thru 84fh to absfile
<RETURN>
```

The command above causes the contents of memory locations 800H-84FH to be stored in the absolute file "absfile.X". Notice that the ".X" extension is appended to the specified filename.

Coordinated Measurements

For information on coordinated measurements and how to use them, refer to the "Coordinated Measurements" chapter in the *Softkey Interface Reference* manual.

A

Using the 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 foreground monitors are supplied with the emulation software and can be found in the following path:

```
/usr/hp64000/monitor/
```

The monitor program is named **fmon70433.s.**

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 is easier to work with. 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. 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 interrupt intensive applications. A foreground monitor is a block of code that runs in the same memory space as your program. Foreground monitors allow the emulator to service real-time events, such as interrupts, while executing in the monitor. For most multitasking, 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 applications. You must also properly configure the emulator to use a foreground monitor (see the "Configuring the Emulator" chapter and the examples in this appendix).

You may link the foreground monitor with your code. However, if possible, linking the monitor separately is preferred. This allows the monitor to be downloaded before the rest of your program. Linking monitor programs separately is more work initially, but it should prove worthwhile overall, since the monitor can then be loaded efficiently during the configuration process at the beginning of a session.

An Example Using the Foreground Monitor

In the following example, we will illustrate how to use a foreground monitor with the demo program from the "Getting Started" chapter. 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 1000H; the demo program will be located at 10000H and 80000H.

```
$ cp /usr/hp64000/monitor/fmon70433.s
<RETURN>
```

Modify EQU Statement

To use the monitor, you must modify the EQU statement near the top of the monitor listing to point to the base address where the monitor will be loaded.

```
$ chmod 644 fmon70433.s <RETURN>
$ vi fmon70433.s <RETURN>
```

Modifying Location of the Foreground Monitor

In this case, we will load the monitor at 1000H, so the modified EQU statement looks like this:

```
MONSEGMENT      EQU   00100H
```

You can load the monitor at any base address on a 2K byte boundary.

Note

You should not load the foreground monitor provided with the 64768 emulator at the base address 0 or 0ff800 hex; the 70433 microprocessor's vector table or SFR are located respectively.

Assemble and Link the Monitor

You can assemble, link and convert the foreground monitor program with the following commands :

```
$ asmv55pi fmon70433.s <RETURN>
$ llink fmon70433.ol -o fmon70433.ab <RETURN>
$ v55cov fmon70433
```

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

Modifying the Emulator Configuration

The following assumes you are modifying the default emulator configuration (that is, the configuration present after initial entry into the emulator or entry after a previous exit using "end release_system"). Enter all the default answers except those shown below.

Modify memory configuration? yes

You must modify the memory configuration so that you can select the foreground monitor and map memory.

Monitor type? foreground

Specifies that you will be using a foreground monitor program.

Reset map (change of monitor type requires map reset)? yes

You must answer this question as shown to change the monitor type to foreground.

Monitor address? 1000h

Specifies that the monitor will reside in the 2K byte block from 1000H through 17FFH.

Monitor file name? fmon70433

Enter the name of the foreground monitor absolute file. This file will be loaded at the end of configuration.

Mapping Memory for the Example

When you specify a foreground monitor and enter the monitor address, all existing memory mapper terms are deleted and a term for the monitor block will be added. Add the additional term to map memory for the demo program, and "end" out of the memory mapper.

```
0h thru 0ffh emulation ram <RETURN>
10000h thru 1ffffh emulation ram <RETURN>
80000h thru 80fffh emulation rom <RETURN>
default  target  ram <RETURN>
end <RETURN>
```

Modify pod configuration? yes

You must answer this question as shown to access and modify the question below.

Reset value for the stack segment? 1000h
Reset value for the stack pointer? 0f000h

When you use foreground monitor, the stack address should be defined in emulation memory or a target system RAM because the foreground monitor program use the user stack pointer.

Modify debug/trace options? yes

You must answer this question as shown to access and modify the question below.

Trace background or foreground operation? both

Later in this chapter, trace examples show transitions from reset into the foreground monitor, from the monitor to the user program, and from the user program back into the monitor. 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. Answering this configuration question as shown allows both foreground and background emulation processor cycles to appear in the trace.

Configuration file name? fmoncfg

If you wish to save the configuration specified above, answer this question as shown.

Load the Program Code

Now it's time to load the demo program. You can load the demo program with the following command:

```
load skdemo <RETURN>
```

Tracing from Reset to Break

We want to see the monitor's transition from the reset state to running in the foreground monitor. First, put the emulator into its reset state with the command:

```
reset <RETURN>
```

The 64768 emulator breaks to the foreground monitor via a few background cycles. You can see the transition between reset and foreground monitor execution. Enter following command.

```
trace <RETURN>
```

After entering the command above, the "Emulation trace started" message appears on the status line. Enter the following command to break into the monitor.

```
break <RETURN>
```

The status line now shows that the emulator is "Running in monitor" and that the "Emulation trace complete". Enter the following command to display the trace.

```
display trace <RETURN>
```

```
Trace List                  Offset=0
Label:  Address   Data                Opcode or Status                time count
Base:     hex     hex                     mnemonic                     relative
after    0FFFF2     0007    0007  fetch                BGM        ------------
+001     000020     0000    0000  write mem            BGM            1.9   uS
+002     000022     FFFF    FFFF  write mem            BGM            1.9   uS
+003     000024     F002    F002  write mem            BGM            1.9   uS
+004     000008     0200    0200  read mem             BGM            3.8   uS
+005     0FFFF4     FE01    FE01  fetch                BGM            1.9   uS
+006     00000A     0100    0100  read mem             BGM            1.9   uS
+007     001200     A32E    A32E  fetch                BGM            2.9   uS
+008     001202     002C    002C  fetch                BGM            3.8   uS
+009     001200     002C  MOV PS:WORD PTR 002C,AW                    320     nS
+010     001204     892E    892E  fetch                BGM            1.6   uS
+011     00102C     F080    F080  write mem            BGM            1.9   uS
+012     001206     2E2E    2E2E  fetch                BGM            1.9   uS
+013     001204     2E2E  MOV PS:WORD PTR 002E,BP                    320     nS
+014     001208     2E00    2E00  fetch                BGM            3.5   uS

STATUS:   n70433--Running in monitor      Emulation trace complete______...R....
display trace


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

The trace listing shows that the processor began executing code; it executed in background monitor. The "BGM"s in the trace listing indicate the background monitor cycles.

To see the transition from background monitor to the foreground monitor, press the <NEXT> key to page down until the background cycles go.

```
Trace List                  Offset=0
Label:  Address   Data                Opcode or Status                 time count
Base:     hex     hex                     mnemonic                      relative
+057     001000     0000    0000  read mem              BGM               1.9   uS
+058     001233     0000  illegal opcode, data = 0F 27                  320     nS
+059     001236     0000    0000  fetch                 BGM               1.6   uS
+060     000024     F002    F002  read mem              BGM               1.9   uS
+061     000022     0100    0100  read mem              BGM               3.8   uS
+062     001238     0400    0400  fetch                 BGM               1.9   uS
+063     000020     0332    0332  read mem              BGM               1.9   uS
+064     001332     F62E    F62E  fetch                                   2.9   uS
+065     001334     1906    1906  fetch                                   3.8   uS
+066     001332     1906  TEST PS:BYTE PTR 0019,01                       320     nS
+067     001336     0100    0100  fetch                                   1.6   uS
+068     001338     C62E    C62E  fetch                                   1.9   uS
+069     00133A     1B06    1B06  fetch                                   1.9   uS
+070     001019     01FF    01xx  read mem                                1.9   uS
+071     001338     01FF  MOV PS:BYTE PTR 001B,02                        320     nS

STATUS:   n70433--Running in monitor      Emulation trace complete______...R....
display trace


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

You will see the transition from the background monitor to the foreground monitor in the display.

Tracing from 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. Enter:

```
trace about __main <RETURN>
```

Because you'd like to see the states leading up to the transition from monitor to user program, trace "about" so that states before the trigger are captured.

Now, run the demo program:

```
run from transfer_address <RETURN>
```

```
Trace List                  Offset=0
Label:  Address   Data                Opcode or Status                  time count
Base:     hex     hex                     mnemonic                       relative
-007     001528     8BCF  RETI                                           960     nS
-006     00152A     140E    140E  fetch                                  960     nS
-005     001016     0080    xx80  write mem                                1.9   uS
-004     01EFFA     00C2    00C2  read mem                                 3.8   uS
-003     00152C     8E00    8E00  fetch                                    1.9   uS
-002     01EFFC     8000    8000  read mem                                 1.9   uS
-001     01EFFE     F002    F002  read mem                                 2.9   uS
about    0800C2     00B8    00B8  fetch                                    3.8   uS
+001     0800C2     00B8  MOV AW,1500                                    320     nS
+002     0800C4     8E15    8E15  fetch                                    1.6   uS
+003     0800C6     33D8    33D8  fetch                                    1.9   uS
+004     0800C5     33D8  MOV DS0,AW                                     320     nS
+005     0800C8     8EC0    8EC0  fetch                                    1.6   uS
+006     0800C7     8EC0  XOR AW,AW                                      320     nS
+007     0800CA     26C0    26C0  fetch                                    3.5   uS

STATUS:   n70433--Running user program    Emulation trace complete______...R....
run from transfer_address


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

The user program began execution at state 0. Now, you will know the processor executed the **RETI** instruction to transfer execution to the user program at state 0.

Tracing from User Program to Break

You can trace the execution from the user program to the foreground monitor due to a break condition. Since the foreground monitor occupies the address range from 1000h through 17ffh, we can simply trigger on any access to that range.

trace about range 1000h ***thru*** 17ffh <RETURN>

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

break <RETURN>

```
Trace List                   Offset=0
Label:  Address   Data                 Opcode or Status                  time count
Base:     hex     hex                      mnemonic                       relative
-007     08054E     FF29    FF29  fetch                                     1.6   uS
-006     000020     015C    015C  write mem              BGM                1.9   uS
-005     000022     803F    803F  write mem              BGM                1.9   uS
-004     000024     F283    F283  write mem              BGM                1.9   uS
-003     000008     0200    0200  read mem               BGM                3.8   uS
-002     080550     6300    6300  fetch                  BGM                1.9   uS
-001     00000A     0100    0100  read mem               BGM                1.9   uS
about    001200     A32E    A32E  fetch                  BGM                4.16  uS
+001     001202     002C    002C  fetch                  BGM                1.9   uS
+002     001200     002C  MOV PS:WORD PTR 002C,AW                         320     nS
+003     001204     892E    892E  fetch                  BGM                1.6   uS
+004     00102C     1009    1009  write mem              BGM                1.9   uS
+005     001206     2E2E    2E2E  fetch                  BGM                1.9   uS
+006     001204     2E2E  MOV PS:WORD PTR 002E,BP                         320     nS
+007     001208     2E00    2E00  fetch                  BGM                3.5   uS

STATUS:   n70433--Running in monitor      Emulation trace complete______...R....
 break


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

Now, the trace listing shows that the processor entered the background state to make the transition.

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 SINGLE_STEP_ENTRY in the foreground monitor. The address of the SINGLE_STEP_ENTRY is 300H plus the beginning of the foreground monitor. To modify the "BRK flag" interrupt vector to point to the SINGLE_STEP_ENTRY, enter the following command:

> `modify memory 4h` ***`words to`*** `0300h,0100h`
> `<RETURN>`

When you load the foreground monitor at the different base address, you should modify the "BRK flag" interrupt vector to point to the identifier SINGLE_STEP_ENTRY with same way.

Software Breakpoint and Foreground Monitor

To use the software breakpoint 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 3" interrupt vector, located at 0CH thru 0FH. Enter the following command:

```
modify memory 0ch words to 1234h,5678h
<RETURN>
```

This address is not change even if you load the foreground monitor at the different base address.

Limitations of Foreground Monitors

Listed below are limitations or restrictions present when using a foreground monitor.

Synchronized MeasurementsCMB

You cannot perform synchronized measurements over the CMB when using a foreground monitor. If you need to make such measurements, use the background monitor.

Instruction Using BRK flag

If user program includes instruction using the BRK flag(in PSW register), you can not use the foreground monitor because foreground monitor uses the BRK flag in "step" command.

Stepping

You can not use "step" command in the following instructions.

HALT/STOP
POP PSW
BRK 3/BRK imm8/BRKV
CHKIND
FPO
TSKSW/BRKCS
RETRBI

Break from Halt/Stop state

When the processor is in halt or stop state, the program counter(PC) indicates the next address of HALT or STOP instruction. If you use commands which require temporary break(display/modify register, or display/modify target system memory or I/O), the program will run from the address that PC indicates

B

Using the Format Converter

Absolute files generated by InterTools language tool can not be loaded into the 70433 emulator directly. Therefore, the 70433 Softkey Interface provides a format converter.

How to use the Converter

The format converter generates HP format files from InterTools format files for 70433.

To execute the converter program, use the following command:

```
$ v55cnv [options] <file_name>
```

<file_name> is the name of InterTools format file(.abs) which is created by the InterTools linking locator(llink). The converter program will read the InterTools format file. It will generate the following HP format files:

- HP Absolute file(with .X suffix)
- HP Linker symbol file(with .L suffix)
- HP Assembler symbol file(with .A suffix)

The converter accepts the following options.

-x	This option specifies to generate HP format absolute file (with .X suffix).
-l	This option specifies to generate HP format linker symbol file (with .L suffix).

-a	This option specifies to generate HP format assembler symbol file for all of source module information available in the <file_name>. (with .A suffix).
-A ***module***	This option specifies to generate HP assembler symbol file specified. This option may appear as many as required. If option -a, described above, is used simultaneously, specifications by this option takes precedence so that assembler symbols for modules specified by this option are generated.
-f ***module_list_file***	This option specifies to read a list of modules to generate HP OMF assembler symbol files from module_list_file. Assembler symbol files associated to modules listed in module_list_file are generated. No other assembler symbol files are not generated. If option -a is used simultaneously, specifications by this option takes precedence so that assembler symbols for modules listed in module_list_file are generated.
-q	Suppress warning messages.
-m ***anonymous***	Use anonymous module name anonymous instead of default "zzzzlib"

Restrictions and Considerations

Listed below are restrictions or considerations present when using the format converter.

The converter can not generate symbols in more than 1M bytes memory space.

The converter uses anonymous module(default: zzzzlib) when the converter generates linker symbol files. When you load absolute file, the emulator displays error message which means that there is not assembler symbol file(default: zzzzlib.A). But, this error will cause no damage on your operation.

You can use the [a-z],[A-Z],[0-9] characters to indicate symbols. Any other characters will be changed to "_". Symbols are truncated to 15 characters.

You can not treat symbols which is defined with "EQU" directive in assembler source file.

As for local symbols of C source file, the converter generates the symbols which are scoped on file.

Assuming that all files(source files and object files) exist in current directory, the converter operates. Therefore, No two files shear the same file name even if they exist in different directory, and if you want to reference C source line, C source files must exist in current directory.

Notes

Index

Notes