HP 64731

70322/70320 Emulator Terminal Interface

User's Guide

HP Part No. 64731-97004
Printed in U.S.A.
March 1993

Edition 3

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

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

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

Printing History

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

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

Edition 1	64731-90901, August 1988	E0888
Edition 2	64731-97000, June 1989	
Edition 3	**64731-97004, April 1993**	

Using this Manual

This manual will show you how to use the HP 64731 (70322/70320) emulator with the firmware resident Terminal Interface.

This manual will:

- Show you how to use emulation commands by executing them on a sample program and describing their results.
- Show you how to configure the emulator for your development needs. Topics include: restricting the emulator to real-time execution, selecting a target system clock source, and allowing the target system to insert wait states.
- Show you how to use the emulator in-circuit (connected to a target system).
- Describe the command syntax which is specific to the 70322/70320 emulator.

This manual will not:

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

Organization

Chapter 1 **Introduction to the 70322/70320 Emulator.** This chapter briefly introduces you to the concept of emulation and lists the basic features of the HP 64731 70322/70320 emulator.

Chapter 2 **Getting Started.** This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to: load programs into the emulator, map memory, display and modify memory, display registers, step through programs, run programs, use software breakpoints, search memory for data, and perform coverage tests on emulation memory.

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

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

Appendix A **70322/70320 Emulator Specific Command Syntax**. This appendix describes the command syntax which is specific to the 70322/70320 emulators. Included are: emulator configuration items, address syntax, display and access modes.

Contents

Illustrations

Tables

1

Introduction to the 70322/70320 Emulator

Introduction

The topics in this chapter include:

- Purpose of the emulator
- Features of the emulator

Purpose of the Emulator

The HP 64731 70322/70320 emulator is designed to replace the 70322/70320 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.

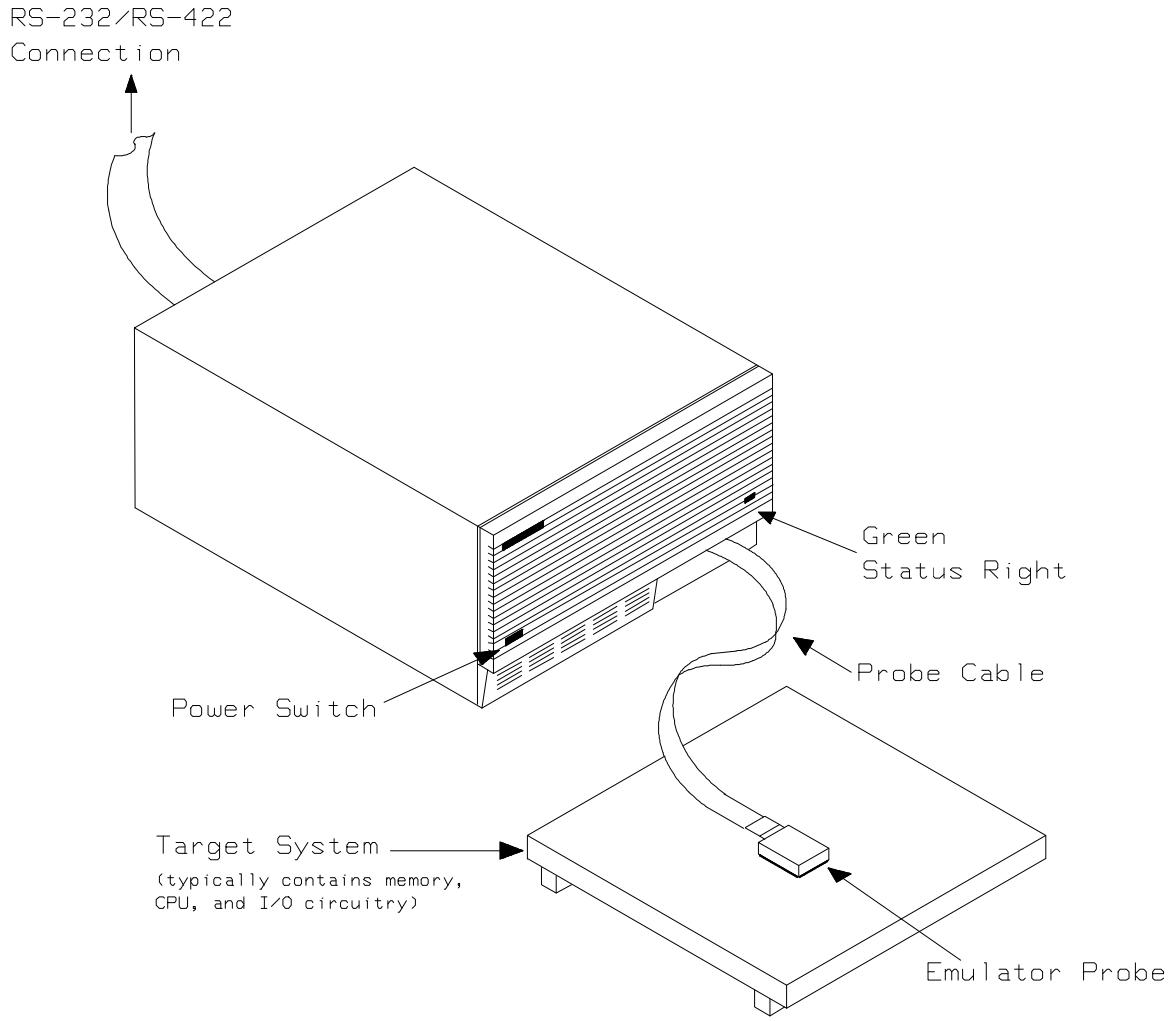


Figure 1-1 The HP 64731A Emulator for 70322/70320

Features of the 70322/70320 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 84 pin PLCC version of 70322/70320 microprocessor is supported.

Clock Speeds

The emulator runs with an internal clock speed of 8 MHz (system clock), or with target system clocks from 4-16 MHz.

Emulation Memory

The HP 64731A 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

You can define up to 16 memory ranges (at 256 byte boundaries and at least 256 bytes in length). The monitor occupies 4K bytes lieaving 124K, 508K 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 as guarded memory. The emulator generates an error message when accesses are made to guarded memory locations. You can configure the emulator so that writes to memory defined as ROM cause emulator execution to break out of target program execution.

Analysis

The analyzer supplied with the emulator, also known as the *emulation analyzer* captures emulator bus cycle information. The emulation analyzer captures bus cycle states synchronously with the emulation clock.

Registers

You can display or modify the 70322/70320 internal register contents.

Single-Step

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

Breakpoints

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

You can also define software breakpoints in your program. The emulator uses the 70322/70320 BRK 3 instruction to provide software breakpoints. When you define a software breakpoint, the emulator places an BRK 3 instruction at the specified address; after the BRK 3 instruction causes emulator execution to break out of your program, the emulator replaces the original opcode.

Reset Support

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

Configurable Target System Interface

You can configure the emulator so that it honors target system wait requests when accessing emulation memory.

Background Emulation Monitor

The monitor program runs in background, the emulator mode in which foreground operation is suspended so that the emulation processor can be used for communication with the emulation controller. 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.

2

Getting Started

Introduction

This chapter will lead you through a basic, step by step tutorial designed to familiarize you with the HP 64731 emulator for the 70322/70320 microprocessor.

This chapter will:

- Describe the sample program used for this chapter's examples.
- Show you how to use the "help" facility.
- Show you how to use the memory mapper.
- Show you how to enter emulation commands to view execution of the sample program. The commands described in this chapter include:
  - displaying and modifying memory
  - stepping
  - displaying registers
  - defining macros
  - searching memory
  - running
  - breaking
  - using software breakpoints
  - copying memory
  - testing coverage.

Before You Begin

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

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

   References: *HP 64700 Series Installation/Service* manual

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

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

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

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

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

A Look at the Sample Program

The sample program used in this chapter is listed in figure 2-1. The program transfers message data to another memory area.

```
FILE: ~o/64731/sampprog  HEWLETT-PACKARD: 70320 Assembler

LOCATION OBJECT CODE LINE     SOURCE LINE

                      1 "70320"
                      2         PROG
    0000 B80000       3 START   MOV     AW,#0000H
    0003 BB0000       4         MOV     BW,OFFSET MESSAGE
    0006 B108         5         MOV     CL,8
    0008 8A07         6 LOOP    MOV     AL,[BW]
    000A 88870002     7         MOV     [BW+200H],AL
    000E 43           8         INC     BW
    000F FEC9         9         DEC     CL
    0011 75F5        10         BNZ     LOOP
    0013 EBFE        11 END     BR      END
                     12
                     13         DATA
    0000 7550443730  14 MESSAGE ASC     "uPD70320"
                     15
                     16         COMN
    0000 2020202020  17 DISPLAY ASC     "        "
                     18         END

Errors=    0
```

Figure 2-1 Sample Program Listing

Using the "help" Facility

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

```
help  - display help information

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

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

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

Commands are grouped into various classes. To see the commands grouped into a particular class, you can use the help command with that group. Viewing the group help information in short form will cause the commands or the grammar to be listed without any description. For example, if you want to get some information for group gram, enter "**help gram**". Following help information should be displayed.

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

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

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

Becoming Familiar with the System Prompts

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

Using "help proc" to View Prompt Description

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

```
--- Address format ---
  Memory address--20 bit physical or 32 bit (seg:off) logical address
  IO addresses--16 bit address

--- Emulation Status Characters ---
  R - emulator in reset state           c - no target system clock
  U - running user program              r - target system reset active
  M - running monitor program           h - processor halted
  W - waiting for CMB to become ready   g - bus granted
  s - processor in stop mode            b - no bus cycles
  ? - unknown state

--- Equates for Analyzer Label stat ---
  extmemfetch - external mem fetch          firstop - first opecode
  extmemrd    - external mem read           grd     - guarded access
  extmemwr    - external mem write          hldack  - hold acknowledge
  dmaack      - dma acknowledge             intack  - interrupt ack
  dmamtom     - dma memory to memory        iord    - io read
  msextmemrd  - macro serv ext mem read     iowr    - io write
  msextmemwr  - macro serv ext mem write    iramrd  - int ram read
  msiramrd    - macro serv int ram read     iramwr  - int ram write
  msiramwr    - macro serv int ram write    refresh - refresh
  msiromrd    - macro serv int rom read     sfrrd   - sfr read
  mssfrrd     - macro serv sfr read         sfrwr   - sfr write
  mssfrwr     - macro serv sfr write        wrrom   - write to rom
  mon         - in monitor
```

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

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

```
U>es
N70322--Running user program
```

Initializing the Emulator

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

To initialize the emulator, enter the following command:

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

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

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

The **init** command does not:

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

Other Types of Initialization

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

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

Mapping Memory

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

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

Note

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

Blocks of memory can also be characterized as guarded memory.

Guarded memory accesses will generate "break to monitor" requests. Writes to ROM will generate "break to monitor" requests if the **rom** break condition is enabled. Memory is mapped with the **map** command. To view the memory mapping options, enter:

```
M>help map
```

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

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

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

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

```
M>map
```

```
# remaining number of terms   : 16
# remaining emulation memory  : 1b000h bytes
map  other tram
```

Which Memory Locations Should be Mapped?

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

```
HP 64000+ Linker

FILE/PROG NAME               PROGRAM    DATA       COMMON     ABSOLUTE
-------------------------------------------------------------------------
sampprog.R                   00001000   00001200   00001400

next address                 00001014   00001208   00001408
XFER address = 00000000  Defined by DEFAULT
Current working directory = /users/whoever/work
Absolute file name = sampprog.X
Total number of bytes loaded = 00000024
```

From the load map listing, you can see that the sample program occupies three address range. The program area, which contains the opcodes and operands which make up the sample program, occupies locations 1000H through 1014H. The data area, which contains the ASCII values of the messages the program transfers, is occupies locations 1200H through 1207H. The destination area, which contains the command input byte and the locations of the message destination, occupies locations 1400H through 1407H.

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

```
R>map 1000..10ff erom
R>map 1200..12ff erom
R>map 1400..14ff eram
R>map

# remaining number of terms   : 13
# remaining emulation memory  : 1ad00h bytes
map  001000..0010ff  erom   # term 1
map  001200..0012ff  erom   # term 2
map  001400..0014ff  eram   # term 3
other tram
```

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

Getting the Sample Program into Emulation Memory

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

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

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

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

Standalone Configuration

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

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

```
R>m -db 1000=0b8,0,0,0bb,0,12,0b1,8
R>m -db 1008=8a,7,88,87,0,2,43,0fe
R>m -db 1010=0c9,75,0f5,0eb,0fe
R>m -db 1200=75,50,44,37,30,33,32,30
R>m -db 1400..1407=20
```

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

```
R>m -dm 1000..1013
```

```
001000  b80000      MOV AW,#0000H
001003  bb0012      MOV BW,#1200H
001006  b108        MOV CL,#08H
001008  8a07        MOV AL,BYTE PTR [BW]
00100a  88870002    MOV BYTE PTR 200H[BW],AL
001003  43          INC BW
00100f  fec9        DEC CL
001011  75f5        BNE 1008H
001013  ebfe        BR 1013H
```

Note

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

Transparent Configuration

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

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

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

HP Absolutes

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

Note

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

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

###
R>
```

Other Supported Absolute Files

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

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

Remote Configuration

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

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

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

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

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

For More Information

For more information on downloading absolute files, refer to the **load** command description in the *HP 64700 Emulators Terminal Interface: User's Reference* manual.

Displaying Memory In Mnemonic Format

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

```
R>m -dm 1000..1013

001000  b80000      MOV AW,#0000H
001003  bb0012      MOV BW,#1200H
001006  b108        MOV CL,#08H
001008  8a07        MOV AL,BYTE PTR [BW]
00100a  88870002    MOV BYTE PTR 200H[BW],AL
001003  43          INC BW
00100f  fec9        DEC CL
001011  75f5        BNE 1008H
001013  ebfe        BR 1013H
```

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

Stepping Through the Program

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

```
R>help s
```

```
s - step emulation processor

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

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

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

```
R>s 1 0:1000
```

```
00000:01000  b80000      MOV AX,#0000H
PC = 00000:01003
```

Displaying Registers

The step command shown above executed a MOV AW,# 0 instruction. Enter the following command to view the contents of the registers.

```
M>reg
```

```
reg ps=0000 pc=1003 psw=f002 aw=0000 bw=0bf8 cw=0000 dw=0000 sp=07f8 bp=0014
reg ix=00f0 iy=0000 ds0=0000 ds1=0000 ss=0000
```

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

You can also display peripheral control block registers or certain classes of peripheral registers. Refer to the "70322/70320 Emulator Specific Command Syntax" appendix for more information on the peripheral register names and classes.

Combining Commands

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

```
M>s;reg

00000:01003  bb0012      MOV BW,#1200H
PC = 00000:01006
reg ps=0000 pc=1006 psw=f002 aw=0000 bw=1200 cw=0000 dw=0000 sp=07f8 bp=0014
reg ix=00f0 iy=0000 ds0=0000 ds1=0000 ss=0000
```

Using Macros

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

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

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

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

```
M>st
```

```
# s;reg
00000:01006  b108        MOV CL,#08H
PC = 0000:01008
reg ps=0000 pc=1008 psw=f002 aw=0000 bw=1200 cw=0008 dw=0000 sp=07f8 bp=0014
reg ix=00f0 iy=0000 ds0=0000 ds1=0000 ss=0000
```

Command Recall

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

Repeating Commands

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

```
M>rep 4 st
```

```
# s;reg
00000:01008  8a07        MOV AL,BYTE PTR [BW]
PC = 00000:0100a
reg ps=0000 pc=100a psw=f002 aw=0075 bw=1200 cw=0008 dw=0000 sp=07f8 bp=0014
reg ix=00f0 iy=0000 ds0=0000 ds1=0000 ss=0000
# s;reg
00000:0100a  88870002    MOV BYTE PTR 0200H[BW],AL
PC = 00000:0100e
reg ps=0000 pc=100e psw=f002 aw=0075 bw=1200 cw=0008 dw=0000 sp=07f8 bp=0014
reg ix=00f0 iy=0000 ds0=0000 ds1=0000 ss=0000
# s;reg
00000:0100e  43          INC BW
PC = 00000:0100f
reg ps=0000 pc=100f psw=f002 aw=0075 bw=1201 cw=0008 dw=0000 sp=07f8 bp=0014
reg ix=00f0 iy=0000 ds0=0000 ds1=0000 ss=0000
# s;reg
00000:0100f  fec9        DEC CL
PC = 00000:01011
reg ps=0000 pc=1011 psw=f002 aw=0075 bw=1201 cw=0007 dw=0000 sp=07f8 bp=0014
reg ix=00f0 iy=0000 ds0=0000 ds1=0000 ss=0000
```

Modifying Memory

The preceding step and register commands show the sample program is executing transfer message data to destination area. Use the **m** (memory) command to modify the source message data bytes.

```
M>m 1200=41,42,43,44,45,46,47,48
```

To verify that above data has been written to 1200H, enter the following command.

```
M>m -db 1200..1207
001200..001207  41 42 43 44 45 46 47 48
```

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

Specifying the Access and Display Modes

There are a couple different ways to modify the display and access modes. One is to explicitly specify the mode with the command you are entering (e.g., m -db 1200). The **mo** (display and access mode) command is another way to change the default mode. For example, to display the current modes, define the display mode as "word", and redisplay 1200H, enter the following commands.

```
M>mo
mo -ab -db
```

```
M>mo -dw
M>m 1200
001200..001200  4241
```

To continue the rest of program ..

```
M>r
```

Display the DISPLAY memory locations (destination of the message, 1400H) to verify that the program moved the correct ASCII bytes. At this time we want to see correct byte value, so "-db" option (display with byte) is used.

```
M>m  -db 1400..1407
001400..001407  41 42 43 44 45 46 47 48
```

Searching Memory for Data

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

```
M>ser 1200..1207="FG "
pattern match at address: 001205
```

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

Breaking into the Monitor

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

```
U>b
M>
```

Note

If DMA transfer is in progress with BURST transfer mode, the emulator can break into monitor after such DMA transfers are completed.

Using Software Breakpoints

Software breakpoints are handled by the 70322/70320 BRK 3 instruction. When you define or enable a software breakpoint (with the **bp** command), the emulator will replace the opcode at the software breakpoint address with a breakpoint interrupt instruction (BRK 3).

Note

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

Caution

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

Note

Because software breakpoints are implemented by replacing opcodes with the BRK 3 instructions, you cannot define software breakpoints in target ROM. You can, however, copy target ROM into emulation memory (see the "Target ROM Debug Topics" section of the "In-Circuit Emulation" chapter).

When the emulator detects a vector fetch from the BRK 3 interrupt vector area (in other words, the BRK 3 instruction has executed or vector area is read from user program), it generates a break to background request. Since the system controller knows the locations of defined software breakpoints, it can determine whether the BRK 3 interrupt was generated by an enabled software breakpoint or by a BRK 3 interrupt 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 the 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 system, execution still breaks to the monitor, and an "undefined breakpoint" status message is displayed. To continue with program execution, you must run or step from the target program’s breakpoint interrupt vector address.

Displaying and Modifying the Break Conditions

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

```
M>bc
```

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

To enable the software break point feature enter

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

Defining a Software Breakpoint

Now that the software breakpoint feature is enabled, you can define software breakpoints. Enter the following command to break on the address of the **LOOP** (address 1000H) label.

```
M>bp 1000
M>bp
```

```
bp 001008 #enabled
```

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

```
M>r 0:1000
```

```
!ASYNC_STAT 615! Software breakpoint: 00000:01008
```

```
M>st
```

```
# s;reg
00000:0100a  8a07        MOV AL,BYTE PRT [BW]R
PC = 00000:0100a
reg ps=0000 pc=100a psw=f686 aw=0041 bw=1200 cw=0008 dw=9f65 sp=38bb bp=28e9
reg ix=00f1 iy=8577 ds0=0000 ds1=0000 ss=0000
```

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

```
                                M>bp

### BREAKPOINT FEATURE IS ENABLED ###
bp 001008 #disabled
                                M>bp -e 001008
                                M>bp

### BREAKPOINT FEATURE IS ENABLED ###
bp 001008 #enabled
                                M>r 1000

!ASYNC_STAT  615! Software breakpoint: 00000:01008
                                M>bp

### BREAKPOINT FEATURE IS ENABLED ###
bp 001008 #disabled
```

Using the Analyzer

Predefined Trace Labels

Three trace labels are predefined in the 70322/70320 emulator. You can view these labels by entering the **tlb** (trace label) command with no options.

```
                                M>tlb

#### Emulation trace labels
tlb addr 0..19
tlb data 20..35
tlb stat 36..46
```

Predefined Status Equates

Common values for the 70322/70320 status trace signals have been predefined. You can view these predefined equates by entering the **equ** command with no options.

```
M>equ
```

```
### Equates ###
equ dmaack=0xxx0xxxxxxx
equ dmamtom=0xxxxx011011y
equ extmemfetch=0xxxxx001001y
equ extmemrd=0xxxxx001011y
equ extmemwr=0xxxxx001101y
equ firstop=0xxxxxxxxxx0y
equ grd=0x0xxxxxxxxy
equ hldack=0xxxx0xxxxxx
equ intack=0xxxxx000001y
equ iord=0xxxxx000011y
equ iowr=0xxxxx000101y
equ iramrd=0xxxxx110001y
equ iramwr=0xxxxx110101y
equ iromrd=0xxxxx010001y
equ mon=0xxxxxxxxxy
equ msextmemrd=0xxxxx101001y
equ msextmemwr=0xxxxx101101y
equ msiramrd=0xxxxx100001y
equ msiramwr=0xxxxx100101y
equ msiromrd=0xxxxx010011y
equ mssfrrd=0xxxxx100011y
equ mssfrwr=0xxxxx100111y
equ refresh=0xxxxx010101y
equ sfrrd=0xxxxx110011y
equ sfrwr=0xxxxx110111y
equ wrrom=0xx0xxxxxxxy
```

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

Specifying a Simple Trigger

The **tg** analyzer command is a simple way to specify a condition on which to trigger the analyzer. Suppose you wish to trace the states of the program after the read of a "E" (45H) ascii data from the source data area. Enter the following commands to set up the trace, run the program, issue the trace, and display the trace status. (Note that the analyzer is to search for a lower byte read of 45H because the address is even.)

```
                                M>tg data=0xx45
                                M>t
                                M>r 0:1000
Emulation trace started
                                U>ts

--- Emulation Trace Status ---
NEW User trace complete
Arm ignored
Trigger in memory
Arm to trigger ?
States 396 (396) -140..255
Sequence term 2
Occurrence left 1
```

The trace status now shows that the trigger condition has been found, and that 396 states have been stored in trace memory. Enter the following command to display the first 20 states of the trace.

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

```
Line   addr,H  7032x mnemonic,H                      xbits,H    count,R  seq
----   ------  ----------------------------------  -------  ---------  ---
   0   01204           45H, mem read  (ext)         0000          ---   +
   1   00089         45xxH, refresh                 0000      0.960 uS   .
   2   0100a   INSTRUCTION--opcode unavailable      0000      0.280 uS   .
   3   0100e           43H, fetch     (ext)         0000      2.760 uS   .
   4   0100f         fexxH, fetch     (ext)         0000      2.000 uS   .
   5   01010           c9H, fetch     (ext)         0000      2.000 uS   .
   6   0008a           c9H, refresh                  0000      0.960 uS   .
   7   0100e   INC BW                                0000      2.280 uS   .
   8   01404           45H, mem write (ext)         0000      0.760 uS   .
   9   0100f   DEC CL                                0000      1.240 uS   .
  10   01011         75xxH, fetch     (ext)         0000      0.760 uS   .
  11   01012           f5H, fetch     (ext)         0000      2.000 uS   .
  12   0008b         f5xxH, refresh                  0000      0.960 uS   .
  13   01011   BNE 1008H                             0000      2.280 uS   .
  14   01013         ebxxH, fetch     (ext)         0000      0.760 uS   .
  15   01014           feH, fetch     (ext)         0000      2.000 uS   .
  16   01015         00xxH, fetch     (ext)         0000      2.000 uS   .
  17   0008c           00H, refresh                  0000      0.960 uS   .
  18   01016           00H, fetch     (ext)         0000      3.040 uS   .
  19   01017         00xxH, fetch     (ext)         0000      2.000 uS   .
```

Line 0 in the trace list above shows the state which triggered the analyzer. The trigger state is always on line 0. To list the next lines of the trace, enter the following command.

```
U>tl

Line   addr,H  7032x mnemonic,H                      xbits,H    count,R  seq
----   ------  ----------------------------------  -------  ---------  ---
  20   0008d         00xxH, refresh                  0000     1.960 uS    .
  21   01008           8aH, fetch     (ext)          0000     3.040 uS    .
  22   01009         07xxH, fetch     (ext)          0000     2.000 uS    .
  23   0100a           88H, fetch     (ext)          0000     2.000 uS    .
  24   0008e           88H, refresh                  0000     0.960 uS    .
  25   01008   MOV AL,BYTE PTR [BW]                  0000     0.280 uS    .
  26   0100b         87xxH, fetch     (ext)          0000     2.760 uS    .
  27   0100c           00H, fetch     (ext)          0000     2.000 uS    .
  28   0100d         02xxH, fetch     (ext)          0000     2.000 uS    .
  29   0008f         02xxH, refresh                  0000     0.960 uS    .
  30   01205         46xxH, mem read  (ext)          0000     3.040 uS    .
  31   0100a   MOV BYTE PTR 0200H[BW],AL             0000     1.240 uS    .
  32   0100e           43H, fetch     (ext)          0000     0.760 uS    .
  33   0100f         fexxH, fetch     (ext)          0000     2.000 uS    .
  34   00090           feH, refresh                  0000     0.960 uS    .
  35   01010           c9H, fetch     (ext)          0000     3.040 uS    .
  36   0100e   INC BW                                0000     1.240 uS    .
  37   01405         46xxH, mem write (ext)          0000     0.760 uS    .
  38   0100f   DEC CL                                0000     1.240 uS    .
  39   01011         75xxH, fetch     (ext)          0000     0.760 uS    .
```

For a Complete Description

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

Copying Memory

The **cp** (copy memory) command gives you the ability to copy the contents of one range of memory to another. This is a handy feature to test things like the relocatability of programs, etc. This also useful to save result of data to another locations. To save transfered data to another memory area.

```
U>cp 1420=1400..1407
U>m 1420..1427
001420..001427  41 42 43 44 45 46 47 48
```

Testing for Coverage

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

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

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

```
U>cov -r
```

Set software breakpoint at 1008 (as in earlier), and run the program from the start address (00000:01000H) and use the **cov** command to display how much of the program is accessed.

```
U>rst
R>cov -r
R>bp -e 1008
R>r 1000
!ASYNC_STAT  615! Software breakpoint: 00000:01008
M>cov -a 1000..1014
# coverage list  -  list of address ranges accessed
001000..00100d

percentage of memory accessed:  % 66.6
```

Now run the rest of programs.

```
M>r
U>cov -a 1000..1014
# coverage list  -  list of address ranges accessed
001000..001014

percentage of memory accessed: % 100.0
```

Resetting the Emulator

To reset the emulator, enter the following command.

```
U>rst
R>
```

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

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

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

3

Emulation Topics

Introduction

Many of the topics described in this chapter involve the commands which are unique to the 70322/70320 emulator such as the **cf** command which allows you to specify emulator configuration. A reference-type description of the 70322/70320 emulator configuration items can be found in the "70322/70320 Emulator Specific Command Syntax" appendix.

This chapter will:

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

Prerequisites

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

Execution Topics

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

Restricting the Emulator to Real-Time Runs

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

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

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

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

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

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

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

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

Default Physical to Logical Run Address Conversion

The run and step commands allow you to enter addresses in either logical form (segment:offset, e.g., 0F000:0FFFF) or physical form (e.g., 0FFFFF).

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. By default, a physical run address is converted such that the low 16 bits of the address become the offset value. Use the **cf** (configuration) command with the **rad** (run address default conversion) configuration item to specify that the low 4 bits of the physical address become the offset. The physical address is right-shifted 4 bits to yield the segment value.

```
R>cf rad=maxseg
```

 # phys_addr =
(phys_addr > > 4):(phys_addr & 0xf)

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

```
R>cf rad=minseg
```

 # phys_addr =
((phys_addr > > 4) & 0xf000):(phys_addr & 0xffff)

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

Setting Up to Break on an Analyzer Trigger

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

Specify the Signal Driven when Trigger is Found

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

```
R>tgout trig1
```

Enable the Break Condition

Enable the "trig1" break condition.

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

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

Making Coordinated Measurements

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

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

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

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

Monitor Option Topics

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

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

Background Monitor

Some things to be aware of when using the background monitor are:

- Guarded memory accesses can occur if no vector table is loaded and the vector table area, 0-3FFH, is mapped as "guarded memory". (If locations 0-3FFH are not mapped, the "map other grd" command specifies these locations as guarded memory.)
- Halt instructions will cause "processor halted" emulation status (the "h> " prompt is shown). A subsequent break command, followed by a run or step command, causes the halt instruction to be executed again.

Locating the Monitor

The default emulator configuration locates the monitor at 000000H. You can relocate the monitor to any 4K byte boundary. The location of the background monitor may be important since background cycles are always visible to the target system. Use the **cf** (configuration) command with the **loc** configuration item to specify the location of the monitor.

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

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

Note

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

Selecting Microprocessor

The HP 64731 70322/70320 emulator have capability to emulate 70320 (ROMless version) microprocessor and 70322 (incorporates internal ROM) microprocessor. You can select either microprocessor. To select 70320 microprocessor.

```
R>cf proc=70320
```

Upon power up, 70322 microprocessor is selected.

Note

Changing microprocessor cause all memory mapper terms to be removed.

Selecting Accept Or Ignore Target System Reset

The 70322/70320 emulator can respond or ingore target system reset. You can ignore reset from target system completely by specifing "**cf urst= dis**". In this configuration emulator ignore any reset from target system. Specifying "**cf urst= en**", this is a default configuration, make the emulator to respond to reset from target system. In this configuration, emulator will accept reset and execute from reset vector (0FFFF0H) as same manner as actual microprocessor after reset is inactivated

Note

This configuration item is true while either running in user program or running in monitor program. If enabled by entering "cf urst= en", emulator will respond to target system reset while running in monitor program.

Other Topics

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

Detecting Illegal Opcodes

Illegal opcode detection is handled by the processor's own detection mechanism. An illegal opcode interrupt will cause execution to be diverted to the address contained in the interrupt vector table. This address should point to your interrupt service routine.

You must prepare illegal opcode interrupt service routine in your program.

Note

You should prepare illegal opcode interrupt service routine in your program code and set appropriate vector.

4

In-Circuit Emulation Topics

Introduction

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

This chapter will:

- Describe the issues concerning the installation of the emulator probe into target systems.
- Show you how to install the emulator probe.
- Describe how to set up the emulator to use a target system clock and how to allow DMA accesses to emulation memory. How you can relocate background monitor program. These topics are related to program execution in general.
- Describe how to use software breakpoints with ROMed code, how to perform coverage testing on ROMed code, and how to test patches to ROMed code. These topics relate to the debugging of target system ROM.
- Describe the ACcharacteristics and the target system interface
- Describe some of restrictions and considerations.

Prerequisites

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

Installing the Emulator Probe into a Target System

The emulator probe has a PLCC connector. The emulator probe is also provided with a non-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 emulation probe contains devices that are susceptible to damage by static discharge. Therefore, precautionary measures should be taken before handling the microprocessor connector attached to the end of the 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 the 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.

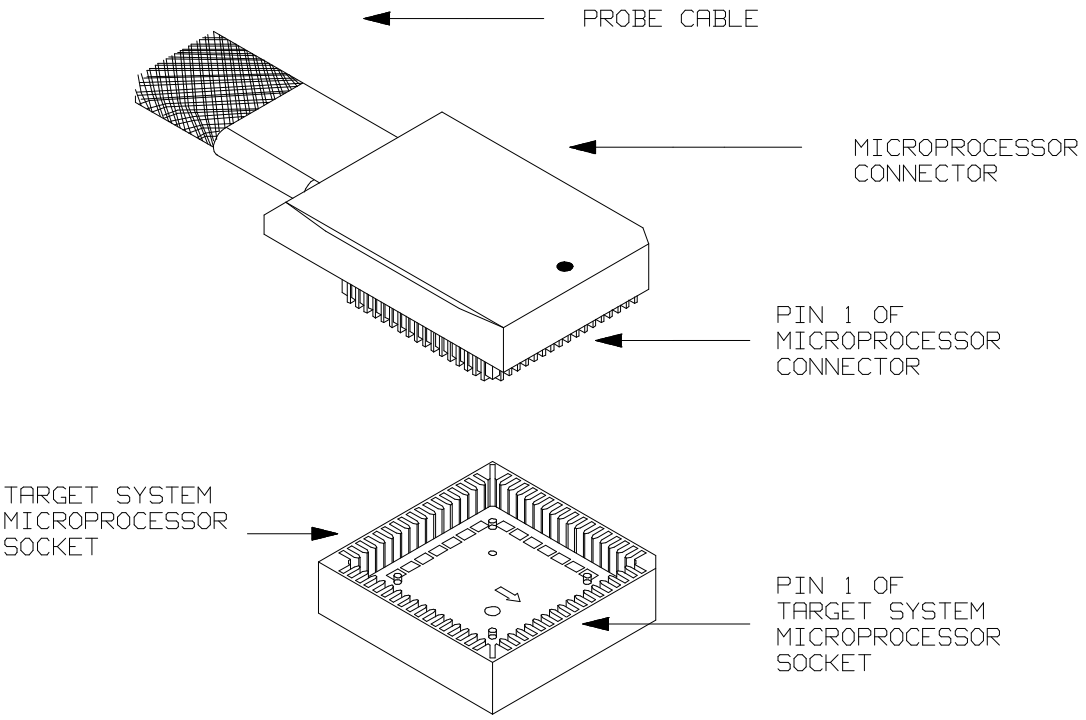


Figure 4-1 Installing into a 70322 PLCC type socket

Execution Topics

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

Specifying the Emulator Clock Source

The default emulator configuration selects the internal 8 MHz (system clock speed) clock as the emulator clock source. You can configure the emulator to select an external target system clock source in the range of 4-16 MHz. You can use external clock or crystal to generate system clock as actual microprocessor does. Use the **cf** (configuration) command and the **clk** configuration item to specify that the emulator use a target system clock.

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

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

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

DMA Cycles

Cycles from the emulation processor's internal DMA controller is allowed to access emulation memory. If DMA controller is programmed to transfer with **Burst mode**, the emulator can not break in monitor until transfers are completed. With other DMA mode, emulator breaks into the monitor (background) upon request, however, DMA transfers are continued. DMA cycles are sent to the analyzer.

Note

Target system DMA controller access to emulation memory is not allowed. Only internal DMA controller can access emulation memory as it's transfer object.

Note

External (target system) DMA controller execution cycles can be traced when following conditions are met.
A. Address and data signals can be read by the emulator. In other words, those signals must reach to emulator probe and timing specification comply with specification of DMA cycles of 70322/70320 microprocessor.
B. Control signals (such as /MREQ, R/W, etc) can be read by the emulator.

Monitor Operation

The 70322/70320 emulator is background type emulator, so that any memory space is not required to perform emulator functions. However, the emulator performs some functions (such as display/modify target system memory) by executing monitor program. It is very important to know where the monitor program is running, since the monitor operation is always visible to target system. Upon power up, the background monitor resides address range 00000H through 00FFFH. Since the monitor operation is always visible to target system, if your target system have memory or some memory mapped I/O devices on the area, it may be problem. In such case, you can relocate monitor program address to another location if desired. You can relocate each 4K byte boundary.

To see default monitor operation address ...

TYPE: **tck -ub**

This allow the emulation analyzer to trace monitor operation including your program operation.

Now, trace the monitor operation by entering "t" (trace) command and display analysis trace by entering "tl" (trace list display) command.

The monitor program reside 00000H through 00FFFH memory space.

Now, let's relocate monitor location to 20000H by issuing "**cf loc= 20000**". And trace monitor operation as did earlier. Analysis trace display should resemble:

```
 Line   addr,H  7032x mnemonic,H                      xbits,H    count,R  seq
-----   ------  ----------------------------------  -------  ---------  ---
    0   0011a           00H, refresh            MON   0000         ---   +
    1   20326           2eH, fetch     (ext)    MON   0000      3.000 uS   .
    2   20327         81xxH, fetch     (ext)    MON   0000      2.000 uS   .
    3   20328           3eH, fetch     (ext)    MON   0000      2.000 uS   .
    4   0011b         3exxH, refresh            MON   0000      1.000 uS   .
    5   20326   CMP PS:WORD PTR 00c4H,#0100H          0000      0.280 uS   .
    6   20327                                         0000      2.000 uS   .
    7   20329         c4xxH, fetch     (ext)    MON   0000      0.720 uS   .
    8   2032a           00H, fetch     (ext)    MON   0000      2.000 uS   .
    9   2032b         00xxH, fetch     (ext)    MON   0000      2.000 uS   .
   10   0011c           00H, refresh            MON   0000      1.000 uS   .
   11   2032c           01H, fetch     (ext)    MON   0000      3.000 uS   .
   12   2032d         74xxH, fetch     (ext)    MON   0000      2.000 uS   .
   13   2032e           02H, fetch     (ext)    MON   0000      2.000 uS   .
   14   0011d         02xxH, refresh            MON   0000      1.000 uS   .
   15   200c4           00H, mem read  (ext)    MON   0000      3.000 uS   .
   16   200c5         03xxH, mem read  (ext)    MON   0000      2.000 uS   .
   17   2032f         ebxxH, fetch     (ext)    MON   0000      2.000 uS   .
   18   0011e           ebH, refresh            MON   0000      1.000 uS   .
   19   2032d   BE 0331H                               0000      0.280 uS   .
```

As you can see, monitor program is relocated to 20000H (physical address).

Emulator Probe Signal Topics

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

Allowing the Target System to Insert Wait States

High-speed emulation memory provides no-wait-state operation. However, the emulator may optionally respond to the target system ready lines while emulation memory is being accessed. Use the **cf** (configuration) command with the **rdy** configuration item to cause emulation memory accesses to honor target system ready signals.

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

When the ready relationship is locked to the target system, emulation memory accesses honor ready signals from the target system (wait states are inserted if requested).

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

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

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

Target ROM Debug Topics

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

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

Using Software Breakpoints with ROMed Code

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

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

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

Coverage Testing ROMed Code

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

Once target ROM is copied into emulation memory, coverage testing may be done normally at addresses in these emulation memory locations.

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

Modifying ROMed Code

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

Electrical Characteristics

The AC characteristics of the HP 64731A emulator are listed in the following table.

Table 4-1 Electrical Specifications

Characteristic	Symbol	uPD70322,70320		HP 64731 A	Unit
		Min	Max	Typical	
CLKOUT Width Low,High	t_{WKH} t_{WKL}	0.5T-15		0.5T-25	ns
CLKOUT Rise andFAll Time	t_{KR},t_{KF}		15	25	ns
Address Valid to Data-in Valid (Read)	t_{DADR}		(n+ 1.5)T-90	(n+ 1.5)T-120	ns

Table 4-1 Electrical Specifications(Cont'd)

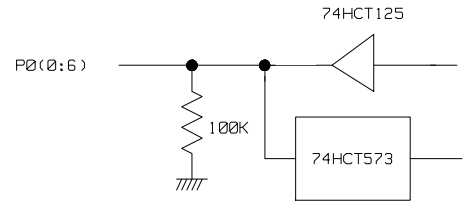
Characteristic	Symbol	uPD70322,70320		HP 64731 A	Unit
		Min	Max	Typical	
CLKOUT Width Low,High	t_{WKH} t_{WKL}	0.5T-15		0.5T-25	ns
$\overline{\text{MREQ}}$ Asserted to Data-in Valid(Read)	t_{DMRD}		(n+ 1)T-75	(n+ 1)T-105	ns
$\overline{\text{MSTB}}$ Asserted to Data-in Valid(Read)	t_{DMSD}		(n+ 0.5)T-75	(n+ 0.5)T-105	ns
$\overline{\text{IOSTB}}$ Asserted to Data-in Valid(Read)	t_{DISD}		(n+ 1)T-90	(n+ 1)T-120	ns
READY Setup Time	t_{SCRY}		(n-1)T-100	(n-1)T-130	ns
HLDRQ Setup Time	t_{SHQK}		30	60	ns
$\overline{\text{INTP0}}$-$\overline{\text{INTP2}}$,DMARQ0-DMARQ Setup Time	t_{SIQK}		30	60	ns
$\overline{\text{POLL}}$ Set up Time	t_{SPLK}		30	60	ns
INT Setup Time	t_{SIRK}		30	60	ns

T: Cycle Time
n: Wait-State Count

Target System Interface

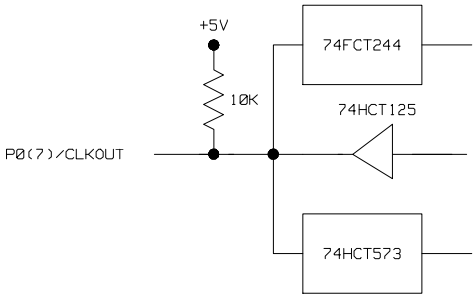
P0(0:6)

These signals are connected to 74HCT125 and 74HCT573 and 100K ohm pull-down register.



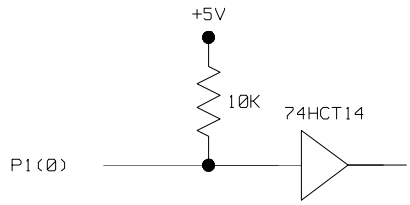
P0(7)/CLKOUT

This signal is connected to 74HCT125 and 74HCT573 and 10K ohm pull-up register in process with PORT mode. In other case, this signal is connected to 74FCT244 and 10K ohm pull-up register.

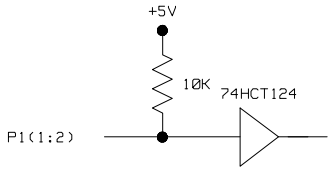


P1(0)

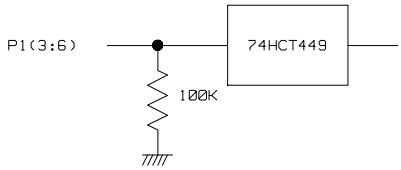
This signal is connected tp 74HCT14 and 10K ohm pull-up register.



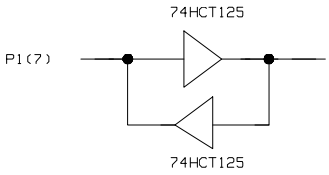
P1(1:2) These signals are connected to 74HCT125 and 10K ohm pull-up register.



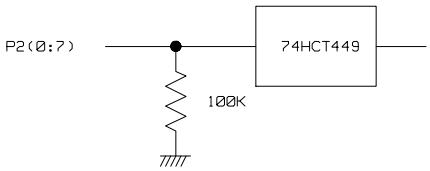
P1(3:6) These signals are connected to 74HCT449 and 100K ohm pull-down register.



P1(7) This signal is connected to 74HCT125.

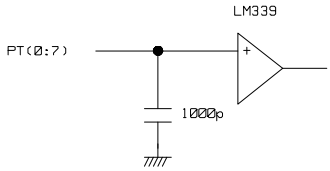


P2(0:7) These signals are connected to 74HCT449 and 100K ohm pull-down register.



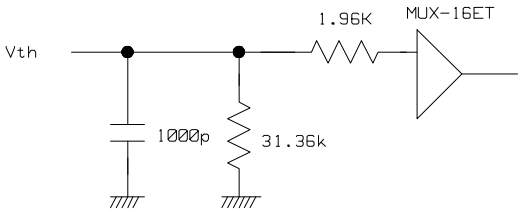
PT(0:7)

These signals are connected to LM339 and 1000p farad condenser.



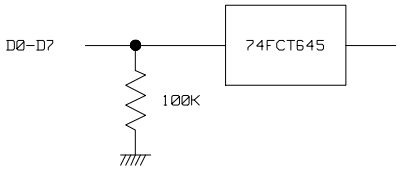
Vth

This signal is connected to MUX-16ET through 1.96K ohm series register and 31.36K ohm pull-down register and 1000p farad condenser.



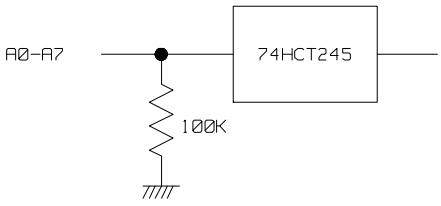
D0-D7

These signals are connected to 74HCT645 and 100K ohm pull-down register.

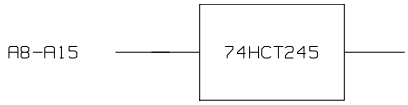


A0-A7

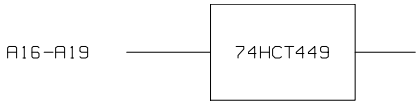
These signals are connected to 74HCT245 and 100K ohm pull-down register.



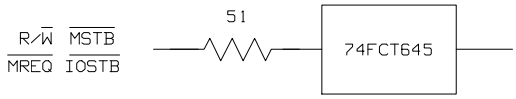
A8-A15 These signals are connected to 74HCT245.



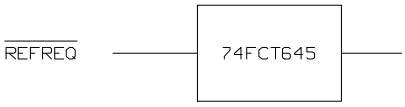
A16-A19 These signals are connected to 74HCT449.



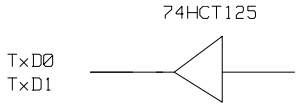
R/$\overline{W}$ $\overline{MSTB}$ $\overline{MREQ}$ $\overline{IOSTB}$ These signals are connected to 74FCT645 through 51 ohm series register.



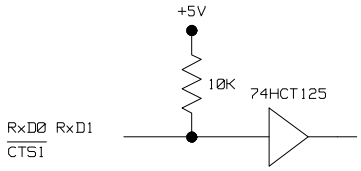
$\overline{REFREQ}$ This signal is connected to 74FCT645.



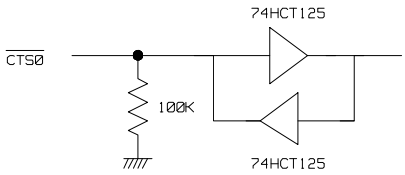
TxD0 TxD1 These signals are connected to 74HCT125.



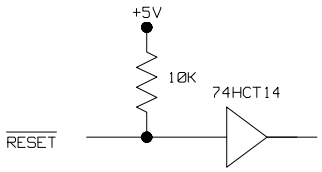
$\underline{RxD0}$ RxD1 CTS1 These signals are connected to 74HCT125 and 10K ohm pull-up register.



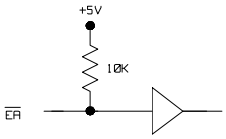
$\overline{\text{CTS0}}$ This signal is connected to 74HCT125 and 100K ohm pull-down register.



$\overline{\text{RESET}}$ This signal is connected to 74HCT14 and 10K ohm pull-up register.



$\overline{\text{EA}}$ This signal is connected to 74HCT125 and 10K ohm pull-up register.



Restrictions And Considerations

Interrupts While Executing Step Command

While executing user program code in stepping, interrupts (include NMI, non-maskable interrupt) are suspended. If the interrupt and certain instruction execution which can not be interrupted occur simultaneously, the interrupt will be ignored completely.

Note

You should not use step command when if target system can generates interrupt. It's possible to be ignored interrupts.

DMA Operation

In case of the 70322/70320 microprocessor, internal ROM/RAM area can not become source/destination of DMA operation. External memory is accessed instead. However, the 70322/70320 emulator accesses internal ROM/RAM area as DMA operation source or destination.

For example, in case of actual processor (70322), if DMA operation from internal ROM area to target I/O is executed, external memory which is overlapping to internal ROM become data source. However, in case of the emulator, internal ROM becomes source instead.

Load/Modify Operation to Internal ROM

When you load the program to internal ROM area, the emulator stores to the internal ROM area. In addition, the emulator performs write operation to the target system. If your target system have memory on address area where the internal ROM resides, the emulator write to the memory when you load the code to internal ROM. This is true when you issue the command which performs the write operation to internal ROM (ie. "m" modify memory command).

Tracing Internal RAM/SFR Access Cycles

The address information when accessing internal RAM/SFR area, only lowest 8 bit is correct. Other higher 12 bit address information is unstable.

If your program accesses internal RAM/SFR, emulation trace display shows you such cycles. However, displayed address information of internal RAM/SFR access cycle may differ from one you expect. In such case, only least 8 bit is certain. Rest 12 bit of address information may be unreliable (have no meaning).

A

70322/70320 Emulator Specific Command Syntax

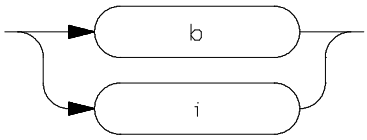
The following pages contain descriptions of command syntax specific to the 70322/70320 emulator. The following syntax items are included (several items are part of other command syntax):

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

ACCESS_MODE

Summary This configuration item have special meaning to the HP 64731 70322/70320 emulator. When your target microprocessor is 70322, there is internal ROM on chip. This internal ROM and internal RAM/SFR (Special Function Registers) can overlap. In such situation, you can specify which resource you want to access to.

Syntax



Function The < **ACCESS_MODE**> for the 70322/70320 emulator has a special meaning. The 70322 microprocessor have internal ROM (16 Kbyte) and internal RAM, SFR (Special Function Register) on memory space. The internal ROM and internal RAM/SFR can overlap each other. Upon reset, Internal ROM and internal RAM/SFR overlaps, since the IDB register is set to FFH (internal RAM start at FFE00H physical address). In such situation, you can specify the emulator to access which resource (internal ROM or internal RAM/SFR).

Note

This configuration item has no meaning when you select 70320 microprocessor because 70320 does not incorporate ROM on chip, or internal RAM/SFR do not overlap with internal ROM. **This configuration has meaning only when you select 70322 microprocessor and internal RAM/SFR overlap with internal ROM**.

Parameters

i	Access Internal RAM/SFR. Select internal RAM/SFR as object of command you requested. Setting this mode allow you to access (display/modify) internal RAM/SFR when it overlap with the internal ROM.
b	Access Internal ROM. Select internal ROM as object of command you requested. Setting this mode allow you to access (display/modify) internal ROM when it overlap with the internal RAM.

Caution

This configuration item also affects the destination of load command. You must set this to "**i**" when you load your programs into internal ROM. Otherwise, if set to "**b**", your programs will be loaded into internal RAM/SFR. Keep this in your mind when loading programs into internal ROM. Since the emulator's default is "**b**" you must modify this configuration item to "**i**" prior to loading the program into internal ROM.

Defaults

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

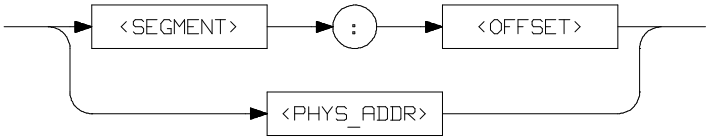
Related Commands

mo (specify display and access modes)

ADDRESS

Summary Address specifications used in emulation commands.

Syntax



Function The **< ADDRESS>** parameter used in emulation commands may be specified as a logical address or as a physical address (though a physical address in a run or step command is converted to a logical address by the emulation system).

Parameters **< SEGMENT>**

This expression is the segment portion of the logical address. The value specified is placed in the 70322/70320 PS register before running or stepping.

< OFFSET>

This expression is the offset portion of the logical address. The value specified is placed in the 70322/70320 PC register before running or stepping.

< PHYS_ADDR>

This expression is a physical address in the 70322/70320 address range. In run and step commands, the emulation system converts this physical address to a logical address as specified by the **rad** (run address default) configuration item (see the **< CONFIG_ITEM>** description).

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

Defaults

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

Examples

The examples below show logical addresses being used in the run and step command as compared to physical addresses.

```
M>s 2 0ffff:0
  ffff:0000  ea00f400f0  JMP FAR PTR ff400
  f000:f400  b800f0      MOV AW,#f000
  PC = f000:f403
M>s 2 0ffff0
  f000:fff0  ea00f400f0  JMP FAR PTR ff400
  f000:f400  b800f0      MOV AW,#f000
  PC = f000:f403
```

Related Commands

r (run command)

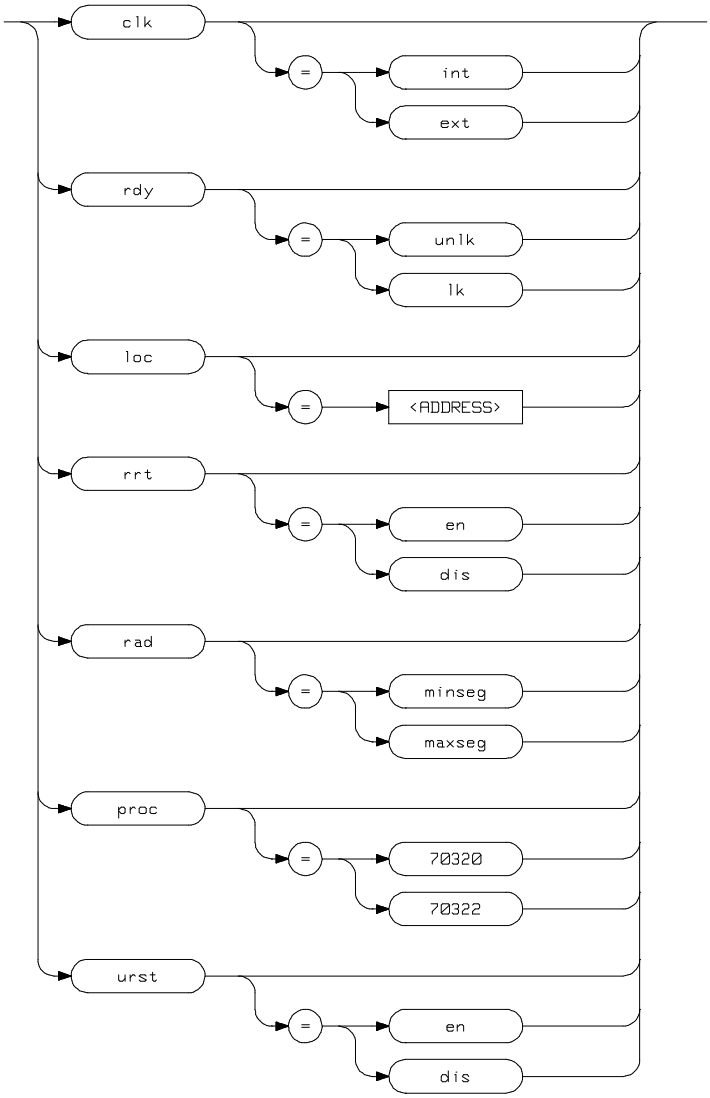
s (step command)

< **CONFIG_ITEMS**> (70322/70320 specific items specified with the **cf** command)

CONFIG_ITEMS

Summary 70322/70320 emulator configuration items.

Syntax



Function

The < **CONFIG_ITEMS**> are the 70322/70320 specific configuration items which can be displayed/modified using the **cf** (emulator configuration) command. If the "= " portion of the syntax is not used, the current value of the configuration item is displayed.

Parameters

clk Clock Source. This configuration item allows you to specify whether the emulator clock source is to be internal (**int**, provided by the emulator) or external (**ext**, provided by the target system). The internal clock speed is 8 MHz (system clock, CLKOUT). The emulator will operate at external clock speed from 4-16 MHz (entered clock or crystal frequency connected).

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

loc Monitor Location. This configuration item allows you to specify the location of the monitor program. The monitor may be located on any 4K byte boundary. If the < ADDRESS> specified is not on a 4K boundary, the 4K boundary below the address is used. The location of background monitors may be important because background cycles of the 70322/70320 emulator are always visible to the target system. In default, the monitor is located on 00000H through 00FFFH (physical address).

Note 

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

rrt Restrict to Real-Time Runs. This configuration item allows you to specify whether program execution should take place in real-time or whether commands should be allowed to cause breaks to the monitor during program execution. To restrict execution to real-time, set **rrt** equal to **en**. To allow breaks to the monitor during program execution, set **rrt** equal to **dis**. When runs are restricted to real-time, commands which access target system resources (display registers, step, or display/modify target system memory or I/O) are not allowed.

rad Physical to Logical Run Address Conversion. This configuration item allows you to specify the default method in which the emulation system will convert physical addresses specified in run and step commands to logical addresses. Setting **rad** equal to **maxseg** specifies that the low nibble of the physical address become the offset value; the high four nibbles become the segment value. Setting **rad** equal to **minseg** specifies that the low four nibbles of the physical address become the offset value; the high nibble and three hex zeros will become the segment value.

proc Microprocessor selection. This configuration item specifies microprocessor to be emulated. 70320 or 70322 are available for microprocessor name. If you select 70322 as a microprocessor to be emulated, 16K byte of emulation memory will be mapped automatically as internal ROM area (however, this does not appear as mapper term). Therefore, rest 108K byte is available for emulation memory. If you select 70320 as target microprocessor, amount of 124K byte emulation memory is available.

urst Specify whether accept or ignore reset signal from target system. If you specify **en** for this configuration item, emulator accept target system reset signal and execution will start from reset vector address as actual microprocessor does. This is true while running background monitor. If disabled with specifying **dis**, emulator always ignore target system reset signal regardless of foreground (executing user program) or background (executing monitor program). Reset signal from target system will be ignored completely.

Defaults The default values of the configuration items are listed below.

```
cf clk=int
cf rdy=unlk
cf loc=0FF800
cf rrt=dis
cf rad=minseg
cf proc=70322
cf urst=en
```

Related Commands **help**

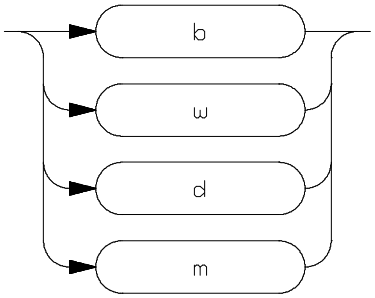
You can get an on line help information for particular configuration items by typing:

R> **help cf < CONFIG_ITEM>**

DISPLAY_MODE

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

Syntax



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

Parameters

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

display mode in the **ser** (search memory for data) command.

Defaults

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

Related Commands

mo (specify access and display modes)

m (memory display/modify)

io (I/O display/modify)

ser (search memory for data)

REGISTERS

Summary 70322/70320 register designators. Table A-1 shows all available register class names and register names.

< REG_CLASS>

< REG_NAME> Description

***(All basic registers)**

ah, al, aw	BASIC registers.
bh, bl, bw	
ch, cl, cw	
dh, dl, dw	
sp, bp, ix, iy	
ds0, ds1, ss	
pc, ps, psw	

port(Port registers)

p0	Port 0	
pm0	Port 0 mode	(Write Only)
pmc0	Port mode control	(Write Only)
p1	Port 1	
pm1	Port 1 mode	(Write Only)
pmc1	Port 1 mode control	(Write Only)
p2	Port 2	
pm2	Port 2 mode	(Write Only)
pmc2	Port 2 mode control	(Write Only)
pt	Port T	(Read Only)
pmt	Port T mode	

intr(Interrupt registers)

intm	External interrupt mode
ems0	External interrupt macro service control 0
ems1	External interrupt macro sercice control 1
ems2	External interrupt macro sercice control 2
exic0	External interrupt request control 0
exic1	External interrupt request control 1
exic2	External interrupt request control 2
ispr	In service priority

cri(Serial I/F registers)

rxb0	Recieve buffer 0 (Read Only)
txb0	Transmit buffer 0 (Write Only)
srms0	Serial recieve macro service control 0
stms0	Serial transmit macro service control 0
scm0	Serial mode 0
scc0	Serial control 0
brg0	Baud rate grnerater 0
sce0	Serial error (Read Only)
seic0	Serial error interrupt request control 0
sric0	Serial recieve interrupt request control 0
stic0	Serial transmit interrupt request control 0
rxb1	Recieve buffer 1 (Read Only)
txb1	Transmit buffer 1 (Write Only)
srms1	Serial recieve macro service control 1
stms1	Serial transmit macro service control 1
scm1	Serial mode 1
scc1	Serial control 1
brg1	Baud rate grnerater 1
sce1	Serial error (Read Only)
seic1	Serial error interrupt request control 1
sric1	Serial recieve interrupt request control 1
stic1	Serial transmit interrupt request control 1

tmr(Timer registers)

tm0	Timer 0
md0	Modulo/Timer 0
tm1	Timer 1
md1	Modulo/Timer 1
tmc0	Timer control 0
tmc1	Timer control 1
tmms0	Timer unit macro service control 0
tmms1	Timer unit macro service control 1
tmms2	Timer unit mcro service control 2
tmic0	Timer unit interrupt request control 0
tmic1	Timer unit interrupt request control 1
tmic2	Timer unit interrupt request control 2

dma(DMA control registers)

dmac0	DMA control 0
dmam0	DMA mode 0
dmac1	DMA control 1
dmam1	DMA mode 1
dic0	DMA interrupt request control 0
dic1	DMA interrupt request control 1

Proc(Processor status register)

stbc	Standby control
rfm	Refresh mode
wtc	Wait control
flag	User flag
prc	Processor control
tbic	Timer base interrupt control
idb	Internal data area base

Function The < **REG_CLASS**> names may be used in the **reg**(register) command to display a class of 70433 registers.

The < **REG_NAME**> names may be used with the **reg** command to either display or modify the contents of 70433 registers.

Refer to your 70433 use's manual for complete details on the use of the 70433 registers.

Related Commands **reg** (register display/modify)

Function The < **REG_CLASS**> names may be used in the **reg** (register) command to display a class of 70322/70320 registers.

The < **REG_NAME**> names may be used with the **reg** command to either display or modify the contents of an 70322/70320 register.

Refer to your 70322/70320 user's manual for complete details on the use of the 70322/70320 registers.

Related Commands **reg** (register display/modify)

Index

Notes