HP 64700-Series

Emulator Softkey Interface

Reference Manual

HEWLETT PACKARD

HP Part No. 64700-97000
Printed in U.S.A.
February, 1990

Edition 3

Notice

Hewlett-Packard Company
Logic Systems Division
8245 North Union Boulevard
Colorado Springs, CO 80920, 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	64740-90905, November 1988 E1188
Edition 2	64740-90901, April 1989 E0489
Edition 3	**64700-97000, February 1990**

Using this Manual

This manual, the *Softkey Interface Reference*, explains the general operation of the Softkey Interface. Use this manual with your *Emulator Softkey Interface User's Guide*. Each *Emulator Softkey Interface User's Guide* contains specific information about for your emulator.

Topics Covered

Topics covered in this manual include:

- Introduction to the Softkey Interface - Chapter 1
- Configuring the Emulator - Chapter 2
- Commands - Chapter 3
- Command Syntax and Descriptions - Chapter 3
- Coordinated Measurements - Chapter 4
- Measurement System Operation - Chapter 4
- Windowing Capabilities - Chapter 5
- Command Files - Chapter 6
- Manual Pages - Chapter 7
- Performance Verification for HP 98659A - Appendix A
- Error Messages - Appendix B

The Index contains terms and corresponding page numbers so that you can locate information quickly.

Understanding HP 64700 Terms

If you do not understand a term in this manual, refer to the *HP 64700 Emulators Glossary Of Terms* for a definition.

Conventions Used

Examples in this manual use these conventions:

ENTER: ***run*** ***from*** START <RETURN>

ENTER:	Instructs you to execute the command that follows.
run from	Softkeys are in bold italic type.
START	Entries you make are in normal text.
step	Bold type signifies commands and options in text.
<RETURN>	Press the keyboard **Return** key.

Using the Manuals

The HP 64700-Series Manual Maps direct you to the appropriate manuals for the various interfaces and information on using your emulator/analyzer. You can find the maps in the package marked *Read Me First*.

Contents

Illustrations

Tables

Notes

1

Introducing the Softkey Interface

Topics in this Chapter

- Description of HP 64700-Series Emulators
- Features of the HP 64700-Series Emulators
- Interactive Operation
- The Emulator and Your Program
- Activity while Programs Run
- The Emulator and Your Target System
- The Emulation Process
- Installing Software and Hardware

Description of HP 64700-Series Emulators

Each HP 64700-Series emulation system is a separate functional module when used with the Softkey Interface in the HP 64000-UX system environment. Each emulation system has three hardware modules packaged in a unit, with the emulation software and technical manuals.

The hardware modules include:

- Emulation Subsystem
- Emulation Memory Subsystem
- Emulation Analyzer (with optional external channels)

HP 64700-Series emulation systems may be used with “external” analysis for more sophisticated measurements. These are available as options to the HP 64700-Series emulators.

The HP 64700-Series emulator/analyzer is a tool used to aid you in the development of your target system hardware and software. Proper use can help ensure that your hardware and software work together. The HP 64700-Series emulator/analyzer can be used with or without target system hardware, or with other products to debug your target system hardware and to integrate your software and hardware. Figure 1-1 shows an example emulation solution.

Emulation Subsystem

The emulation subsystem allows you to access and modify internal microprocessor registers, and locations or blocks of memory. In addition, you can access code instruction-by-instruction by stepping through a program.

Emulation Controller

The emulation controller controls the interaction between the HP 64000-UX operating system software and the HP 64700-Series emulation hardware. This board is the major interface between the emulator and memory and analysis. The HP 64000-UX operating software can control the emulation microprocessor “reset” and the bus activity directly through the controller. The two main functions of the controller are to convert emulator timing signals to compatible memory and analysis bus signals and to provide a channel to the emulator for hardware configuration.

Emulation Memory Subsystem

The emulation system includes emulation memory implemented in static RAM. Emulation memory can be used in place of your target system ROM or RAM. Program modules that will ultimately reside in target system ROM can be developed and thoroughly tested before being permanently stored in ROM.

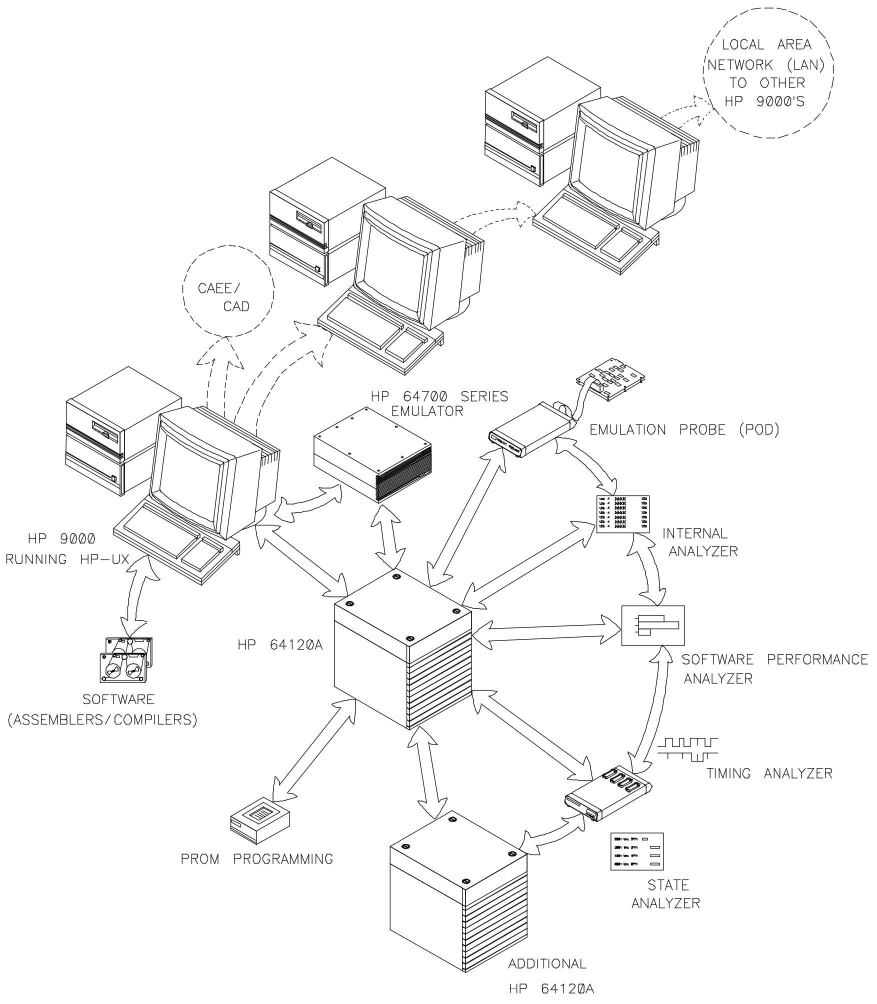


Figure 1-1. Integrated Products Solution

The memory controller supervises emulation and target memory. The controller monitors the emulation memory bus to determine the type of memory that is to be accessed (emulation memory or target system memory). The memory controller is the interface between emulation memory and the HP 64000-UX operating system. The operating software communicates with the emulation microprocessor, transferring data to and from emulation memory. This transfer is accomplished through the memory controller board. Besides providing an access port into emulation memory, the memory controller contains a hardware mapper programmed to map the memory resources into emulation or target system RAM or ROM, or guarded memory spaces.

Emulator Probe

The emulator probe is at the end of the probe cable. This flexible cable extending from the lower right part of the HP 64700-Series emulator front panel allows you to connect the emulator to the target system. You can then use the emulator to help in the design and debugging of target system hardware and software.

Emulation Analyzer

The emulator can be used for software development before you finish target system hardware development. Program modules can be run by the emulator, and trace measurements can be made by the emulation analyzer.

The emulation analyzer is the equivalent of a logic analyzer. It accepts trigger specifications, then monitors the emulation analysis bus to determine if the specified event has occurred. When the event occurs, the analyzer traces 1024 states of program execution and stores them in a trace memory. Trace data can then be displayed on the terminal. When you specify counts, the analyzer will capture 512 states.

External Analysis Channels

Your emulator may contain an optional external analyzer. The external analyzer provides sixteen external trace signals, with 2 external clock inputs. You can use the external analyzer as an extension to the emulation analyzer, as an independent state analyzer, or as an independent timing analyzer.

Figure 1-2 shows a basic block diagram of the HP 64700 emulation system.

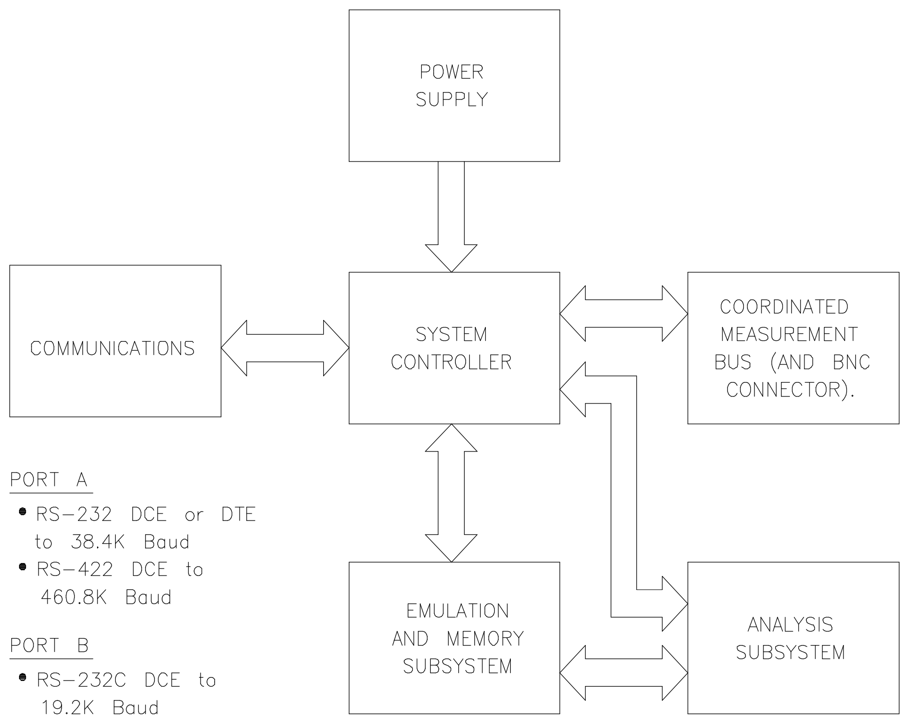

POWER SUPPLY
COMMUNICATIONS
SYSTEM CONTROLLER
COORDINATED MEASUREMENT BUS (AND BNC CONNECTOR).
PORT A
• RS−232 DCE or DTE to 38.4K Baud
• RS−422 DCE to 460.8K Baud
PORT B
• RS−232C DCE to 19.2K Baud
EMULATION AND MEMORY SUBSYSTEM
ANALYSIS SUBSYSTEM

Figure 1-2. HP 64700 Emulation System Block Diagram

Features of the HP 64700-Series Emulators

The HP 64700-Series emulator/analyzer is just one tool available for embedded microprocessor design, test, and debug. The tasks simplified by the HP 64700-Series emulators include software debugging, hardware debugging, and hardware and software integration. You can do these using the basic emulator features described in table 1-1.

Note

The asterisk (*) listed by the following features indicate that the feature does not exist on all emulators. Refer to your *Emulator Softkey Interface User's Guide* for details.

Table 1-1. Emulator Softkey Interface Features

Program Loading and Execution	The code you develop using the editor, compilers, assembler, and linker can be loaded into memory and executed by the emulator.
Run and Stop Controls	Programs may be run from address or symbolic locations. Emulation can be stopped by breaking into the emulation foreground/background monitor or by resetting the microprocessor.
Memory Display and Modification	You can display locations or blocks of memory and modify any that can be changed.
I/O Ports Display and Modification*	You can display and modify input/output (I/O) port address locations and values.
Global and Local Symbols Display	You can display the addresses associated with your program's global and local symbols while working in emulation.

Table 1-1. Emulator Softkey Interface Features (Cont'd)

Registers Display and Modification	You can display and modify internal registers of the microprocessor.
Analysis (With Optional External Channels)	You can use the analyzer to observe real-time activity on the emulation microprocessor bus. With external channels, you can observe real-time activity on the external signals where you have connected the probes.
Program Stepping	You can execute code instruction-by-instruction, at the assembly level or by source lines, gaining access to the internal machine states between instructions.
Memory Mapping	You can use emulation memory and target memory at the same time by defining the characteristics of the blocks of memory.
Memory Characterization	You can assign emulation and target system memory as ROM or RAM. By doing this, you can test “ROM” code without using ROM hardware.
Breakpoint Generation	You can transfer program execution to a monitor routine with the occurrence of a particular machine state or range of states.
Clock Source Selection	You can select the target system clock for the emulation clock source. If a target system clock is not available, use the internal clock in the emulator.
Simulated I/O	You can set up your emulation system to communicate with HP 9000 files, the workstation keyboard, and the workstation display using simulated I/O. In addition, you can use the emulation system to execute HP-UX commands, which allows the emulator to communicate with other HP 9000 input/output devices, such as printers, plotters, and modems.

Interactive Operation

Emulation and analysis features can be used in an interactive manner between an emulator and another module through the HP 64000-UX measurement system. That module could be another emulator or a state or timing analyzer. Interaction allows the integration of development work on multi-processor designs, or more elaborate and detailed analysis of a design, or both.

Using the CMB

You can use the Coordinated Measurement Bus (CMB) between two HP 64700 Series emulators to synchronize a measurement. Or, you can use one or more HP 64700 Series emulators, with the HP 64306A IMB/CMB Interface board, to make coordinated measurements with instruments in the HP 64120 Logic Development Station. Functions provided include:

- cross-triggering of analyzers
- coordinated emulation starts
- simultaneous emulation breaks
- simultaneous emulation resume operations

Chapter 4 contains additional information about the CMB.

The Emulator and Your Program

The emulator does not permanently alter your program, but it may affect the execution of your program. The way in which the emulator affects your program depends on the emulation operations you select.

Real-Time versus Nonreal-Time

You can configure the emulator to operate in either of two modes: real-time or nonreal-time.

"Real-time" refers to continuous execution of your target system program without interaction with the HP 64000-UX host computer (except as instructed by you).

Interaction occurs when a break to the emulation monitor program is initiated either manually by you or automatically. The emulation monitor is the tool that allows you access to the internal registers of the microprocessor and target system memory.

Whenever the emulator is running under control of the emulation monitor program, your program is no longer executing in real-time. The emulation monitor program may be described in more detail in your *Emulator Softkey Interface User's Guide.*

Activity while Programs Run

While your program is running, the emulation microprocessor generates address information for each cycle. The hardware differentiates between your target system and the emulation system resources based on that address information.

If the emulator identifies a target system resource having the current address, data path buffers between your target system and the emulation processor are enabled. If the address is mapped to emulation memory space, data path buffers between the emulation processor and the emulation bus are enabled.

As your program runs, the emulation analyzer monitors activity on the emulation analysis bus. You can instruct the analyzer to store this program flow for later display without interrupting the real-time flow of the program.

Emulation Monitor Program Control

Emulation functions are implemented by seizing control of the emulation processor from your program, and transferring control to the monitor program.

The emulation monitor program is the link between the emulation processor and the host system. The foreground emulation monitor is a program written in assembly code. It is located within emulation memory, because this is the only memory directly accessible in the HP 64700-Series emulators. The background monitor is in background memory, which you cannot access.

The monitor program is constructed of several separate routines. Some routines execute automatically whenever the monitor program is entered. These routines extract the internal microprocessor information that existed at the time of entry. You can then display this information on your terminal. If, for instance, the monitor program is entered after the execution of one of your program instructions, the internal machine state that exists then is available.

The Emulator and Your Target System

The goal of the HP 64700-Series emulator is to appear just like the microprocessor that will eventually control your target system. The emulator can give you complete and immediate insight into the clock-by-clock operation of the target system. The function, signal quality, drive capacity, and other factors at the emulation probe should be indistinguishable from those of the actual microprocessor. This characteristic is called *transparency*.

Functional Transparency

Functional transparency refers to the ability of the emulator to function in the same way as the microprocessor you will use in your target system. Functional transparency requires that the emulator execute your program, generate outputs, and respond to inputs in the same manner as the actual target microprocessor.

Timing Transparency

Timing transparency refers to the timing relationships between signals at the location where you plug the emulator in to the target system. There may be a difference between the timing of signals at the emulation probe, and the timing of signals in the target system. Execution in the emulation environment is designed to run at the maximum speed of the microprocessor.

Timing diagrams for your emulator and target system microprocessor may be included in the *Terminal Interface User's Guide* for your emulator.

Electrical Transparency

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

The Emulation Process

There are three steps to the entire emulation process:

1. Prepare the software.

2. Prepare the emulator.

3. Use the emulator.

Prepare the Software

Preparing the software consists of creating a program, assembling or compiling the program, and linking the assembled or compiled program modules. Refer to the appropriate *Assembler/Linker Manual* or *Compiler Manual* for more information.

Prepare the Emulator

Preparing the emulator consists of starting the emulator using the **emul700** command (or using the HP 64000-UX measurement system). The emulator device file named */usr/hp64000/etc/64700tab* must contain the emulator name. Optionally, you can initialize and define a measurement system for each HP 64700-Series emulator. This is described in chapter 4 . After defining the emulator, you configure it for your particular application. Configuration details are covered in each *Emulator Softkey Interface User's Guide.*

Use the Emulator

Using the emulator consists of loading absolute code (generated by linking program modules) into the emulator. Then you use the emulator to observe the program as it runs, display the contents of the registers and/or memory, and to debug your hardware and software. Use of the emulator with the Softkey Interface is described in your *Emulator Softkey Interface User's Guide.*

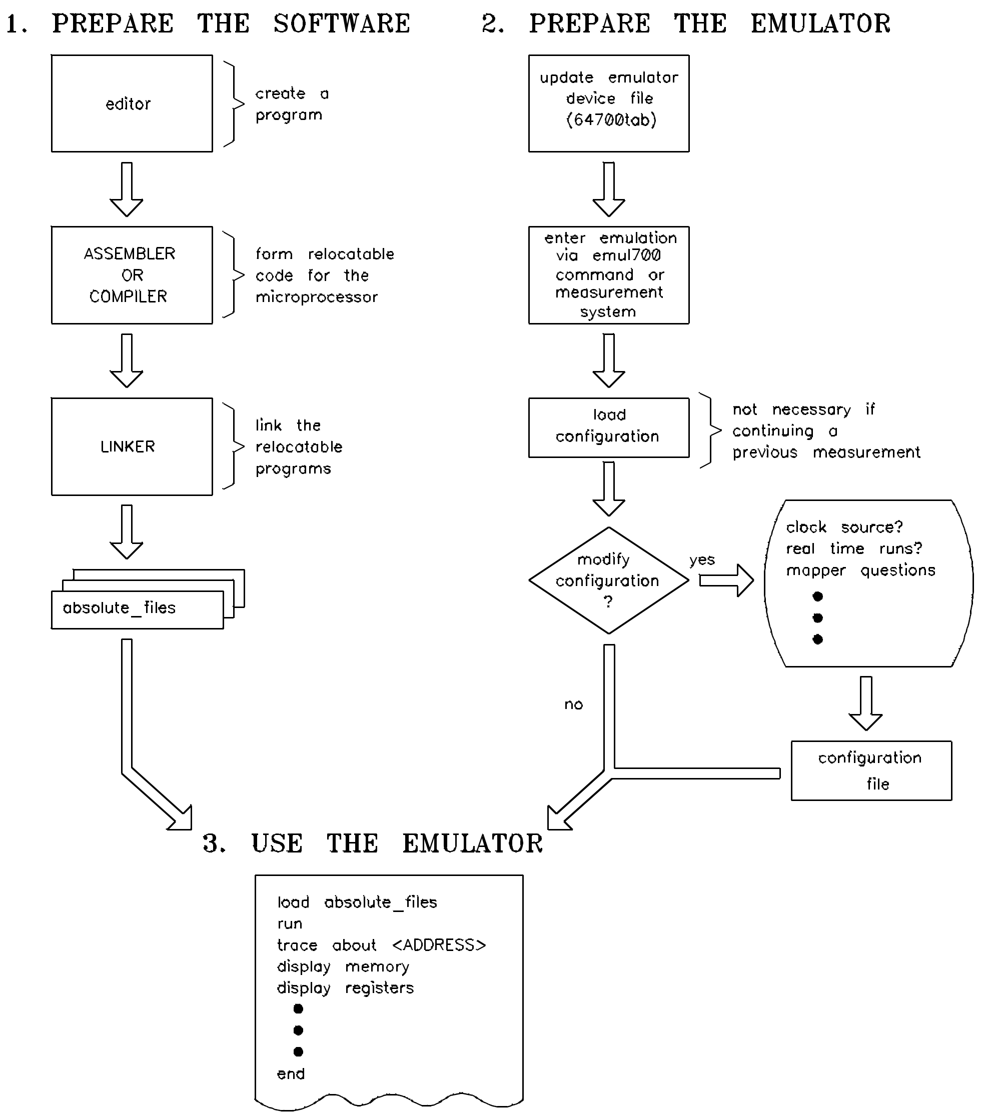


Figure 1-3. Steps for Using the Emulator

Installing Software and Hardware

A *Softkey Interface Installation Notice* is supplied with your HP 64700-Series emulator Softkey Interface documentation. This notice describes what you should do to install and/or update the Softkey Interface software for your emulation system. It also describes the interface hardware you must install in the host computer.

Installing Software

As described in the *Softkey Interface Installation Notice*, you must install software for:

- HP 64801 Operating System
- HP 64700-Series Emulator
- Interface Card driver (for HP 98628A, HP 98642A, or HP 98659A)

Depending on your host computer system configuration and current software, you may need to update the host operating system. Refer to the *Softkey Interface Installation Notice* supplied with your HP 64700-Series product for more information.

The *Softkey Interface Installation Notice* also describes how to make a device file for the interface card you are using, and how to modify the “64700tab” file.

Installing Hardware

After you install software, you must install an interface card to allow the HP 9000 to communicate with your HP 64700-Series emulator. Then you can connect your HP 64700-Series Emulator to the interface card.

2

Emulation Configuration

Overview

This chapter describes:

- Address Conventions for some Emulators
- Real-Time and Nonreal-Time Operation
- Emulation Configuration Questions

The emulator needs to know about the clock and memory resources available on your target system. The emulation configuration questions allow you to define your target system microprocessor for the emulator.

Because emulation memory provides memory to be used if your target system memory is not yet available, you must define the mapping of memory resources. Also, you must define the mode of operation (real-time or non-real-time) for the emulator, and whether you want to be notified of attempts to write to ROM.

Your *Emulator Softkey Interface User's Guide* contains details about configuring your emulator.

Address Conventions for Some Emulators

You must understand the address conventions for your microprocessor to map the emulation and target system memory later in this chapter. Depending on your emulator, the emulation software may use two different memory address conventions (physical address and logical address). If your emulator uses both physical and logical addressing, it will accept either address convention after configuration is complete.

Physical Address

You use physical addresses for specifying the memory map during configuration setup. The address takes the form 0XXXXb, where "b" is the number base (B = binary, Q or O = octal, D = decimal, and H = hexadecimal). A leading zero is only required where the leading character is a non-numeric hexadecimal character (as in 0FFFH).

Logical Address

Logical addresses have a segment number, a colon separator, and an offset number within the given segment. The logical address takes the form 0XXXXb:YYYYb, where "b" is the number base (B = binary, O = Octal, D = decimal, and H = hexadecimal). The "X" term is the segment identifier, and the "Y" term is the offset identifier. A leading zero is only required if the leading character is a non-numeric hexadecimal character (as in 0FFFH).

Real-Time and Non-Real Time Operation

The emulator allows you to restrict operation to real-time program execution. "Real-time" here is not based on whether wait states are inserted or not, because none are needed. Instead, real-time refers to the continuous execution of your program without interaction from the host computer, except as instructed by you.

Caution

POSSIBLE DAMAGE TO CIRCUITRY !

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

Configuration Questions

To modify the emulator configuration after you have entered the emulator Softkey Interface. . .

```
PRESS:  modify  config  <RETURN>
```

The following questions and options appear sequentially when you choose default values. Some options may be different for your particular emulator.

```
Microprocessor clock source?  internal
Enter monitor after configuration?  yes
Restrict to real-time runs?  no
Modify memory configuration?  no
Modify emulator pod configuration?  no
Modify debug/trace options?  no
Modify simulated I/O configuration?  no
Modify external analyzer configuration?  no
Modify interactive measurement specification?  no
Configuration file name?
```

Depending on the answers you supply, you may enter other levels of the configuration process. The options are described on the screen as you go through the configuration process.

For details about the configuration items for your emulator, refer to your *Emulator Softkey Interface User's Guide*. For information about configuring the analyzer, refer to the *Analyzer Softkey Interface User's Guide*.

The rest of this chapter describes the emulation configuration questions and available options.

Microprocessor clock source?

internal	When you select "internal", the emulation processor will use the oscillator contained in the emulator as its clock source. The oscillator speed may be different for each emulator, so check your *Emulator Softkey Interface User's Guide* for details.

external	When you select “external”, the emulation processor will use the clock contained in the target system.

When you change this part of the emulation configuration, the emulator will enter the reset state.

Enter monitor after configuration?

yes	When you choose “yes”, the emulator will enter the monitor after you modify the emulation configuration. If this process fails, the previous configuration will be restored. The process could fail if you select an external clock, but don’t provide one.
no	When you choose “no”, the emulator will not enter the monitor after you modify the emulation configuration.

Restrict to real-time runs?

no	If runs are not restricted to real time, the emulation software performs all commands upon request, and detects entry to the emulation monitor at any time.

To allow this to happen, the emulation system must be capable of entering the monitor program at any time. The monitor program enables the emulation system controller to access the memory mapped as user (target system) memory. User memory is accessed when a command to display, list, modify, load, or store user memory is processed.

In nonreal-time mode, the emulation system forces entry into the emulation monitor program whenever a command that requires access to the microprocessor registers, target system I/O, or target system memory is processed. If your program was executing at the time of the request, the emulation system forces entry into the emulation monitor. When the monitor obtains the necessary information, your program resumes execution.

Entering the emulation monitor program extends the execution time of your program. If your system is dependent on execution time, try restricting operation to real-time.

yes	While your program is executing, emulation commands that require the monitor program are restricted.

The commands that are restricted during real-time runs are listed below:

```
copy data
copy memory
copy registers
display data
display memory
display registers
load <file>
load user_mem
modify memory
modify register
modify software_breakpoints
store memory
The above commands are restricted if applied to user (target system) memory.
```

Following a program run, the emulator remains in the real-time mode until a break from one of the following sources is detected by the emulation software. The break conditions can be:

1. A memory break caused by a write to ROM (if this configuration item is enabled) or an access to guarded memory.

2. An analysis break from a **trace** command that includes a **break_on_trigger** specification.

3. A **break** command.

4. A **run from** command.

5. A **step** command.

6. The lack of a READY signal on the Coordinated Measurement Bus (CMB) if CMB operation is enabled.

Once a break is detected, the emulator enters the emulation monitor. Once the emulation monitor is detected, the commands listed above in steps 1 through 6 are enabled. The emulation system returns to the real-time mode when execution returns to your program with a **run** command.

Modify memory configuration?

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

When you begin an initial emulation session, the emulator starts in the default emulation configuration. The default configuration assigns some blocks of memory as emulation RAM. You must configure (map) the memory space used by your program.

Base your decisions about memory mapping on the length and features of your target system program(s). As you progress with your program development, your memory map requirements probably will change.

For example, additional memory in the target system may become available. Rather than start a new configuration session from the beginning, you can modify your present configuration. You can then either keep the same configuration file name (by writing over the current file) or assign the new configuration a new file name.

If you assign a new configuration file name, and you use a command file to enter the emulation session, remember to change the name of the configuration file in the command file.

Options to the “Modify memory configuration?” question are:

yes	This response allows you to alter the way in which emulation and target system memory is defined and used. The microprocessor is reset, and the configuration questions are presented one at a time with their current default values. Each default response can be

	entered as listed by pressing **< RETURN>** . Or, you can modify the response for the current emulation session, then enter it using the **< RETURN>** key.
no	This response skips modifications to the memory definitions. A response of “no” configures the memory as specified in the current emulation configuration file.

Mapping Memory

To perform emulation, the memory mapper must be set up to use emulation memory and/or target system memory resources. The memory mapper allows you to divide the microprocessor address space into several blocks that can be individually assigned any one of the five available descriptors: emulation RAM, emulation ROM, target RAM, target ROM, or guarded memory.

During emulation, the mapper monitors the address bus and gives the descriptor for the address present at any given time. The emulator hardware uses this information to control data and program activity between the emulation microprocessor and the memory resources.

Memory Map Definition

The map has several address range definition entries and a choice for default memory. The number of ranges depends on the emulator type. Each entry defines a particular address range as a possible memory type. Any address range not defined by an entry maps to the memory default specification.

Entries do not need to be an integral multiple of the block size. Once the mapper software processes the inputs, the boundaries round to integral multiples of the block size. Therefore, assuming a block size of 4 kilobytes, if you enter an address range of 0 through 07FH, one entire 4 kilobyte block of memory is allocated (0 thru 0FFFH). The block size for your emulator is listed at the top of the memory map display as shown in figure 2-1.

The final boundaries include all the memory space specified, plus the remainder of any partially specified blocks. The remaining parts of your microprocessor address range, not covered by an entry, map to the memory default.

When you specify target memory for a given address range, all memory cycles within that address range are sent to the target system. All **memory load** and **display** operations for target system memory are done using the emulation monitor program.

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

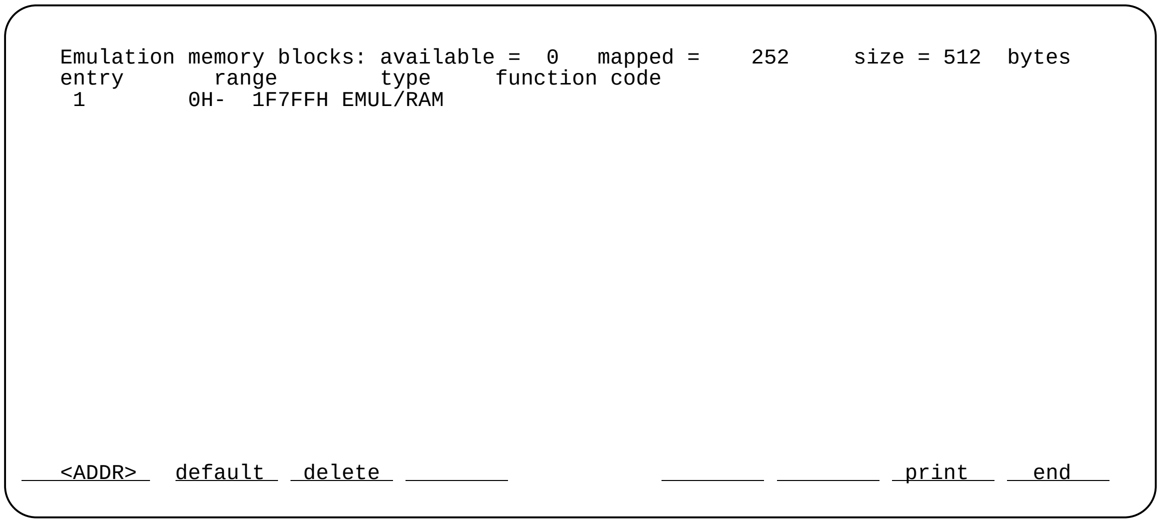
Guarded memory is memory that the emulation system cannot access. Examples of this may occur where there is a memory shadow from another memory block in the same address space (due to partial address decoding in your target system memory). Or, memory in that range is either not developed or not available to your system. The block of memory may not even exist.

Memory Map Organization

The default memory map is shown in figure 2-1. The top line of the display shows the number of emulation memory blocks available for mapping, the number of blocks currently mapped, and the size of the blocks. Each new mapper entry updates the “available” and “mapped” block numbers to reflect the current values. The number of available blocks depends on the amount of emulation memory in the emulator.

If you enter emulation without loading a configuration file, the map contains the default map entries. Any attempt to end the emulation session while the memory map is blank causes an error message to be displayed.

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



```
Emulation memory blocks: available =  0   mapped =    252     size = 512  bytes
entry       range        type     function code
 1        0H-  1F7FFH EMUL/RAM
```

Figure 2-1 Default Memory Map (68000 Emulator)

Entering Mapper Blocks

All mapper entries consist of an address or address range and a descriptor, which defines the type of memory within the specified addresses. Once you enter the desired address or address range, the available descriptors appear as softkeys.

You must select one of five memory descriptors for each memory address range that you map. The descriptors are target ROM, target RAM, emulation ROM, emulation RAM, and guarded.

Define the mapper blocks using the syntax shown in figure 2-2.

The memory mapper options are defined as:

target	This refers to memory supplied by your target system. Mapping an address range to target memory space does not require any emulation memory. Therefore, the number of available memory blocks listed at the top of the mapper screen does not change when specifying target space.

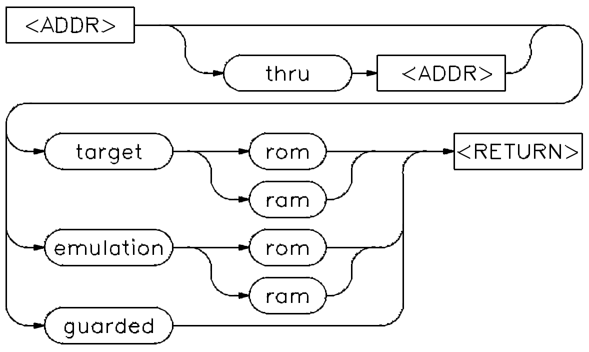


Figure 2-2. Memory Mapper Block Syntax

emulation	This refers to memory supplied by the emulation system. When specifying emulation memory, the number of available blocks of emulation memory decreases by the number of blocks required for the assignment.
guarded	This option designates an address range that you do not plan to access. Any microprocessor access to a location within such a range results in a break of the program execution. No emulation memory is used when specifying an address range as “guarded”.
rom	ROM defines memory that can be read but cannot be modified by the processor. The emulator can detect an error on the occurrence of write cycles to this memory. Emulation memory that is RAM but is mapped as ROM performs as ROM during emulation.
ram	RAM defines memory that can be read from or written to without restriction.

< ADDR>	The address specifying a particular memory location can be a pattern of 32 bits or less. The pattern can be represented by a binary, octal, decimal, or hexadecimal number.

The first < ADDR> of a range specification can be the starting address of a block boundary, or an address within the memory block. If you enter an address within the memory block, the system converts this address to the starting address of the block prior to its mapping. If the most significant digit in the address is numeric, you do not have to include a leading zero.

If you specify a single address, rather than a range of addresses, only the block containing that address is mapped. Because the entire block is automatically used, the “thru < ADDR> ” portion of the syntax does not need to be entered. Enter only a single address and a descriptor.

Default Memory

Any address ranges that are unmapped when the mapping session ends are assigned to the memory type specified as the default. The default descriptor can be defined as target RAM, target ROM, or guarded by using the **default** command. If no default descriptor is specified, all unmapped memory blocks are defined as guarded memory.

The syntax for the default memory type command is shown in figure 2-3.

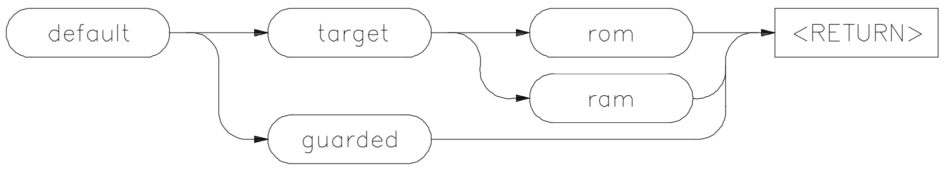


Figure 2-3. Default Memory Syntax

Deleting Blocks

One or all of the memory map entries can be removed by using the **delete** command. The syntax for the **delete** command is shown in figure 2-4.

Ending the Mapping Session

You can exit the memory map configuration session by pressing the **end** softkey followed by **< RETURN>** . If you try to end the mapping session while the memory map is blank, an error message is displayed.

Modify emulator pod configuration?

When you select **yes** to modify the emulator configuration, you see all the emulator-specific configuration questions. These will differ for each emulator, so refer to your *Emulator Softkey Interface User's Guide* for details.

yes	The emulator-specific set of configuration questions are accessed, allowing you to view and/or modify the emulator configuration items.
no	When you answer **no** to this question, you bypass modifying the emulator-specific configuration questions.

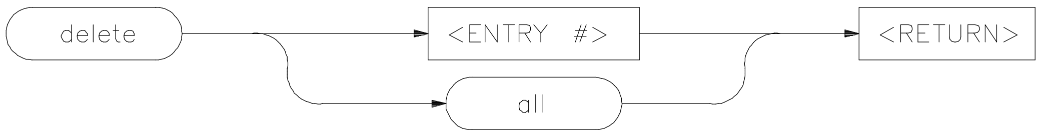


Figure 2-4. Deleting Memory Map Blocks

Note

Do not use **pod_command** to modify the emulator configuration. If you do this, you will not see the new configuration changes reflected when you use the **modify configuration** command.

Modify debug/trace options?

Answering **yes** allows you to change the way the emulation or external analyzer debugs programs and captures trace information.

yes	When you choose **yes**, you see questions about breaking the emulation processor on writes to ROM. In addition, depending on the emulator you are using, you can redefine the trap number for software breakpoints, and define whether to trace foreground or background operation, or both.
no	Answering **no** leaves the debug/trace options as previously defined.

Modify simulated I/O configuration?

This configuration question allows you to simulate various functions of the HP 64700-Series emulator running on the host computer.

no	By answering **no** to the “Modify simulated I/O configuration?” question, you bypass all modifications to the simulated I/O features.
yes	Once you have responded with **yes** to this question, you are asked the following question:

Enable polling for simulated I/O?

no	Answering **no** bypasses modification to the simulated I/O control addresses.
yes	Answering **yes** allows you to define addresses for control address 1 through 6. Once you have answered those questions, you can

	specify names for standard input, output, and error files.

The last simulated I/O question to appear is:

Enable simio status messages?

yes	If you enable display of simulated I/O status messages, the command and return code will be shown on screen.
no	If you disable simulated I/O status messages, simulated I/O will run faster than if the status messages are enabled.

Refer to the *HP 64000-UX Simulated I/O Manual* for details about using simulated I/O.

Modify external analyzer configuration?

The external analyzer is used to capture information on signals external to the HP 64700-Series emulator. If you want to modify the external analyzer configuration, answer **yes** to this question.

no	When you answer **no** to this question, all modifications to the analyzer configuration are bypassed.
yes	When you answer **yes**, a specific set of analyzer configuration questions will be presented for your viewing and/or modification. The specific questions are described in the *Analyzer Softkey Interface User's Guide.*

Modify interactive measurement specification?

When you choose **yes** to modify the interactive measurement specification, you can define drivers and receivers for the internal trigger signals in the emulation analyzer (trig1 and trig2).

You can configure trig1 to drive or receive the BNC port trigger and CMB trigger. You can configure trig2 to drive or receive the BNC port trigger, or receive CMB trigger. In addition, trig2 can be configured to drive the emulator or analyzer, or can be received from the analyzer.

yes	If you want interaction or to modify a previously defined specification, answering **yes** to this question allows you to review and modify this specification as necessary.
no	You bypass access to the specific set of interactive measurement configuration questions when you answer **no** to this question.

For details about making measurements, see the chapter on *Coordinated Measurements* in this manual.

Configuration file name?

You can save modifications to the emulator configuration in a file that can be loaded into the emulator at another time. To do this, when this question appears, type in the name of a file where you want the configuration stored. You can include multiple levels of subdirectories.

The first time you go through the configuration process, you will not see a default file name. If you modify the configuration again during the emulation session, the file name specified last will appear as the selection to this question.

An example default configuration file (for the 68000 emulator) is shown in figure 2-5.

```
BEGIN MEMORY MAP
default guarded
0H thru 01F7FFH emulation ram
END MEMORY MAP
Micro-processor clock source?   internal
Restrict to real-time runs?   no
Enter monitor after configuration?  yes
Inverse assembly syntax to use?   64845
Monitor type?   background
Monitor address?   0FFF800H
Monitor function code?   none
Enable bus arbitration?   yes
Tag bus arbitration for analyzer?   no
Interlock emulator DTACK with user DTACK?   no
Enable Bus Error on emulation memory accesses?   no
Respond to target system interrupts?   yes
Reset value for Supervisor Stack Pointer?   1FFEH
Target memory access size?   bytes
Drive background cycles to target system?   yes
Value for address bits A23-A16 during background cycles?   0
Function code for background cycles?   supr prog
Break processor on write to ROM?   yes
Enable software breakpoints?   yes
Trap number for software breakpoint (0..0FH)?   0000FH
Trace background or foreground operation?   foreground
Should BNC drive or receive Trig1?   neither
Should CMBT drive or receive Trig1?   neither
Should BNC drive or receive Trig2?   neither
Should CMBT drive or receive Trig2?   neither
Should Emulator break receive Trig2?   no
Should Analyzer drive or receive Trig2?   neither
Should emulation control the external bits?   yes
Threshold voltage for bits 0-7 and J clock?   TTL
Threshold voltage for bits 8-15 and K clock?   TTL
External analyzer mode?   emulation
Slave clock mode for external bits?   off
Edges of J clock used for slave clock?   none
Edges of K clock used for slave clock?   none
Edges of L clock used for slave clock?   none
Edges of M clock used for slave clock?   none
First external label name?   xbits
First external label start bit?   0
First external label width?   16
First external label polarity?   positive
Define a second external label?   no
Second external label name?   low_byte
Second external label start bit?   0
Second external label width?   8
Second external label polarity?   positive
```

Figure 2-5. Example 68000 Configuration File

```
Define a third external label?   no
Third external label name?   hi_byte
Third external label start bit?   8
Third external label width?   8
Third external label polarity?   positive
Define a fourth external label?   no
Fourth external label name?   bit0
Fourth external label start bit?   0
Fourth external label width?   1
Fourth external label polarity?   positive
Define a fifth external label?   no
Fifth external label name?   bit1
Fifth external label start bit?   1
Fifth external label width?   1
Fifth external label polarity?   positive
Define a sixth external label?   no
Sixth external label name?   bit2
Sixth external label start bit?   2
Sixth external label width?   1
Sixth external label polarity?   positive
Define a seventh external label?   no
Seventh external label name?   bit3
Seventh external label start bit?   3
Seventh external label width?   1
Seventh external label polarity?   positive
Define an eighth external label?   no
Eighth external label name?   bit4
Eighth external label start bit?   4
Eighth external label width?   1
Eighth external label polarity?   positive
Enable polling for simulated I/O?   no
Simio control address 1?   SIMIO_CA_ONE
Simio control address 2?   SIMIO_CA_TWO
Simio control address 3?   SIMIO_CA_THREE
Simio control address 4?   SIMIO_CA_FOUR
Simio control address 5?   SIMIO_CA_FIVE
Simio control address 6?   SIMIO_CA_SIX
File used for standard input?   /dev/simio/keyboard
File used for standard output?   /dev/simio/display
File used for standard error?   /dev/simio/display
Enable simio status messages?   yes
```

Figure 2-5. Example 68000 Configuration File (Cont'd)

Notes

3

Commands

Overview

This chapter describes:

- Softkey Interface Features
- Syntax Conventions
- Summary of Commands
- A Syntax for all Emulators
- Command Descriptions

Softkey Interface Features

Softkeys

You enter Softkey Interface commands by pressing softkeys whose labels appear at the bottom of the screen. Softkeys provide for quick command entry, and minimize the possibility of errors.

Command Completion

You can type the first few characters of a command (enough to uniquely identify the command) and then press **Tab**. The Softkey Interface completes the command word for you.

Command Word Selection

If you entered a command, but want to make a change or correction, press the **Tab** key to position the cursor at that word. Pressing **Tab** moves the cursor to the next word on the command line. Pressing **Shift Tab** moves the cursor to the previous word.

Command Line Recall

Softkey Interface commands that you enter are stored in a buffer and may be recalled by pressing **CTRL r**. Pressing **CTRL b** cycles forward through the recall buffer.

Command Line Erase

Instead of pressing the **Back space** key to erase command lines, press **CTRL u**. You can then reenter the command. Pressing **Clear line** erases the command line from the cursor position to the end of the line.

Multiple Commands on one Line

You can enter more than one command at a time by separating the commands with a semicolon (;).

Change Directory

You can change your working directory while in emulation using the **cd** command. This command does not appear on the softkey labels. Typing **pwd** on the command line will display the name of your current working directory on the status line.

Working Symbol

The Symbolic Retrieval Utilities (SRU) handle symbol access within emulation. SRU maintains trees representing the symbol structure and scoping within your program code. You can specify a specific path in the tree using the **cws** (current working symbol) command. After you specify a symbol in this way, other symbol accesses are assumed to be relative to this symbol unless you specify complete paths. You can display the working symbol in use with the **pws** (print working symbol) command. The working symbol will be displayed on the emulator status line.

Refer to the **--SYMB--** syntax pages and the *HP 64000-UX System User's Guide* for more information on symbols and the Symbolic Retrieval Utilities.

Name of Emulation Module

While operating your emulator Softkey Interface, you can verify the name of the emulation module. This is also the logical name of the emulator in the emulator device file. To find the name of your emulation module, enter **name_of_module** **< RETURN>** . The name of the emulation module appears on the Status line.

Set Environment Variables

You can set an HP-UX shell environment variable from within the Softkey Interface. To do this, use the format:

```
set <environment variable> = <value>
```

For example, you could enter:

```
set PRINTER = "lp -s" <RETURN>
export PRINTER
```

After you set an environment variable from within the Softkey Interface, you can verify the value of it by entering **!set** < **RETURN**> . Be sure to export the value of a variable after setting it. This ensures the variable is visible to application programs.

Filters and Pipes

You can specify HP-UX filters and pipes as the destination for information while using the **copy** command. See the description of the **copy** command in this chapter for details.

Command Files

You can execute a series of commands that have been stored in a command file. You can create command files using the **log_commands** command or by using an editor on your host computer. Once you create a command file, you can execute the file in the emulation environment by typing the name of the file on the command line and pressing < RETURN> . See the chapter on *Command Files* for more information.

Help Command

A **help** command is available to you within an emulation session. Several methods are available for displaying help information about a command. You can use any of these methods:

1. ENTER: **help** and press a softkey that appears

2. ENTER: **?** and press a softkey that appears

3. ENTER: **pod_command** "help emul"

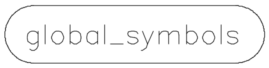
   ENTER: **display pod_command**

Syntax Conventions

Conventions used in the command syntax diagrams are defined below.

Oval-shaped Symbols

Oval-shaped symbols show options available on the softkeys and other commands that are available, but do not appear on softkeys (such as **log_commands** and **wait**). These appear in the syntax diagrams as:

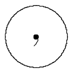


Rectangular-shaped Symbols

Rectangular-shaped symbols contain prompts or references to other syntax diagrams. Prompts are enclosed with angle brackets (< and >). References to other diagrams are shown in all capital letters. Also, references to expressions are shown in all capital letters. Examples of expressions are —EXPR— and —SYMB— (see those syntax diagrams). These appear in the following syntax diagrams as:

Circles

Circles indicate operators and delimiters used in expressions and on the command line as you enter commands. These appear in the syntax diagrams as:



The —NORMAL— Key

The softkey labeled **—NORMAL—** allows you exit the —SYMB— definition, and access softkeys that are not displayed when defining expressions. You can press this key after you have defined an expression to view other available options.

Summary of Commands

Softkey Interface commands are summarized in table 3-1.

Table 3-1. Summary of Commands

!HP-UX_COMMAND	display local_symbols_in	performance_measure-ment_run
break	display memory[4]	pod_command
cd (change directory)[3]	display registers[1]	pwd (print working directory)[3]
cmb_execute	display simulated_io[2]	pws (print working symbol)[3]
<command file>[3]	display software_breakpoints	reset
copy data[4]	display status	run
copy display	display trace	set
copy error_log	end	specify
copy event_log	help[3]	step
copy global_symbols	load <absolute_file>	stop_trace
copy help	load configuration	store memory
copy io_port[1]	load emul_mem	store trace
copy local_symbols_in	load trace	store trace_spec
copy memory[4]	load trace_spec	trace
copy pod_command	load user_memory	wait[3]
copy registers[1]	log_commands[3]	
copy software_breakpoints	modify configuration	
copy status	modify io_port[1]	
copy trace	modify keyboard_to_simio[2]	
cws(change working symbol)[3]	modify memory[4]	
display data[4]	modify register[1]	
display error_log	modify software_breakpoints[1]	
display event_log	name_of_module[3]	
display global_symbols	performance _measurement_end	
display io_port[1]	performance_measurement_init	

1 This option is not available in real-time mode.
2 This is only available when simulated I/O is defined.
3 These commands are not displayed on softkeys.
4 This option is not available in real-time mode if addresses are in user memory.

A Syntax for all Emulators

This syntax chapter contains information that is applicable to all HP 64700-Series emulators. In certain cases, you may want to refer to your *Emulator Softkey Interface User's Guide* for details about your emulator.

Function Codes

Function codes may be mentioned in some of the following syntax diagrams. When you see a reference to function codes, you should refer to your *Emulator Softkey Interface User's Guide,* or your *Emulator Terminal Interface User's Guide* to determine whether your emulator supports function codes.

Note

Not all HP 64700-Series emulators support the use of function codes.

break

This command causes the emulator to leave user program execution and begin executing in the monitor.

Syntax

Function

The behavior of **`break`** depends on the state of the emulator:

running	Break diverts the processor from execution of your program to the emulation monitor.
reset	Break releases the processor from reset, and diverts execution to the monitor.
running in monitor	The **`break`** command does not perform any operation while the emulator is executing in the monitor.

Default Value

none

Parameters

none

Example

```
break <RETURN>
```

Related Commands

```
help break
run
step
```

Notes

cmb_execute

This command causes an EXECUTE signal to be put on the Coordinated Measurement Bus (CMB), and starts a trace measurement on receipt of a CMB EXECUTE signal.

Syntax

cmb_execute
<RETURN>

Function

The **cmb_execute** command causes the emulator to emit an EXECUTE pulse on its rear panel CMB connector. All emulators connected to the CMB (including the one sending the CMB EXECUTE pulse) and configured to respond to this signal will take part in the measurement.

Default Value

none

Parameters

none

Example

```
cmb_execute  <RETURN>
```

Related Commands

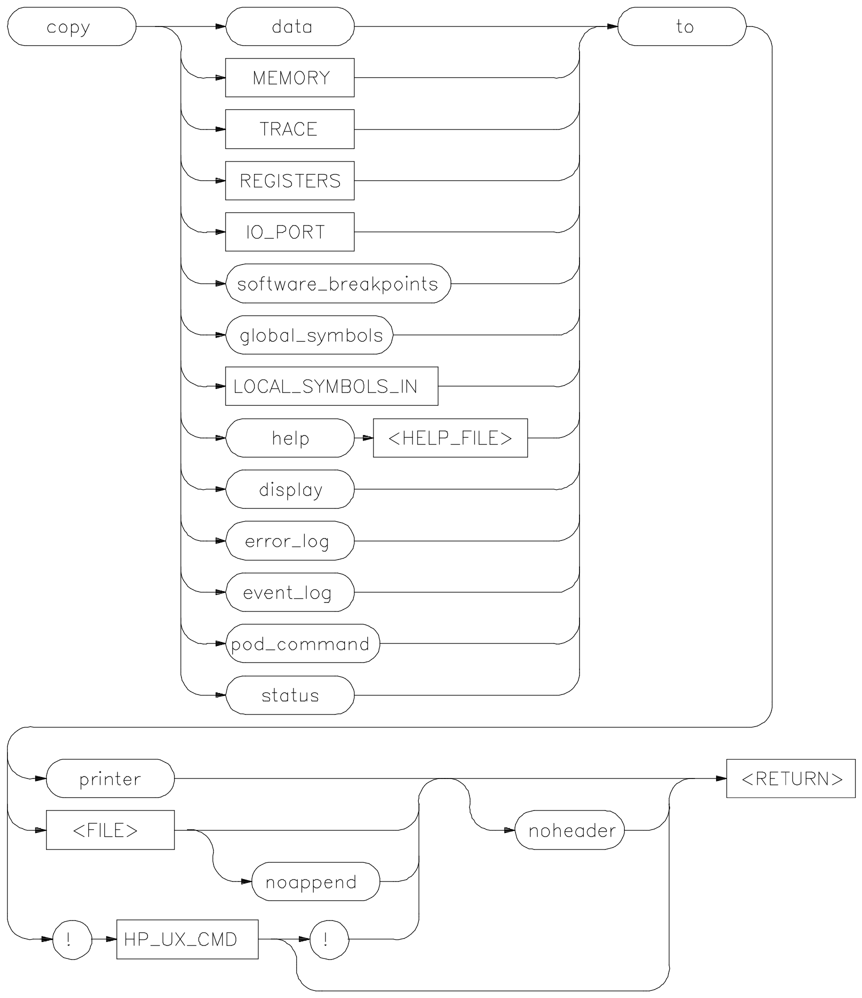
```
help cmb
help cmb_execute
help specify
specify run
specify trace
```

Notes

copy

Use this command with various parameters to save or print emulation and analysis information.

Syntax



Note

The **copy io_port** command is not used by all HP 64700 emulators because some do not have I/O ports. Refer to your *Emulator Softkey Interface User's Guide* for details.

Function

The **copy** command copies selected information to your system printer or listing file, or directs it to an HP-UX process.

Default Values

Depending on the information you choose to copy, default values may be options selected for the previous execution of the **display** command. For example, if you display memory locations 10h through 20h, then issue a **copy memory to myfile** command, myfile will list only memory locations 10h through 20h.

Parameters

data	This allows you to copy a list of memory contents formatted in various data types (see display data).
display	This allows you to copy the display to a selected destination.
error_log	This allows you to copy the most recent errors that occurred.
event_log	This allows you to copy the most recent events that occurred.
< FILE>	This prompts you for the name of a file where you want the specified information to be copied. If you want to specify a file name that begins with a number, you must precede the file name with a backslash. For example: **copy display to \12.10 < RETURN>**
global _symbols	This lets you copy a list of global symbols to the selected destination.

help	This allows you to copy the contents of the emulation help files to the selected destination.
< HELP _FILE>	This represents the name of the help file to be copied. Available help file names are displayed on the softkey labels.
HP-UX CMD	This represents an HP-UX filter or pipe where you want to route the output of the **copy** command. HP-UX commands must be preceded by an exclamation point (!). An exclamation point following the HP-UX command continues Softkey Interface command line execution after the HP-UX command executes. Emulation is not affected when using an HP-UX command that is a shell intrinsic.
io_port	This lets you copy a list of the I/O port contents to the selected destination. Not all HP 64700-Series emulators have I/O ports. Refer to your *Emulator Softkey Interface User's Guide* for details.
local _symbols_in	This lets you copy all the children of a given symbol to the selected destination. See the **--SYMB--** syntax page and the *HP 64000-UX User's Guide* for information on symbol hierarchy.
memory	This allows you to copy a list of the contents of memory to the selected destination.
noappend	This causes any copied information to overwrite an existing file with the same name specified by < FILE> . If this option is not selected, the default operation is to append the copied information to the end of an

	existing file with the same name that you specify.
noheader	This copies the information into a file without headings.
pod_command	This allows you to copy the most recent commands sent to the HP 64700-Series emulator/analyzer.

printer	This option specifies your system printer as the destination device for the **copy** command. Before you can specify the printer as the destination device, you must define PRINTER as a shell variable. For example, you could enter the text shown below after the “$” symbol:
	
	`$  PRINTER= lp`
	`$  export PRINTER`
	
	If you don’t want the print message to overwrite the command line, execute:
	
	`$ set PRINTER =  "lp -s"`

registers	This allows you to copy a list of the contents of the emulation processor registers to the selected destination.
software _breakpoints	This option lets you copy a list of the current software breakpoints to a selected destination.
status	This allows you to copy emulation and analysis status information.
to	This allows you to specify a destination for the copied information.

trace	This lets you copy the current trace listing to the selected destination.
!	An exclamation point specifies the delimiter for HP-UX commands. An exclamation point must precede all HP-UX commands. A trailing exclamation point should be used if you want to return to the command line and specify noheader. Otherwise, the trailing exclamation point is optional. If an exclamation point is part of the HP-UX command, a backslash (\) must precede the exclamation point.

Note If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

Examples

See the following pages on various **copy** syntax diagrams.

Related Commands

```
help copy
```

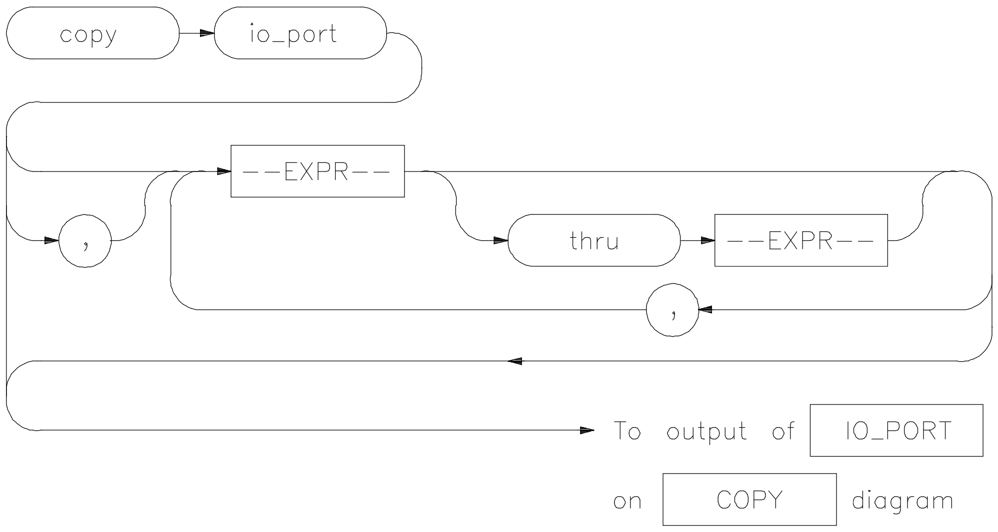
See the following pages on various **copy** syntax diagrams.

Notes

copy io_port

This command copies the current values at the emulator I/O ports to the selected destination.

Syntax



Function

This command can be executed only when the emulator is running in the monitor or running a user program.

Note

Some HP 64700-Series emulators do not have I/O ports. Refer to your *Emulator Softkey Interface User's Guide* for information about whether your emulator has I/O ports.

Default Values

Initial values are the same as those specified by the command **display io_port 0 absolute bytes**. Defaults are to values specified in the previous **display io_port** command.

Parameters

--EXPR--	This is a combination of numeric values, symbols, operators, and parentheses, specifying I/O port addresses. See the EXPR syntax diagram.
thru	This allows you to specify a range of I/O port locations to be copied.
,	A comma used immediately after **io_port** in the command line appends the current **copy io_port** command to the preceding **display io_port** command. The data specified in both commands is copied to the destination selected in the current command. Formatting is specified by the current command. The comma is also used as a delimiter between I/O port address values.

Examples

```
copy  io_port  1h , 45h , 60h thru 80h ,
0FFH  to  printer  <RETURN>
```

```
copy  io_port , CLEAR  thru  OUTPUT  to
iofile  <RETURN>
```

Related Commands

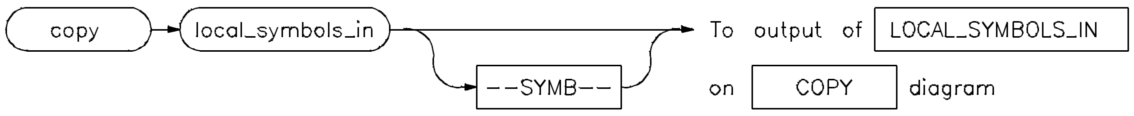
display io_port

help ***copy***

copy local_symbols_in

This command lets you copy local symbols contained in a source file and relative segments (program, data, or common) to the selected destination.

Syntax



Function

Local symbols are symbols that are children of the particular file or symbol defined by **--SYMB--**, that is, they are defined in that file or scope.

For additional information on symbols, refer to the **--SYMB--** syntax pages and the *HP 64000-UX System User's Guide.*

Default Value

--SYMB-- is the current working symbol.

Parameters

--SYMB--	This option represents the symbol whose children are to be listed. See the **--SYMB--** syntax diagram and the *HP 64000-UX System User's Guide* for information on symbol hierarchy.

Examples

```
copy  local_symbols_in  prog68k.S:  to
printer  <RETURN>
copy  local_symbols_in  cmd_rdr.s:  to
myfile  <RETURN>
```

The result may resemble:

```
Symbols in cmd_rdr.s:
Static symbols
Symbol name ____________________ Address range __ Segment _____________ Offset
Again                                000450       PROG                    0050
Cmd_A                                000428       PROG                    0028
Cmd_B                                000434       PROG                    0034
Cmd_I                                000440       PROG                    0040
Cmd_Input                            000600       DATA                    0000
End_Msgs                             000534       COMM                    0034
Exe_Cmd                              000416       PROG                    0016
Fill_Dest                            000456       PROG                    0056
Init                                 000400       PROG                    0000
Msg_A                                000500       COMM                    0000
Msg_B                                000512       COMM                    0012
Msg_Dest                             000602       DATA                    0002
Msg_I                                000524       COMM                    0024
Msgs                                 000500       COMM                    0000
Read_Cmd                             000406       PROG                    0006
Scan                                 00040E       PROG                    000E
Stack                                0006FA       DATA                    00FA
Write_Msg                            00044A       PROG                    004A
```

Related Commands

display local_symbols_in <SYMB>

`help` ***copy***

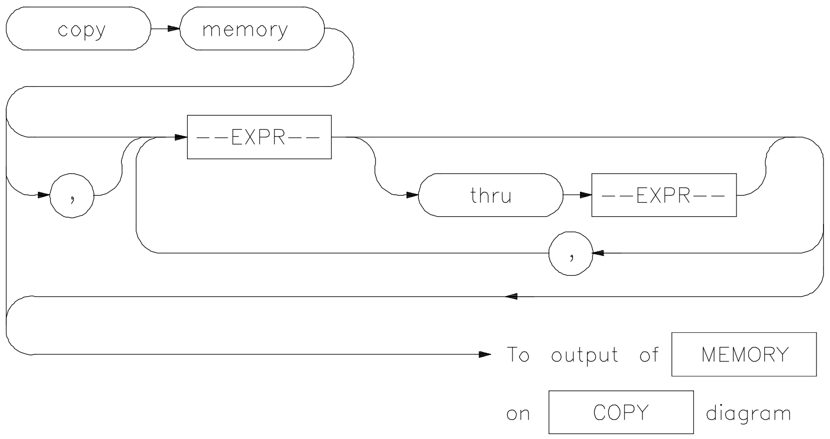
`cws`

`pws`

copy memory

This command copies the contents of a memory location or series of locations to the specified output.

Syntax



Function

The memory contents are copied in the same format as specified in the last display memory command.

Contents of memory can be displayed if program runs are not restricted to real-time. Memory contents are listed as an asterisk (*) under the following conditions:

1. The address refers to guarded memory.

2. Runs are restricted to real-time, the emulator is running a user program, and the address is located in user memory.

Values in emulation memory can always be displayed.

Default Values

Initial values are the same as those specified by the command **display memory 0 blocked bytes offset_by 0**.

Defaults are to values specified in the previous **display memory** command.

Parameters

--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying a memory address or offset value. See the EXPR syntax diagram.
,	A comma used immediately after **memory** in the command line appends the current **copy memory** command to the preceding **display memory** command. The data specified in both commands is copied to the destination specified in the current command. Data is formatted as specified in the current command. The comma is also used as a delimiter between values when specifying multiple memory addresses.

Note

If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide.*

Examples

```
copy  memory  START to  printer <RETURN>
copy  memory  0  thru  100H , START  thru
+5 , 500H , TARGET2 to  memlist  <RETURN>
```

```
copy memory 2000h thru 204fh to
memlist <RETURN>
```

The result of the last command could resemble:

```
Memory  :bytes :blocked
address    data       :hex                                  :ascii
---------------------------------------------------------------------------
002000-07    24   79   00   00   10   00   26   79      $ y . . . . & y
002008-0F    00   00   10   04   14   BC   00   00      . . . . . . . .
002010-17    10   12   0C   00   00   00   67   F8      . . . . . . g .
002018-1F    0C   00   00   41   67   00   00   0E      . . . A  g . . .
002020-27    0C   00   00   43   67   00   00   14      . . . C  g . . .
002028-2F    60   00   00   1E   10   3C   00   11      ‘ . . . . . .
002030-37    20   7C   00   00   10   08   60   00        | . . . . ‘ .
002038-3F    00   1A   10   3C   00   11   20   7C      . . .  . .  |
002040-47    00   00   10   19   60   00   00   0C      . . . . ‘ . . .
002048-4F    10   3C   00   0F   20   7C   00   00      . . .   | . .
```

Related Commands

display memory

help ***copy***

modify memory

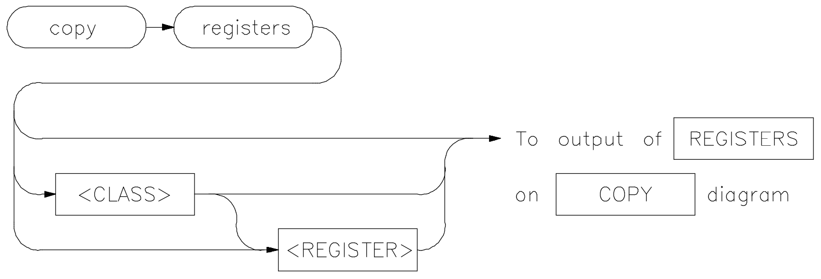
store memory

Notes

copy registers

This command copies the contents of the processor program counter and registers to a file or printer.

Syntax



Function

The **copy register** process does not occur in real-time. The emulation system must be configured for nonreal-time operation to list the registers while the processor is running.

Note

Refer to your *Emulator Softkey Interface User's Guide* for details about your emulator registers.

Default Values

With no options specified, the basic register class is displayed. This will differ for each emulator type.

Parameters

< CLASS>	Specifies a particular class of the emulator registers.
< REGISTER>	Specifies the name of an individual register.

Examples

copy registers BASIC ***to printer*** <RETURN>

copy registers to reglist <RETURN>

The results of the last command could resemble:

```
Registers
 ______________________________________________________________________________
Next_PC 002012@sp
PC 00002012       STATUS 2704   z          USP 00000000      SSP 00005000
D0-D7  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-A7  00000000 00000000 00003000 00004000 00000000 00000000 00000000 00005000
```

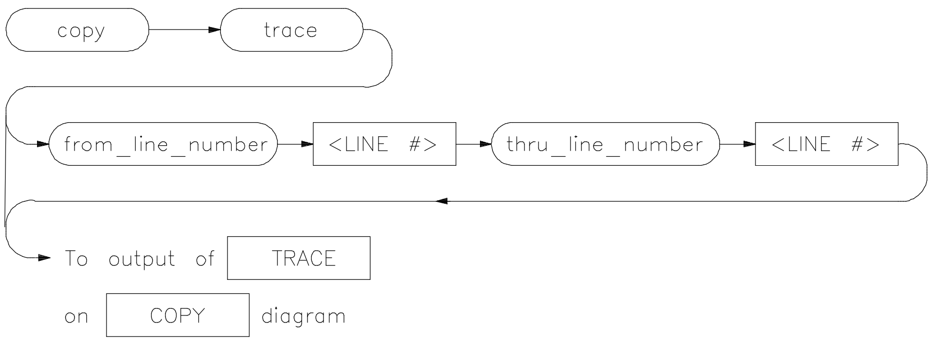
Related Commands

display registers
help ***copy***
help ***registers***
modify registers

copy trace

This command copies the contents of the trace buffer to a file or to the printer.

Syntax



Function

Trace information is copied in the same format as specified in the last display trace command.

Default Values

Initial values are the same as specified by the last **display trace** command.

Parameters

from_line _number	This specifies the trace list line number from which copying will begin.
< LINE# >	Use this with **from_line_number** and **thru_line_number** to specify the starting and ending trace list lines to be copied.
thru_line_number	Specifies the last line number of the trace list to include in the copied range.

Examples

```
copy  trace  to  tlist  <RETURN>
copy  trace from_line_number  0
thru_line_number  5
to  longtrac  <RETURN>
```

Related Commands

display trace

help ***copy***

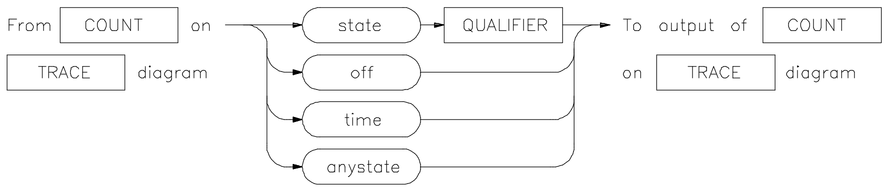
help ***trace***

store trace

COUNT

Allows you to select whether the emulation analyzer counts time, states, or nothing during a trace measurement.

Syntax


From COUNT on TRACE diagram
state
QUALIFIER
off
time
anystate
To output of COUNT on TRACE diagram

Function

A state is a unique combination of address, data, and status values occurring on the emulation bus simultaneously.

Default Value

The analyzer will count time by default.

Parameters

anystate	This option allows you to set up the **counting** parameter for the analyzer to count on any state.
off	This option turns off trace counting capability. This provides a larger trace depth. See the note below.
QUALIFIER	This is defined by you and used with the **state** option to define the states to be captured by the analyzer. External labels are described in the STATE definition. See the QUALIFIER and STATE syntax diagrams for details.

state	This causes the emulation analyzer to count occurrences of the specified state during a trace measurement.
time	This option causes the emulation analyzer to count the time between states acquired in the trace measurement.

Note

When **counting** is specified, the analyzer will capture 512 states. It will capture 1024 states when **counting** is off.

By default, the number of states displayed is 256. You can display all the captured states by increasing the trace display depth. For example, you would execute:

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

Examples

```
trace after START counting state LOOP2
<RETURN>
trace counting time <RETURN>
```

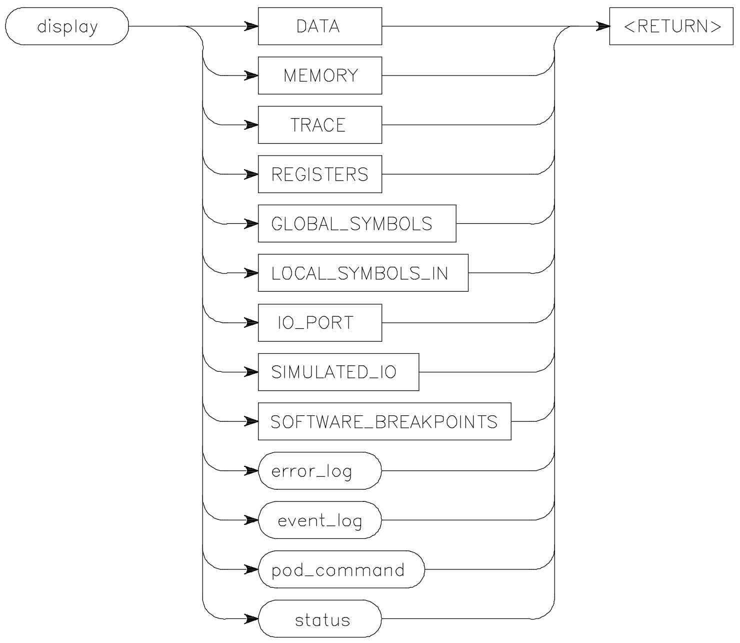
Related Commands

```
help trace
trace
```

display

This command displays selected information on your screen.

Syntax



Function

You can use the **up arrow**, **down arrow**, **PREV PAGE**, and **NEXT PAGE** keys to view the displayed information. For software_breakpoints, data, memory, and trace displays you can use the **CTRL g** and **CTRL f** keys to scroll left and right if the information goes past the edge of the screen.

Default Values

Depending on the information you select, defaults may be the options selected for the previous execution of the **display** command.

Parameters

data	This allows you to display a list of memory contents formatted in various data types (see the **display data** pages for details).
error_log	This option displays the recorded list of error messages that occurred during the emulation session.
event_log	This option displays the recorded list of events.
global _symbols	This option lets you display a list of all global symbols in memory.
io_port	This option allows you to display the contents of emulator I/O port locations.

Note

Not all HP 64700-Series emulators have I/O ports. Refer to the *Emulator Softkey Interface User's Guide* for details.

local _symbols_in	This option lets you display all the children of a given symbol. See the **--SYMB--** syntax page and the *HP 64000-UX System User's Guide* for details on symbol hierarchy.
memory	This option allows you to display the contents of memory.
pod_command	This option lets you display the output of previously executed emulator pod commands.
registers	This allows you to display the contents of emulation processor registers.
simulated_io	This lets you display data written to the simulated I/O display buffer after you have

	enabled polling for simulated I/O in the emulation configuration.
software _breakpoints	This option lets you display the current list of software breakpoints.
status	This displays the emulator and trace status.
trace	This displays the current trace list.

Examples

display event_log <RETURN>

The result of this command may resemble:

```
Event Log
Time     Type           Message
 ____________________________________________________________________________________
11:17:36 SYSTEM  cd: /users/yourself
11:17:40 OTHER   Loaded configuration file: /usr/hp64000/inst/emul/tmp/.C000002.pod
/default
11:17:40 PROC    M68000--Running in monitor
12:12:23 OTHER   Loaded configuration file: /users/yourself/newconfig
12:23:23 OTHER   Loaded configuration file: /usr/hp64000/inst/emul/tmp/.C000002.pod
/config
```

display local_symbols_in cmd_rdr.s: <RETURN>

The result of this command may resemble:

```
Symbols in cmd_rdr.s:
Static symbols
Symbol name ____________________ Address range __ Segment _____________ Offset
Again                                000450       PROG                    0050
Cmd_A                                000428       PROG                    0028
Cmd_B                                000434       PROG                    0034
Cmd_I                                000440       PROG                    0040
Cmd_Input                            000600       DATA                    0000
End_Msgs                             000534       COMM                    0034
Exe_Cmd                              000416       PROG                    0016
Fill_Dest                            000456       PROG                    0056
Init                                 000400       PROG                    0000
Msg_A                                000500       COMM                    0000
Msg_B                                000512       COMM                    0012
Msg_Dest                             000602       DATA                    0002
Msg_I                                000524       COMM                    0024
Msgs                                 000500       COMM                    0000
Read_Cmd                             000406       PROG                    0006
Scan                                 00040E       PROG                    000E
Stack                                0006FA       DATA                    00FA
Write_Msg                            00044A       PROG                    004A
```

Related Commands

help ***display***

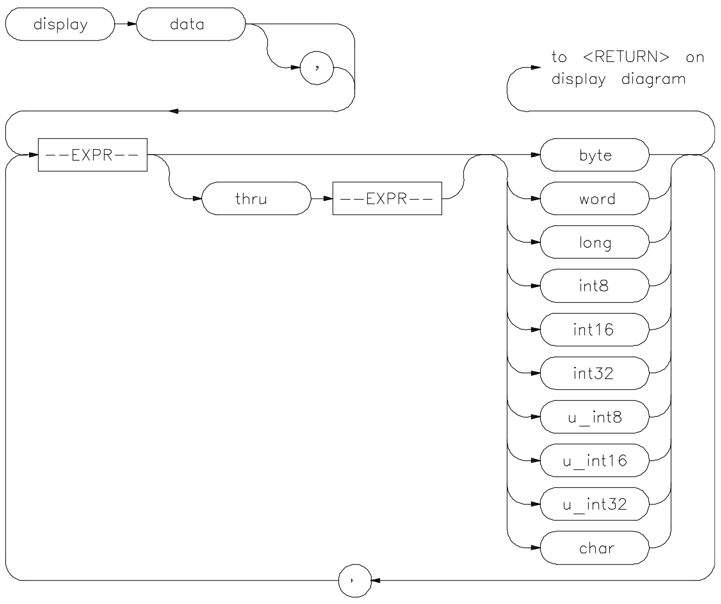
copy

The following pages describe various **display** syntax diagrams.

display data

Displays the values of variables of simple data types from your program.

Syntax



Function

display data can display the values of simple data types in your program. Using this command can save you time—you would otherwise need to search through memory displays for the location and value of a particular variable.

The address, identifier, and data value of each symbol may be displayed. You must issue the command **set symbols on** to see the symbol names displayed.

Default Value

For the first display data command after you begin an emulation session, you must supply at least one expression specifying the data item(s) to display.

Thereafter, the display data command defaults to the expressions specified in the last display data command, unless new expressions are supplied or appended (with a leading comma).

Symbols are normally set off until you give the command **set symbols on**. Otherwise, only the address, data type, and value of the data item will be displayed.

Parameters

,	A leading comma allows you to append additional expressions to the previous display data command. Commas between expression/data type specifications allow you to specify multiple variables and types for display with the current command.
--EXPR--	Prompts you for an expression specifying the data item to display. The expression can include various math operators and program symbols. See the --EXPR-- and --SYMB-- syntax pages for more information.

Note

If your processor supports function codes, you can specify additional --EXPR-- qualifiers using the function code information. Refer to the *Emulator Softkey Interface User's Guide* for more information.

thru --EXPR--	Allows you to specify a range of addresses for which you want data display. Typically, you use this to display the contents of an array. You can display both single-dimensioned and multi-dimensioned arrays. Arrays are displayed in the order specified by the language definition, typically

	row major order for most Algol-like languages.
< TYPE>	Specifies the format in which to display the information. (Data type information is not available from the symbol database, so you must specify.)
byte	Hex display of one 8 bit location.
word	Hex display of one 16 bit location.
long	Hex display of one 32 bit location.

Note Byte ordering in word and long displays is determined by the conventions of the processor in use.

int8	Display of one 8 bit location as a signed integer using two's complement notation.
int16	Display of two bytes as a signed integer using two's complement notation.
int32	Display of four bytes as a signed integer using two's complement notation.
u_int8	Display of one byte as an unsigned positive integer.
u_int16	Display of two bytes as an unsigned positive integer.
u_int32	Display of four bytes as an unsigned positive integer.
char	Displays one byte as an ASCII character in the range 0..127. Control characters and

values in the range 128..255 are displayed as a period (.).

Examples

display data `Msg_A` ***thru*** `+17` ***char***`, Stack` ***long***

```
 Data
   address  type        data
--------------------------------------------------------------------------------
    000500  char[]      Command A entered
    0006FA  long           48790000







STATUS:   M68000--Running in monitor____________________________________........
```

set symbols on

set width label 30

display data , Msg_B ***thru*** +17 ***char***, Msg_Dest ***thru*** +17 ***char***

```
 Data  :noupdate
   address  label                           type        data
    000500           COMM|Msgs              char[]      Command A entered
    0006FA  DATA|davek/68k/cmd_rdr.s:Stack  long           48790000
    000512  COMM|davek/68k/cmd_rdr.s:Msg_B  char[]      Entered B command
    000602          DATA|Msg_Dest           char[]      p...@.(5|...,Hx..







STATUS:   M68000--Running in monitor____________________________________........
```

Related Commands

copy data

help ***copy***

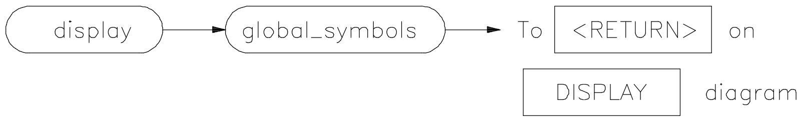
set

Notes

display global_symbols

This command displays the global symbols defined for the current absolute file.

Syntax



Function

Global symbols are symbols declared as global in the source file. They include procedure names, variables, constants, and file names. When the **display global_symbols** command is used, the listing will include the symbol name and its logical address.

Default Value

none

Examples

display global_symbols <RETURN>

Related Commands

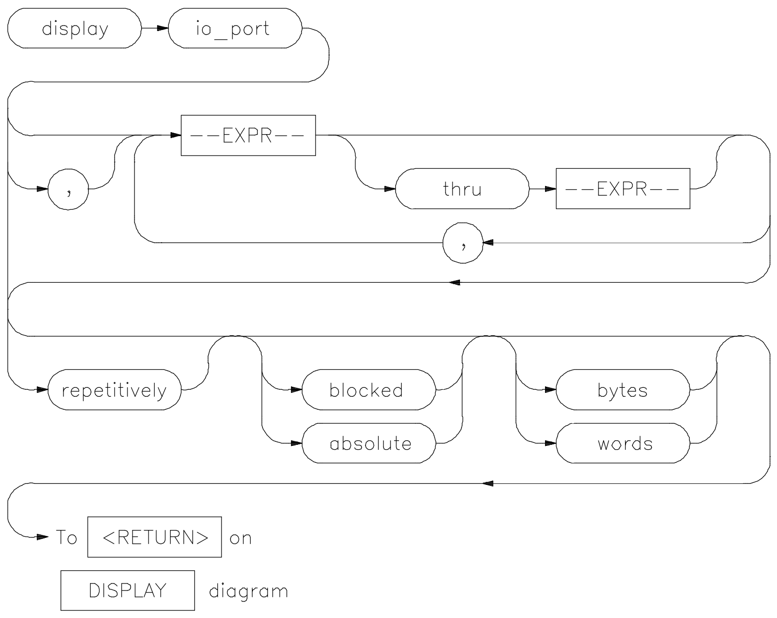
copy global_symbols
help ***display***

Notes

display io_port

This command displays the current values at the I/O ports.

Syntax



Function

I/O port values can be displayed in formats you define. The values at the I/O ports also can be displayed repetitively.

Note

Some HP 64700-Series emulators do not have I/O ports. Refer to your *Emulator Softkey Interface User's Guide* for more information about your emulator.

Default Values

The address list defaults to any previously specified list or to 0 if no value is specified. The format of the list is the previously specified format or in absolute bytes.

Parameters

absolute	This formats the list of I/O ports in a single column.
blocked	This formats the list of I/O ports in multiple columns.
bytes	This displays the absolute or blocked I/O ports listing as byte values.
--EXPR--	This is a combination of numeric values, symbols, operators, and parentheses, specifying an I/O port address. See the EXPR syntax diagram.
repetitively	Continuously updates the I/O port display listing.
thru	This allows you to specify a range of I/O ports to be displayed. Only 16 lines of information can be displayed on the screen at a time. To display additional lines, use the **up arrow**, **down arrow**, **NEXT PAGE**, or **PREV PAGE** keys.
words	Displays the I/O ports listing as word values.
,	A comma immediately after **io_port** in the command line appends the current **display io_port** command to the preceding **display io_port** command. The data specified in both commands is displayed. The data is formatted as specified in the current command. The comma also is a delimiter between values when specifying multiple I/O port addresses.

Examples

display io_port 1h , 45h , 60h ***thru*** 80h, 0FFH ***blocked words*** <RETURN>

display io_port 1 , 45 , 60 ***thru*** 80 , 0FFH ***absolute bytes*** <RETURN>

display io_port START ***thru*** READ_INPUT <RETURN>

Related Commands

copy io_port

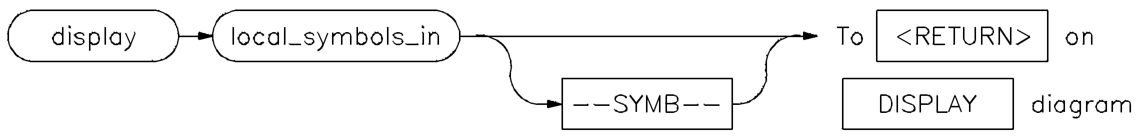
help ***display***

Notes

display local_symbols_in

Displays the local symbols in a specified source file and their relative segment (program, data, or common).

Syntax



Function

Local symbols of **--SYMB--** are the ones which are children of the file and/or scope specified by **--SYMB--**. That is, they are defined in that file or scope.

See the **--SYMB--** syntax pages and the *HP 64000-UX System User's Guide* for further explanation of symbols.

Displaying the local symbols sets the current working symbol to the one specified.

Default Value

--SYMB-- is the current working symbol.

Parameters

--SYMB--	This option represents the symbol whose children are to be listed. See the **--SYMB--** syntax diagram and the *HP 64000-UX System User's Guide* for more information on symbol hierarchy and representation.

Examples

display ***local_symbols_in*** `temp1.S:` `<RETURN>`

display ***local_symbols_in*** `prog68k.S:main`
<RETURN>

Related Commands

copy ***local_symbols_in*** `<FILE>`

`help` ***display***

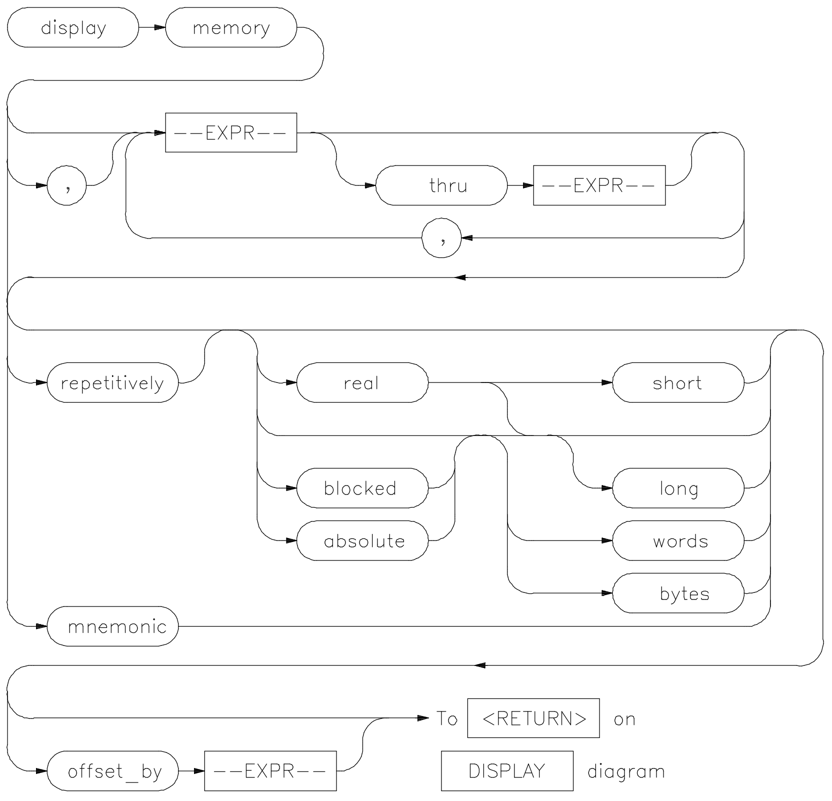
`cws`

`pws`

display memory

This command displays the contents of the specified memory location or series of locations.

Syntax



Function

The memory contents can be displayed in mnemonic, hexadecimal, or real number format. In addition, the memory addresses can be listed offset by a value, which allows the information to be easily compared to the program listing.

When displaying memory mnemonic and stepping, the next instruction that will step is highlighted. The memory mnemonic display autopages to the new address if the next PC goes outside the currently displayed address range. This feature works even if stepping is performed in a different emulation window than the one displaying memory mnemonic (see chapter on windowing capabilities).

Pending software breakpoints are shown in the memory mnemonic display by an asterisk (*) in the leftmost column of the assembly instruction or source line that has a pending breakpoint.

A label column (symbols) may be displayed for all memory displays except blocked mode. Memory mnemonic may be displayed with source and assembly code intermixed, or with source code only. Symbols also can be displayed in the memory mnemonic string. (See the set command.)

Note

If your emulator uses function codes, refer to the *Emulator User's Guide* for details.

Default Values

Initial values are the same as specified by the command:

```
display memory 0 blocked bytes offset_by 0
```

Defaults are values specified in a previous **display memory** command.

The symbols and source defaults are:

```
set source off symbols off
```

Parameters

absolute	Formats the memory listing in a single column.
blocked	Formats the memory listing in multiple columns.
bytes	Displays the absolute or blocked memory listing as byte values.

--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying a memory address or memory offset value. See the EXPR syntax diagram.
long	Displays the memory listing as long word values. When used with the **real** parameter, **long** displays memory in a 64-bit real number format.
mnemonic	This causes the memory listing to be formatted in assembly language instruction mnemonics with associated operands. When specifying mnemonic format, you should include a starting address that corresponds to the first byte of an operand to ensure that the listed mnemonics are correct. If **set source only** is on, you will see only the high level language statements and corresponding line numbers.
offset_by	This option lets you specify an offset that is subtracted from each of the absolute addresses before the addresses and corresponding memory contents are listed. You might select the offset value so that each module appears to start at address 0000H. The memory contents listing will then appear similar to the assembler or compiler listing. This option is also useful for displaying symbols and source lines in dynamically relocated programs.
real	Formats memory values in the listing as real numbers. (NaN in the display list means “Not a Number.”)

repetitively	Updates the memory listing display continuously. You should only use this to monitor memory while running user code, since it is very CPU intensive. To allow updates to the current memory display whenever memory is modified, a file is loaded, software breakpoint is set, etc., use the **set update** command.
short	Formats the memory list as 32-bit real numbers.
thru	This option lets you specify a range of memory locations to be displayed. Use the **up arrow**, **down arrow**, **NEXT PAGE**, and **PREV PAGE** keys to view additional memory locations.
words	Displays the memory listing as word values.
,	A comma after **memory** in the command line appends the current **display memory** command to the preceding **display memory** command. The data specified in both commands is displayed. The data will be formatted as specified in the current command. The comma is also a delimiter between values when specifying multiple addresses.

Examples

display memory `2000h` ***thru*** `204fh` ***blocked words*** `<RETURN>`

The result of this command may resemble:

```
Memory  :words :blocked
address    data       :hex                                 :ascii
_________________________________________________________________________________
002000-0E    2479 0000 1000 2679 0000 1004 14BC 0000   $y....&y  ........
002010-1E    1012 0C00 0000 67F8 0C00 0041 6700 000E   ......g. ...Ag...
002020-2E    0C00 0043 6700 0014 6000 001E 103C 0011   ...Cg... ‘......
002030-3E    207C 0000 1008 6000 001A 103C 0011 207C    |....‘. ..... |
002040-4E    0000 1019 6000 000C 103C 000F 207C 0000   ....‘... ... |..
```

display memory `2000h` ***thru*** `202fh , 2100h` ***real long*** `<RETURN>`

The result of this command may resemble:

```
Memory  :long real
address      data  :real
_____________________________________________________
002000      5.50328431726029E-133
002008      0.00000000000000E+000
002010      2.90606516754831E-231
002018      6.98394306836813E-251
002020      6.98395638835160E-251
002028      2.68163846825574E+154
002100     -5.49484035779135E+152
```

display memory 400h ***mnemonic***

set symbols on

```
 Memory  :mnemonic :file = cmd_rdr.s:
   address  label                data
    000400       PROG|Init        2E7C000006  MOVEA.L #0000006FA,A7
    000406  P|cmd_rdr.s:Read_Cmd  13FC000000  MOVE.B  #000,DATA|Cmd_Input
    00040E  PROG|/cmd_rdr.s:Scan  1039000006  MOVE.B  DATA|Cmd_Input,D0
    000414                        67F8        BEQ.B   P|cmd_rdr.s:Scan
    000416  PR|cmd_rdr.s:Exe_Cmd  0C000041    CMPI.B  #041,D0
    00041A                        6700000C    BEQ.W   |cmd_rdr.s:Cmd_A
    00041E                        0C000042    CMPI.B  #042,D0
    000422                        67000010    BEQ.W   |cmd_rdr.s:Cmd_B
    000426                        6018        BRA.B   |cmd_rdr.s:Cmd_I
    000428  PROG|cmd_rdr.s:Cmd_A  323C0011    MOVE.W  #00011,D1
    00042C                        207C000005  MOVEA.L #000000500,A0
    000432                        6016        BRA.B   cmd_rd:Write_Msg
    000434  PROG|cmd_rdr.s:Cmd_B  323C0011    MOVE.W  #00011,D1
    000438                        207C000005  MOVEA.L #000000512,A0
    00043E                        600A        BRA.B   cmd_rd:Write_Msg
    000440  PROG|cmd_rdr.s:Cmd_I  323C000F    MOVE.W  #0000F,D1

STATUS:   M68000--Running in monitor____________________________________........
```

set source on

display memory main mnemonic

```
 Memory  :mnemonic :file = main.c:
   address  label          data
      99    extern void    update_state_of_system();
     100    extern void    get_operator_input();
     101
     102    main()
     103    {
    00106A   PROG|_main   4E560000    LINK    A6,#00000
    00106E                2F0A        MOVE.L  A2,-(A7)
    001070                247C000601  MOVEA.L #0000601AA,A2
     104         initialize_system();
    001076                4EB9000011  JSR     |_initialize_sys
     105
     106         while (system_is_running)
    00107C                600000B0    BRA.W   main.c:continue1
    001080                4E71        NOP
     107         {
     108              if (time_to_update_system)

STATUS:   M68000--Running in monitor____________________________________........
```

Related Commands

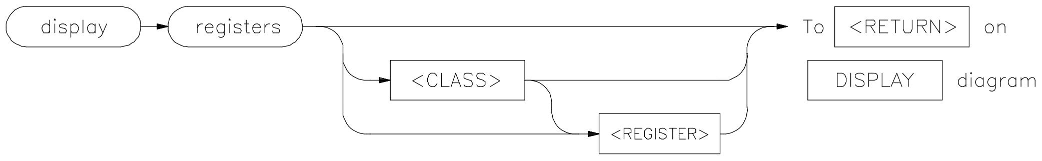
copy memory
`cws`
`help` ***display***
modify memory
`pws`
set
store memory

Notes

display registers

This command displays the current contents of the emulation processor program counter and registers.

Syntax



Function

If a **step** command just executed, the mnemonic representation of the last instruction is also displayed, if the current display is the register display. This process does not occur in real-time. The emulation system must be configured for nonreal-time operation to display registers while the processor is running. Symbols also may be displayed in the register step mnemonic string (see **set symbols**).

Default Values

With no options specified, the basic register class is displayed as the default. This differs for each emulator type.

Parameters

< CLASS>	This allows you to display a particular class of emulation processor registers. The classes differ for each emulator.
< REGISTER>	This displays an individual register.

Examples

display registers <RETURN>
display registers BASIC <RETURN>

Related Commands

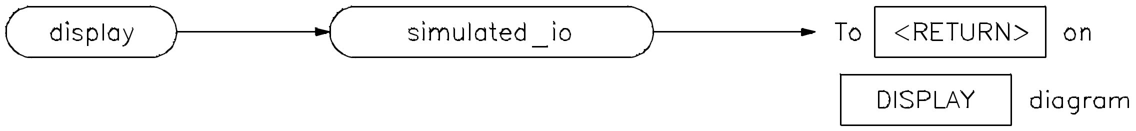
copy registers
help ***display***

help ***registers***
modify registers
set
step

display simulated_io

This command displays information written to the simulated I/O display buffer.

Syntax



Function

After you have enabled polling for simulated I/O during the emulation configuration process, six simulated I/O addresses can be defined. You then define files used for standard input, standard output, and standard error.

For details on setting up simulated I/O, refer to the question “Modify simulated I/O configuration?” in the *Emulation Configuration* chapter.

Default Value

none

Parameters

none

Example

display simulated_io <RETURN>

Related Commands

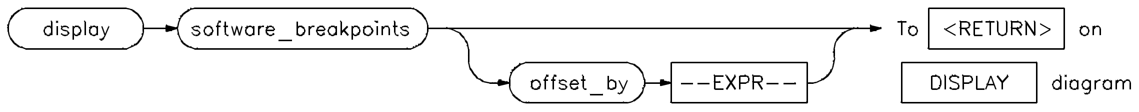
help ***display***

Notes

display software _breakpoints

This command displays the currently defined software breakpoints and their status.

Syntax



Function

If the emulation session is continued from a previous session, the listing will include any previously defined breakpoints. The column marked “status” shows whether the breakpoint is pending, inactivated, or unknown.

An “unknown” breakpoint status will occur if you set the breakpoint, then remap the breakpoint address as guarded. A pending breakpoint causes the processor to enter the emulation monitor or background memory upon execution of that breakpoint. Executed breakpoints are listed as inactivated. Entries that show an inactive status can be reactivated by executing the following command:

modify software_breakpoints set <RETURN>

A label column also may be displayed for addresses that correspond to a symbol. See the **set** command for details.

Default Value

none

Parameters

--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying an offset value for the breakpoint address. See the **--EXPR--** syntax diagram.

offset_by	This option allows you to offset the listed software breakpoint address value from the actual address of the breakpoint. By subtracting the offset value from the breakpoint address, the system can cause the listed address to match that given in the assembler or compiler listing.

Examples

display software_breakpoints `<RETURN>`

display software_breakpoints offset_by `1000H <RETURN>`

Related Commands

copy software_breakpoints

`help` ***display***

`help` ***software_breakpoints***

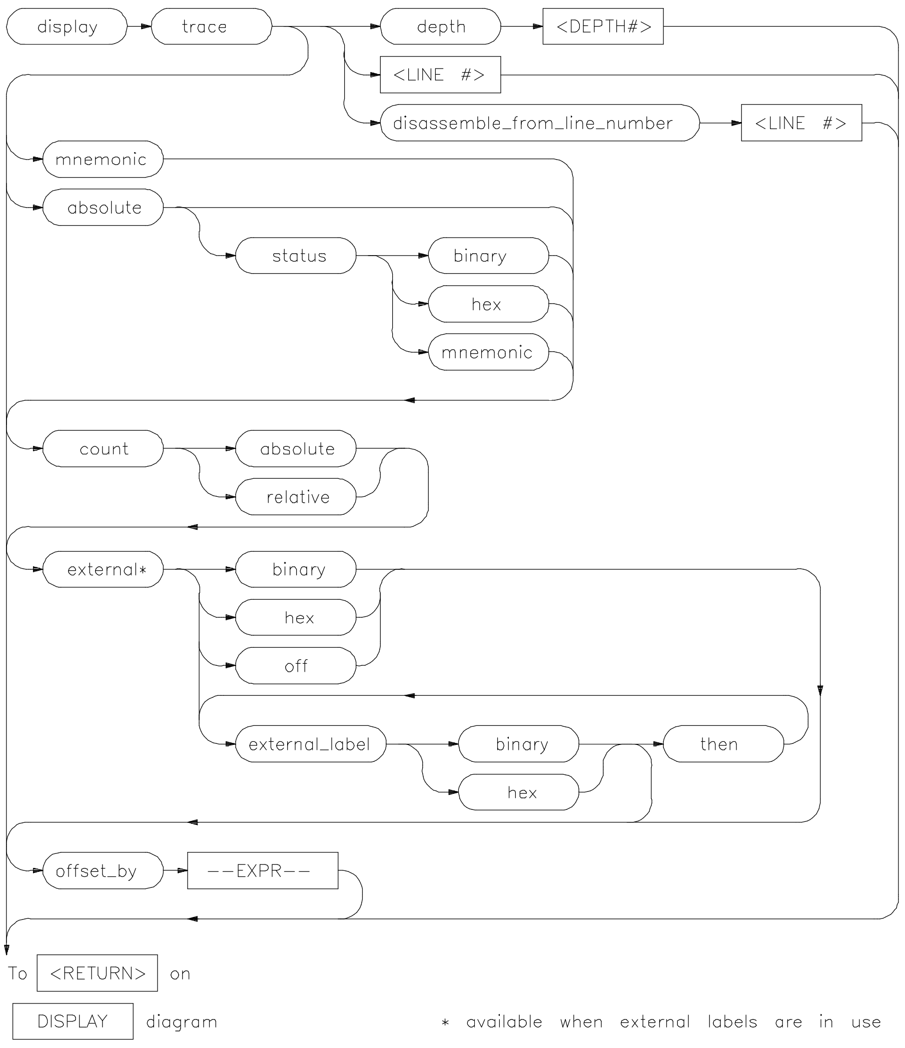
modify software_breakpoints

set

display trace

This command displays the contents of the trace buffer.

Syntax



Function Captured information can be presented as absolute hexadecimal values or in mnemonic form. The processor status values captured by the analyzer can be listed mnemonically or in hexadecimal or binary form.

Addresses captured by the analyzer are physical addresses.

The **offset_by** option subtracts the specified offset from the addresses of the executed instructions before listing the trace. With an appropriate entry for **offset**, each instruction in the listed trace will appear as it does in the assembled or compiled program listing.

The **count** parameter lists the current trace of time or state either relative to the previous event in the trace list or as an absolute count measured from the trigger event. If time counts are currently selected, the **count** parameter causes an absolute or relative time count to be listed. If the current trace contains state counts, a relative or absolute state count results.

The **source** parameter allows display of source program lines in the trace listing, enabling you to quickly correlate the trace list with your source program.

Default Values

Initial values are the same as specified by the command:

```
display trace mnemonic count relative
offset_by 0 <RETURN>
```

Parameters

absolute	Lists trace information in hexadecimal format, rather than mnemonic opcodes.
count	
absolute	This lists the state or time count for each event of the trace as the total count measured from the trigger event.
relative	This lists the state or time count for each event of the trace as the count measured relative to the previous event.

depth	
< DEPTH# >	This defines the number of states to be uploaded by the Softkey Interface.

Note

After you have changed the trace depth, execute the command **wait measurement_complete** before displaying the trace. Otherwise the new trace states will not be available.

disassemble _from_line _number	This causes the inverse assembly software to begin disassembling the trace code from the specified line number. This feature is required for processors where the inverse assembler cannot uniquely identify the first state of an instruction on the processor bus. The command is not available on emulators where the corresponding inverse assembler can identify instructions on the processor bus.
--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying an offset value to be subtracted from the addresses traced by the emulation analyzer. See the EXPR syntax diagram.
external	
binary	Displays the external analyzer trace list in binary format.
< external _label>	This option displays a defined external analyzer label.
hex	Displays the external analyzer trace list in hexadecimal format.

off	Use this option to turn off the external trace list display.
then	This allows you to display multiple external analysis labels. This option appears when more than one external analyzer label is in use.
< LINE# >	This prompts you for the trace list line number to be centered in the display. Also, you can use < LINE# > with **disassemble_from_line_number**. < LINE# > prompts you for the line number from which the inverse assembler attempts to disassemble data in the trace list.
mnemonic	Lists trace information with opcodes in mnemonic format.
offset_by	This option allows you to offset the listed address value from the address of the instruction. By subtracting the offset value from the physical address of the instruction, the system makes the listed address match that given in the assembler or compiler listing. This option is also useful for displaying symbols and source lines in dynamically relocated programs.

Note ☝

When using the **set source only** command, the analyzer may operate more slowly than when using the **set source on** command. This is an operating characteristic of the analyzer:

> When you use the command **set source on**, and are executing only assembly language code (not high-level language code), no source lines are displayed. The trace list will then fill immediately with the captured assembly language instructions.
>
> When using **set source only**, no inverse assembled code is displayed. Therefore, the emulation software will try to fill the display with high-level source code. This requires the emulation software to search for any captured analysis data generated by a high-level language statement.

In conclusion, you should not set the trace list to **set source only** when tracing assembly code. This will result in optimum analyzer performance.

status

binary	Lists absolute status information in binary form.
hex	Lists absolute status information in hexadecimal form.
mnemonic	Lists absolute status information in mnemonic form.

Examples

display trace count absolute <RETURN>

The result of this command may resemble:

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                Opcode or Status               time count
Base:     hex     hex                     mnemonic                    absolute
______  _______   ______  _________________________________________  ____________
after    004FFA    2700    2700  supr data rd word                   ------------
+001     004FFC    0000    0000  supr data rd word                   + 520     nS
+002     004FFE    2000    2000  supr data rd word                   +   1.0   uS
+003     002000    2479  MOVEA.L 0001000,A2                          +   1.5   uS
+004     002002    0000    0000  supr prog                           +   2.0   uS
+005     002004    1000    1000  supr prog                           +   2.5   uS
+006     002006    2679  MOVEA.L 0001004,A3                          +   3.0   uS
+007     001000    0000    0000  supr data rd word                   +   3.5   uS
+008     001002    3000    3000  supr data rd word                   +   4.00  uS
+009     002008    0000    0000  supr prog                           +   4.52  uS
+010     00200A    1004    1004  supr prog                           +   5.00  uS
+011     00200C    14BC  MOVE.B  #000,[A2]                           +   5.52  uS
+012     001004    0000    0000  supr data rd word                   +   6.00  uS
+013     001006    4000    4000  supr data rd word                   +   6.52  uS
+014     00200E    0000    0000  supr prog                           +   7.00  uS
```

display trace absolute status binary
<RETURN>

The result of this command may resemble:

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                 Absolute Status               time count
Base:     hex     hex                       binary                    absolute
______  _______   ______  _________________________________________  ____________
after    004FFA    2700  11101110                                    ------------
+001     004FFC    0000  11101110                                    + 520     nS
+002     004FFE    2000  11101110                                    +   1.0   uS
+003     002000    2479  10110110                                    +   1.5   uS
+004     002002    0000  10110110                                    +   2.0   uS
+005     002004    1000  10110110                                    +   2.5   uS
+006     002006    2679  10110110                                    +   3.0   uS
+007     001000    0000  11101110                                    +   3.5   uS
+008     001002    3000  11101110                                    +   4.00  uS
+009     002008    0000  10110110                                    +   4.52  uS
+010     00200A    1004  10110110                                    +   5.00  uS
+011     00200C    14BC  10110110                                    +   5.52  uS
+012     001004    0000  11101110                                    +   6.00  uS
+013     001006    4000  11101110                                    +   6.52  uS
+014     00200E    0000  10110110                                    +   7.00  uS
```

set source on

display trace mnemonic

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:      Address       Data        Opcode or Status w/ Source Lines      time
Base:       symbols       hex                mnemonic w/symbols              rel
+009   sysstack:+003FC2     0738    0738  supr data wr word                  520
+010   sysstack:+003FC0     0006    0006  supr data wr word                  480
+011   main:main+00000A     01AA    01AA  supr prog                          520
      ##########main.c - line   104 ##########################################
           initialize_system();
+012   main:main+00000C     4EB9  JSR     |_initialize_sys                   480
+013   main:main+00000E     0000    0000  supr prog                          520
+014   main:main+000010     114A    114A  supr prog                          480
      ##########initSystem.c - line     1 thru    38 ##########################
      void refresh_menu_window();

      void
      initialize_system()
      {
+015   |_initialize_sys     4E56  LINK    A6,#00000                          520

STATUS:   M68000--Running user program    Emulation trace complete______........
```

set source only

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:                          Source Lines Only                           time
Base:                                                                        rel
+012    ##main.c - line   104 ##########################################   480
           initialize_system();
+015    ##initSystem.c - line     1 thru    38 ##########################   520
      void refresh_menu_window();

      void
      initialize_system()
      {
+038    ##initSystem.c - line    39 thru    45 ##########################   480
              buffered. */
           /* setvbuf(stdin, NULL, _IONBF, 1); */

           /* Initialize system clock. */
           time = &time_struct;
+041    ##initSystem.c - line    46 ######################################   520

STATUS:   M68000--Running user program    Emulation trace complete______........
```

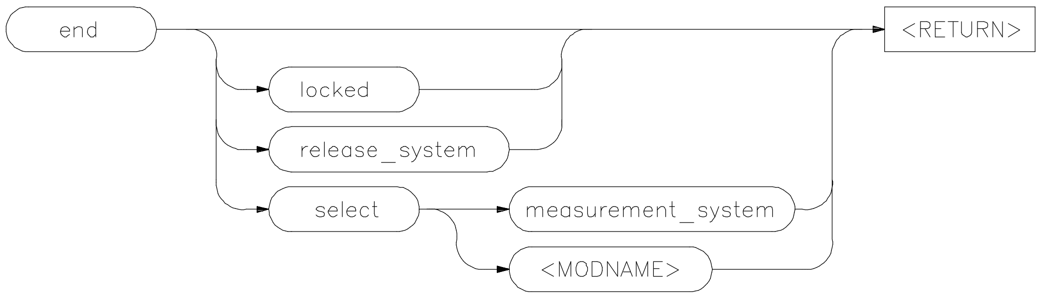
Related Commands

copy trace
help ***display***
help ***trace***
store trace
set

end

This command terminates the current emulation session.

Syntax



Function

You can end the emulation session and keep the emulator in a locked state. The current emulation configuration is stored, so that you can continue the emulation session on reentry to the emulator. If you choose, you can select another measurement system when ending the current session. You also can release the emulation system when ending the session so that others may use the emulator.

Note

Pressing **CNTL d** performs the same operation as pressing **end < RETURN>** . Pressing **CNTL** \ or **CNTL |** performs the same as **end release_system < RETURN>** .

Default Value

When the emulation session ends, control returns to the HP-UX shell without releasing the emulator.

Parameters

locked	This option allows you to stop all active instances of an emulator Softkey Interface session in one or more windows and/or

	terminals. This option is not available when operating the emulator in the measurement system.
measurement _system	This is used with the **select** option, and represents another emulation system in the HP 64000-UX measurement system. This option is only available when operating the HP 64700-Series emulator in the measurement system.
< MODNAME>	Choose this option with **select** to enter another module in the measurement system after ending the current one. < MODNAME> appears when other measurement system modules are defined in the HP 64000-UX measurement system. This option is only available when operating the HP 64700-Series emulator in the measurement system.
release _system	This option stops all instances of the Softkey Interface in one or more windows or terminals. The emulation system is released for other users. If you do not release the emulation system when ending, others cannot access it.
select	This option lets you choose another defined emulation measurement system when you end the current emulation session. One or more different measurement systems must be active for this option to appear.

Examples

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

Related Commands

`emul700 <emulator_name>`

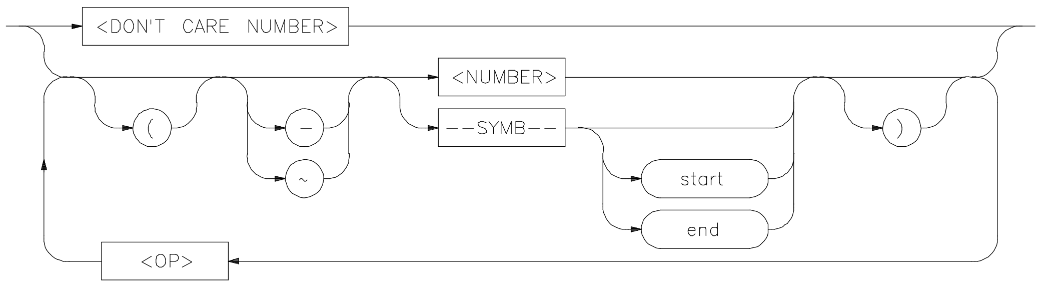
`help` ***end***

Notes

--EXPR--

An expression is a combination of numeric values, symbols, operators, and parentheses used to specify address, data, or status, or any other value used in the emulation commands.

Syntax



Function

The function of an expression (--EXPR--) is to let you define the address, data, or status expression that fits your needs. You can combine multiple values to define the expression.

Certain emulation commands will allow the option of < + EXPR> after pressing a thru softkey. This allows you to enter a range without retyping the original base address or symbol. For example, you could specify the address range

```
disp_buf thru disp_buf + 25
```

as

```
disp_buf thru +25
```

Default Value

none

Parameters

DON'T CARE NUMBER	You can include "don't care numbers" in expressions. These are indicated by a number containing an "x." These numbers may be defined as binary, octal, decimal, or

hexadecimal. For example: 1fxxh, 17x7o, and 011xxx10b are valid.

Note “Don’t care numbers” are not valid for all commands.

--NORMAL-- This appears as a softkey label to enable you to return to the **--EXPR--** key. The **--NORMAL--** label can be accessed whenever defining an expression, but is only valid when “C” appears on the status line, which indicates a valid expression has been defined.

< NUMBER> This can be an integer in any base (binary, octal, decimal, or hexadecimal), or can be a string of characters enclosed with quotation marks.

< OP> This represents an algebraic or logical operand and may be any of the following (in order of precedence):

mod	modulo
*	multiplication
/	division
&	logical AND
+	addition
-	subtraction
|	logical OR

--SYMB-- This allows you to define symbolic information for an address, range of addresses, or a file. See the **--SYMB--** syntax pages and the *HP 64000-UX System User’s Guide* for more information on symbols.

end	This displays the last location where the symbol information may be located. For example, if a particular symbol is associated with a range of addresses, **end** will represent the last address in that range.
start	This displays first memory location where the symbol you specify may be located. For example, if a particular symbol is associated with a range of addresses, **start** will represent the first address in that range.
< UNARY>	This defines either the algebraic negation (minus) sign (**-**) or the logical negation (NOT) sign (~).
( )	Parentheses may be used in expressions to enclose numbers. For every opening parenthesis, a closing parenthesis must exist.

Note

When “C” appears on the right side of the status line, a valid expression exists. The **--NORMAL--** key can be accessed at any time, but is only valid when “C” is on the command line.

Examples

```
05fxh
DISP_BUF + 5
SYMB_TBL + (OFFSET / 2)
START
cprog.C: line 15 end
```

Related Commands

help ***expressions***
SYMB

Note

When a **thru** softkey has been entered, a < + EXPR> prompt appears. This saves you from tedious repeated entry of long symbols and expressions. For example:

```
disp_buf thru +25
```

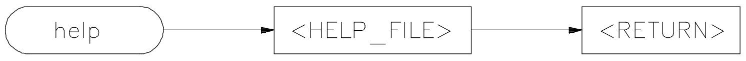
is the same as

```
disp_buf thru disp_buf + 25
```

help

Displays information about system and emulation features during an emulation session.

Syntax



Function

Typing **help** or **?** displays softkey labels that list the options on which you may receive help. When you select an option, the system will list the information to the screen.

The **help** command is not displayed on the softkeys. You must enter it into the keyboard. You may use a question mark in place of **help** to access the help information.

Default Value

none

Parameters

< HELP _FILE>	This represents one of the available options on the softkey labels. You can either press a softkey representing the help file, or type in the help file name. If you are typing in the help file name, make sure you use the complete syntax. Not all of the softkey labels reflect the complete file name.

Examples

```
help  system_commands  <RETURN>
?  run  <RETURN>
```

This is a summary of the commands that appear on the softkey labels when you type **help** or press **?**:

system_commands
run
trace
step
display
modify
break
end
load
store
stop_trace
copy
reset
specify
software_breakpoints
cmb_execute
expressions (--EXPR--)
symbols (--SYMB--)
registers
cmb
wait
pod_command
bbaunload
coverage
performance_measurement_initialize
performance_measurement_run
performance_measurement_end
set

For example, to display information about the command named **pod_command**, enter:

```
help  pod_command  <RETURN>
```

The result resembles:

```
 ---Syntax---

pod_command

---Function---

This command allows you to send commands directly to the HP 64700 emulation pod
and view the results in the "pod_command" display.

                            --- WARNING ---
_____________________________________________________________________________

Care should be taken when using the "pod_command."  The user  interface,  and
the  configuration  files in particular, assume that the configuration of the
64700 pod is NOT changed except by the user interface.  Be  aware  that  what
you   see  in  "modify  configuration"  will  NOT  reflect  the  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
_____________________________________________________________________________

---Parameters---

STRING                A quoted string to send to the HP 64700 pod for execution.
                 ~    Quote characters are matched pairs of double quotes ("),
                   ~  single quotes (’), or carats (^).

---Examples---

pod_command "map"     Display the memory map in the pod.
pod_command ’cf’      Display the configuration settings for the emulator.

---See Also:---

1) Terminal Interface User’s Manual
```

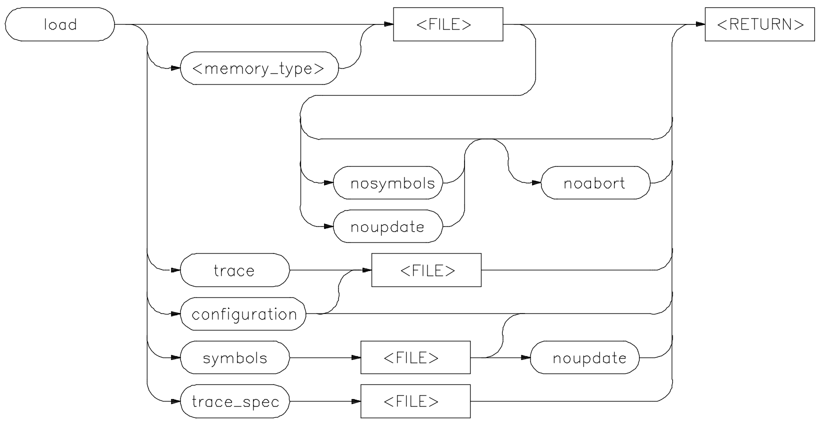
Related Commands See the list under Examples.

Notes

load

This command transfers absolute files from the HP 9000 into emulation or target system RAM. With other parameters, the load command can load emulator configuration files, trace records, trace specifications, or symbol files.

Syntax



Function

The absolute file contains information about where the file is stored. The memory map specifies that the locations of the file are in user (target system) memory or emulation memory. This command also allows you to access and display previously stored trace data, load a previously created configuration file, and load absolute files with symbols.

Note

Any file specified by < FILE> cannot be named “configuration”, “emul_mem”, “user_mem”, “symbols”, “trace”, or “trace_spec” because these are reserved words, and are not recognized by the HP 64000-UX system as ordinary file names. Other reserved words may exist for your emulator (for example, “bkg_mon” and “fg_mon” are reserved words for the 80186 emulator).

Note

If your emulator uses function codes, refer to your *Emulator Softkey Interface User's Guide* for details.

Default Value

The absolute file is loaded into emulation memory by default.

Parameters

configuration	This option specifies that a previously created emulation configuration file will be loaded into the emulator. You can follow this option with a file name. Otherwise the previously loaded configuration will be reloaded.
< FILE>	This represents the absolute file to be loaded into either target system memory, emulation memory (.X files are assumed), or the trace memory (.TR files are assumed).
< memory _type>	This indicates the type of memory that you choose for the load operation. The memory type can be emulation or user memory. You also can load a background monitor file.
noabort	This option allows you to load a file even if part of the file is located at memory mapped as "guarded" or "target ROM" (trom).
nosymbols	This option causes the file specified to be loaded without symbols.
noupdate	This option suppresses rebuilding of the symbol data base when you load an absolute file. If you load an absolute file, end emulation, then modify the file (and relink it), the symbol database will not be updated

	upon reentering emulation and reloading the file. The default is to rebuild the database.
symbols	This option causes the file specified to be loaded with symbols.
trace	This option allows you to load a previously generated trace file.
trace_spec	This option allows you to load a previously generated trace specification.

Note

The current trace specification will be modified, but a new trace will not be started. To start a trace with the newly loaded trace specification, enter **trace again** or **specify trace again** (not **trace**). If you specify **trace**, a new trace will begin with the default trace specification, not the one you loaded.

Examples

```
load  sort1  <RETURN>
load  configuration  config3  <RETURN>
load  trace  trace3  <RETURN>
```

Related Commands

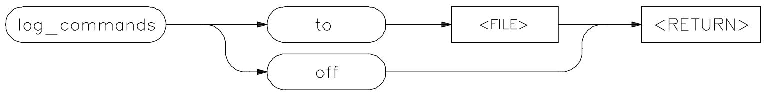
```
display  trace
help  load
```

Notes

log_commands

This command allows you to record commands that are executed during an emulation session.

Syntax



Function

Commands executed during an emulation session are stored in a file until this feature is turned off. This is a handy method for creating command files.

To execute the saved commands after the file is closed, type the filename on the command line.

Default Value

Commands are not logged (stored) in a file.

Parameters

< FILE>	This represents the file where you want to store commands that are executed during an emulation session.
off	This option turns off the capability to log commands.
to	This allows you to specify a file for the logging of commands.

Examples

log_commands ***to*** logfile
log_commands ***off***

Related Commands

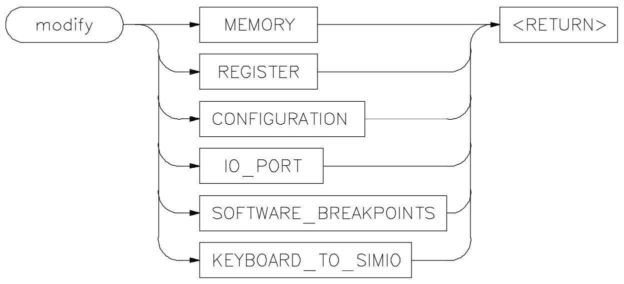
help ***system_commands***

Notes

modify

This command allows you to observe or change information specific to the emulator.

Syntax



Function

The **modify** command is used to:

- View or edit the current emulation configuration.
- Modify contents of memory (as integers, strings, or real numbers).
- Modify the contents of the processor registers.
- Write specified values to I/O port addresses.
- Modify the software breakpoints table.

Note

If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

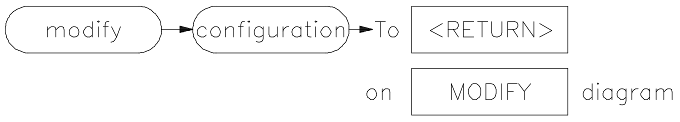
The following pages contain detailed information about the various **modify** syntax diagrams.

Notes

modify configuration

This command allows you to view and edit the current emulation configuration items.

Syntax



Function

The configuration questions are presented in sequence with either the default response, or the previously entered response. You can select the currently displayed response by pressing **< RETURN>** . Otherwise, you can modify the response as you desire, then press **< RETURN>** .

Default Value

For each emulator, default responses defined on powerup are displayed. For details on these default configuration question responses, refer to your *Emulator Softkey Interface User's Guide* and chapter 2 of this manual.

Parameters

none

Example

```
modify  configuration  <RETURN>
```

Related Commands

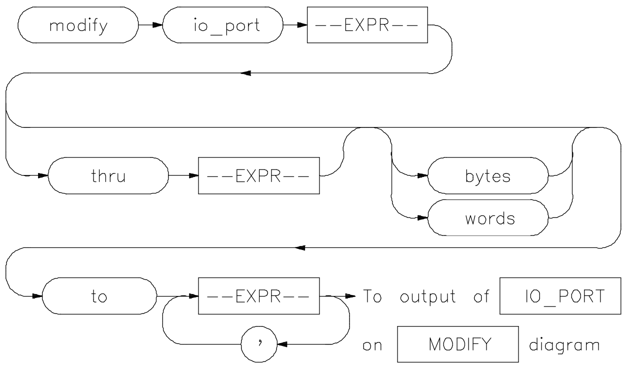
```
help  modify
```

Notes

modify io_port

This command allows you to write a value to a specified I/O address or to a range of I/O addresses.

Syntax



Function

Data may be written as bytes or words, and may be specified as a single entry or as a list of entries. Modifying large ranges may take longer than you expect.

Default Value

The default for modification is to the current display I/O port mode, if one is in effect. Otherwise the default is to “byte.”

Note

Not all HP 64700-Series emulators support the use of I/O ports. Refer to your *Emulator Softkey Interface User’s Guide* for details about your emulator.

Parameters

bytes	Modify the I/O ports with byte values.
--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses

	specifying an I/O port address or I/O port value. See the **--EXPR--** syntax diagram.
thru	This option enables you to specify a range of I/O locations to be modified.
to	This allows you to specify the values to which the selected I/O port locations will be changed.
words	This option allows you to select I/O locations to be modified as word values.
,	A comma is a delimiter between values when multiple I/O port locations are modified.

Examples

```
modify  io_port  0  to  12H  <RETURN>
modify  io_port  PRINTER  words  to  0F3H
<RETURN>
modify  io_port  DISPLAY  thru  DISPLAY+60H
bytes  to  1 , 2 , 3 , 4 , 5 , 6  <RETURN>
```

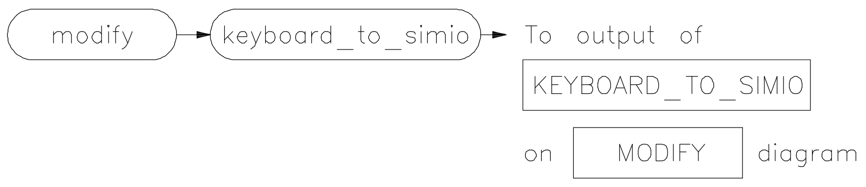
Related Commands

```
copy  io_port
display  io_port
help  modify
```

modify keyboard _to_simio

This command allows the keyboard to interact with your program through the simulated I/O software.

Syntax



Function

When the keyboard is activated for simulated I/O, its normal interaction with emulation is disabled. The emulation softkeys are blank and the softkey labeled “suspend” is displayed on your screen. Pressing **suspend < RETURN>** will deactivate keyboard simulated I/O and return the keyboard to normal emulation mode. For details about setting up simulated I/O on your HP 9000 host computer system, refer to the *HP 64000-UX Simulated I/O Manual.*

Note

This feature is not available on all HP 64700-Series emulators. Refer to your *Emulator Softkey Interface User’s Guide* for more information.

Default Value

none

Parameters

none

Example

```
modify  keyboard_to_simio  <RETURN>
```

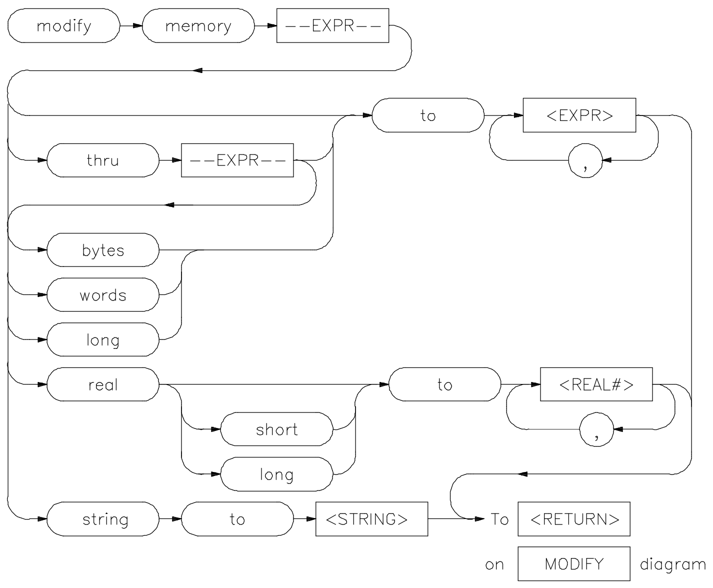
Related Commands

`help` ***`modify`***

modify memory

This command lets you modify the contents of selected memory locations.

Syntax



Function

You can **modify** the contents of individual memory locations to individual values. Or, you can modify a range of memory to a single value or a sequence of values.

Modify a series of memory locations by specifying the address of the first location in the series to be modified, and the values to which the contents of that location and successive locations are to be changed. The first value listed will replace the contents of the first memory location. The second value replaces the contents of the next memory location in the series, and so on, until the list is exhausted. When more than one value is listed, the value representations must be separated by commas. (See the examples for more information.)

A range of memory can be modified such that the content of each location in the range is changed to the single specified value, or to a single or repeated sequence. This type of memory modification is done by entering the limits of the memory range to be modified (--EXPR-- thru --EXPR--) and the value or list of values (--EXPR--, ... , --EXPR--) to which the contents of all locations in the range are to be changed.

Note

If the specified address range is not large enough to contain the new data, only the specified addresses are modified.

If the address range contains an odd number of bytes and a word operation is being executed, the last word of the address range will be modified. Thus the memory modification will stop one byte after the end of the specified address range.

If an error occurs in writing to memory (to guarded memory or target memory with no monitor) the modification is aborted at the address where the error occurred.

Note

If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

Default Values

For integer memory modifications, the default is to the current display memory mode, if one is in effect. Otherwise the default is to "byte."

For real memory modifications, the default is to the current display memory mode, if one is in effect. Otherwise the default is "short."

Parameters

bytes	Modify memory in byte values.
--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying a memory address. See the EXPR syntax diagram.

long	Modify memory as long word values. When used with the **real** parameter, **long** specifies that memory be modified as a 64-bit real number value.
real	Modify memory as real number values.
< REAL# >	This prompts you to enter a real number as the value.
short	Modify memory values as 32-bit real numbers.
string	Modify memory values to the ASCII character string given by < STRING> .
< STRING>	Quoted ASCII string including special characters as follows:

null	\0
newline	\n
horizontal tab	\t
backspace	\b
carriage return	\r
form feed	\f
backslash	\\
single quote	\’
bit pattern	\ooo (where ooo is an octal number)

thru	This option lets you specify a range of memory locations to be modified.
to	This lets you specify values to which the selected memory locations will be changed.

words	Modify memory locations as word values.
,	A comma is used as a delimiter between values when modifying multiple memory addresses.

Examples

modify memory `00A0H` ***words to*** `1234H <RETURN>`

modify memory `DATA1` ***bytes to*** `0E3H , 01H , 08H <RETURN>`

modify memory `DATA1` ***thru*** `DATA100` ***to*** `0FFFFH <RETURN>`

modify memory `0675H` ***real to*** `-1.303 <RETURN>`

modify memory `TEMP` ***real long to*** `0.5532E-8 <RETURN>`

modify memory `buffer` ***string to*** `"This is a test \n\0"`

The following pages show some additional examples of **modify memory**, with screen displays shown to help you see the effects of a particular modification.

display memory blocked bytes

modify memory Msg_Dest ***thru*** +50 ***to*** 41h, 42h, 43h

```
 Memory  :bytes :blocked :update
   address       data       :hex                                   :ascii
    000602-09    41   42   43   41   42   43   41   42      A B C A  B C A B
    00060A-11    43   41   42   43   41   42   43   41      C A B C  A B C A
    000612-19    42   43   41   42   43   41   42   43      B C A B  C A B C
    00061A-21    41   42   43   41   42   43   41   42      A B C A  B C A B
    000622-29    43   41   42   43   41   42   43   41      C A B C  A B C A
    00062A-31    42   43   41   42   43   41   42   43      B C A B  C A B C
    000632-39    41   42   43   53   41   E8   03   FC      A B C S  A . . .
    00063A-41    28   88   20   6B   00   04   41   E8      ( .   k  . . A .
    000642-49    03   FC   29   48   00   04   70   18      . . ) H  . . p .
    00064A-51    D0   82   29   40   00   08   33   FC      . . ) @  . . 3 .
    000652-59    00   0C   00   06   10   A8   33   FC      . . . .  . . 3 .
    00065A-61    03   FC   00   06   10   AA   42   B9      . . . .  . . B .
    000662-69    00   06   04   90   42   79   00   06      . . . .  B y . .
    00066A-71    04   60   42   B9   00   06   04   9A      . ' B .  . . . .
    000672-79    13   FC   00   01   00   06   04   98      . . . .  . . . .
    00067A-81    70   01   23   C0   00   06   04   94      p . # .  . . . .

STATUS:   M68000--Running in monitor____________________________________........
```

modify memory Msg_Dest ***string to*** "HP 64000 Softkey Interface"

```
 Memory  :bytes :blocked :update
   address       data       :hex                                   :ascii
    000602-09    48   50   20   36   34   30   30   30      H P   6  4 0 0 0
    00060A-11    20   53   6F   66   74   6B   65   79        S o f  t k e y
    000612-19    20   49   6E   74   65   72   66   61        I n t  e r f a
    00061A-21    63   65   43   41   42   43   41   42      c e C A  B C A B
    000622-29    43   41   42   43   41   42   43   41      C A B C  A B C A
    00062A-31    42   43   41   42   43   41   42   43      B C A B  C A B C
    000632-39    41   42   43   53   41   E8   03   FC      A B C S  A . . .
    00063A-41    28   88   20   6B   00   04   41   E8      ( .   k  . . A .
    000642-49    03   FC   29   48   00   04   70   18      . . ) H  . . p .
    00064A-51    D0   82   29   40   00   08   33   FC      . . ) @  . . 3 .
    000652-59    00   0C   00   06   10   A8   33   FC      . . . .  . . 3 .
    00065A-61    03   FC   00   06   10   AA   42   B9      . . . .  . . B .
    000662-69    00   06   04   90   42   79   00   06      . . . .  B y . .
    00066A-71    04   60   42   B9   00   06   04   9A      . ' B .  . . . .
    000672-79    13   FC   00   01   00   06   04   98      . . . .  . . . .
    00067A-81    70   01   23   C0   00   06   04   94      p . # .  . . . .

STATUS:   M68000--Running in monitor____________________________________........
```

modify memory `Msg_Dest` ***thru*** `+50` ***to*** `0`

```
 Memory  :bytes :blocked :update
   address       data       :hex                                    :ascii
    000602-09    00   00   00   00   00   00   00   00      . . . .  . . . .
    00060A-11    00   00   00   00   00   00   00   00      . . . .  . . . .
    000612-19    00   00   00   00   00   00   00   00      . . . .  . . . .
    00061A-21    00   00   00   00   00   00   00   00      . . . .  . . . .
    000622-29    00   00   00   00   00   00   00   00      . . . .  . . . .
    00062A-31    00   00   00   00   00   00   00   00      . . . .  . . . .
    000632-39    00   00   00   53   41   E8   03   FC      . . . S  A . . .
    00063A-41    28   88   20   6B   00   04   41   E8      ( .   k  . . A .
    000642-49    03   FC   29   48   00   04   70   18      . . ) H  . . p .
    00064A-51    D0   82   29   40   00   08   33   FC      . . ) @  . . 3 .
    000652-59    00   0C   00   06   10   A8   33   FC      . . . .  . . 3 .
    00065A-61    03   FC   00   06   10   AA   42   B9      . . . .  . . B .
    000662-69    00   06   04   90   42   79   00   06      . . . .  B y . .
    00066A-71    04   60   42   B9   00   06   04   9A      . ‘ B .  . . . .
    000672-79    13   FC   00   01   00   06   04   98      . . . .  . . . .
    00067A-81    70   01   23   C0   00   06   04   94      p . # .  . . . .

STATUS:   M68000--Running in monitor____________________________________........
```

Related Commands

copy memory

display memory

help ***modify***

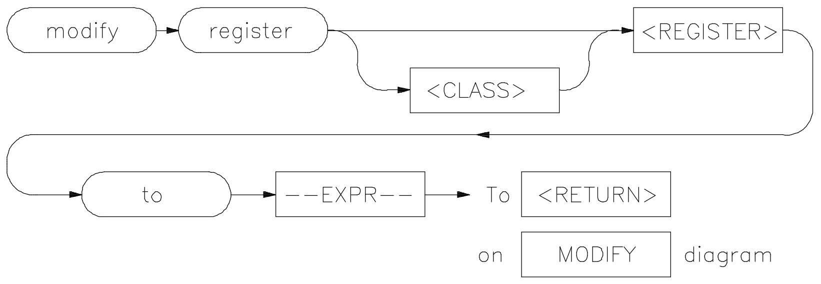
store memory

Also see the **m** syntax pages in the *HP 64700 Emulators Terminal Interface Reference* manual for more information on memory handling and byte ordering in memory modifications.

modify register

This command allows you to modify the contents of one or more of the emulation processor internal registers.

Syntax



Function

The entry you specify for < REGISTER> determines which register is modified.

Register modification cannot be performed during real-time operation of the emulation processor. A **break** command or condition must occur before you can modify the registers.

Default Value

none

Parameters

< CLASS>	This represents the name of a processor register class. Register classes are displayed on the softkey labels.
--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying a register value. See the **--EXPR--** syntax diagram.
< REGISTER>	This represents the name of a register that you specify.

to	This allows you to specify the values to which the selected registers will be changed.

Examples

modify register D0 ***to*** 9H <RETURN>

modify register BASIC PC to 2000H <RETURN>

Note These examples apply to the 68000 emulator. If you are not using a 68000 emulator, these specific examples will not work with your emulator.

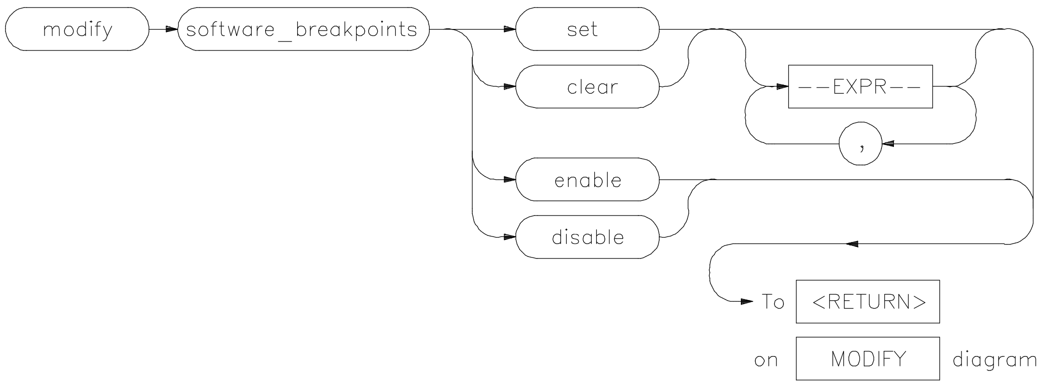
Related Commands

copy registers
display registers
help ***modify***
help ***registers***
modify registers

modify software _breakpoints

Change the specification of software breakpoints.

Syntax



Function

Software breakpoints allow you to break program execution when the breakpoint address is encountered. Any valid address (number, label, or expression) may be specified as a breakpoint. Valid addresses identify the first byte of valid instructions. Operation of the program can be resumed after the breakpoint is encountered, by specifying either a **run** or **step** command.

If you modify software breakpoints while the memory mnemonic display is active, the new breakpoints are indicated by a “*” in the leftmost column of the instruction containing the breakpoint.

Note

Do not modify software breakpoints while the user program is running. If you do, program execution may be unpredictable.

Note If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

You must enable software breakpoints before you can perform an action on them.

Default Values

When you set software breakpoints, the emulator will search through the existing software breakpoint list and reactivate all entries that are inactivated.

When you clear software breakpoints, the entire software breakpoint list is deleted and memory is restored to its original values.

Parameters

clear	This option erases the specified breakpoint address and restores the original content of the memory location. (The location must not have changed (by loading a file or modifying memory) after the breakpoint was set.) If no breakpoints are specified in the command, all currently specified breakpoints are cleared and the memory locations restored to their original values.
disable	This option turns off the software breakpoint capability.
enable	This option allows you to modify the software breakpoint specification.
--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying a software breakpoint address. See the EXPR syntax diagram.
set	This option allows you to activate software breakpoints in your program. If no

breakpoint addresses are specified in the command, all breakpoints that have been inactivated (executed) are reactivated.

,	A comma is used as a delimiter between specified breakpoint values.

Examples

modify software_breakpoints enable `<RETURN>`

modify software_breakpoints clear `99H , 1234H <RETURN>`

modify software_breakpoints set `LOOP1 END , LOOP2END , 0EH <RETURN>`

modify software_breakpoints clear `<RETURN>`

modify software_breakpoints set `<RETURN>`

Related Commands

copy software_breakpoints

display memory mnemonic

display software_breakpoints

`help` ***modify***

`help` ***software_breakpoints***

Notes

performance _measurement _end

This command stores data previously generated by the **performance_measurement_run** command, in a file named “perf.out" in the current working directory.

Syntax

performance_measurement_end
<RETURN>

Function

The file named “perf.out” is overwritten each time this command is executed. Current measurement data existing in the emulation system is not altered by this command.

Default Value

none

Parameters

none

Example

```
performance_measurement_end  <RETURN>
```

Related Commands

```
help  performance_measurement_initialize
help  performance_measurement_run
performance_measurement_initialize
performance_measurement_run
```

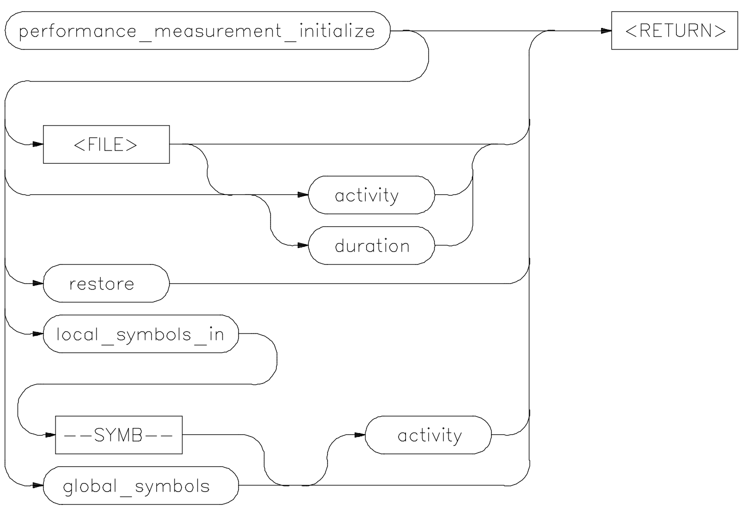
Refer to the *Analyzer Softkey Interface User’s Guide* for examples of performance measurement specification and use.

Notes

performance _measurement _initialize

This command sets up performance measurements.

Syntax



Function The emulation system will verify whether a symbolic database has been loaded. If a symbolic database has been loaded, the performance measurement is set up with the addresses of all global procedures and static symbols. If a valid database has not been loaded, the system will default to a predetermined set of addresses, which covers the entire emulation processor address range.

Default Value The measurement will default to “activity” mode.

Default values will vary, depending on the type of operation selected, and whether symbols have been loaded.

Parameters

activity	This option causes the performance measurement process to operate as though an option is not specified.
duration	This option sets the measurement mode to “duration.” Time ranges will default to a predetermined set (unless a user-defined file of time ranges is specified).
< FILE>	This represents a file you specify to supply user-defined address or time ranges to the emulator.
global _symbols	This option specifies that the performance measurement will be set up with the addresses of all global symbols and procedures in the source program.
local _symbols_in	This causes addresses of the local symbols to be used as the default ranges for the measurement.
restore	This option restores old measurement data so that a measurement can be continued when using the same **trace** command as previously used.
--SYMB--	This represents the source file that contains the local symbols to be listed. This also can be a program symbol name, in which case all symbols that are local to a function or procedure are used. See the SYMB syntax diagram.

Examples

performance_measurement_initialize `<RETURN>`

performance_measurement_initialize `duration <RETURN>`

performance_measurement_initialize local_symbols_in `prog68k.S: <RETURN>`

Related Commands

`help` ***performance_measurement_initialize***

`help` ***performance_measurement_run***

performance_measurement_run

performance_measurement_end

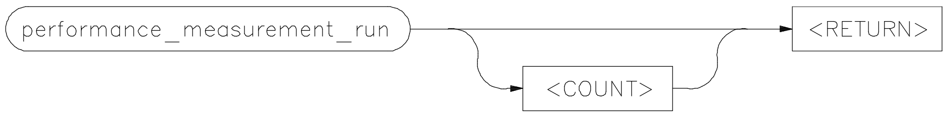
Refer to the *Analyzer Softkey Interface User's Guide* for examples of performance measurement specification and use.

Notes

performance _measurement _run

This command begins a performance measurement.

Syntax



Function

This command causes the emulation system to reduce trace data contained in the emulation analyzer, which will then be used for analysis by the performance measurement software.

Default Value

The default is to process data presently contained in the analyzer.

Parameters

< COUNT>	This represents the number of consecutive traces you specify. The emulation system will execute the trace command, process the resulting data, and combine it with existing data. This sequence will be repeated the number of times specified by the **COUNT** option.

Note

The **trace** command must be set up correctly for the requested measurement. For an activity measurement, you can use the default **trace** command (**trace counting time** < **RETURN**>).

For a duration measurement, you must set up the trace specification to store only the points of interest. To do this, for example, you could enter:

```
trace  only  <symbol_entry> or <symbol_exit>
```

Examples

```
performance_measurement_run  10  <RETURN>
performance_measurement_run  <RETURN>
```

Related Commands

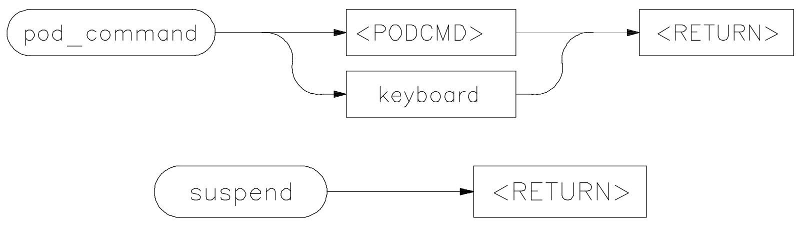
```
help  performance_measurement_initialize
help  performance_measurement_run
performance_measurement_end
performance_measurement_initialize
```

Refer to the *Analyzer Softkey Interface User's Guide* for examples of performance measurement specification and use.

pod_command

Allows you to control the emulator through the direct HP 64700 Terminal Interface.

Syntax



Function

The HP 64700 Series emulators contain a low-level Terminal Interface, which allows you to control the emulator's functions directly. You can access this interface using **pod_command**. The options to **pod_command** allow you to supply only one command at a time. Or, you can select a keyboard mode which gives you interactive access to the Terminal Interface.

The *Terminal Interface Reference* and *User's Guide* for your emulator are excellent sources of information on using the Terminal Interface to control the emulator. But, there are certain commands that you should avoid while using the Terminal Interface through **pod_command**.

stty, po, xp	Do not use. These commands will change the operation of the communications channel, and are likely to hang the Softkey Interface and the channel.
echo, mac	Using these may confuse the communications protocols in use on the channel.
wait	Do not use. The pod will enter a wait state, blocking access by the Softkey Interface.
init, pv	These will reset the emulator pod and force an end release_system command.
t	Do not use. The trace status polling and unload will become confused.

To see the results of a particular **pod_command** (the information returned by the emulator pod), you use **display pod_command**.

Default

None. You must specify either a particular Terminal Interface command as a quoted string or enter the **keyboard** mode.

Parameters

keyboard	Enters an interactive mode where you can simply type Terminal Interface commands (unquoted) on the command line. Use **display pod_command** to see the results returned from the emulator.
< POD_CMD>	Prompts you for a Terminal Interface command as a quoted string. Enter the command in quotes and press < RETURN> .
suspend	This command is displayed once you have entered keyboard mode. Select it to stop interactive access to the Terminal Interface and return to the Softkey Interface.

Examples

This example shows a simple interactive session with the Terminal Interface.

display pod_command

pod_command keyboard

`cf`

`tsq`

`tcq`

```
Pod Commands
  Time                Command
  cf lfc=x
  cf mon=bg
  cf rrt=dis
  cf rssp=01ffe
  cf swtp=0f
  cf ti=en

16:18:37 tsq

  tif 1 any
  tsto all
  telif never

16:18:44 tcq

  tcq time

STATUS:   M68000--Running in monitor____________________________________........
```

Enter **suspend** to return to the Softkey Interface.

Related Commands

display pod_command

`help` ***pod_command***

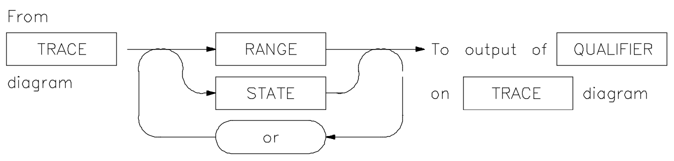
Also see the *Terminal Interface Reference* and *User's Guide* manuals for your emulator.

Notes

QUALIFIER

The **QUALIFIER** parameter is used with **trace only**, **trace prestore**, **TRIGGER**, and **trace counting** to specify states captured during the trace measurement.

Syntax



Function

You may specify a range of states (RANGE) or specific states (STATE) to be captured. You can continue to “or” states until the analyzer resources are depleted. You can use only one RANGE statement in the entire **trace** command.

You can include “don’t care numbers.” These contain an “x” preceded and/or followed by a number. Some examples include 1fxxh, 17x7o, and 011xxx10b. “Don’t care numbers” may be entered in binary, octal, or hexadecimal base.

Default Value

The default is to qualify on all states.

Parameters

or	This option allows you to specify multiple states (STATE) to be captured during a trace measurement. See the STATE syntax diagram.
RANGE	This allows you to specify a range of states to be captured during a trace measurement. See the RANGE syntax diagram.
STATE	This represents a unique state that can be a combination of address, data, status values,

and external labels. See the STATE syntax diagram.

Examples

trace only address `prog68k.S:READ_INPUT <RETURN>`

trace only address range `prog68k.S:READ_INPUT` ***thru*** `OUTPUT <RETURN>`

trace only address range `prog68k.S:CLEAR` ***thru*** `READ_INPUT <RETURN>`

Related Commands

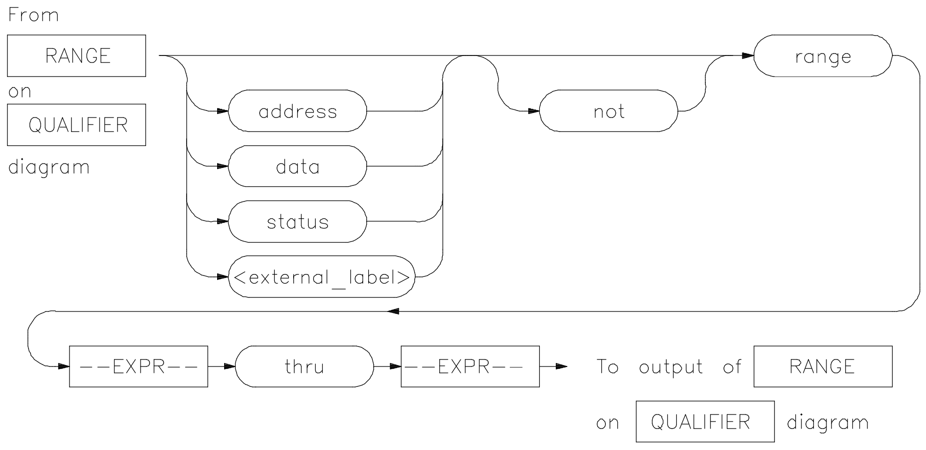
`help` ***trace***

trace

RANGE

The **RANGE** parameter allows you to specify a condition for the trace measurement, made up of one or more values.

Syntax



Function

The **range** option can be used for address, data, status, and external labels. **Range** can only be used once in a trace measurement.

Default Value

Expression types are “address” when none is chosen.

Parameters

address	This specifies that the expression that follows is an address value.
data	This specifies that the expression that follows is a data value on the emulation processor data bus.

--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying an address, data or status value. See the EXPR syntax diagram for details.
< external _label>	This represents a defined external analyzer label.
not	This specifies that the analyzer search for the logical "not" of the specified range (this includes any addresses not in the specified range).
range	This indicates a range of addresses to be specified (--EXPR-- thru --EXPR--).
status	This allows the analyzer to trace status information, such as read operations.
thru	This indicates that the following address expression is the upper address in a range.

Examples

See the **trace** command examples.

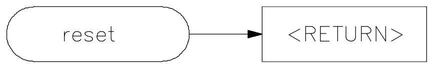
Related Commands

help ***trace***
QUALIFIER
trace

reset

This command suspends target system operation and reestablishes initial emulator operating parameters, such as reloading control registers.

Syntax



Function

The reset signal is latched when the reset command is executed and released by either the **run** or **break** command.

Default Value

The emulator is reset to background.

Parameters

none

Example

```
reset  <RETURN>
```

Related Commands

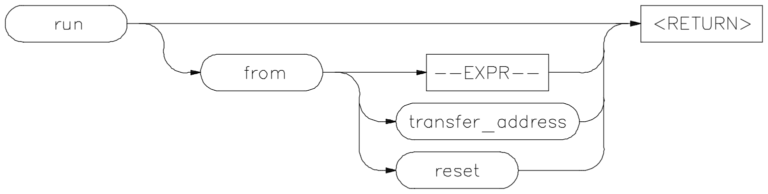
```
help  reset
```

Notes

run

This command causes the emulator to execute a program.

Syntax



Function

If the processor is in a reset state, **run** releases the reset condition. If you specify **run from --EXPR--** or **run from transfer_address**, the processor is directed to the particular address. If the processor is running in the emulation monitor or background memory, a **run** command causes the processor to exit into your program. The program can either run from a specified address (--EXPR--), from the address stored in the emulation processor program counter, or from a label specified in the program.

For an explanation of how your emulator runs from a reset condition (using the **run from reset** command), refer to your *Emulator Terminal Interface User's Guide.*

Note

If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

Default Value

If you omit the address option (--EXPR--), the emulator begins program execution at the current address specified by the emulation processor program counter. If an absolute file containing a transfer address has just been loaded, execution starts at that address.

Parameters

--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying a memory address. See the EXPR syntax diagram.
from	This specifies the address from which program execution is to begin.
reset	This option starts the processor executing from the reset address, or when a reset signal is initiated in the target system. Refer to the *Emulator Softkey Interface User's Guide* for details about the **run from reset** operation.
transfer _address	This represents the starting address of the program loaded into emulation or target memory. The transfer address is defined in the linker map.

Examples

```
run  <RETURN>
run  from  810H  <RETURN>
run  from  COLD_START  <RETURN>
```

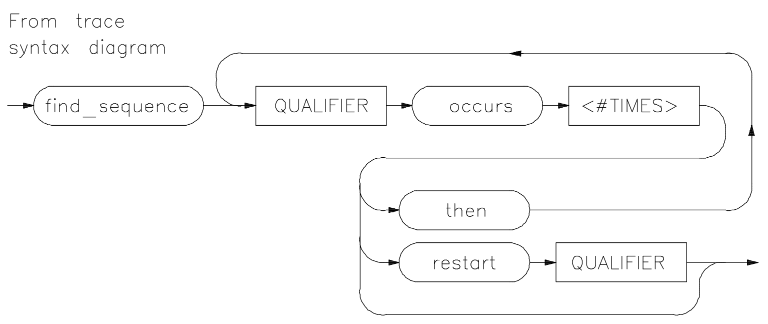
Related Commands

```
help  run
help  step
step
```

SEQUENCING

Lets you specify complex branching activity that must be satisfied to trigger the analyzer.

Syntax



Function

Sequencing provides you with parameters for the **trace** command that let you define branching conditions for the analyzer trigger.

You are limited to a total of seven sequence terms, including the trigger, if no windowing specification is given. If windowing is selected, you are limited to a total of four sequence terms.

Default Value

The analyzer default is no sequencing terms. If you select the sequencer using the find_sequence parameter, you must specify at least one qualifying sequence term.

Parameters

find_sequence	Specifies that you want to use the analysis sequencer. You must enter at least one qualifier.
QUALIFIER	Specifies the address, data, or status value or value range that will satisfy this sequence term if looking for a sequence (find_sequence), or will restart at the beginning of the sequence (restart). See the

	QUALIFIER syntax pages for further information.
occurs	Selects the number of times a particular qualifier must be found before the analyzer proceeds to the next sequence term or the trigger term. This option is not available when trace windowing is in use. See the **WINDOW** syntax pages.
< # TIMES>	Prompts you for the number of times a qualifier must be found.
then	Allows you to add multiple sequence terms, each with its own qualifier and occurrence count.
restart	Selects global restart. If the analyzer finds the restart qualifier while searching for a sequence term, the sequencer is reset and searching begins for the first sequence term.

Examples

The following example uses the "Getting Started" program from the 68000/68010 Emulator User's Guide.

The program reads a one-byte command location and writes one of three messages to an output area depending on what was input.

We want to trace only when we see the program startup, followed by clearing the command input, and an access to the address for a particular message. We want to restart the analyzer if the compare is passed for that particular message, indicating that it was not the message input.

```
display trace
trace find_sequence Init then Read_Cmd
restart Exe_Cmd + 8 trigger after Msg_A
modify Cmd_Input to 41h
```

The result is shown in the following display:

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data               Opcode or Status               time count
Base:     hex     hex                    mnemonic                    relative
after    000500     436F    43    supr data rd byte                   480     nS
+001     000602     4343    43    supr data wr byte                   520     nS
+002     000454     FFFC  Unimplemented Instruction:0FFFC             480     nS
+003     000450     12D8  MOVE.B  [A0]+,[A1]+                         760     nS
+004     000452     57C9  DBEQ    D1,0000450                          520     nS
+005     000501     436F      6F  supr data rd byte                   480     nS
+006     000603     6F6F      6F  supr data wr byte                   520     nS
+007     000454     FFFC    FFFC  supr prog                           480     nS
+008     000450     12D8  MOVE.B  [A0]+,[A1]+                         760     nS
+009     000452     57C9  DBEQ    D1,0000450                          480     nS
+010     000502     6D6D    6D    supr data rd byte                   520     nS
+011     000604     6D6D    6D    supr data wr byte                   480     nS
+012     000454     FFFC    FFFC  supr prog                           520     nS
+013     000450     12D8  MOVE.B  [A0]+,[A1]+                         760     nS
+014     000452     57C9  DBEQ    D1,0000450                          480     nS

STATUS:   M68000--Running user program    Emulation trace complete______........
```

Related Commands

trace

QUALIFIER

WINDOW

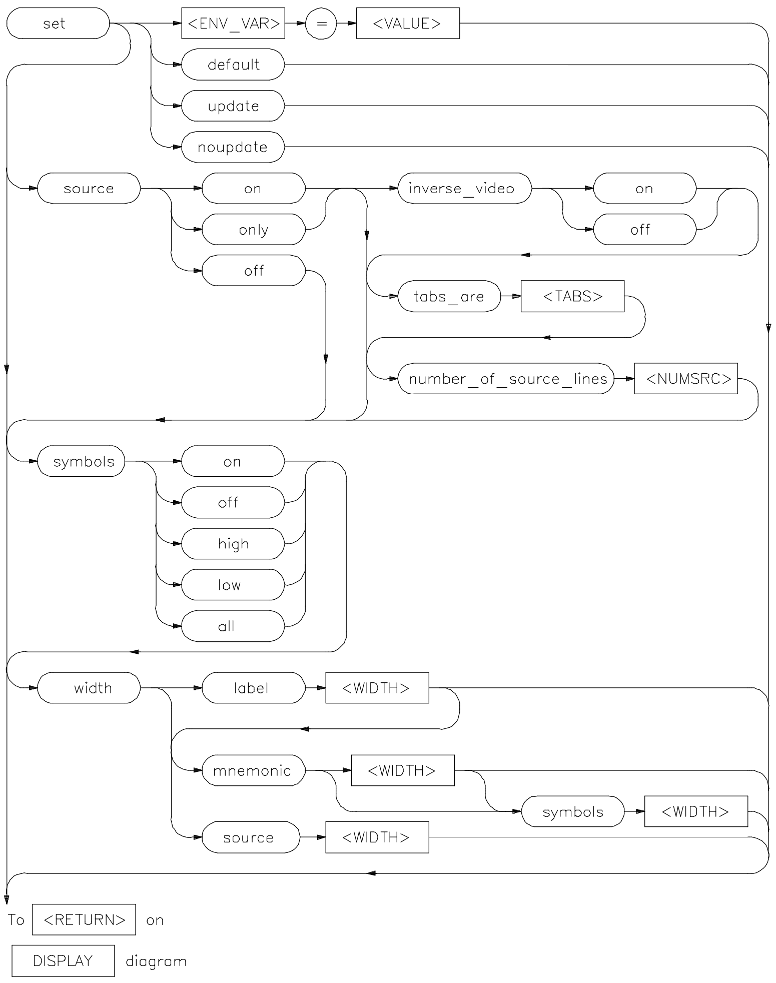
help ***trace***

Notes

set

Controls the display format for the data, memory, register, software breakpoint, and trace displays.

Syntax



Function With the set command, you can adjust the display format results for various measurements, making them easier to read and interpret.

Formatting of source lines, symbol display selection and width, and update after measurement can be modified to your needs.

The display command uses the set command specifications to format measurement results for the display window.

Another option to the set command, **< ENV_VAR> = < VALUE>** , allows you to set and export system variables to the HP-UX and HP 64000-UX environments.

Default Values

The default display format parameters are the same as those set by the commands:

```
set update
set source off symbols off
```

You can return the display format to this state by simply using the command:

```
set default
```

Parameters

default	This option restores all the set options to their default settings.
< ENV_VAR>	Specifies the name of an environment variable to be set within the HP 64000-UX environment or the system environment.
=	The equals sign is used to equate the < ENV_VAR> parameter to a particular value represented by < VALUE> .
inverse video	
off	This displays source lines in normal video.
on	This highlights the source lines on the screen (dark characters on light background) to differentiate the source lines from other data on the screen.

noupdate	When using multiple windows or terminals, and specifying this option, the display buffer in that window or terminal will not update when a new measurement completes. Displays showing memory contents are not updated when a command executes that could have caused the values in memory to change (modify memory, load, etc.).
number_of_ source_lines	This allows you to specify the number of source lines displayed for the actual processor instructions with which they correlate. Only source lines up to the previous actual source line will be displayed. Using this option, you can specify how many comment lines are displayed preceding the actual source line. The default value is 5.
< NUMSRC>	This prompts you for the number of source lines to be displayed. Values in the range 1..50 may be entered.
source	
off	This option prevents inclusion of source lines in the trace and memory mnemonic display lists.
on	This option displays source program lines preceding actual processor instructions with which they correlate. This enables you to correlate processor instructions with your source program code. The option works for both the trace list and memory mnemonic displays.
only	This option displays only source lines. Processor instructions are only displayed in memory mnemonic if no source lines correspond to the instructions. Processor

	instructions are never displayed in the trace list.
symbols	
off	This prevents symbol display.
on	This displays symbols. This option works for the trace list, memory, software breakpoints, and register step mnemonics.
high	Displays only high level symbols, such as those available from a compiler. See the HP 64000-UX System User's Guide for a detailed discussion of symbols.
low	Displays only low level symbols, such as those generated internally by a compiler, or an assembly symbol.
all	Displays all symbols.
tabs_are	This option allows you to define the number of spaces inserted for tab characters in the source listing.
< TABS>	Prompts you for the number of spaces to use in replacing the tab character. Values in the range of 2..15 may be entered.
update	When using multiple windows or terminals, and specifying this option, the display buffer in that window or terminal will be updated when a new measurement completes. This is the default. Note that for displays that show memory contents, the values will be updated when a command executes that changes memory contents (such as modify memory, load, and so on).

< VALUE>	Specifies the logical value to which a particular HP-UX or HP 64000-UX system environment variable is to be set.
width	
source	This allows you to specify the width (in columns) of the source lines in the memory mnemonic display. To adjust the width of the source lines in the trace display, increase the widths of the label and/or mnemonic fields.
label	This lets you specify the address width (in columns) of the address field in the trace list or label (symbols) field in any of the other displays.
mnemonic	This lets you specify the width (in columns) of the mnemonic field in memory mnemonics, trace list and register step mnemonics displays. It also changes the width of the status field in the trace list.
symbols	This lets you specify the maximum width of symbols in the mnemonic field of the trace list, memory mnemonic, and register step mnemonic displays.
< WIDTH>	This prompts you for the column width of the source, label, mnemonic, or symbols field.

Note

< CTRL> f and < CTRL> g may be used to shift the display left or right to display information which is off the screen.

Examples

set noupdate

set source on inverse_video on tabs_are 2

set symbols on width label 30 ***mnemonic*** 20

set PRINTER = "lp -s"

set HP64KSYMBPATH=".file1:proc1 .file2:proc2:code_block_1"

Related Commands

display data

display memory

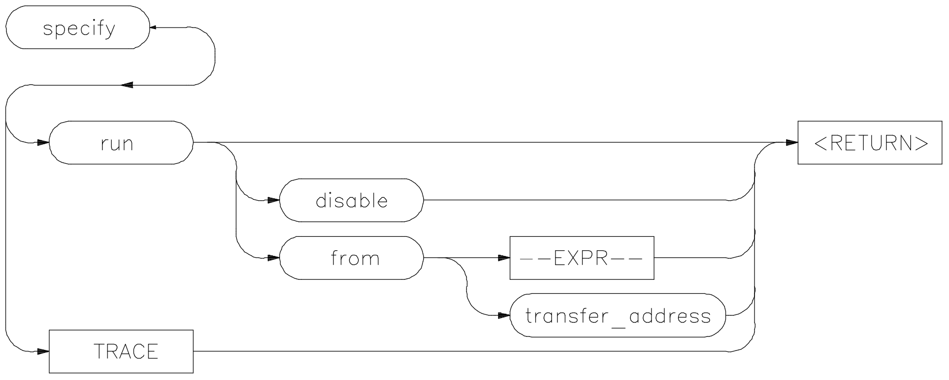
display software_breakpoints

display trace

specify

This command prepares a **run** or **trace** command for execution, and is used with the **cmb_execute** command.

Syntax



Function

When you precede a **run** or **trace** command with **specify**, the system does not execute your command immediately. Instead, it waits until you enter a **cmb_execute** command.

If the processor is reset and no address is specified, a **cmb_execute** command will run the processor from the "reset" condition.

If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

Note

The **run** specification is active until you enter **specify run disable**. The trace specification is active until you enter another **trace** command without the **specify** prefix.

Default Value

The emulator will run from the current program counter address.

Parameters

disable	This option turns off the specify condition of the **run** process.
from	
--EXPR--	This is used with the **specify run from** command. An expression is a combination of numeric values, symbols, operators, and parentheses, specifying a memory address. See the EXPR syntax diagram.
transfer _address	This is used with the **specify run from** command, and represents the address from which the program will begin running.
run	This option specifies that the emulator will run from either an expression or from the transfer address when a CMB EXECUTE signal is received.
TRACE	This option specifies that a trace measurement will be taken when a CMB EXECUTE signal is received.

Examples

specify run from `START <RETURN>`

specify trace after address `1234H <RETURN>`

Related Commands

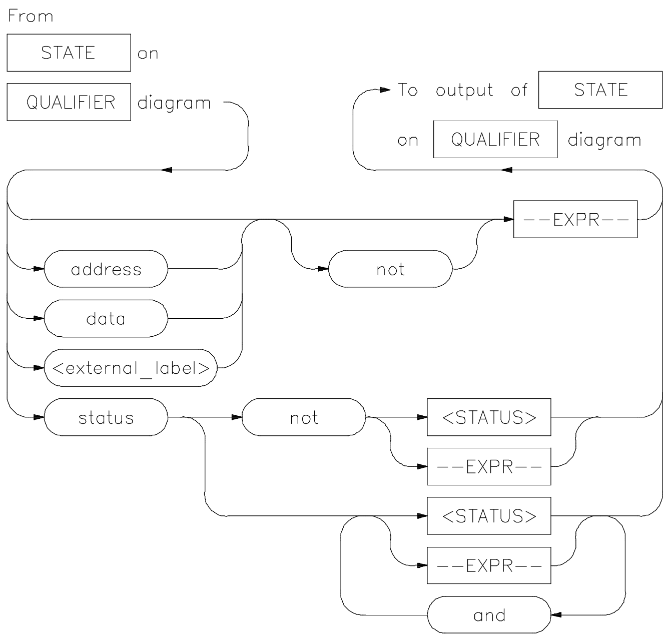
cmb_execute

`help` ***specify***

STATE

This parameter lets you specify a trigger condition as a unique combination of address/data/ status values and external analyzer labels.

Syntax



Function

The STATE option is part of the QUALIFIER parameter to the **trace** command, and allows you to specify a condition for the trace measurement.

Default Value

The default STATE expression type is address.

Parameters

address	This specifies that the expression following is an address value. This is the default, and is

	therefore not required on the command line when specifying an address expression.
and	This lets you specify a combination of status and expression values when **status** is specified in the state specification.
data	This specifies that the expression following is a data value on the processor data bus.
--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses, specifying an address, data, or status value. See the EXPR syntax diagram.
< external _label>	This specifies an external analyzer label to be included in the trace measurement.
not	This specifies that the analyzer will search for the logical "not" of a specified state (this includes any address that is not in the specified state).
status	This specifies that the expression following, or status word, is a status value for the processor.
< STATUS>	This prompts you to enter a status value in the command line. Status values can be entered from softkeys or typed into the keyboard. Numeric values may be entered using symbols, operators, and parentheses to specify a status value. See the EXPR syntax diagram.

Examples

See the **trace** command examples.

Related Commands

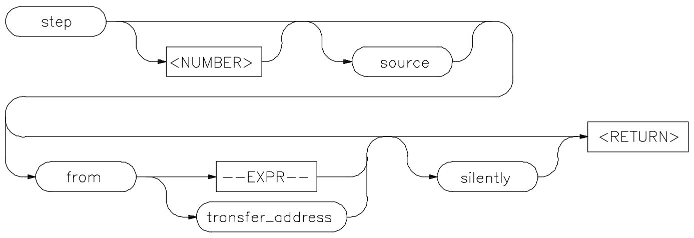
`help` ***`trace`***
`QUALIFIER`
`trace`

Notes

step

The **step** command allows sequential analysis of program instructions by causing the emulation processor to execute a specified number of assembly instructions or source lines.

Syntax



Function

You can display the contents of the processor registers, trace memory, and emulation or target memory after each **step** command.

Source line stepping is implemented by single stepping assembly instructions until the next PC is beyond the address range of the current source line. When attempting source line stepping on assembly code (with no associated source line), stepping will complete when a source line is found. Therefore, stepping only assembly code may step forever. To abort stepping, type < CTRL> c.

When displaying memory mnemonic and stepping, the next instruction that will step is highlighted. The memory mnemonic display autopages to the new address if the next PC goes outside of the currently displayed address range. This feature works even if stepping is performed in a different emulation window than one displaying memory mnemonic (see the chapter on windowing capabilities).

Note

If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

Default Values

If no value is entered for < NUMBER> times, only one **step** instruction is executed each time you press **< RETURN>** . Multiple instructions can be executed by holding down the **< RETURN>** key. Also, the default step is for assembly code lines, not source code lines.

If the **from** address option (defined by **--**EXPR**--** or transfer_address) is omitted, stepping begins at the next program counter address.

Parameters

--EXPR--	An expression is a combination of numeric values, symbols, operators, and parentheses specifying a memory address. See the EXPR syntax diagram.
from	Use this option to specify the address from which program stepping begins.
< NUMBER>	This defines the number of instructions that will be executed by the **step** command. The number of instructions to be executed can be entered in binary (B), octal (O or Q), decimal (D), or hexadecimal (H) notation.
silently	This option updates the register step mnemonic only after stepping is complete. This will speed up stepping of many instructions. The default is to update the register step mnemonic after each assembly instruction (or source line) executes (if stepping is performed in the same window as the register display).
transfer _address	This represents the starting address of the program you loaded into emulation or target memory. The transfer_address is defined in the linker map.

source	This option performs stepping on source lines.

Examples

step <RETURN>

step from 810H <RETURN>

step 20 ***from*** 0A0H <RETURN>

step 5 ***source*** <RETURN>

step 20 ***silently*** <RETURN>

step 4 ***from*** main

```
Registers

 A0-A7  00000000 00000000 00000000 00000000 00000000 00000000 00043FFC 00043FF8


Step_PC 001070@sp  MOVEA.L #0000601AA,A2
Next_PC 001076@sp
 PC 00001076       STATUS 2704  s  z           USP 00000000      SSP 00043FF8
 D0-D7  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 A0-A7  00000000 00000000 000601AA 00000000 00000000 00000000 00043FFC 00043FF8


Step_PC 001076@sp  JSR     |_initialize_sys
Next_PC 00114A@sp
 PC 0000114A       STATUS 2704  s  z           USP 00000000      SSP 00043FF4
 D0-D7  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 A0-A7  00000000 00000000 000601AA 00000000 00000000 00000000 00043FFC 00043FF4


STATUS:   M68000--Stepping complete_____________________________________........
```

Related Commands

help ***step***

display registers

display memory mnemonic

set symbols

Notes

stop_trace

This command terminates the current trace and stops execution of the current measurement.

Syntax

Function

The analyzer stops searching for trigger and trace states. If trace memory is empty (no states acquired), nothing will be displayed.

Default Value

none

Parameters

none

Example

stop_trace `<RETURN>`

Related Commands

`help` ***stop_trace***

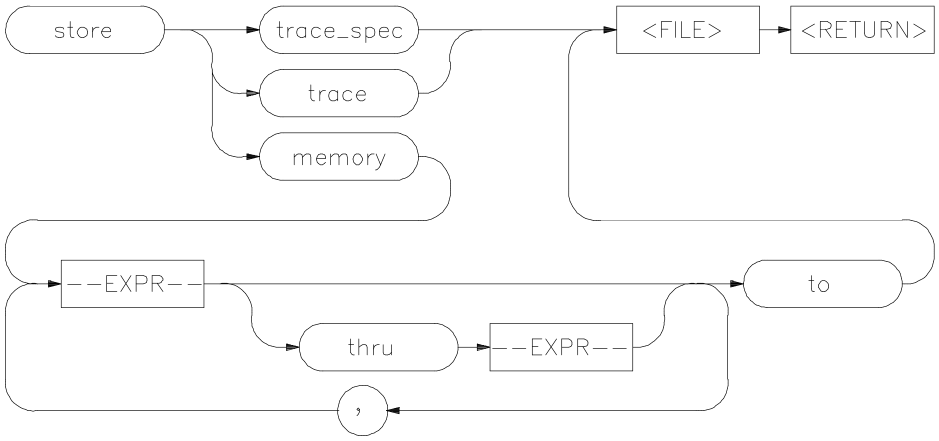
trace

Notes

store

This command lets you save the contents of specific memory locations in an absolute file. You also can save trace memory contents in a trace file.

Syntax



Function

The **store** command creates a new file with the name you specify, if there is not already an absolute file with the same name. If a file represented by < FILE> already exists, you must decide whether to keep or delete the old file. If you respond with **yes** to the prompt, the new file replaces the old one. If you respond with **no**, the **store** command is canceled and no data is stored.

Note

If your emulator uses function codes, refer to the *Emulator Softkey Interface User's Guide* for details.

Default Value

The transfer address of the absolute file is set to zero.

Parameters

--EXPR--	This is a combination of numeric values, symbols, operators, and parentheses, specifying a memory address. See the EXPR syntax diagram.
< FILE>	This represents a file name you specify for the absolute file identifier or trace file where data is to be stored. If you want to name a file beginning with a number, you must precede the file name with a backslash (\) so the system will recognize it as a file name.
memory	This causes selected memory locations to be stored in the specified file with a **.X** extension.
thru	This allows you to specify that ranges of memory be stored.
to	Use this in the **store memory** command to separate memory locations from the file identifier.
trace	This option causes the current trace data to be stored in the specified file with a **.TR** extension.
trace_spec	This option stores the current trace specification in the specified file with a **.TS** extension.
,	A comma separates memory expressions in the command line.

Examples

```
store memory 800H thru 20FFH to TEMP2
<RETURN>
store memory EXEC thru DONE to \12.10
<RETURN>
store trace TRACE <RETURN>
store trace_spec TRACE <RETURN>
```

Related Commands

display memory

display trace

help ***store***

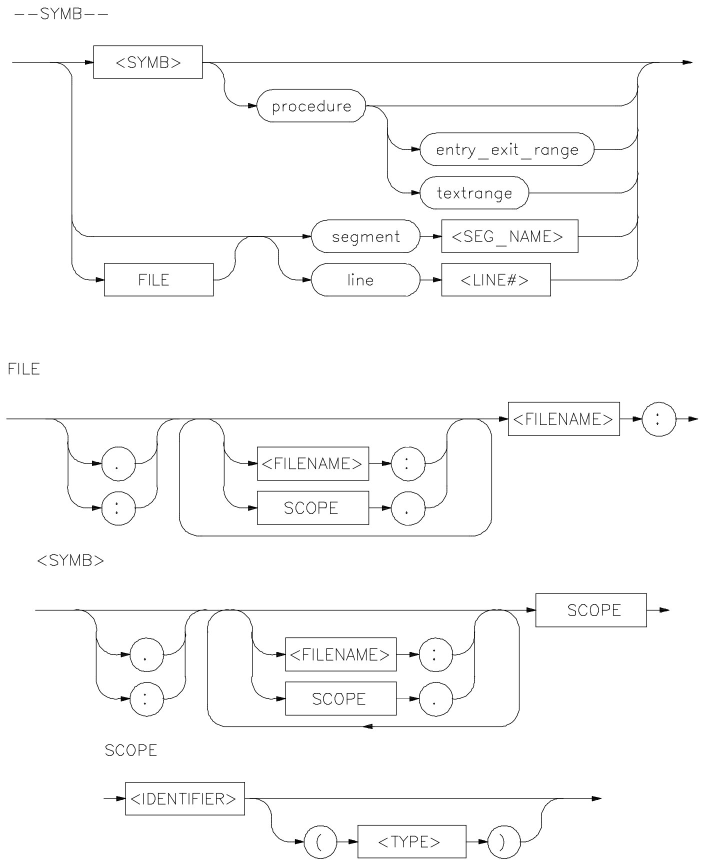
load

Notes

--SYMB--

This parameter is a symbolic reference to an address, address range, file, or other value.

Syntax



Note

If no default file was defined by executing the command **display local_symbols_in --SYMB--**, or with the **cws** command, a source file name (< **FILE**>) must be specified with each local symbol in a command line.

Function

Symbols may be:

- Combinations of paths, filenames, and identifiers defining a scope, or referencing a particular identifier or location (including procedure entry and exit points).
- Combinations of paths, filenames, and line numbers referencing a particular source line.
- Combinations of paths, filenames, and segment identifiers identifying a particular PROG, DATA or COMN segment or a user-defined segment.

The Symbolic Retrieval Utilities (SRU) handle symbol scoping and referencing. These utilities build trees to identify unique symbol scopes.

If you use the SRU utilities to build a symbol database before entering the emulation environment, the measurements involving a particular symbol request will occur immediately. If you then change a module and reenter the emulation environment without rebuilding the symbol database, the emulation software rebuilds the changed portions of the database in increments as necessary.

Further information regarding the SRU and symbol handling is available in the *HP 64000-UX System User's Guide.* Also refer to that manual for information on the **HP64KSYMBPATH** environment variable.

Default Value

The last symbol specified in a **display local_symbols_in --SYMB--** command, or with the **cws** command, is the default symbol scope. The default is “none” if no current working symbol was set in the current emulation session.

You also can specify the current working symbol by typing the cws command on the command line and following it with a symbol name. The **pws** command displays the current working symbol on the status line.

Display memory mnemonic also can modify the current working symbol.

Parameters

< FILENAME>	This is an HP-UX path specifying a source file. If no file is specified, and the identifier referenced is not a global symbol in the executable file that was loaded, then the default file is assumed (the last absolute file specified by a display local_symbols_in command). A default file is only assumed when other parameters (such as **line**) in the **--SYMB--** specification expect a file.
line	This specifies that the following numeric value references a line number in the specified source file.
< LINE# >	Prompts you for the line number of the source file.
< IDENTIFIER>	Identifier is the name of an identifier as declared in the source file.
SCOPE	Scope is the name of the portion of the program where the specified identifier is defined or active (such as a procedure block).
segment	This indicates that the following string specifies a standard segment (such as PROG, DATA, or COMN) or a user-defined segment in the source file.
< SEG_NAME>	Prompts you for entry of the segment name.

(<TYPE>)	When two identifier names are identical and have the same scope, you can distinguish between them by entering the type (in parentheses). Do not type a space between the identifier name and the type specification. The type will be one of the following:
filename	Specifies that the identifier is a source file.
fsegment	Used by the 80386 emulator only; holds information about code or data fsegments in the global descriptor table.
module	These refer to module symbols. For the 80386 C compiler, these names derive from the source file name. For Ada, they are packages. Other language systems may allow user-defined module names.
procedure	Any procedure or function symbol. For languages that allow a change of scope without explicit naming, SRU assigns an identifier and tags it with type procedure.
static	Static symbols, which includes global variables. The logical address of these symbols will not change.
task	Task symbols, which are specifically defined by the processor and language system in use.
:	A colon is used to specify the HP-UX file path from the line, segment, or symbol specifier. When following the file name with a line or segment selection, there must be a space after the colon. For a symbol, there must not be a space after the colon.

Examples

The following short C code example should help illustrate how symbols are maintained by SRU and referenced in your emulation commands.

```
int *port_one;
main ()
{
int port_value;

  port_one = 255;
  port_value = 10;

  process_port (port_one, port_value);

} /* end main */
```

/users/dave/control.c

```
#include "utils.c"

process_port (int *port_num, int port_data)
{
static int i;
static int i2;

  for (i = 0; i <= 64; i++) {
    *port_num = port_data;
    delay ();
      {
      static int i;
      i = 3;
      port_data = port_data + i;
      }
    }
} /* end of process_port */
```

/system/project1/porthand.c

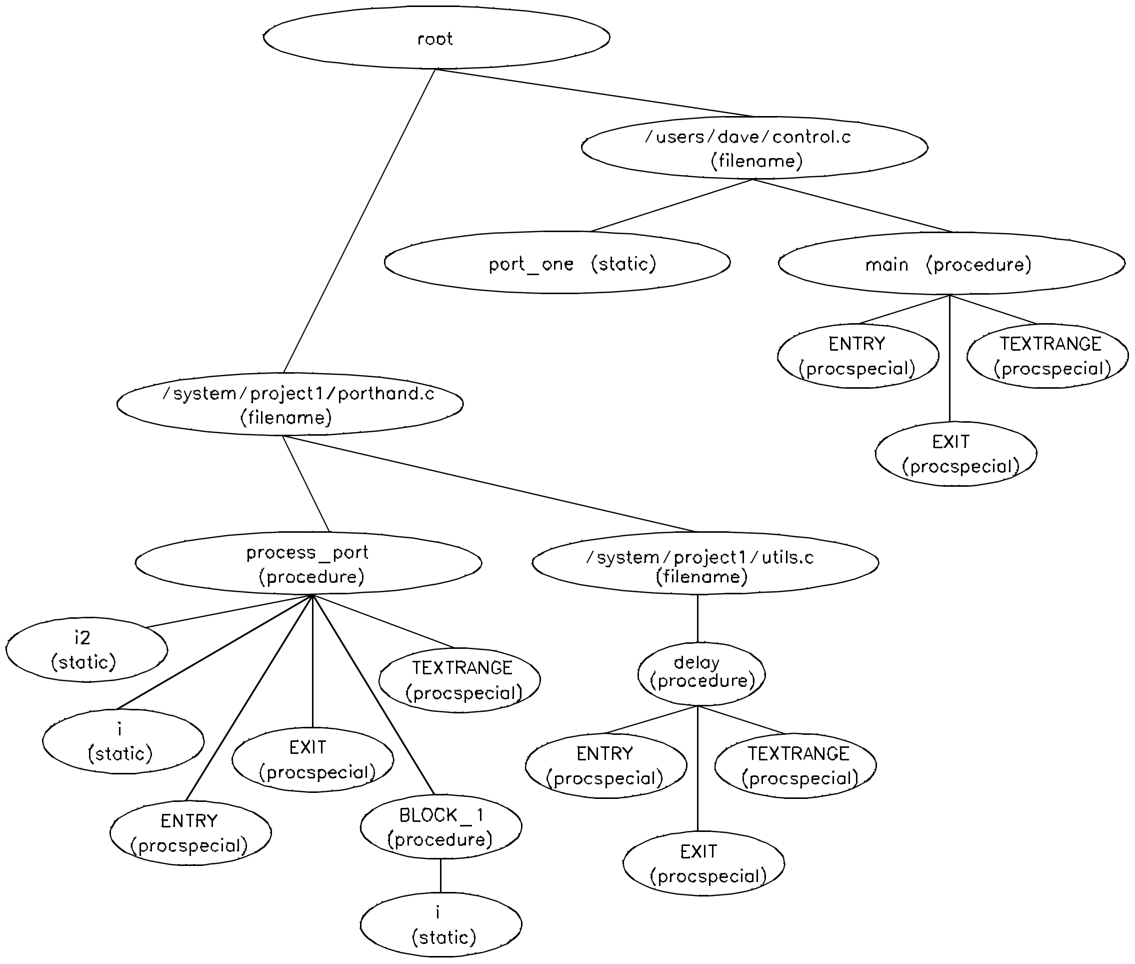
```
delay()
{
int i,j;
int waste_time;

  for (i = 0; i <= 256000; i++)
    for (j = 0; j <= 256000; j++)
      waste_time = 0;

} /* end delay */
```

/system/project1/utils.c

The symbol tree as built by SRU might appear as follows, depending on the object module format (OMF) and compiler used:



Note that SRU does not build tree nodes for variables that are dynamically allocated on the stack at run-time, such as i and j within the delay () procedure. SRU has no way of knowing where these variables will be at run time and therefore cannot build a corresponding symbol tree entry with run time address.

Here are some examples of referencing different symbols in the above programs:

```
control.c:main
control.c:port_one
porthand.c:utils.c:delay
```

The last example above only works with IEEE-695 object module format; the HP object module format does not support referencing of include files that generate program code.

```
porthand.c:process_port.i
porthand.c:process_port.BLOCK_1.i
```

Notice how you can reference different variables with matching identifiers by specifying the complete scope. You also can save typing by specifying a scope with cws. For example, if you are making many measurements involving symbols in the file porthand.c, you could specify:

```
cws porthand.c:process_port
```

Then:

```
i
BLOCK_1.i
```

are prefixed with porthand.c: process_port before the database lookup.

If a symbol search with the current working symbol prefix is unsuccessful, the last scope on the current working symbol is stripped. The symbol you specified is then retested with the modified current working symbol. Note that this does not change the actual current working symbol.

For example, if you set the current working symbol as

```
cws porthand.c:process_port.BLOCK_1
```

and made a reference to symbol i2, the retrieval utilities attempt to find a symbol called

```
porthand.c:process_port.BLOCK_1.i2
```

which would not be found. The symbol utilities would then strip BLOCK_1 from the current working symbol, yielding

```
porthand.c:process_port.i2
```

which is a valid symbol.

You also can specify the symbol type if conflicts arise. Although not shown in the tree, assume that a procedure called port_one is also defined in control.c. This would conflict with the identifier port_one which declares an integer pointer. SRU can resolve the difference. You must specify:

```
control.c:port_one(static)
```

to reference the variable, and

```
control.c:port_one(procedure)
```

to reference the procedure address.

Related Commands

`copy  local_symbols_in`

`cws`

`display  local_symbols_in`

`help` ***`symbols`***

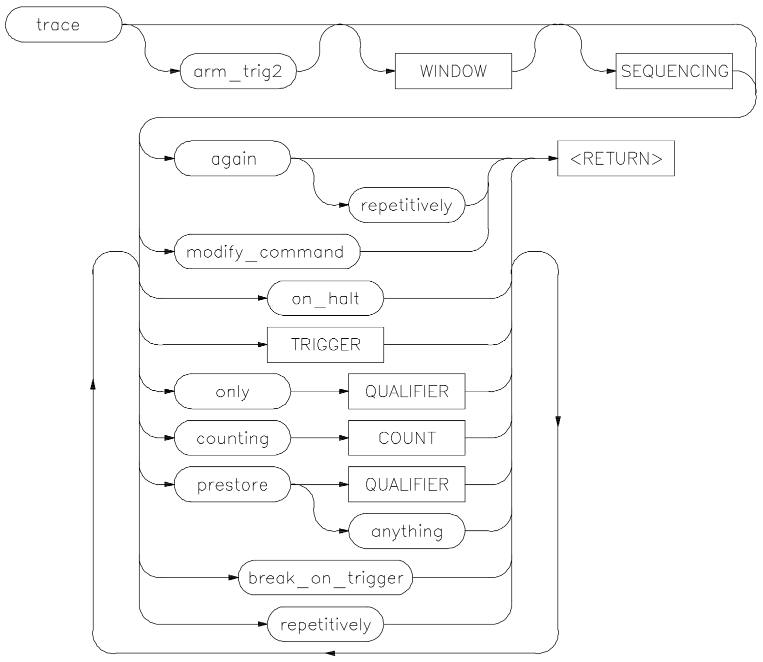
`pws`

Also refer to the *HP 64000-UX System User's Guide* for further information on symbols.

trace

This command allows you to trace program execution using the emulation analyzer.

Syntax



Note

The options shown can be executed once for each **trace** command. Refer to the TRIGGER, QUALIFIER, and COUNT diagrams for details on setting up a trace.

Function

You can perform analysis tasks either by starting a program run and then specifying the trace parameters, or by specifying the trace parameters first and then initiating the program run. Once a **trace** begins, the analyzer monitors the system busses of the emulation processor to detect the states specified in the **trace** command.

When the trace specification is satisfied and trace memory is filled, a message will appear on the status line indicating the trace is complete. You can then use display trace to display the contents of the trace memory. If a previous trace list is on screen, the current trace automatically updates the display. If the trace memory contents exceed the page size of the display, the **NEXT PAGE**, **PREV PAGE**, **up arrow**, or **down arrow** keys may be used to display all the trace memory contents. You also can press **CTRL f** and **CTRL g** to move the display left and right.

You can set up trigger and storage qualifications using the **specify trace** command. The analyzers will begin tracing when a **cmb_execute** command executes, which causes an EXECUTE signal on the Coordinated Measurement Bus.

Default Value

The analyzer will trace any state, counting time by default.

Parameters

again	This option repeats the previous trace measurement. It also begins a trace measurement with a newly loaded trace specification. (Using **trace** without the **again** parameter will start a trace with the default specification rather than the loaded specification.)
anything	This causes the analyzer to capture any type of information.
arm_trig2	This option allows you to specify the external trigger as a trace qualifier, for coordinating measurements between multiple HP 64700-Series emulators, or an

	HP 64700-Series emulator and another instrument. Before arm_trig2 can appear as an option, you must modify the emulation configuration interactive measurement specification. When doing this, you must specify that either BNC or CMBT drive trig2, and that the analyzer receive trig2. See the chapter on “Coordinated Measurements” for more information.
break_on _trigger	This stops target system program execution when the trigger is found. The emulator begins execution in the emulation monitor. When using this option, the **on_halt** option cannot be included in the command.
COUNT	This specifies whether time or state occurrences, or nothing, will be counted during the trace. See the COUNT syntax diagram for details.
counting	This option specifies whether the analyzer will count time or occurrences of states during a trace, or whether the option is to be turned off.
modify _command	This recalls the last trace command that was executed.
on_halt	When using this option, the analyzer will continue to capture states until the emulation processor halts or until a **stop_trace** command is executed. When this option is used, the **break_on_trigger**, **repetitively**, and **TRIGGER** options cannot be included in the command.
only	This option allows you to qualify the states that are stored, as defined by **QUALIFIER**.

prestore	This option instructs the analyzer to save specific states that occur prior to states that are stored (as specified with the “only” option).
QUALIFIER	This determines which of the traced states will be stored or prestored in the trace memory for display upon completion of the trace. Events can be selectively saved by using **trace only** to enter the specific events to be saved. When this is used, only the indicated states are stored in the trace memory. See the QUALIFIER syntax.
repetitively	This initiates a new trace after the results of the previous trace are displayed. The trace will continue until a **stop_trace** or a new **trace** command is issued. When using this option, you cannot use the **on_halt** option.
SEQUENCING	Allows you to specify up to seven sequence terms including the trigger. The analyzer must find each of these terms in the given order before searching for the trigger. You are limited to four sequence terms if windowing is enabled. See the **SEQUENCING** syntax pages for more details.
TRIGGER	This represents the event on the emulation bus to be used as the starting, ending, or centering event for the trace. See the **TRIGGER** syntax diagram. When using this option, you cannot include the **on_halt** option.
WINDOW	Selectively enables and disables analyzer operation based upon independent enable and disable terms. This can be used as a simple storage qualifier. Or, you may use it to further qualify complex trigger

specifications. See the **WINDOW** syntax pages for details.

Examples

trace after 1000H <RETURN>

trace only address range 1000H ***thru*** 1004H <RETURN>

trace counting state address 1004H <RETURN>

trace after address 1000H ***occurs*** 2 ***only address range*** 1000H ***thru*** 1004H ***counting time break_on_trigger*** <RETURN>

Related Commands

copy trace

display trace

help ***trace***

load trace

load trace_spec

specify trace

store trace

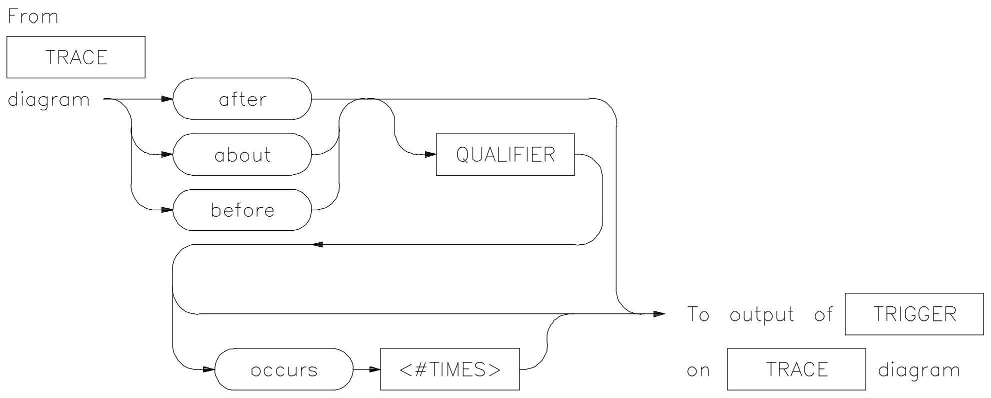
store trace_spec

Notes

TRIGGER

This parameter lets you define where the analyzer will begin tracing program information during a trace measurement.

Syntax



Function

A trigger is a QUALIFIER. When you include the **occurs** option, you can specify the trigger to be a specific number of occurrences of a QUALIFIER (see the QUALIFIER syntax diagram).

Default Value

The default is to trace after any state occurs once.

Parameters

about	This option captures trace data leading to and following the trigger qualifier. The trigger is centered in the trace listing.
after	Trace data is acquired after the trigger qualifier is found.
before	Trace data is acquired prior to the trigger qualifier.

occurs	This specifies a number of qualifier occurrences of a range or state on which the analyzer is to trigger.
QUALIFIER	This determines which of the traced states will be stored in trace memory.
< # TIMES>	This prompts you to enter a number of qualifier occurrences.

Examples

```
trace  after  MAIN  <RETURN>
trace  after  1000H  then  data  5 <RETURN>
```

Also see the **trace** command examples.

Related Commands

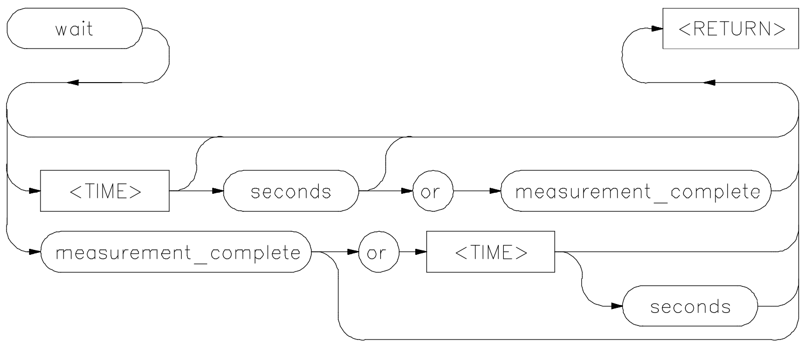
```
help  trace
trace
```

Also refer to the chapter on *Coordinated Measurements*.

wait

This command allows you to present delays to the system.

Syntax



Function

The **wait** command can be an enhancement to a command file, or to normal operation at the main emulation level. Delays allow the emulation system and target processor time to reach a certain condition or state before executing the next emulation command.

The **wait** command does not appear on the softkey labels. You must type the **wait** command into the keyboard. After you type **wait**, the command parameters will be accessible through the softkeys.

Default Value

The system will pause until it receives a **< CTRL> c** signal.

Note

If **set intr < CTRL> c** was not executed on your system, < **CRTL> c** normally defaults to the backspace key. See your HP-UX system administrator for more details regarding keyboard definitions.

Parameters

measurement _complete	This causes the system to pause until a pending measurement completes (a trace data upload process completes), or until a **< CTRL> c** signal is received. If a measurement is not in progress, the **wait** command will complete immediately.
or	This causes the system to wait for a < **CTRL> c** signal or for a pending measurement to complete. Whichever occurs first will satisfy the condition.
seconds	This causes the system to pause for a specific number of seconds.
< TIME>	This prompts you for the number of seconds to insert for the delay.

Note

A **wait** command in a command file will cause execution of the command file to pause until a **< CTRL> c** signal is received, if **< CTRL> c** is defined as the interrupt signal. Subsequent commands in the command file will not execute while the command file is paused.

You can verify whether the interrupt signal is defined as **< CTRL> c** by typing **set** at the system prompt.

Examples

```
wait  <RETURN>
wait 5; wait  measurement_complete  <RETURN>
```

Related Commands

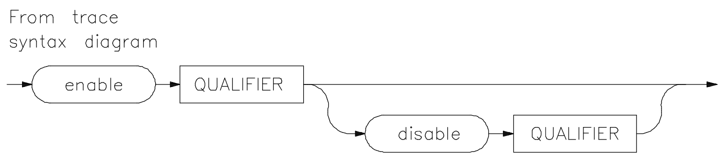
```
help  system_commands
help  wait
```

Notes

WINDOW

Lets you select which states are stored by the analyzer.

Syntax



Function

WINDOW allows you to selectively toggle analyzer operation. When enabled, the analyzer will recognize sequence terms, trigger terms, and will store states. When disabled, the analyzer is effectively off, and only looks for a particular enable term.

You specify windowing by selecting an enable qualifier term; the analyzer will trigger or store all states after this term is satisfied. If the disable term occurs after the analyzer is enabled, the analyzer will then stop storing states, and will not recognize trigger or sequence terms. You may specify only one enable term and one disable term.

Default

The analyzer defaults to recognizing all states. If you specify enable, you must supply a qualifier term. If you then specify disable, you must specify a qualifier term.

Parameters

disable	Allows you to specify the term which will stop the analyzer from recognizing states once the enable term has been found.
enable	Allows you to specify the term which will enable the analyzer to begin monitoring states.
QUALIFIER	Specifies the actual address, data, status value or range of values that cause the analyzer to enable or disable recognition of

states. Note that the enable qualifier can be different from the disable qualifier. Refer to the QUALIFIER syntax pages for further details on analyzer qualifier specification.

Examples

The following example uses the sample program from the *68000/68010 Softkey Interface User's Guide.*

The program reads a command input byte and writes one of three messages to an output area based upon the value of that byte. We want to capture only the data writes to the output message area.

display trace

trace enable `Write_Msg` ***disable*** `Read_Cmd` ***only status write***

modify `Cmd_Input` ***to*** `42h`

The display will appear as follows:

```
Trace List                 Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data               Opcode or Status              time count
Base:     hex     hex                    mnemonic                    relative
after    00044C     0000  ORI.B   #**,D0                             520     nS
+001     000602     4545    45    supr data wr byte                    2.5   uS
+002     000603     6E6E      6E  supr data wr byte                    2.8   uS
+003     000604     7474    74    supr data wr byte                    2.8   uS
+004     000605     6565      65  supr data wr byte                    2.7   uS
+005     000606     7272    72    supr data wr byte                    2.8   uS
+006     000607     6565      65  supr data wr byte                    2.8   uS
+007     000608     6464    64    supr data wr byte                    2.8   uS
+008     000609     2020      20  supr data wr byte                    2.7   uS
+009     00060A     4242    42    supr data wr byte                    2.8   uS
+010     00060B     2020      20  supr data wr byte                    2.8   uS
+011     00060C     6363    63    supr data wr byte                    2.8   uS
+012     00060D     6F6F      6F  supr data wr byte                    2.7   uS
+013     00060E     6D6D    6D    supr data wr byte                    2.8   uS
+014     00060F     6D6D      6D  supr data wr byte                    2.8   uS

STATUS:   M68000--Running user program    Emulation trace started_______........
```

Related Commands

help ***trace***

SEQUENCING

trace

QUALIFIER

Also refer to the *Analyzer Softkey Interface User's Guide.*

4

Coordinated Measurements

Introduction

The Coordinated Measurement Bus (CMB) allows coordinated measurements between multiple HP 64700-Series emulators, between HP 64700-Series and HP 64000-UX emulators, or between an HP 64700 and another instrument, using the BNC connector. The CMB has a cable and software, and depending on the configuration, additional hardware may be required (as in IMB to CMB measurements).

Coordinated measurements made between an emulator and another instrument use the BNC connector labeled TRIGGER IN/OUT on the rear panel of the HP 64700. For example, an HP 64700 analyzer can trigger or be triggered by an HP 1630 Logic Analyzer via the trigger signal on the BNC connector. Another instrument also can cause the emulator to break into the monitor by driving the BNC connector.

Target Systems with Multiple Microprocessors

For target systems that contain multiple microprocessors, multiple HP 64700-Series emulators can perform synchronous runs and emulator breaks into the monitor. HP 64700-Series analyzers can cross-trigger on the CMB.

Example Measurements

Some example coordinated measurements include:

- Two or more HP 64700 emulators start and stop executing user programs simultaneously.
- An HP 64700 analyzer triggers another HP 64700 analyzer when it finds its specified trigger.

- An HP 64700 analyzer causes another HP 64700 emulator to break into the monitor.
- An HP 64700 emulator and an HP 64000-UX emulator begin executing user programs simultaneously.
- An HP 64000-UX analyzer triggers an HP 64700 analyzer when it finds its specified trigger.
- An external analyzer causes an HP 64700 emulator to break into the monitor by driving the BNC connector.

Detailed examples are given at the end of this chapter.

Getting Started

If you plan to make coordinated measurements between multiple HP 64700-Series emulators/analyzers, you must physically connect them with one or more CMB cables. For details, refer to the *CMB User's Guide* and the *Hardware Installation And Configuration* manual for HP 64700-Series emulators. Then return here.

About the HP 64306A Interface

To make measurements between HP 64700-Series and HP 64000-UX emulators you must use the HP 64306A IMB/CMB Interface Board. The HP 64306A IMB/CMB Interface allows the Coordinated Measurement Bus (CMB) and InterModule Bus (IMB) to operate together. With the HP 64306A, you can cross-trigger HP 64000-UX and HP 64700-Series analyzers, and can coordinate the start of HP 64000-UX and HP 64700-Series emulators. You must physically connect the HP 64700 emulator(s) and the HP 64306A Interface with the CMB cable. Refer to the chapter on *Using the IMB and CMB* in the *CMB User's Guide* for HP 64700-Series emulators.

To find out more about installing the HP 64306A, refer to the *Installation Notice* supplied with that board.

Background Information on the CMB

There are three bi-directional signal lines on the CMB, and an associated BNC connector on the rear panel of the emulator. These CMB signals are:

Trigger

The CMB TRIGGER line is low true. This signal can be driven or received by any HP 64700 or HP 64306A IMB/CMB Interface connected on the CMB. This signal can be used to trigger an analyzer. It can be used as a break source for the emulator.

Ready

The CMB READY line is high true. It is an open collector and performs an ANDing of the ready state of enabled emulators on the CMB. Each emulator on the CMB releases this line when it is ready to run. This line goes true when all enabled emulators are ready to run, providing for a synchronized start.

When CMB is enabled, each emulator is required to break to background when CMB READY goes false, and will wait for CMB READY to go true before returning to the run state. When an enabled emulator breaks, it will drive the CMB READY false and will hold it false until it is ready to resume running. When an emulator is reset, it also drives CMB READY false.

Execute

The CMB EXECUTE line is low true. Any HP 64700 on the CMB can drive this line. It serves as a global interrupt and is processed by both the emulator and the analyzer. This signal causes an emulator to run from a specified address when CMB READY returns true.

Note

You must use a background emulation monitor when using the CMB READY and CMB EXECUTE signals to make coordinated measurements. Refer to chapter on *Configuring the Emulator* in your *Emulator Softkey Interface User's Guide* for more information about the emulation monitor.

BNC Trigger Signal

The BNC trigger signal is edge-sensitive. This signal can either drive or receive an analyzer trigger, or receive a break request for the emulator.

Comparison Between CMB and BNC Triggers

The CMB trigger and BNC trigger lines have the same logical purpose: to provide a means for connecting the internal trigger signals (trig1 and trig2) to external instruments. The CMB and BNC trigger lines are bi-directional. Either signal may be used directly as a break condition. Configure both by modifying the Interactive Measurement Specification using the **modify configuration** command.

The CMB trigger is level-sensitive, while the BNC trigger is edge-sensitive. The CMB trigger line puts out a true pulse following receipt of EXECUTE, despite the commands used to configure it. This pulse is internally ignored.

Note

If you use the EXECUTE function, the CMB trigger should not be used to trigger external instruments, because a false trigger will be generated when EXECUTE is activated.

Where to Find More Information

You may need to use other sources of information when setting up and starting to use your HP 64700-Series emulator with the HP 64000-UX Measurement System. The following references should help.

For details about using the Coordinated Measurement Bus, refer to the *HP 64700 Emulators Terminal Interface CMB User's Guide.*

This manual describes using the CMB separately, and with the HP 64000-UX InterModule Bus (IMB).

For additional information about using an emulator with the HP 64000-UX measurement system, refer to the *HP 64000-UX Measurement System Operating Manual.* This manual is part of the HP 64000-UX manual set for the HP 64801 operating software (which runs on the HP 9000).

You can use the HP 64808 User Interface Software with HP 64700-Series emulators. For details, refer to the *User Interface Software Operating Manual* for HP 64000-UX.

Before Continuing

Make sure that you have:

- Installed the appropriate software.
- Installed the HP 64700 emulation hardware.
- Installed the HP 64306A IMB/CMB Interface Board if you are going to make measurements between HP 64700-Series and HP 64000-UX emulators.
- Made the correct CMB connections.

Modifying the Interactive Measurement Specification

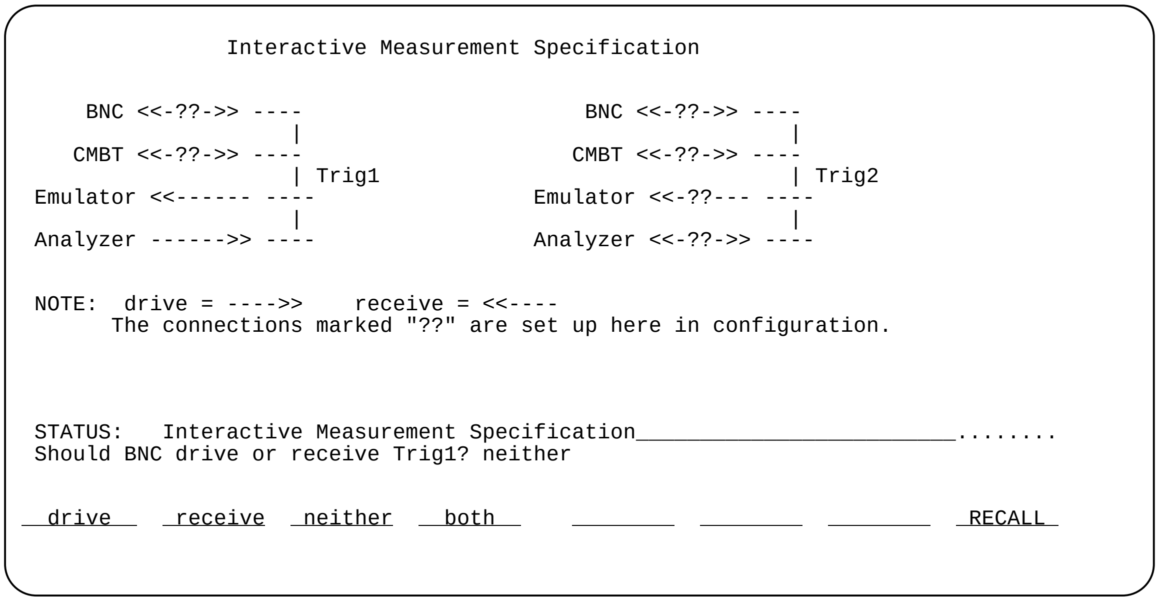
Before an HP 64700-Series emulator/analyzer can drive or receive an external trigger, you must modify the interactive measurement specification. To begin . . .

```
ENTER:  modify configuration <RETURN>
```

Now, press the < RETURN> key, until you see the following question.

Modify interactive measurement specification? no

Press the **yes** softkey to present the interactive measurement configuration questions. After answering **yes**, the information shown in figure 4-1 is displayed. It illustrates the possible connections between the internal trigger signals (trig1 and trig2) and external devices.



```
                 Interactive Measurement Specification


      BNC <<-??->> ----                      BNC <<-??->> ----
                      |                                      |
     CMBT <<-??->> ----                     CMBT <<-??->> ----
                      | Trig1                                | Trig2
 Emulator <<------ ----                 Emulator <<-??--- ----
                      |                                      |
 Analyzer ------>> ----                 Analyzer <<-??->> ----


 NOTE:  drive = ---->>    receive = <<----
       The connections marked "??" are set up here in configuration.



 STATUS:   Interactive Measurement Specification_________________________........
 Should BNC drive or receive Trig1? neither


__drive__  _receive_ _neither_ __both__   ________ ________ ________ _RECALL_
```

Figure 4-1. Interactive Measurement Specification

Notice in figure 4-1 that the analyzer always drives trig1, and the emulator always receives trig1. This provides for the **break_on_trigger** syntax of the **trace** command.

Using the Analyzer Trigger to Drive External Signals

The analyzer can be configured to drive an external trigger signal when it finds its trigger condition. Do this by setting up the analyzer to drive an internal trigger and setting up an external signal to receive it. Either or both of the internal triggers can be used.

Analyzer Drives CMB Trigger

For example, you might want the analyzer to drive the CMB TRIGGER signal when it finds its specified trigger using trig2. Set up the analyzer to drive trig2, and the CMB TRIGGER to receive it by responding to these questions as follows:

Should CMBT drive or receive Trig2? receive

Should Analyzer drive or receive Trig2? drive

All other interactive measurement questions should be answered **neither** or **no**.

Analyzer Drives BNC Trigger

Perhaps the analyzer is to drive BNC TRIGGER when it finds its specified trigger. Set up the analyzer to drive trig2 and BNC TRIGGER to receive trig2 by responding to these questions as follows:

Should BNC drive or receive Trig2? receive

Should Analyzer drive or receive Trig2? drive

All other interactive measurement questions should be answered **neither** or **no**.

Analyzer Drives Both CMB and BNC Triggers

If the analyzer is to drive both CMB TRIGGER and BNC TRIGGER using trig1, set up both the CMB TRIGGER and BNC TRIGGER to receive trig1 by responding to these questions as follows:

Should CMBT drive or receive Trig1? receive

Should BNC drive or receive Trig1? receive

Notice that you do not need to set up the analyzer to drive trig1 since it always drives trig1 as explained previously. All other interactive measurement questions should be answered **neither** or **no**.

Using External Signals to Trigger the Analyzer

The analyzer can be configured to receive its trigger from an external trigger signal. Do this by setting an external trigger signal to drive trig2 and setting the analyzer to receive it. Notice that trig2 must be used here, because the analyzer cannot receive trig1.

Using CMB Trigger

For example, if the analyzer is to be triggered by the external CMB TRIGGER signal generated by another analyzer on the CMB, set up CMB TRIGGER to drive trig2 and the analyzer to receive trig2 by responding to these questions as follows:

Should CMBT drive or receive Trig2? drive

Should Analyzer drive or receive Trig2? receive

All other interactive measurement questions should be answered **neither** or **no**.

Using BNC Trigger

If the analyzer is to be triggered by the external BNC TRIGGER signal generated by another instrument, set up the BNC TRIGGER to drive trig2 and the analyzer to receive it by responding to these questions as follows:

Should BNC drive or receive Trig2? drive

Should Analyzer drive or receive Trig2? receive

All other interactive measurement questions should be answered **neither** or **no**.

Using External Signals to Break the Emulator

Besides using the external trigger signals to trigger the analyzer, you can use these signals to cause the emulator to break into background. Do this by configuring one of the external trigger signals to drive trig1 or trig2 and having the emulator receive it.

CMB Trigger Causes Emulator to Break

For example, if you want the CMB TRIGGER signal generated by another analyzer on the CMB bus, to cause the emulator to break into background, set up the CMB TRIGGER to drive trig2 and set up the emulator to receive trig2 by responding to these questions as follows:

Should CMBT drive or receive Trig2? drive

Should Emulator break receive Trig2? yes

All other interactive measurement questions should be answered **neither** or **no**.

BNC Trigger Causes Emulator to Break

If you want the BNC TRIGGER signal to cause the emulator to break into background using trig1, set up the BNC TRIGGER to drive trig1 by responding to this question as follows:

Should BNC drive or receive Trig1? drive

All other interactive measurement questions should be answered **neither** or **no**. Notice that the emulator is already set up to receive trig1 as explained above.

Accessing the Emulator via Measurement System

You can put your HP 64700-Series emulator in a measurement system. Once you configure that emulator into a measurement system, the HP 64700-Series emulator will appear just as an HP 64000-UX emulator appears in a measurement system. If you have used HP 64000-UX emulators before, this may be a familiar process.

Before you can create a measurement system, you must run the **msinit** command to set up HP 64700 measurement system modules.

```
ENTER:  msinit <RETURN>
```

Besides searching for HP 64120 Instrumentation Cardcages, this command searches for all HP 64700-Series emulators listed in the “64700tab” file, and will set up HP 64700 measurement system modules.

Next you can configure your HP 64700-Series emulator into a measurement system. To do this . . .

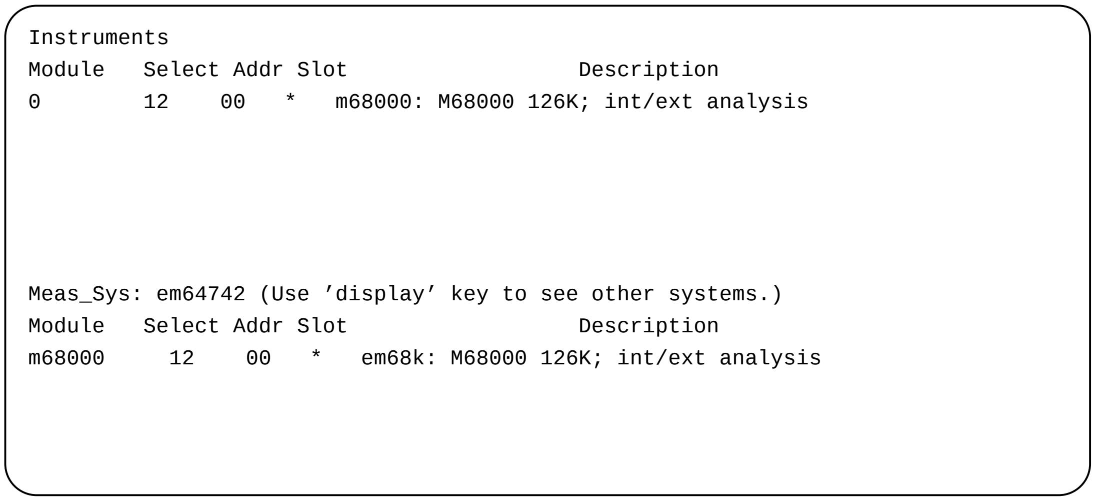
```
ENTER:  msconfig  <RETURN>
```

This command displays a list of modules that can be configured into measurement systems. The HP 64700 measurement system modules are indicated by an asterisk (*) in the “slot” field. The process of creating a measurement system is described in the *HP 64000-UX Measurement System Manual*.

Note

Each HP 64700-Series emulator must be put in a measurement system by itself. If you are using two HP 64700-Series emulators, you must create an individual measurement system for each.

After you have created a measurement system with your HP 64700-Series emulator, the “msconfig” display should be similar to figure 4-2. Your display probably will differ.



```
Instruments
Module   Select Addr Slot                  Description
0        12    00   *   m68000: M68000 126K; int/ext analysis





Meas_Sys: em64742 (Use ’display’ key to see other systems.)
Module   Select Addr Slot                  Description
m68000     12    00   *   em68k: M68000 126K; int/ext analysis
```

Figure 4-2. Creating a Measurement System

You can then access the emulator by typing the name of the measurement system followed by the module name. For example:

```
ENTER:  em64742  m68000  <RETURN>
```

Other Commands

Other measurement system commands are available. To display the status of a measurement system you can use the **msstat** command. To unlock a measurement system you can use the **msunlock** command. You may want to refer to the *HP 64000-UX Measurement System Manual* for details on how to set up a measurement system, and more about these commands.

Accessing the Emulator via the emul700 Command

Another way to access the emulator quickly and directly (without entering the measurement system) is to use the **emul700** command. The **emul700** command starts the emulator directly from the HP-UX system level.

To start the emulator using this command, for example . . .

```
ENTER:  emul700  m68000  <RETURN>
```

The emulator name (m68000) is defined in the emulator device table file named */usr/hp64000/etc/64700tab*.

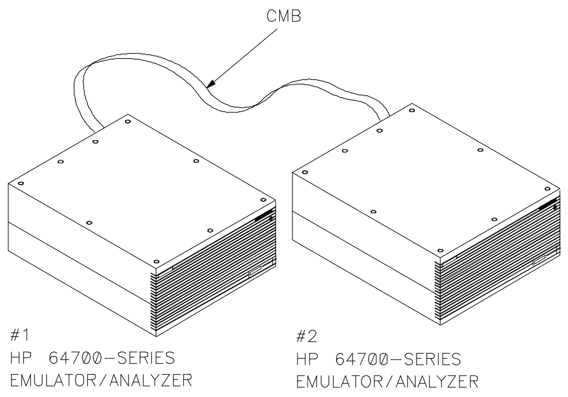
For more information about the **emul700** command, refer to chapter 7 or the on-line manual page. To use the on-line manual page, just type: **man emul700**

Example Measurements

Following are some example measurements. These may help you understand how an HP 64700-Series emulator/analyzer operates in the measurement system.

Example # 1

Two or more HP 64700 emulators start and stop executing user programs simultaneously.



This example uses two HP 64700s connected to the CMB. Each emulator should be reset or executing in the monitor, with a user program loaded into memory.

1. Using the **specify run from** command, set up each emulator to run from a specified address in the user program. This command sets up the stack and program counter, and enables the emulator to start once the CMB becomes ready and the CMB EXECUTE signal is received. The CMB will become ready after both emulators connected on the CMB are set to run.

2. To generate the CMB EXECUTE signal and synchronously start both emulators running, enter the **cmb_execute** command on either # 1 or # 2.

   Observe the status on both emulators to make sure they are running.

3. To synchronously stop both emulators, press **break** on either # 1 or # 2.

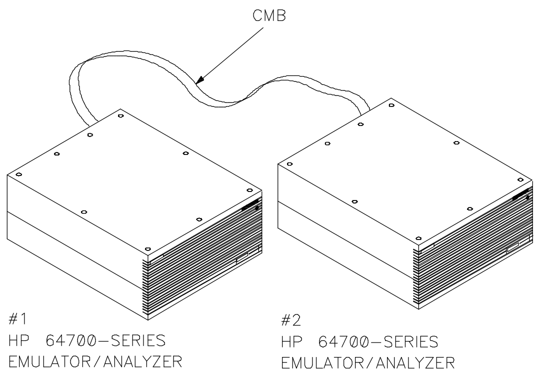
Observe the status of each emulator to see that they have stopped running. The one on which you entered **break** will be executing in the monitor, and the other will be waiting for CMB to become ready.

4. To start both emulators running again, type **run** on the emulator that is currently executing in the monitor.

   You can observe the status of each emulator to see that they are both running again.

Example # 2

An HP 64700 analyzer triggers another HP 64700 analyzer when it finds its specified trigger.



This example uses two HP 64700s connected to the CMB. Each emulator should be reset or executing in the monitor, with a user program loaded into memory. Analyzer # 1 will find its trigger condition, then trigger analyzer # 2.

Cross-triggering is done using the CMB TRIGGER (CMBT) signal, driven by analyzer # 1 and received by analyzer # 2. Though both analyzers connect through the CMB cable, neither can participate in CMB measurements until you enable them.

1. To enable analyzer # 1 to drive the CMBT signal, enter the **modify configuration** command and answer **yes** to the "Modify interactive measurement specification?" question.

The interactive measurement specification display shows two (internal) trigger terms named trig1 and trig2, which can be configured in various combinations. For this example, select CMBT to receive trig1. Because the analyzer is always set up to drive trig1, this is all you need to do to configure the analyzer to drive the CMBT via trig1. Alternatively, you could use the trig2 trigger term by selecting the analyzer to drive trig2 and CMBT to receive trig2.

2. To enable analyzer # 2 to receive CMBT, enter the **modify configuration** command and answer **yes** to the “Modify interactive measurement specification?” question.

   Select CMBT to drive trig2 and the analyzer to receive trig2. This is all you need to do to configure the analyzer to receive CMBT via trig2. Note that trig2 must be used here since the analyzer cannot be configured to receive trig1.

3. Using the **specify run from** command, set up each emulator to run from a specified address in the user program, as described earlier in example # 1.

4. Using the **specify trace** command, set up the trace specification for both analyzers. This command sets up the triggering conditions and enables the analyzer to start once the CMB EXECUTE signal is received.

   For analyzer # 1, the **specify trace** command will include a trigger condition qualified by some address in the user program. For example:

```
ENTER:  specify  trace  after  MAIN  <RETURN>
```

5. For analyzer # 2, the **specify trace** command will include a trigger condition, which is driven from trig2. For example:

```
specify trace after arm_trig2
```

6. To generate the CMB EXECUTE signal, synchronously start both emulators and analyzers by executing the **cmb_execute** command from either # 1 or # 2.

Observe the status on both emulators to make sure they are running and that the analysis traces are in progress or completed.

Once analyzer # 1 finds its trigger condition, it will drive the CMBT signal, triggering analyzer # 2. Depending on the amount of trace data stored, both traces should soon complete.

Notes

Windowing Capabilities

Using Windows

A window environment can give you a variety of views of the measurements made with your emulator/analyzer.

If you have a window environment installed on your host computer, you can operate your emulator in multiple windows on your screen. The window environment allows you to view multiple events occurring in a single emulator on a single screen.

Using Multiple Terminals

If you do not have a window environment installed on your host computer, you can still obtain the benefits of multiple windows by logging into the HP-UX system from several terminals, and starting the emulator on each terminal, just as described here for several windows.

Examples of Using Windows

You may want to start two windows so that you can observe your source program while stepping the emulation processor through the program. You also may want to use multiple windows to compare an old trace measurement with a new one.

For example, you may start the emulator in four windows such that:

1. Window # 1 shows your original program in memory.
2. Window # 2 shows the simulated I/O keyboard and display used as a “virtual” console to your target system.
3. Window # 3 shows the emulation processor stepping through your program.
4. Window # 4 shows a trace of your program execution.

Two supported window environments run on the HP 9000 host computer. These are HP Windows and X Windows. If you have either of these windowing environments installed and operating on an HP 9000 host computer, you can use your HP 64700 emulator in the windows.

You also can operate your emulator in the HP Windows-X environment. This is an X Window program that emulates HP Windows.

Window Environment Documentation

For additional information about HP Windows, refer to the *HP Windows/9000 User's Manual*. For additional details about X Windows or HP Windows-X, refer to the manual titled *Getting Started With the X Window System* or the manual titled *Using the X Window System* (if you are using the X11 windowing environment).

Maximum Number of Windows

Four is the maximum number of windows that you can use to view HP 64700 emulator/analyzer operation. You can start up more than four windows, but if you try to start the emulator in a fifth window, the system will display the following message:

```
ERROR:  No more processes may be attached to this session________........
```

Start the Window Environment and the Emulator

Suppose you have the HP Windows software installed and operating on your host computer, and want to use your emulator in that environment. To start the HP Window environment . . .

```
ENTER:  wmstart  <RETURN>
```

You will see the first window appear on screen labeled “wconsole.”

Using other Window Environments

The examples in this chapter are oriented to HP Windows. If necessary, refer to any other appropriate window documentation for details about how to start up and use another window environment on your host computer.

Start the Emulator

Start up the emulator as you would at the host computer level. For example, if you are using the 68000 emulator, and you have the emulator name defined in the */usr/hp64000/etc/64700tab* file as "em68k", this is how you would start the emulator in the first window:

```
ENTER:  emul700  em68k  <RETURN>
```

In several moments, you will see the emulation session appear in the window.

Start another Window

Start another window. To start the emulator in the new window, execute the same command as you did before.

```
ENTER:  emul700  em68k  <RETURN>
```

In several moments, you will see another emulation session appear in the new window. The status will show that this session is joining the session already in progress. The event log will be displayed in this window also.

Note

Additional windows may be added to the emulation session at any time. You do not have to add them only when starting the emulator.

Activities that Occur in the Windows

When using an HP 64700-Series emulator in a window environment (or with multiple terminals), the following activities occur in the windows where the emulator is currently operating.

Event Log is Displayed

After starting the emulator, the event log is displayed in the window. This lists all events that have occurred in the emulator since you began the emulation session.

Commands Complete in Sequence

When you execute commands that access the emulator (in multiple windows) the first command you specified will complete before the second command begins executing.

Status Line is Updated

When you perform an emulation task in one window that updates the status line, status lines are updated in all other windows where the emulator is operating. The event log is also updated in each window.

Ending the Emulation Session

When you are using the emulator in multiple windows, you can choose to either release the emulation session in a single window, or in all the windows. The **end** command by itself just ends the window where the command is executed. When you choose to end the session in all windows, control of the system returns to the host computer.

6

Using Command Files

Topics in this Chapter

This chapter describes how to create and use command files with the Softkey Interface. Topics included are:

- What are Command Files?
- How to Create Command Files
- How to Use Command Files

What are Command Files?

Command files are ASCII files created by you that contain Softkey Interface commands. They allow you to accomplish and duplicate activities without entering all the commands manually. For example, let's assume you start the Softkey Interface and want to:

1. Load an existing configuration file.
2. Load a program.
3. Set up a trace specification.
4. Run the program you loaded.
5. Display the trace results.

You can save time and keystrokes by creating a command file, and then using that command file each time you want this set of commands executed. You need only create the command file once and the bulk of your work is done.

Any Softkey Interface commands that you execute at the emulation system level can be included in a command file. Commands that you cannot include in command files are those that require you to specify parameters, such as the emulator configuration, memory map, and CMB and BNC trigger specifications. These must be stored in a configuration file.

Nesting Command Files

You can nest a maximum of eight levels of command files. Nesting command files means one command file calls another.

For More Information

The *HP 64000-UX User's Guide* contains additional details about using command files (including parameter passing).

How to Create Command Files

You can create command files by:

1. Using an editor.

2. Using the Softkey Interface **log_commands** command.

Using an Editor to Create a Command File

You can use any editor on your host computer to create a command file. Create the command file as you would any text file. When you finish, name the file “goemul,” or whatever other name you choose.

This is an example command file created using the “vi” editor:

```
load  configuration  /yourproject/config4
load  /yourproject/program4
trace  after  START
run  from  2000h
wait 10
display  trace
```

Logging Commands to Create a Command File

Rather than using an editor to create a command file, you can use the **log_commands** command. This allows you to record all commands that you execute.

Logging both commands and output can be helpful when executing a set of commands that you are not sure will produce the results you are seeking. By logging commands that you type, you can record everything you try. By logging the resulting output, you can verify that the expected results occurred. When you have finished logging commands, you can use the “log” file as a command file.

Note

You do not have to modify the “log” file to use it as a command file, because all commands and output are stored in a format that the Softkey Interface can read when you load the file. You may want to edit the log file to remove any unwanted commands or results, or add any other commands or comments.

If a file exists with the same name in the present working directory, the **log_commands** command will try to overwrite the existing file.

Using the wait Command

You can use the **wait** command in command files. This allows you to pause execution of the command file between commands.

Use the **wait measurement_complete** command after changing the trace depth. By doing this, when you copy or display the trace after changing the trace depth, the new trace states will be available. Otherwise the new states won’t be available.

How to Use Command Files

Suppose you want to use the command file you created in this chapter. Enter the Softkey Interface. Then just enter the command file name. For example:

```
ENTER:  goemul  <RETURN>
```

All the commands contained in the command file named “goemul” would execute in sequence.

You also can start the Softkey Interface and immediately begin a command file. There are two ways to do this.

```
emul700 <emul_name> < <cmd_file>
```

starts the interface and redirects input from the command file to the interface.

```
echo ’<cmd_file>’ | emul700 <emul_name>
```

starts the softkey interface, which then starts the command file.

7

Manual Pages

This chapter contains manual pages for some relevant HP 64700-Series emulator commands and files.

Commands

This command is described:

emul700

Files

This file is described:

64700tab

Notes

emul700

The **emul700** command starts an emulation session using an HP 64700-Series emulation pod.

Synopsis

emul700 [-d] [-u userinterface] logicalname

emul700 -l [-v] [logicalname [logicalname . . .]]

emul700 -U [-v] logicalname

Description

The **emul700** command starts a user interface that controls an HP 64700-Series emulator. It may be used to obtain information about one or more HP 64700-Series emulators, or may be used to unlock an emulator locked by a previous user.

The command arguments and options are:

logicalname	This is a logical name defined in the “64700tab” file. These names uniquely identify a specific emulator.
-d	This option defaults the emulator configuration if you are starting a session. This option has no effect if joining a session already in progress.
-l	This option lists the status of the emulators defined in the “64700tab” file, or the emulator(s) specified by the last argument(s) to the **emul700** command.
-u userinterface	This option selects the user interface to control the requested emulator. To find which user interfaces are available for one or more emulators, use the **-v** option with the **emul700 -l** command. The default user interface, when more than one is available, is **skemul**, the softkey-driven user interface for the emulator.

-v	This option reports actions and/or information verbosely with the **-l** or **-U** option.
-U	This option unlocks the emulator(s) specified by the last argument(s) to **emul700**, if there is no current session in progress.

Related Files

/usr/hp64000/etc/64700tab

This emulator device table file associates a name for an HP 64700-Series emulator (its logical name) with the actual I/O device used to communicate with that emulator. Channel attributes such as baud rate and protocol are also specified in this file.

/usr/hp64000/inst/emul/< productID>

This directory contains user interface support for specific HP 64700-Series emulators. The < productID> represents the directory for your product.

See Also

64700tab(4)

64700tab

The “64700tab” file describes the format of HP 64700-Series emulator information.

Description

The “64700tab” file lists the Hewlett-Packard 64700-Series emulators attached (or that may be attached from time to time) to this HP-UX host.

The file is originally installed as:

/usr/hp64000/etc/newconfig/64700tab

The first installation also copies this file to:

/usr/hp64000/etc/64700tab

This file must be modified by the system administrator or other knowledgeable person. It must include the information described below for each HP 64700-Series emulator that may be attached to the host. After modification, the file should remain at:

/usr/hp64000/etc/64700tab

This file must be a Configuration Dependent File if this host is a node in an HP-UX diskless cluster. (The installation process should take care of this. See also **makecdf(1M)**).

This file is the reference used by HP 64000-UX software when an HP 64700-Series emulator is accessed. Each HP 64700-Series emulator connects to an HP-UX host via its own dedicated serial interface channel. This file allows you to define a unique name for each emulator, its logical name, the channel to which it connects, and the channel settings used to communicate with the emulator. Each uncommented line in the “64700tab” file describes one emulator. Each line has 9 fields separated by blanks or tabs:

logical name	This is the name associated with this emulator/serial channel combination. Each logical name must conform to the following syntax (using the notation of regular expressions): [a-zA-Z][a-zA-Z0-9_]*

For example, “emul68k” and “emul_68k” are valid names, whereas “68000” and “emul-68k” are not.

Each logical name in this file must be unique. The logical name also must be unique among the module names defined in **msconfig** on this host. Only the first 14 characters in logical name are used.

processor type	This represents the type of processor in the emulator. This field is used to locate all emulators supporting a specific type of processor.
physical device	The name of the device file controlling the serial I/O channel connected to the emulator (for example, /dev/emcom15). Names such as /dev/tty15 should be avoided since some system administration tools (like reconfig) rely on the prefix “tty” to automatically configure login ports. The device file should have read/write permission for all users who will be using the emulator (crw-rw-rw-, for example). There should **not** be a getty running on the channel. Each emulator should have its own dedicated channel. If a channel must be shared among several emulators, only one session for one emulator may be active at any time.
xpar mode	HP 64700-Series emulation pods support a “transparent mode” of operation on the two serial ports built into the pod. This mode is not supported in HP 64000-UX software. The only setting allowed for this field is OFF.

baud rate	This is the serial channel baud rate. Supported values are: 1200 2400 9600 19200 57600 230400
parity	This refers to the parity mode in use on the serial channel. The only valid setting is NONE.
flow	This enables or disables flow (pacing) control in the serial channel. Two protocols are supported: XON/XOFF and RTS/CTS. The difference between these is that XON/XOFF uses characters in the data stream for pacing, whereas RTS/CTS uses separate signals. Valid settings for flow are: XON RTS

Caution

RTS/CTS is preferred over XON/XOFF because it is a more reliable means of pacing. It is supported for only the HP 98659 card.

Without some form of pacing or flow control, overrun may occur during non-binary transfers from the HP 64700-Series emulation pod to the HP-UX host. (Binary transfers, such as uploading trace data or emulation memory, have a record protocol with built-in flow control.)

stop bits	This is the number of stop bits used in the serial channel protocol. The only supported setting is 2.

char size	This option refers to the number of bits used for each character (or byte) transferred on the serial channel. The only supported size is 8.

Note

The switches on the rear panel of the HP 64700-Series emulator must match the settings for baud rate, parity, flow, stop bits, and char size when the emulator is turned on. You should not change these settings without cycling power on the emulator pod.

The file */usr/hp64000/etc/64700tab* must be kept up to date as HP 64700-Series emulators are added to (or removed from) the host. The contents of this file are the reference data for all HP 64000-UX software that uses the emulators.

Example The following is an example of the relevant portion of a completed “64700tab” file:

```
#---------------------------------------------------------------------------------
#                                         xpar    baud    parity flow   stop  char
#logical name   processor    physical     mode    rate                  bits  size
# (14 chars)      type        device                               XON
#                                         OFF              NONE    RTS    2     8
#---------------------------------------------------------------------------------
#exampleRS232   m68000     /dev/emcom15   OFF    19200    NONE    XON    2     8
#exampleRS422   i80186     /dev/emcom23   OFF    230400   NONE    RTS    2     8
#
 emul68k        m68000     /dev/emcom14   OFF    9600     NONE    XON    2     8
 scarecrow      z80        /dev/emcom25   OFF    230400   NONE    RTS    2     8
```

Hardware Dependencies The only interface cards characterized for support by HP 64000-UX software are the HP 98628 RS-232, HP 98642

RS-232 MUX, and HP 98659 RS-422 cards. Of these, the HP 98659A is the only interface card recommended for connection to HP 64700-Series emulators. (See the table on the next page.)

Caution

The other two interface cards (HP 98628 and HP 98642) are designed primarily for host-to-peripheral transfers. They are less capable for the large peripheral-to-host transfers required by HP 64700-Series emulators. You may see dropped characters or other communications failures with these cards. If these occur, you may need to press **CTRL c** (send a SIGINT signal) in the user interface to recover control. Some information may be lost.

Each card has limits to the configurations that are supported. The following table contains an “x” for the supported fields, and a “.” for the unsupported fields. For example, the HP 98628 card will support operation at 19200 baud, but not at 57600 baud.

The following table shows the compatibility of these cards with the available options.

Fields in 64700tab File	**Supported Interface Cards**		
	HP 98659	**HP 98628**	**HP 98642**
baud rate:			
1200	x	x	x
2400	x	x	x
9600	x	x (2)	x (2) (3)
19200	x	x (2)	x (2) (3)
57600	x	.	.
230400	x	.	.
parity:			
NONE	x	x	x
flow:			
XON	x	x	x
RTS	x	.	.

Fields in 64700tab File	**Supported Interface Cards**		
	HP 98659	**HP 98628**	**HP 98642**
stop bits: **2**	x	x	x
char size: **8**	x	x	x
(1) These baud rates must use the RTS/CTS pacing protocol. (2) The HP 98628 and HP 98642 cards may have an unacceptable error rate in some applications. Reducing the baud rate and/or the load on the host computer may help. (3) The HP 98642 card has four ports. The “modem” port has the lowest error rate when used with an HP 64700 emulator. Port 3 has the highest error rate.			

Related Files

/usr/hp64000/etc/newconfig/64700tab

This is the template file shipped with the software for an HP 64700-Series emulator. This file must be modified and installed as:

/usr/hp64000/etc/64700tab

/usr/hp64000/etc/64700tab

This is the customized version of the original “64700tab” file. The contents of this file should reflect the device(s) and settings used to communicate with HP 64700-Series emulator(s) on this host. If this host is configured as a node in an HP-UX diskless cluster, this file must be a Configuration Dependent File.

See Also

emul700(1), **makecdf(1M)**

A

Performance Verification for the HP 98659A

Introduction

The HP 98659A High-Speed RS-422 Interface has two levels of performance verification (PV). The first level is a series of built-in ROM-based power up tests which perform a checksum on the ROM, test the on-board RAM, and test other local resources. These tests are the same as those embedded in the other programmable datacomm interface products. Other programmable datacomm interfaces include the HP 98628, HP 98629, HP 98641, and HP 98649.

The second level is an HP-UX based test, which is unique for this interface. It exercises the drivers and receivers and provides data integrity testing.

If Powerup Tests Fail

If any of the powerup tests fail, the bootrom initialization in the HP 9000 Series 300 during powerup or rebooting of the HP-UX system will display the following message:

```
HP98659 at "sc" Failed
```

“sc” is the select code of the interface.

During the HP-UX booting process, if the powerup tests fail, the error message will read:

```
HP98691 at select code "sc" ignored; unrecognized card option
```

This error message occurs after the powerup test fails, because the kernel.s procedure (**data_com_type**) returns a value of -1 instead of the protocol ID of the board. The HP 98691 message comes from the kernel.s procedure (**last_make_entry**). These failures prevent the driver from being linked, and therefore the HP-UX based tests cannot be run. The failure mode is available on the card but there is no present way under HP-UX to access it.

The second level of HP-UX based performance verification can be run as a field verification test. The field verification test is described in this appendix. The HP-UX based tests check the external interface logic and the user cable, if connected. It will use a test hood to feed back all used signals for verification of both drivers and receivers.

Customer/Field Test Hood Requirements

Table A-1 shows the interconnections in the test hood used with the HP 98659A High-Speed RS-422 Interface. This test hood is for customer and Hewlett-Packard Customer Engineer (CE) testing.

Part Numbers

The part number of the test hood is 98659-67950. The test hood has a 25-pin connector, which plugs into the cable assembly (part number 98659-63001) that connects the interface card to the emulator.

Table A-1. Customer/Field Test Hood Requirements

Signal Type	Signal Definition	Pin Direction	Data Flow

Data	SD(A) --> RD(A)	9 --> 25	SD= Send Data (from cable)
			RD= Receive Data (to cable)

	SD(B) --> RD(B)	10 --> 18	

Control	RS(A) --> CS(A)	17 --> 21	RS= Request to Send (from cable)
			CS= Clear to Send (to cable)

	RS(B) --> CS(B)	19 --> 23	

	DM(A),RR(A) < -- TR	12 < -- 20	DM= Data Mode, RR= Receiver Ready (to cable)
			TR= Terminal Ready (from cable)
			SG= Signal Ground

	DM(B),RR(B) < -- SG	13 < -- 7	

Timing	TT(A) --> RT(A),ST(A)	24 --> 11	TT= Terminal Timing (from cable)
			RT= Receive Timing, ST= Send Timing (to cable)

	TT(B) --> RT(B),ST(B)	14 --> 16	

Note: Some pins in the connector have two names (representing functions). For example, pin 12 serves as both Data Mode and Receiver Ready.

Invocation of the HP-UX based PV is through an executable file named `/usr/CE.utilities/98659/98659pv`. Executing the program without any options displays the following:

```
HP 98659A Performance Verification Test:
Usage
98659pv /dev/xxx [loop]
"/dev/xxx" is the special device file for the HP 98659A card
under test.  "loop" is an optional request to loop on a
portion of the test.  Adding a loop option provides a
regular stimulus response which is useful for repair of
a faulty interface card. "DTR" or "DSR" will toggle the DTR
output/DSR input.  "DATA" will loop at 19200 baud with
no handshake, toggling the data lines.  "RTS" or "CTS"
will toggle the RTS/CTS Handshake, in a 230400 baud data loop.
A CTRL C will exit from any of the loops.
```

Test without Looping

If the test starts with a valid special device file, no looping options, and a working card (and cable if connected), the following is displayed:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for /dev/ody25, fildes = 3
Passed DTR Toggle Test
sending a 5K byte file out at 19200 baud,
No pacing enabled, checking data out/in lines
returned string same as send string
Passed Data Transfer Test at 19200 Baud
sending a 5K byte file out at 460800 baud,
RTS/CTS pacing enabled, checking clock and RTS/CTS lines
returned string same as send string
Passed Data Transfer Test at 460800 Baud
Passed EXTERNAL CLOCK and RTS/CTS check
Driver closed for /dev/ody25
98659A Performance Verification PASSED for device /dev/ody25
```

If the test starts with an invalid special device file, the following is displayed:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Failure to OPEN
errno = 20
ERROR EXIT
```

If you specify an invalid looping option, the following is displayed:

```
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Unsupported Option: xxx
ERROR EXIT
```

“xxx” represents the invalid option entered.

If any failures occur during these tests, an error message is printed and the test terminates. The fault can be in either the card or the cable.

Error Messages

Various failures display the following error messages:

DTR or DSR failure

The DTR or DSR error occurs when DTR or DSR will not go true. The result on screen resembles:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for  /dev/ody25,  fildes = 4
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 DTR will not go true (false)
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Failure in modem line test
errno = 25
ERROR EXIT
```

Data Read Failure

The Data Read error occurs when the test fails to read the data. The result on screen resembles:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for /dev/ody25, fildes = 3
Passed DTR Toggle Test
sending a 5K byte file out at 19200 baud,
No pacing enabled, checking data out/in lines
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Failure to read
errno = 5
ERROR EXIT
```

Data Character Loss Failure

The Data Character Loss Failure error occurs when the test fails to read all the data. The result on screen resembles:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for /dev/ody25, fildes = 3
Passed DTR Toggle Test
sending a 5K byte file out at 19200 baud,
No pacing enabled, checking data out/in lines
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Failure to read all of data
ERROR EXIT
```

Data Corruption Failure

The Data Corruption Failure error occurs when the test returns a string that differs from the string sent. The result on screen resembles:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for /dev/ody25, fildes = 3
Passed DTR Toggle Test
sending a 5K byte file out at 19200 baud,
No pacing enabled, checking data out/in lines
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
return string not same as sent string
ERROR EXIT
```

RTS/CTS and Clock Test Failure

The RTS/CTS and Clock Test Failure error occurs when the test fails to read the data when RTS/CTS pacing is enabled. The result on screen resembles:

```
returned string same as send string
Passed Data Transfer Test at 19200 Baud
sending a 5K byte file out at 460800 baud,
RTS/CTS pacing enabled, checking clock and RTS/CTS lines
ERROR!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Failure to read
errno = 5
ERROR EXIT
```

Looping Options

If you specify the “DTR” or “DSR” option, the following is displayed:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for /dev/ody25, fildes = 3
toggling DTR out, DSR in
```

The lines will be toggling so that any open or short circuits in the path can be located.

If you specify the “DATA” option, the following is displayed:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for /dev/ody25, fildes = 3
Passed DTR Toggle Test
Loop is sending out 5K binary 8 bit file
Reading back as much as possible
```

This will allow the data path to be traced so that any problems in the data loop can be isolated.

If invoked with the “RTS” or “CTS” option, the following is displayed:

```
98659A Performance Verification Test:
Test Requires Test Hood # 98659-67950
Driver opened for /dev/ody25, fildes = 3
Passed DTR Toggle Test
sending a 5K byte file out at 19200 baud,
No pacing enabled, checking data out/in lines
returned string same as send string
Passed Data Transfer Test at 19200 Baud
Loop is sending out 5K binary 8 bit file
at 460800 baud using RTS/CTS handshaking and
the external X1 clock
Reading back as much as possible
Failures could be in data, clock, or RTS/CTS
```

This provides stimulus for checking the clock path and the RTS output to CTS input path.

These tests, when used without a looping option, provide a simple yes/no test for use in field testing. With the addition of the looping option, they also provide a tool for repair.

Additional Information

For additional information about the HP 98659A, refer to the *Installation Guide* supplied with that card. Also, you can refer to the *Installation Notice* for the HP 64700-Series Emulators Softkey Interface, supplied with your Emulator Softkey Interface documentation.

Notes

Error Messages

Introduction

This appendix contains a list of error messages that may occur while operating your HP 64700-Series Emulator.

Messages Recorded in Error Log

The *error log* records error messages received during the emulation session. You may want to display the error log to view the error messages. Sometimes several messages will be displayed for a single error to help you locate a problem quickly. To prevent overrun, the error log purges the oldest messages to make room for the new ones.

To display the error log. . .

```
ENTER:  display  error_log  <RETURN>
```

Terminal Interface Error Messages

Terminal Interface error messages may be displayed. If you don't find the error message in this appendix, try locating it in the *Terminal Interface Reference* or *Emulator Terminal Interface User's Guide*. These messages have designated numbers under 10000.

Organization of the Messages

The Softkey Interface Error Messages are listed in alphabetical order in groups according to the type of task you are performing.

Groups of Messages

These are the groups of error messages listed in alphabetical order:

- Analyzer Usage
- Configuration File Building
- Display Workarea

- Emulator Grammar Usage
- Emulator Initialization
- emul700dmn Communications
- Ending the Emulation Session
- Expression Usage
- Fatal to the Emulation Session
- Initialize/Load/Modify Emulation Configuration
- Inverse Assembler Initialization
- Load/Store Absolute File
- Memory Display
- Miscellaneous
- Miscellaneous Numbered
- Performance Measurement
- Processor Control
- Symbol Usage
- Trace Display/Load

Analyzer Usage

These messages can occur while using the emulation analyzer.

Messages

Emulation analyzer defaulted to delete label

Slave clock requires at least one edge

Configuration File Building

These messages can occur while the host system is building a configuration file.

Messages

Configuration process caught SIGQUIT

Could not create < CONFIGURATION BINARY FILENAME>

Could not exec configuration process

Could not fork configuration process

Error in configuration process

Error starting configuration process

Insufficient emulation memory, memory map may be incomplete

Invalid answer in < CONFIGURATION FILENAME> ignored

Module file missing or invalid, end and run msinit

Question file missing or invalid

Display Workarea

This message can occur when a window cannot be opened.

Message Can’t open a display workarea

Emulator Grammar Usage

These messages can occur while specifying commands.

Messages

Illegal status combination

Illegal symbol name

Inverse assembly not available

No address label defined

Number of lines not in range: 1 < = valid lines < = 50

Number of spaces not in range: 2 < = valid spaces < = 15

Performance tool must be initialized

Performance tool not initialized

Warning: at least one integer truncated to 32 bits

Width not in range: 1 < = valid width < = 80

Emulator Initialization

These messages can occur during emulator initialization.

Messages

Cannot create module file:

Cannot start. Ending previous session, try again

Cannot start. Pod initialization failed

Connecting to < LOGICAL NAME>

Continue load failed

Continuing previous session, continue file loaded

Continuing previous session, user interface defaulted

Could not create default configuration

Could not create new default configuration

Could not initialize with default config

Could not initialize with new default config

Could not load default configuration

Could not load new default configuration

Emul700dmn continuation failed

Error: display size is < LINES> lines by < COLUMNS> columns.
It must be at least 24 by 80.

Initialization failed

Initialization load failed

Initializing emulator with default configuration

Initializing user interface with default config file

Joining session already in progress, continue file loaded

Joining session already in progress, user interface defaulted

Measurement system not found

Memory block list unreadable

No more processes may be attached to this session

Starting new session, continue file loaded

Starting new session, user interface defaulted

unknown hardware id: < HARDWARE ID>

emul700dmn Communications

These messages can occur when the host system cannot communicate properly with the HP 64700 emulation daemon.

Messages

Emul700dmn failed to start

Emul700dmn message too large

Emul700dmn message too small

Emul700dmn queue and/or semaphores missing

Emul700dmn queue failure

Emul700dmn queue full

Timeout in emul700dmn communication

Unexpected message from emul700dmn

Note

The messages listed above are all fatal to the emulation session and require you to press **end_release_system**. You must exit this emulation session completely, then start a new session.

Emul700dmn executable not found

Emul700dmn sem op failed, perhaps kernel limits too low

Emul700dmn version incompatible with this product

HP 64700 I/O error; communications timeout

Note

The messages listed above are encountered when starting up the emulation session.

Ending the Emulation Session

These messages can occur while ending the emulation session.

Messages

Ending released

Fatal error from function < ADDRESS OF FUNCTION>

< LOGICAL NAME> : End, continuing

< LOGICAL NAME> : End, released

Memory allocation failed, ending released

Session aborted

Session cannot be continued, ending released

Status unknown, run "emul700 -l < LOGICAL NAME> "

Expression Usage

These messages can occur while defining expressions.

Messages

Don’t care number unexpected

Unknown expression type

Fatal to the Emulation Session

The following messages appear due to fatal errors in the emulation session. If you cannot recover from any one of these error messages, call your HP Response Center or HP Representative to answer any questions.

Recovery Action

Recovery action for these error messages is the same: you must end the current emulation session by pressing the **end_release_system** softkey to exit the emulation session completely. Then you must start a new session.

Messages

Cannot start. Ending previous session, try again

Cannot start. Pod initialization failed

Emul700dmn continuation failed

Emul700dmn error in file operation

Emul700dmn executable not found

Emul700dmn failed to start

Emul700dmn message too large

Emul700dmn message too small

Emul700dmn parameter unknown

Emul700dmn queue and/or semaphores missing

Emul700dmn queue failure

Emul700dmn queue full

Emul700dmn sem op failed, perhaps kernel limits too low

Emul700dmn version incompatible with this product

Measurement system not found

Memory allocation fault

No error

No more processes may be attached to this session

Session aborted

Timeout in emul700dmn communication

Unexpected message from emul700dmn

64700 command failed

64700 error

Initialize/Load/ Modify Emulation Configuration

These messages can occur while the host system is initializing, loading, or modifying the emulation configuration.

Messages

Cannot build < CONFIGURATION FILENAME>

Cannot modify < CURRENT CONFIGURATION FILENAME>

< CONFIGURATION FILENAME> does not exist

Configuration not valid, restoring previous configuration

Configuration update failed, previous one restored

Inverse Assembler Initialization

These messages can occur while initializing the inverse assembler.

Messages

Inverse assembly file < INVERSE ASSEMBLER FILENAME> could not be loaded

Inverse assembly file < INVERSE ASSEMBLER FILENAME> not found, < filename>

Warning: No inverse assembly file specified

Load/Store Absolute File

These messages can occur while loading or storing absolute files.

Messages

File could not be opened

File is not executable or absolute - load aborted

File name too long, truncated to: < BASENAME OF ABSOLUTE FILE NAME>

Load aborted

Load completed with errors

Position must be -500 through 500

Read memory failed at < PHYSICAL ADDRESS> - store aborted

Store aborted

Memory Display

These messages can occur when controlling memory.

Messages

Address range too small for request - request truncated

Memory range overflow

opcode extends beyond specified address range

Range crosses segment boundary

Starting address greater than ending address

Miscellaneous

This message can occur if the host system cannot determine the current time.

Message

Wait time failure, could not determine system time

Miscellaneous Numbered

These numbered messages can occur because of various problems with the emulator/analyzer.

Messages

10315 Logical emulator name unknown; not found in 64700tab file

10316 < FILENAME> file error; open failed

< FILENAME> file error; permission denied

< FILENAME> file error; read failed

< FILENAME> file error; incompatible file format

< FILENAME> file error; invalid %s on line %s

< FILENAME> file error; duplicate logical name \(%s\) on line %s

< FILENAME> file error; expected type %s on line %s

< FILENAME> = "64700tab" or "lockinfo"

10326 Emulator locked by another user

10327 Cannot lock emulator; failure in obtaining the accessid

Cannot lock emulator; failure in < ERRNO MSG>

10328 Cannot unlock emulator; emulator not locked

Cannot unlock emulator; lock file missing

Cannot unlock emulator; semaphore missing

Cannot unlock emulator; emulator in use by user:
< USER NAME>

10329 Emulator locked by user: < USER NAME>

10330 Emulator locked by another user interface

10331 HP64700 I/O channel in use by emulator:
< LOGICAL NAME>

10332 Cannot default emulator; already in use

10340 HP64700 I/O channel semaphore failure:
< SEMOP ERROR>

Cannot open HP64700 I/O channel; channel already open

Cannot open HP64700 I/O channel; channel does not exist

Cannot open HP64700 I/O channel; RS232 RTS/CTS unsupported

Cannot open HP64700 I/O channel; Cannot stat HP64700 I/O channel; < DEVICE NAME>

Cannot close HP64700 I/O channel;
< ERRNO MSG>

HP64700 I/O error; communications timed out

HP64700 I/O error; channel not open

HP64700 I/O error; power down detected

HP64700 I/O error; invalid channel name

HP64700 I/O error; channel locking aborted

Note

The messages listed above are all fatal to the emulation session and require you to press **end_release_system**. You must exit this emulation session completely, then start a new session.

10350 Cannot interpret emulator output

10351 Exceeded maximum 64700 command line length

10352 Incompatible with 64700 firmware version

10353 Unexpected emulator %s, expected %s

10360 Analyzer limitation; all range resources in use

Analyzer limitation; all pattern resources in use

Analyzer limitation; all expression resources in use

Performance Measurement

These messages can occur while using the Software Performance Measurement Tool (SPMT).

Messages

File could not be opened

File perf.out does not exists

File perf.out not generated by measurement software

No address(es) entered

Perfinit - Absolute file (database) must be loaded line < LINE NUMBER>

Perfinit - error in input file line < LINE NUMBER> invalid symbol

Perfinit - error in input file line < NUMBER>

Perfinit < ---EXPR--- ERROR> line < LINE NUMBER>

Perfinit - File could not be opened

Perfinit - No events in file

perf.out file could not be opened - created

Processing trace complete

Symbols not accessible, symbol database not loaded

Processor Control

These messages can occur while you are controlling the emulation processor.

Messages

No symbols loaded

< PROCESSOR NAME> --Stepping aborted; number steps completed: < STEPS TAKEN>

< PROCESSOR NAME> --Stepping complete

Step count must be 1 through 999

Stepping aborted; number steps completed: < STEPS TAKEN>

Symbol Usage

These messages can occur while specifying symbols.

Messages

Rebuilding symbol data base

Symbols not accessible, Symbol database not loaded

Unknown type of scope

Warning: Can’t build symbol data base

Trace Display/Load

These messages can occur while displaying a trace or loading information from a trace file.

Messages

File could not be opened

Inverse assembly not available

No valid trace data

Not a valid trace file - load aborted

Not compatible trace file - load aborted

Software break: < PHYSICAL ADDRESS>

Symbols not accessible; Symbol database not loaded

Trace file not found

Unload trace data failed

Notes

Index

Notes