HP 64000-UX
CASE Solutions for Microprocessors

HP 64700-Series Analyzer

Softkey Interface User's Guide

HEWLETT PACKARD

HP Part No. 64700-97005
Printed in U.S.A.
September 1992

Edition 4

Notice

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

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

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

Printing History

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

A software code may be printed before the date; this indicates the version level of the software product at the time the manual was issued. Many product updates and fixes do not require manual changes and, conversely, 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	64740-90910 E1188, November 1988
Edition 2	64740-97002, August 1989
Edition 3	64700-97001, February 1990
Edition 4	**64700-97005, September 1992**

Using this Manual

This manual will show you how to use the HP 64700-Series analyzer with the host computer Softkey Interface.

This manual will:

- Briefly introduce the analyzer and its features.
- Show you how to use the analyzer in its simplest, power-up condition. From there, it will progressively show you how and why you would use additional trace commands.
- Show you how to use the trace display options.
- Show you how to connect the external analyzer probe to target system signals and how to configure and use the external analyzer.
- Show you how to set up the analyzer trigger to break the emulator into the monitor program.
- Show you how to drive external Coordinated Measurement Bus (CMB) or BNC trigger signals with the analyzer trigger.
- Show you how to use the Timing Analyzer Softkey Interface.

This manual will not:

- Show you how to use every Softkey Interface command and option; the Softkey Interface is described in the *Softkey Interface Reference.*
- Show you how to use coordinate measurements between multiple emulators; specifics on background, specifications and use are described in the "Coordinated

Measurements" chapter of the *HP64700-Series Emulators Softkey Interface Reference*.

Organization

Chapter 1 **Introducing the Analyzer**. This chapter lists the basic features of the analyzer. The following chapters show you how to use these features.

Chapter 2 **Getting Started**. This chapter shows you how to use the analyzer from its simplest power-up condition to specifying trigger conditions, storage qualifiers, prestore qualifiers, and count qualifiers.

Chapter 3 **Displaying Traces**. This chapter describes options available when displaying the trace.

Chapter 4 **Making Software Performance Measurements**. This chapter describes software performance measurements, describes the steps in making measurements with the Software Performance Measurement Tool (SPMT), and shows you example measurements made on the demo program.

Chapter 5 **Using the External Analyzer**. This chapter shows you how to connect the external analyzer probe to target system signals and how to configure and use the external analyzer.

Chapter 6 **Timing: Introduction**. This chapter introduces the external timing analyzer and describes its features.

Chapter 7 **Timing: Getting Started**. This chapter shows you how to start up the timing analyzer Softkey Interface and how to do a simple timing measurement.

Chapter 8 **Timing: Using the Analyzer**. This chapter reviews the functions of the timing analyzer, gives specific information on the use of each of the functions, and gives examples.

Chapter 9 **Timing: Commands**. This chapter furnishes a reference for each of the timing analyzer Softkey Interface commands, describes the command using syntax diagrams, provides a detailed description for each of the parameters, and follows up with examples for the use of the command.

Appendix A **External Analyzer Specifications**.

Appendix B **Timing Output and Display**.

Appendix C **Timing Messages**.

Appendix D **Accurate Timing Measurements**.

Conventions

Example commands throughout the manual use the following conventions:

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

Contents

9 Timing : Commands

C Timing Messages

D Accurate Timing Measurements

Illustrations

Tables

Notes

1

Introducing the Analyzer

This manual describes the HP 64700-Series analyzer. Each HP 64700-Series emulator contains an emulation analyzer. Your emulator may optionally contain an external analyzer. (If your emulator contains an external analyzer, an "L" suffix appears on the serial number tag, for example, "64742AL SN ...".)

The *emulation analyzer* captures emulator bus cycle information synchronously with the processor's clock signal. A *trace* is a collection of these captured states. The *trigger* state specifies when the trace measurement is taken. The *external analyzer* captures activity on signals external to the emulator, typically other target system signals.

The analyzer commands are the same in every emulator; consequently, this manual is shipped with every HP 64700-Series emulator ordered with the Softkey Interface.

Analyzer Features

This chapter lists basic features of the HP 64700-Series analyzer. The chapters which follow show you how to use these features.

Simple Measurements

The default condition of the analyzer allows you to perform a simple measurement by entering a simple "trace" command. You can add qualifiers to the trace command to specify when execution should be traced and which bus cycle states should be stored.

Trace Storage, Prestore, and Count

The analyzer can store up to 1024 states in trace memory. These states can be normal storage states or prestore states (states which precede normal storage states). A count qualifier may be associated with normal storage states; you can specify that the analyzer count in either time or the occurrences of some state. When counts are specified, only 512 states can be stored.

Sequencer and Windowing

The analyzer's sequencer allows you to trigger after a sequence of states are captured. Up to 7 sequence terms (the last of which is the trigger term), each with an option occurrence count, are available. A sequence restart term is also available.

Windowing refers to the ability to capture activity between one state and another. Up to 4 sequence terms are available when windowing is in effect.

Coordinated Measurements

When multiple HP 64700-Series emulators are connected via the Coordinated Measurement Bus (CMB), you can use the analyzer to trigger the analyzers of other emulators. You can also use the analyzer to trigger instruments connected to the BNC port. Conversely, analyzer measurements may be started by other emulators and instruments.

Performance Measurements

The Software Performance Measurement Tool (SPMT), which is part of the Softkey Interface, allows you to make measurements on the performance of your programs. You can measure activity in address ranges, or you can measure the average time it takes during and between execution of a program module.

External Analysis

Your HP 64700-Series emulator may optionally contain an external analyzer. The external analyzer provides 16 external trace signals and two 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.

Timing Analyzer

The timing analysis information is available through the Softkey Interface timing analyzer software residing on a HP-UX host. The software allows you to control the external timing analyzer, acquire timing measurements, analyze the trace measurements for specific occurrences, display the information in graphic or tabular form, and save the trace information for subsequent comparisons.

The list above is only a basic description of the HP 64700-Series analyzer features. The chapters which follow show you how to use these features.

Notes

2

Getting Started

Introduction

This chapter shows you how to use the emulation analyzer from within the Softkey Interface.

This chapter describes:

- The sample program on which example measurements are made.
- The default, power-up condition of the analyzer (including how to begin the trace measurement and display the trace).
- Expressions in trace command qualifiers.

This chapter shows you how to:

- Specify a simple trigger (and change the trigger position).
- Specify a storage qualifier.
- Use trace prestore.
- Change the count qualifier.
- Trigger on the Nth occurrence of some state.
- Trigger on multiple states.
- Use the sequencer.
- Stop a trace measurement.

Prerequisites

Before reading this chapter you should already know how the emulator operates. You should know how to use the Softkey Interface, and how to control the emulator from within the Softkey Interface. Refer to the appropriate *Emulator Softkey Interface User's Guide* manual to learn about the emulator; then, return to this manual.

The Sample Program

The sample program is used to illustrate analyzer examples. The sample program is written in assembly language so the disassembled trace listings will be more meaningful.

The examples in this chapter have been generated using a 68000 (HP 64742) emulator. The sample program is written in 68000 assembly language. (A similar program written in 80186 assembly language can be found in the *HP 64700-Series Emulators Terminal Interface Reference*.)

It is not important that you know 68000 assembly language; however, you should understand what the various sections of the program do and associate these tasks with the labels used in the program.

You are encouraged to rewrite the sample program in the assembly language appropriate for your emulator. Then, you can use your analyzer to perform the examples shown in this chapter. Of course, the output of your commands will be different than those shown here.

Description of the Sample Program

A pseudo-code algorithm of the sample program is shown in figure 2-1.

```
              Initialize the stack pointer.
       AGAIN: Save the two previous random numbers.
              Call the RAND random number generator subroutine.
              Test the two least significant bits of the previous random number.
                  If 00B then goto CALLER_0.
                  If 01B then goto CALLER_1.
                  If 10B then goto CALLER_2.
                  If 11B then goto CALLER_3.
    CALLER_0: Call the WRITE_NUMBER subroutine.
              Goto AGAIN (repeat program).
    CALLER_1: Call the WRITE_NUMBER subroutine.
              Goto AGAIN (repeat program).
    CALLER_2: Call the WRITE_NUMBER subroutine.
              Goto AGAIN (repeat program).
    CALLER_3: Call the WRITE_NUMBER subroutine.
              Goto AGAIN (repeat program).

WRITE_NUMBER: Write the random number to a 256 byte data area, using the second
              previous random number as an offset into that area.
              RETURN from subroutine.

        RAND: Pseudo-random number generator which returns a random number
              from 0-0FFH.
              RETURN from subroutine.
```

Figure 2-1. Pseudo-Code Algorithm of Sample Program

The sample program is not intended to represent a real routine. The program uses four different callers of the WRITE_NUMBER subroutine to simulate situations in real programs where routines are called from many different places. An example later in this chapter shows you how to use the analyzer prestore feature to determine where a routine is called from.

An assembler listing of the sample program is shown in figure 2-2. It is provided so that you can see the addresses associated with the program labels. The program area, which contains the instructions to be executed by the microprocessor, is located at 400H when linking the program. The RESULTS area, to which the random numbers are written, is located at 500H. The area which contains a variable used by the RAND subroutine and memory locations for the stack is located at 600H.

```
Cmdline - as68k -Lh anly.s
Line Address
1                                                XDEF   START,AGAIN
2                                                XDEF   RESULTS,RAND_SEED
3                                                SECT    PROG,,C
4    00000000 2E7C 0000 01FC  R START            MOVE.L  #STACK,A7
5                              *********************************************
6                               * The next two instructions move the second
7                               * previous random number into A1 (offset to
8                               * RESULTS area, and the previous random
9                               * number into D1.
10                             *********************************************
11   00000006 2241              AGAIN            MOVE.L  D1,A1
12   00000008 2200                               MOVE.L  D0,D1
13                             *********************************************
14                              * RAND returns random number in D0.
15                             *********************************************
16   0000000A 6100 0044                          BSR     RAND
17                             *********************************************
18                              * The following instructions determine which
19                              * caller calls WRITE_NUMBER (depends on last
20                              * two bits of the previous random number).
21                             *********************************************
22   0000000E 0801 0001                          BTST    #1,D1
23   00000012 6700 0006                          BEQ     ZERO_ONE
24   00000016 6000 000E                          BRA     TWO_THREE
25   0000001A 0801 0000         ZERO_ONE         BTST    #0,D1
26   0000001E 6700 0012                          BEQ     CALLER_0
27   00000022 6000 0014                          BRA     CALLER_1
28   00000026 0801 0000         TWO_THREE        BTST    #0,D1
29   0000002A 6700 0012                          BEQ     CALLER_2
30   0000002E 6000 0014                          BRA     CALLER_3
31                             *********************************************
32                              * The WRITE_NUMBER routine is called from
33                              * four different places.  The program is
34                              * repeated after the subroutine return.
35                             *********************************************
36   00000032 6100 0016         CALLER_0         BSR    WRITE_NUMBER
37   00000036 60CE                               BRA     AGAIN
38   00000038 6100 0010         CALLER_1         BSR    WRITE_NUMBER
39   0000003C 60C8                               BRA     AGAIN
40   0000003E 6100 000A         CALLER_2         BSR    WRITE_NUMBER
41   00000042 60C2                               BRA     AGAIN
42   00000044 6100 0004         CALLER_3         BSR    WRITE_NUMBER
43   00000048 60BC                               BRA     AGAIN
44                             *********************************************
45                              * The WRITE_NUMBER routine writes the random
46                              * number to the RESULTS area. The second
47                              * previous number is the offset in this area.
48                             *********************************************
49   0000004A 1340 0000       R WRITE_NUMBER     MOVE.B D0,RESULTS(A1)
50   0000004E 4E75                               RTS
51                             *********************************************
52                              * The RAND routine generates a pseudo-random
53                              * number from 0-0FFH, and leaves the result
54                              * in D0.
55                             *********************************************
56   00000050 2039 0000 0100  R RAND             MOVE.L RAND_SEED,D0
```

Figure 2-2. Sample Program Listing

```
00000056 C1FC 4E6D                          MULS.W  #4E6DH,D0
0000005A 2040                               MOVEA.L D0,A0
0000005C 41E8 0339                          LEA    339H(A0),A0
00000060 2008                               MOVE.L  A0,D0
00000062 23C0 0000 0100  R                  MOVE.L D0,RAND_SEED
00000068 4240                               CLR.W   D0
0000006A 4840                               SWAP    D0
0000006C 0280 0000 00FF                     ANDI.L #000000FFH,D0
00000072 4E75                               RTS

                                            SECT    DATA,,D
                         *********************************************
                          * Random numbers written to this area.
                         *********************************************
00000000                  RESULTS           DS.B    100H
                         *********************************************
                          * Variable used in RAND subroutine and
                          * stack area.
                         *********************************************
00000100 0000 0001        RAND_SEED         DC.L    1
00000104                                    DS.L    3EH
000001FC                  STACK             DS.W    2
                                            END     START
```

Figure 2-3. Sample Program Listing (Cont'd)

The sample program is assembled and linked with the following HP 64870 68000/10/20 Assembler/Linker/Librarian commands (which assume that **/usr/hp64000/bin** is defined in the PATH environment variable):

```
$ as68k -Lh anly.s > anly.lis <RETURN>
$ ld68k -c anly.k -Lh > anly.map <RETURN>
```

The linker command file, anly.k, contains the information below.

```
name anly
sect PROG=400h
sect DATA=500h
load anly.o
end
```

Before You Can Use the Analyzer

Before you can use the analyzer to perform measurements on the sample program, you must load and run the sample program. The following examples assume you are using the default emulator configuration which maps locations 0 through 1F7FFH as emulation RAM and which specifies a reset stack pointer value of 1FFEH.

Load the Program

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

load anly <RETURN>

Run the Program

To start the emulator executing the sample program, enter:

run from transfer_address <RETURN>

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

The Default Trace Command

The default trace command (shown below) will trigger on any state, store all captured states, and count time. To trace the states currently executing, enter:

trace <RETURN>

A message will flash on the status line to show you that the "Emulation trace [has] started", and another message will show you when the "Trace [is] complete".

Displaying the Trace

To display the trace, enter:

display trace <RETURN>

```
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    000424     0014  ORI.B   #**,[A4]                            ------------
+001     000438     6100  BSR.W   000044A                               600     nS
+002     00043A     0010    0010  supr prog                             400     nS
+003     0006F8     0000    0000  supr data wr word                     600     nS
+004     0006FA     043C    043C  supr data wr word                     400     nS
+005     00044A     1340  MOVE.B  D0,00500[A1]                          400     nS
+006     00044C     0500    0500  supr prog                             400     nS
+007     00044E     4E75  RTS                                           400     nS
+008     0005E0     B5B5    B5    supr data wr byte                     400     nS
+009     000450     2039  MOVE.L  0000600,D0                            400     nS
+010     0006F8     0000    0000  supr data rd word                     400     nS
+011     0006FA     043C    043C  supr data rd word                     400     nS
+012     00043C     60C8  BRA.B   0000406                               400     nS
+013     00043E     6100    6100   unused prefetch                      400     nS
+014     000406     2241  MOVEA.L D1,A1                                 600     nS

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


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

The first column on the trace list contains the line number. The trigger is always on line 0.

The second column contains the address information associated with the trace states. Addresses in this column may be locations of instruction opcodes on fetch cycles, or they may be sources or destinations of operand cycles.

The third column contains the data information associated with the trace states.

The fourth column shows mnemonic information about the emulation bus cycle. The disassembled instruction mnemonic is shown for opcode fetch cycles. The data and mnemonic status ("0010 supr prog", for example) are shown for operand cycles.

The fifth column shows the count information (time is counted by default). The trace list header indicates that each count is "relative" to the previous state.

If your emulator contains an external analyzer, a sixth column shows the external data captured by the external analyzer. On 80-column display terminals, the external data will be off screen; use **< CTRL> -F** and **< CTRL> -G** to move the screen left and right.

Sometimes, the trace will show opcode fetches for instructions which are not executed because of a transfer of execution to other addresses (see line 13 in the previous trace list). This can happen with microprocessors like the 68000 and the 80186 because they have pipelined architectures or instruction queues which allow them to prefetch the next few instructions before the current instruction is finished executing.

You can use the **<NEXT>** and **<PREV>** keys to scroll through the trace list a page at a time. The **<uparrow>** and **<downarrow>** keys will scroll through the trace list a line at a time. You can also display the trace list starting with a specific line number (for example, **display trace 100** <RETURN>). Refer to the "Displaying Traces" chapter for more information the trace list display.

Note

When a trigger condition is found but not enough states are captured to fill trace memory, the status line will show that the trace is still running. You can display all but the last captured state in this situation; you must halt the trace to display the last captured state.

Expressions in Trace Commands

When modifying the analysis specification (as shown throughout the remainder of this chapter), you can enter expressions which consist of values, symbols, and operators.

Values

Values are numbers in hexadecimal, decimal, octal, or binary. These number bases are specified by the following characters:

B b	Binary (example: 10010110b).
Q q O o	Octal (example: 377o or 377q).
D d (default)	Decimal (example: 2048d or 2048).
H h	Hexadecimal (example: 0a7fh). You must precede any hexadecimal number that begins with an A, B, C, D, E, or F with a zero.

Don't care digits may be included in binary, octal, or hexadecimal numbers and they are represented by the letters **X** or **x**. A zero must precede any numerical value that begins with an "X".

Symbols

A symbol database is built when the absolute file is loaded into the emulator. Both global and local symbols can be used when entering expressions. Global symbols are entered as they appear in the source file or in the global symbols display. When specifying a local symbol, you must include the name of the source file (anly.s) as shown below.

```
anly.s:START
```

Operators

Analysis specification expressions may contain operators. All operations are carried out on 32-bit, two's complement integers. (Values which are not 32 bits will be sign extended when expression evaluation occurs.)

The available operators are listed below in the order of evaluation precedence. Parentheses are also allowed in expressions to change the order of evaluation.

-, **~**	Unary two's complement, unary one's complement. The unary two's complement operator is not allowed on constants containing don't care bits.
*****, **/**, **%**	Integer multiply, divide, and modulo. These operators are not allowed on constants containing don't care bits.
+ , **-**	Addition, subtraction. These operators are not allowed on constants containing don't care bits.
&	Bitwise AND.
|	Bitwise inclusive OR.

Values, symbols, and operators may be used together in analysis specification expressions. For example, if the local symbol exists, the following is a valid expression:

```
file.c:symb+0b67dh&0fff00h
```

Qualifying the Trigger Condition

Suppose you want to look at the execution of the sample program after the address of AGAIN label (406H) occurs. To trigger on the address of label AGAIN, enter:

trace after AGAIN <RETURN>

Trigger Position

The "after" option in the command above supplies trigger position information. It says that states captured after the trigger should be saved; in other words, the trigger is positioned at the top of the trace. You can also specify that the trigger be positioned in the middle of the trace (about) or at the end of the trace (before).

```
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    000406     2241  MOVEA.L D1,A1                                 600     nS
+001     000408     2200  MOVE.L  D0,D1                                 400     nS
+002     00040A     6100  BSR.W   0000450                               400     nS
+003     00040C     0044    0044  supr prog                             400     nS
+004     0006F8     0000    0000  supr data wr word                     600     nS
+005     0006FA     040E    040E  supr data wr word                     400     nS
+006     000450     2039  MOVE.L  0000600,D0                            400     nS
+007     000452     0000    0000  supr prog                             400     nS
+008     000454     0600    0600  supr prog                             400     nS
+009     000456     C1FC  MULS.W  #04E6D,D0                             400     nS
+010     000600     064D    064D  supr data rd word                     400     nS
+011     000602     8FD0    8FD0  supr data rd word                     400     nS
+012     000458     4E6D    4E6D  supr prog                             400     nS
+013     00045A     2040  MOVEA.L D0,A0                                 400     nS
+014     00045C     41E8  LEA.L   00339[A0],A0                          400     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  after AGAIN


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

Trace List Description

In the preceding trace list, line 0 (labeled "after") shows the beginning of the program loop and line 2 shows the call of the RAND subroutine. The disassembled mnemonics on lines 6, 9, 13, and 14 show instructions which are executed in the RAND subroutine.

Press the **< NEXT>** key to see more lines of the trace.

```
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
+015     00045E     0339    0339  supr prog                            5.80  uS
+016     000460     2008  MOVE.L  A0,D0                              400     nS
+017     000462     23C0  MOVE.L  D0,0000600                         400     nS
+018     000464     0000    0000  supr prog                          400     nS
+019     000466     0600    0600  supr prog                          400     nS
+020     000468     4240  CLR.W   D0                                 400     nS
+021     000600     DDA1    DDA1  supr data wr word                  400     nS
+022     000602     9EC9    9EC9  supr data wr word                  400     nS
+023     00046A     4840  SWAP.W  D0                                 400     nS
+024     00046C     0280  ANDI.L  #0000000FF,D0                      400     nS
+025     00046E     0000    0000  supr prog                          400     nS
+026     000470     00FF    00FF  supr prog                          400     nS
+027     000472     4E75  RTS                                        400     nS
+028     000474     0000    0000   unused prefetch                   400     nS
+029     0006F8     0000    0000  supr data rd word                  800     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  after AGAIN


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

In the trace list above you see the last few instructions executed by the RAND subroutine (the RTS is the last instruction). To see more lines of the trace, press **< NEXT>** once again.

```
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
+030     0006FA     040E    040E  supr data rd word                  400     nS
+031     00040E     0801  BTST.L  #01,D1                             400     nS
+032     000410     0001    0001  supr prog                          400     nS
+033     000412     6700  BEQ.W   000041A                            400     nS
+034     000414     0006    0006  supr prog                          400     nS
+035     00041A     0801  BTST.L  #00,D1                             800     nS
+036     00041C     0000    0000  supr prog                          400     nS
+037     00041E     6700  BEQ.W   0000432                            400     nS
+038     000420     0012    0012  supr prog                          400     nS
+039     000422     6000  BRA.W   0000438                              1.0   uS
+040     000424     0014    0014  supr prog                          400     nS
+041     000438     6100  BSR.W   000044A                            600     nS
+042     00043A     0010    0010  supr prog                          400     nS
+043     0006F8     0000    0000  supr data wr word                  600     nS
+044     0006FA     043C    043C  supr data wr word                  400     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  after AGAIN


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

Line 31 shows the first instruction executed after return from the RAND subroutine. The instructions shown in the previous trace list decide which caller will call the WRITE_NUMBER subroutine. Line 41 shows the disassembled mnemonic of the instruction which calls the WRITE_NUMBER subroutine. The address information shows that the caller is CALLER_1. To view the instruction cycles of the WRITE_NUMBER subroutine, press < **NEXT**> again.

```
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
+045     00044A     1340  MOVE.B  D0,00500[A1]                            400     nS
+046     00044C     0500    0500  supr prog                               400     nS
+047     00044E     4E75  RTS                                             400     nS
+048     00054E     A1A1    A1    supr data wr byte                       400     nS
+049     000450     2039  MOVE.L  0000600,D0                              400     nS
+050     0006F8     0000    0000  supr data rd word                       400     nS
+051     0006FA     043C    043C  supr data rd word                       400     nS
+052     00043C     60C8  BRA.B   0000406                                 400     nS
+053     00043E     6100    6100   unused prefetch                        400     nS
+054     000406     2241  MOVEA.L D1,A1                                   600     nS
+055     000408     2200  MOVE.L  D0,D1                                   400     nS
+056     00040A     6100  BSR.W   0000450                                 400     nS
+057     00040C     0044    0044  supr prog                               400     nS
+058     0006F8     0000    0000  supr data wr word                       600     nS
+059     0006FA     040E    040E  supr data wr word                       400     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  after AGAIN


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

Line 45 shows the MOVE.B instruction associated with the WRITE_NUMBER subroutine.

Line 47 in the trace list above shows the RTS instruction associated with the WRITE_NUMBER subroutine. Line 48 shows the random number 0A1H is written to address 54EH.

Line 54 shows the AGAIN address associated with the next loop of the program.

Modifying Previous Trace Commands

Many of the examples presented in this chapter build on previous examples. If you are entering the trace commands shown, you will sometimes find it easier to modify a previous trace command than to enter the new command. If the command you wish to modify was the last command entered, it is still on the command line and you may edit it using the command line editing features (for example, using the left arrow and right arrow keys, using type-over, insert, delete, etc.). If the command you wish to modify was not the last command entered, you will have to recall the command. There are two ways to recall trace commands: command recall and the "trace modify_command" command.

Command Recall

If the command you wish to modify has been recently entered (within the last 20 commands), you can use the command recall feature. Press **< CTRL> -R** to recall commands. If you pass up the command of interest, you can use **< CTRL> -B** to move forward through the list.

Trace Modify Command

The "trace modify_command" command recalls the last trace command. The advantage of this command over command recall is that you do not have to move forward and backward over other commands to find the last trace command; also, the last trace command is always available, no matter how many commands have been entered since.

Specifying Storage Qualifiers

By default, all captured states are stored; however, you can qualify which states get stored. For example, to store *only* the states which write random numbers to the RESULTS area, enter:

```
trace after AGAIN
only range RESULTS thru RESULTS+0ffh <RETURN>
```

Note

You can only select one range in the emulation analysis specification. If you store states in a range, for example, you will not be allowed to select a range in any of the other analyzer specifications.

```
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    000406     2241  MOVEA.L D1,A1                                 2.6   uS
+001     0005F3     4949      49  supr data wr byte                    26.6   uS
+002     0005FD     9999      99  supr data wr byte                    29.2   uS
+003     000549     4242      42  supr data wr byte                    29.2   uS
+004     000599     3A3A      3A  supr data wr byte                    29.2   uS
+005     000542     0505    05    supr data wr byte                    29.2   uS
+006     00053A     5757    57    supr data wr byte                    29.2   uS
+007     000505     B6B6      B6  supr data wr byte                    30.4   uS
+008     000557     A0A0      A0  supr data wr byte                    29.2   uS
+009     0005B6     2B2B    2B    supr data wr byte                    28.0   uS
+010     0005A0     BDBD    BD    supr data wr byte                    30.4   uS
+011     00052B     E1E1      E1  supr data wr byte                    29.2   uS
+012     0005BD     1919      19  supr data wr byte                    29.2   uS
+013     0005E1     D9D9      D9  supr data wr byte                    29.2   uS
+014     000519     7D7D      7D  supr data wr byte                    29.2   uS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace   after AGAIN only  range RESULTS thru RESULTS+0ffh


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

Notice that the trigger state (line 0, labeled "after") is included in the trace list; trigger states are always stored.

This trace shows that the last two hex digits of the address in the RESULTS area are the same as the random number which gets written two states earlier (see the data in the "mnemonic" column of the trace list). This is expected because the sample program writes the current random number using the second previous random number as an offset into the RESULTS area.

Prestoring States

Suppose you find a defect in a subroutine, but you determine that the problem is actually due to something set up by the calling routine. If the subroutine is called from a variety of places in your program, you need to find out which callers cause the problem. Prestore can be used to find the callers of the subroutine.

Prestore allows you to save up to two states which precede a normal store state. Prestore is turned off by default. However, you can include a prestore qualifier in the command line to qualify the states which are prestored.

As an example, let's use a prestore qualifier to show which caller of WRITE_NUMBER corresponds to each value written to the RESULTS area. Because the BSR assembly language instruction is used to call a subroutine, you can qualify prestore states as states whose data equals the BSR opcode (6100H). For example:

```
trace after AGAIN
only range RESULTS thru RESULTS+0ffh
prestore data 6100h <RETURN>
```

```
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    000406     2241  MOVEA.L D1,A1                                    2.6   uS
pstore   00040A     6100  BSR.W********
pstore   000438     6100  BSR.W********
+003     000530     3737    37    supr data wr byte                       26.6   uS
pstore   00040A     6100  BSR.W********
pstore   000444     6100  BSR.W********
+006     000545     7070      70  supr data wr byte                       30.4   uS
pstore   00040A     6100  BSR.W********
pstore   000432     6100  BSR.W********
+009     000537     8E8E      8E  supr data wr byte                       28.0   uS
pstore   00040A     6100  BSR.W********
pstore   00043E     6100  BSR.W********
+012     000570     3434    34    supr data wr byte                       29.2   uS
pstore   00040A     6100  BSR.W********
pstore   000432     6100  BSR.W********

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  after AGAIN only  range RESULTS thru RESULTS+0ffh prestore  data 6100h


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

The prestore state immediately preceding each write state shows the address of the caller.

The analyzer uses the same resource to save prestore states as it does to save count tags. Consequently, no count appears for prestore states. Time counts are relative to the previous normal storage state.

States which satisfy the prestore qualifier and the storage qualifier at the same time are stored as normal states.

Changing the Count Qualifier

Suppose now that you are interested in only one address in the RESULTS area, say 5C2H. You wish to see how many loops of the program occur between each write of a random number to this address. You can enter a trace command that triggers on address 5C2H (since it's the only address of interest), stores only writes to address 5C2H, and counts the address of the AGAIN label (406H). For example:

> ***trace after*** 5c2h
> ***only*** 5c2h
> ***counting state*** AGAIN <RETURN>

```
Trace List                   Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                Opcode or Status              state count
Base:     hex     hex                     mnemonic                    relative
after    0005C2     0606    06    supr data wr byte                 ------------
+001     0005C2     5353    53    supr data wr byte                           51
+002     0005C2     3333    33    supr data wr byte                         1343
+003     0005C2     E2E2    E2    supr data wr byte                          291
+004     0005C2     D3D3    D3    supr data wr byte                          351
+005     0005C2     3B3B    3B    supr data wr byte                          793
+006     0005C2     2E2E    2E    supr data wr byte                           61
+007     0005C2     1818    18    supr data wr byte                          432
+008     0005C2     1A1A    1A    supr data wr byte                           40
+009     0005C2     B3B3    B3    supr data wr byte                         1245
+010     0005C2     5C5C    5C    supr data wr byte                           87
+011     0005C2     BCBC    BC    supr data wr byte                          660
+012     0005C2     3F3F    3F    supr data wr byte                          425
+013     0005C2     BABA    BA    supr data wr byte                          345
+014     0005C2     8686    86    supr data wr byte                          729

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  after 5c2h only 5c2h counting  state AGAIN


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

The trace listing above shows that the number of times the program executes between writes to 5C2H varies.

Turning Counting Off

Turning the count off allows the analyzer to store 1024 states instead of 512 states. For example:

trace after 5c2h
only 5c2h
counting off <RETURN>

(The default trace depth is 256, which means that only 256 states are uploaded and are available to be displayed. You can increase the trace depth so that more states can be viewed; however, it takes longer to upload more states. Refer to the "Changing the Trace Depth" section in the "Displaying Traces" chapter.)

Triggering on the Nth Occurrence of a State

In the trace command, you can specify a trigger on the Nth occurrence of a state. To specify an occurrence count, enter:

trace about 5c2h ***occurs*** 5
only 5c2h
counting state AGAIN <RETURN>

```
Trace List                    Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                 Opcode or Status              state count
Base:     hex     hex                      mnemonic                    relative



-004     0005C2     4141    41    supr data wr byte                 ------------
-003     0005C2     4B4B    4B    supr data wr byte                          447
-002     0005C2     E0E0    E0    supr data wr byte                           81
-001     0005C2     8F8F    8F    supr data wr byte                            9
about    0005C2     EFEF    EF    supr data wr byte                          150
+001     0005C2     9797    97    supr data wr byte                          601
+002     0005C2     1E1E    1E    supr data wr byte                           71
+003     0005C2     2F2F    2F    supr data wr byte                          534
+004     0005C2     0E0E    0E    supr data wr byte                          511
+005     0005C2     0505    05    supr data wr byte                           35
+006     0005C2     F7F7    F7    supr data wr byte                          101
+007     0005C2     5E5E    5E    supr data wr byte                           81

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  about 5c2h occurs 5 only 5c2h counting  state AGAIN


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

Notice that the trigger position has been moved to the center of the trace so that states which occur before the trigger are saved.

Usually, when you enter a "trace about" command, the trigger state (line 0) is labeled "about". However, if there are three or fewer states before the trigger, the trigger state is labeled "after". Likewise, if there are 3 or fewer states after the trigger, the trigger state is labeled "before".

Triggering on Multiple States

The analysis specification allows you to trigger on multiple states. That is, when one state, or a second state, or a third state, etc., occur the specified number of times, the analyzer triggers. For example, suppose you wish to change the previous analysis specification to trigger on the fifth occurrence of address 5C2H or 5C3H. Enter:

trace about 5c2h ***or*** 5c3h ***occurs*** 5
only 5c2h
counting state AGAIN <RETURN>

Press the < **downarrow**> key a few times to view the states that are stored before the trigger.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                Opcode or Status             state count
Base:     hex     hex                     mnemonic                   relative

-002     0005C2     5656    56    supr data wr byte                 ------------
-001     0005C2     6A6A    6A    supr data wr byte                          136
after    0005C3     9393      93  supr data wr byte                           35
+001     0005C2     2626    26    supr data wr byte                          177
+002     0005C2     D3D3    D3    supr data wr byte                          213
+003     0005C2     0707    07    supr data wr byte                          300
+004     0005C2     9797    97    supr data wr byte                          125
+005     0005C2     E3E3    E3    supr data wr byte                          452
+006     0005C2     4747    47    supr data wr byte                           79
+007     0005C2     1010    10    supr data wr byte                          225
+008     0005C2     6E6E    6E    supr data wr byte                          280
+009     0005C2     7474    74    supr data wr byte                          617
+010     0005C2     C0C0    C0    supr data wr byte                          194
+011     0005C2     0202    02    supr data wr byte                          990

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  about 5c2h  or 5c3h occurs 5 only 5c2h counting  state AGAIN


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

Notice, in the preceding display, that only two accesses of address 5C2H occur before the trigger. Because the occurrence count is still five, two accesses of 5C3H must have also occurred before the trigger. The trace command above causes the analyzer to trigger on the fifth occurrence of either address 5C2H or 5C3H.

Using Address, Data, and Status Qualifiers

So far, the examples have not used address, data, and status qualifiers in combination. When an address qualifier is specified, additional data and status specifications serve to further qualify a state.

For example, specifying a trigger on variable RAND_SEED (address 600H in the sample program) will cause the analyzer to trigger on the first access of 600H, regardless of the value being read or written to this address.

However, suppose you wish to trigger on the read of a specific value from 600H, say 0XX5AH (where "X"s are "don't cares"). To trigger on the occurrence of this state, you can include data and status qualifiers along with the address qualifier.

Suppose also that you want to go back to storing addresses in the range 500H through 5FFH, but that you only want to store the writes of bytes whose first hex digit is "5". To do this, enter:

```
trace about RAND_SEED data 0xx5ah status
read
only range RESULTS thru RESULTS+0ffh data
0xx5xh <RETURN>
```

Note

Status qualifiers will differ from one emulator to another. For more information on the status qualifiers refer to your *Emulator Softkey Interface User's Guide.*

```
Trace List                 Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data               Opcode or Status              state count
Base:     hex     hex                    mnemonic                    relative
-007     00054E     5858    58    supr data wr byte                          31
-006     0005F8     5C5C    5C    supr data wr byte                           6
-005     000565     5E5E      5E  supr data wr byte                          20
-004     000576     5D5D    5D    supr data wr byte                          16
-003     0005F2     5454    54    supr data wr byte                          12
-002     000588     5151    51    supr data wr byte                           7
-001     0005A6     5A5A    5A    supr data wr byte                          17
about    000600     F45A    F45A  supr data rd word                           1
+001     0005A7     5B5B      5B  supr data wr byte                           3
+002     000502     5757    57    supr data wr byte                          11
+003     000564     5454    54    supr data wr byte                          16
+004     000520     5555    55    supr data wr byte                          10
+005     000516     5555    55    supr data wr byte                          12
+006     00059E     5A5A    5A    supr data wr byte                           9
+007     000584     5A5A    5A    supr data wr byte                          23

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  about RAND_SEED data 0xx5ah status  read  only  range RESULTS thru RESUL
TS+0ffh data 0xx5xh

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

Using the Sequencer

The examples shown previously in this chapter are all of trace specifications in which the analyzer is triggered on a single state. However, if you use the analyzer's sequencer, you can specify traces that trigger on a series, or sequence, of states.

The analyzer's sequencer has several levels (also called *sequence terms*). Each state in the series of states to be found before triggering, as well as the trigger state, is associated with a sequence term.

The sequencer works like this: the analyzer searches for the state associated with the first sequence term. When that state is captured, the analyzer starts searching for the state associated with the second term, and so on. The last sequence term used is associated with the trigger state. When the trigger state is captured the analyzer is triggered. Up to seven sequence terms and an optional occurrence count for each term are available.

Here is an example trace command that uses the sequencer:

trace find_sequence STATE_1 ***occurs*** 2 ***then*** STATE_2 ***occurs*** 5 ***then*** STATE_3 ***then*** STATE_4 ***then*** STATE_5 ***then*** STATE_6 ***trigger after*** TRIGGER_STATE <RETURN>

In the "Specifying Storage Qualifiers" section earlier in this chapter, the example trace specification triggered on an address and stored only states in which values were written to the RESULTS area. Now you'd like to trigger after a series of states while continuing to store only the states that write to the RESULTS area.

For example, suppose you'd like to trigger the flow of execution from TWO_THREE to CALLER_3 after CALLER_0 has occurred 8 times. To do this, you would enter the following commands. (The "cws" command saves you from having to include "anly.s:" when specifying local symbols in the trace command.)

cws anly.s: <RETURN>

trace find_sequence CALLER_0 ***occurs*** 8
then TWO_THREE
then CALLER_3
trigger about WRITE_NUMBER
only range RESULTS ***thru*** RESULTS+0ffh <RETURN>

```
Trace List                 Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                Opcode or Status              state count
Base:     hex     hex                     mnemonic                    relative
-007     000571     D9D9      D9  supr data wr byte                           1
-006     000523     A0A0      A0  supr data wr byte                           1
sq adv   000432     6100  BSR.W********                                       1
-004     0005D9     5E5E      5E  supr data wr byte                           0
sq adv   000426     0801  BTST.L  #**,D1                                      1
-002     0005A0     E3E3    E3    supr data wr byte                           0
sq adv   000444     6100  BSR.W********                                       0
about    00044A     1340  MOVE.B  D0,****[A1]                                 1
+001     00055E     F2F2    F2    supr data wr byte                           0
+002     0005E3     9898      98  supr data wr byte                           1
+003     0005F2     0E0E    0E    supr data wr byte                           1
+004     000598     5A5A    5A    supr data wr byte                           1
+005     00050E     9292    92    supr data wr byte                           1
+006     00055A     8383    83    supr data wr byte                           1
+007     000592     2929    29    supr data wr byte                           1

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  find_sequence CALLER_0 then TWO_THREE then CALLER_3 trigger  about WRITE
_NUMBER only  range RESULTS thru RESULTS+0ffh

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

Notice the states that contain "sq adv" in the line number column. These are the states associated with (or captured for) each sequence term. Just as the trigger state is always stored in trace memory, the states captured in the sequence are always stored if the trace buffer is deep enough.

Because the trigger is the last sequence term, the remaining states stored after the trigger state are writes to the RESULTS area.

Specifying a Restart Term

When using the analyzer's sequencer, an additional sequence restart term is also allowed. This restart is a "global restart"; that is, it applies to all the sequence terms.

The restart term is a state which, when captured before the analyzer has found the trigger state, causes the sequencing to start from the beginning again. You can use the restart term to make certain some state does not occur in the sequence that triggers the analyzer.

For example, you may have noticed in the previous trace that a write to the RESULTS area occurred between the TWO_THREE and CALLER_3 states in the sequence and that the state count associated with WRITE_NUMBER state shows that AGAIN occurred before the trigger. What was actually captured in the previous trace was the flow of execution where TWO_THREE occurs, then CALLER_2, then WRITE_NUMBER, then a prefetch of CALLER_3 on the return from WRITE_NUMBER, and then the capture of WRITE_NUMBER occurred the next time through the program. By specifying a restart on CALLER_2, you can fix the previous trace command so that only the flow of execution from TWO_THREE to CALLER_3 is captured.

```
trace find_sequence CALLER_0 occurs 8
then TWO_THREE
then CALLER_3
restart CALLER_2
trigger about WRITE_NUMBER
only range RESULTS thru RESULTS+0ffh <RETURN>
```

```
Trace List                   Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                 Opcode or Status              state count
Base:     hex     hex                      mnemonic                    relative
sq adv   00043E     6100  BSR.W********                                        0
-006     000568     DFDF    DF    supr data wr byte                            0
-005     000596     6C6C    6C    supr data wr byte                            1
sq adv   000432     6100  BSR.W********                                        1
-003     0005DF     CBCB      CB  supr data wr byte                            0
sq adv   000426     0801  BTST.L  #**,D1                                       1
sq adv   000444     6100  BSR.W********                                        0
about    00044A     1340  MOVE.B  D0,****[A1]                                  0
+001     00056C     2121    21    supr data wr byte                            0
+002     0005CB     0E0E      0E  supr data wr byte                            1
+003     000521     2A2A      2A  supr data wr byte                            1
+004     00050E     C3C3    C3    supr data wr byte                            1
+005     00052A     5A5A    5A    supr data wr byte                            1
+006     0005C3     2929      29  supr data wr byte                            1
+007     00055A     ADAD    AD    supr data wr byte                            1

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  find_sequence CALLER_0 then TWO_THREE then CALLER_3 restart CALLER_2 tri
gger  about WRITE_NUMBER only  range RESULTS thru RESULTS+0ffh

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

Notice in the resulting trace (you may have to press **< PREV>** in order to see the states captured prior to the trigger) that "sq adv" is also shown next to states which cause a sequencer restart.

The sequencing capabilities from within the Softkey Interface are not as powerful or flexible as they are from within the Terminal Interface. If you do not find the sequencing flexibility you need from within Softkey Interface, refer to the *Terminal Interface: Analyzer User's Guide.*

Tracing "Windows" of Activity

Windowing refers to the analyzer feature that allows you to turn on, or enable, the capturing of states after some state occurs then to turn off, or disable, the capturing of states when another state occurs. In effect, windowing allows you capture segments, or windows, of code execution.

Windowing is different than storing states in a range (the "only range" option in the trace command syntax) because it allows you to capture execution of all states in a window of code whereas storing states in a range won't capture the execution of subroutine that are called in that range or reads and writes to locations outside that range.

When you use the windowing feature of the analyzer, the trigger state must be in the window or else the trigger will never be found.

If you wish to combine the windowing and sequencing functions of the analyzer, there are some restrictions:

- Up to four sequence terms are available when windowing is in effect.
- Global restart is not available when windowing is in effect.
- Occurrence counts are not available.

Suppose that you are only interested in the execution that occurs within the RAND subroutine of the sample program. You could specify the address of the subroutine call as the window enable state and the address of the subroutine's last instruction as the window disable state (and you could trigger on any state in that window by not specifying a trigger). For example:

trace enable 40ah ***disable*** 472h <RETURN>

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                Opcode or Status              state count
Base:     hex     hex                     mnemonic                    relative
+015     000464     0000    0000  supr prog                                    0
+016     000466     0600    0600  supr prog                                    0
+017     000468     4240  CLR.W   D0                                           0
+018     000600     091B    091B  supr data wr word                            0
+019     000602     576B    576B  supr data wr word                            0
+020     00046A     4840  SWAP.W  D0                                           0
+021     00046C     0280  ANDI.L  #0000000FF,D0                                0
+022     00046E     0000    0000  supr prog                                    0
+023     000470     00FF    00FF  supr prog                                    0
sq adv   000472     4E75  RTS                                                  0
sq adv   00040A     6100  BSR.W   0000450                                      1
+026     00040C     0044    0044  supr prog                                    0
+027     0006F8     0000    0000  supr data wr word                            0
+028     0006FA     040E    040E  supr data wr word                            0
+029     000450     2039  MOVE.L  0000600,D0                                   0

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 trace  enable 40ah disable anly.s:RAND+022h


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

Notice in the resulting trace (you may have to press **< NEXT>**) that the enable and disable states have the "sq adv" string in the line number column. This is because the windowing feature uses the analyzer's sequencer.

The windowing capabilities from within the Softkey Interface are not as powerful or flexible as they are from within the Terminal Interface. For example, in the Terminal Interface, you can set up a windowing trace specification where the trigger does not have to be in the window. If you do not find the windowing flexibility you need from within Softkey Interface, refer to the *Terminal Interface: Analyzer User's Guide.*

Storing and Loading Trace Commands

You can save a trace command to a "trace specification" file and reload it at a later time. To store the current trace command, enter:

```
store trace_spec tspecfile <RETURN>
```

The trace command is saved in a file named "tspecfile.TS" in the current directory. The extension ".TS" is appended to trace specification files if no extension is specified in the "store trace_spec" command. Enter another trace command to overwrite the existing trace command:

```
trace <RETURN>
```

To bring back the trace command saved in "tspecfile.TS" and perform a trace measurement using it, enter the commands:

```
load trace_spec tspecfile <RETURN>
trace again <RETURN>
```

Trace commands that have just been loaded will always work, even if symbols have been changed; however, once you modify the trace command, it may no longer work.

Trace Commands in the Event Log Display

The event log display shows the previous trace commands. To view the event log display, enter:

display event_log <RETURN>

```
Event Log
  Time     Type           Message
11:58:52 TRACE   Emulation trace started
11:58:52 TRACE   Emulation trace complete
12:00:25 TRACE   trace find_sequence CALLER_0 then TWO_THREE then CALLER_3 rest
                 art CALLER_2 trigger about WRITE_NUMBER only range RESULTS thr
                 u RESULTS+0ffh
12:00:26 TRACE   Emulation trace started
12:00:26 TRACE   Emulation trace complete
12:01:27 TRACE   trace enable 40ah disable 472h
12:01:28 TRACE   Emulation trace started
12:01:28 TRACE   Emulation trace complete
12:02:19 TRACE   trace
12:02:20 TRACE   Emulation trace started
12:02:20 TRACE   Emulation trace complete
12:02:40 TRACE   trace enable 40ah disable 472h
12:02:48 TRACE   Emulation trace started
12:02:49 TRACE   Emulation trace complete

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  event_log


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

Storing and Loading Traces

You can save a trace to a trace file and reload it at a later time. To store the current trace, enter the command:

store trace trcfile <RETURN>

The trace is saved in a file named "trcfile.TR" in the current directory. The extension ".TR" is appended to trace files if it is not specified in the "store trace" command. Enter another trace command to overwrite the existing trace:

trace <RETURN>

To bring back the trace of the previous section, enter the command:

```
load trace trcfile <RETURN>
display trace <RETURN>
```

The trace information stored in "trcfile.TR" is restored, and you can view the trace as you would any other trace.

The restored trace depth is the depth specified when the trace was stored and cannot be increased. You may want to increase the trace depth before storing traces.

When a trace is loaded, the trace command is not restored. A "trace again" or "trace modify" command will use the last trace command entered, not the command which resulted in the loaded trace. Also, the trace status shown by the "display status" command does not reflect the loaded trace.

Stopping the Trace

You can, and most likely will, specify traces whose trigger or storage states are never found. When this happens, the "Trace complete" message is never shown, and the trace continues to run ("Trace running"). When these situations occur, you can choose to halt the trace measurement with the following command:

```
stop_trace <RETURN>
```

The "stop_trace" command is also useful to deactivate signals which are driven when the trigger is found (refer to the "Coordinated Measurements" chapter in the *Softkey Interface Reference* manual).

Tracing on Halt

The "trace on_halt" command allows you to prevent triggering. In other words, the trace runs until you enter the "stop_trace" command (described in the next section). The "trace on_halt" command is the same as tracing "before" a state that never occurs.

The "trace on_halt" command is useful, for example, when you wish to trace the states leading up to a break into the monitor (provided that you are using the background monitor and tracing only foreground operation). Suppose your program breaks on an access to guarded memory. To trace the states that lead up to the break, enter the "trace on_halt" command, and run the program. When the break occurs, the emulator is running in the background monitor, and the analyzer is no longer capturing states. To display the states leading up to the break, enter the "stop_trace" command (and the "display trace" command if traces are not currently being displayed).

When the "on_halt" option is used in a trace command, the trigger condition (and position) options, as well as the "repetitively" and "break_on_trigger" options, cannot be included in the command.

Conclusion

This concludes the "Getting Started" chapter. You have learned about the basic trace commands and how to specify trigger conditions, storage, prestore, and count qualifiers, occurrence qualifiers, and address, data, and status qualifiers.

3

Displaying Traces

Introduction

This chapter describes the options available when displaying trace lists.

The trace list can contain high-level source file information.

The trace used to illustrate the various display options throughout this chapter was generated in the following manner. (The commands shown below assume that **/usr/hp64000/bin** is specified in the PATH environment variable.)

First, the program shown in figure 3-1 was compiled and linked with the following HP 64902 C Cross Compiler command:

$ **cc68000 -hvONr** hp64742 **-o** cprg cprg.c <RETURN>

Then, the default emulator configuration for the HP 64742 environment was copied to the current directory with the following command:

$ **cp** /usr/hp64000/env/hp64742/config.EA config.EA <RETURN>

Next, the emulation system (Softkey Interface) was entered:

$ **emul700** <emul_name> <RETURN>

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

After the emulation system was entered, the default configuration was loaded:

load configuration config <RETURN>

```
unsigned short dest[0x7f];
unsigned short *dest_ptr;

main ()
{
/****************************************/
/* This is a comment block              */
/* to demonstrate the "number           */
/* of source lines" trace               */
/* display option.                      */
/****************************************/

char *message;

     for (;;)
     {
          message = "This message is to be written indefinitely. ";
          dest_ptr = dest;
          while (*message != '\0')
          {
               *dest_ptr = *message;
               dest_ptr++;
               message++;
          }
     }
}
```

Figure 3-1. Program Used for Example Displays

Then, the absolute file was loaded:

load cprg <RETURN>

The following trace command was entered, and the program was run.

trace after main <RETURN>
run from transfer_address <RETURN>

Display Positioning

The trace display command displays 256 states, not all of which can appear on the screen at the same time. However, you can reposition the display on the screen with the keys described below.

Up/Down

The < **uparrow**> and < **downarrow**> (or roll up and roll down) keys move the display up or down on the screen one line at a time.

The page up and page down keys allow you to move the display up or down a page at a time.

Left/Right

The **< CTRL> -F** and **< CTRL> -G** keys allow you to move the display left or right, respectively. These keys are used if the emulator contains an external analyzer and you have an 80 column display (The default trace display includes the external bits information in this case, but this information cannot be displayed on an 80 column display.)

The **< CTRL> -F** and **< CTRL> -G** keys are also used when the width of the address or mnemonic/absolute columns is increased so that not all the trace display data can be displayed across the screen.

Changing the Trace Depth

The "display trace depth" command allows you to specify the number of states that are displayed. By reducing the trace depth, you can shorten the time it takes for the Softkey Interface to upload the trace information. You can increase the trace depth to view more states of the current trace.

The maximum number of trace states is 1024 when counting is turned off, 512 when counting is on. When you initially enter the Softkey Interface, the trace depth is 256. The minimum trace depth is 9.

If you wish to reduce the number of states that are displayed, the "display trace depth" command must be entered before the trace is displayed. You cannot use this command to reduce the number of states displayed in the current trace.

Displaying About a Line Number

The "< LINE # > " trace display option allows you to specify the line number to be centered in the 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
+193     06017C     008E    008E  supr data rd word                   520     nS
+194     000B00     5493    5493  supr prog                           480     nS
+195     06008E     0061    0061  supr data wr word                   520     nS
+196     000B02     528A    528A  supr prog                           480     nS
+197     06017A     0006    0006  supr data rd word                   520     nS
+198     06017C     008E    008E  supr data rd word                   480     nS
+199     000B04     4A12    4A12  supr prog                           520     nS
+200     06017C     0090    0090  supr data wr word                   480     nS
+201     06017A     0006    0006  supr data wr word                   520     nS
+202     000B06     66F0    66F0  supr prog                           480     nS
+203     000B16     6765    67    supr data rd byte                     1.0   uS
+204     000B08     60E0    60E0  supr prog                           520     nS
+205     000AF8     1012    1012  supr prog                           760     nS
+206     000AFA     4880    4880  supr prog                           480     nS
+207     000B16     6765    67    supr data rd byte                   520     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  trace 200


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

Disassembling the Trace Information

The "disassemble_from_line_number" trace display option causes the inverse assembler to attempt to begin disassembling the trace information from the specified line number. This option is required for inverse assemblers that cannot uniquely identify opcode fetch states on the processor bus. This option is not present for emulators whose inverse assembler can determine opcode fetch states on the processor bus.

If the line number specified is not an opcode fetch state, the disassembled information will be incorrect.

```
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
+200     06017C     0090    0090  supr data wr word                      480     nS
+201     06017A     0006    0006  supr data wr word                      520     nS
+202     000B06     66F0  BNE.B   0000AF8                                480     nS
+203     000B16     6765    67    supr data rd byte                        1.0   uS
+204     000B08     60E0  BRA.B   0000AEA                                520     nS
+205     000AF8     1012  MOVE.B  (A2),D0                                720     nS
+206     000AFA     4880  EXT.W   D0                                     520     nS
+207     000B16     6765    67    supr data rd byte                      480     nS
+208     000AFC     2053  MOVEA.L (A3),A0                                520     nS
+209     000AFE     3080  MOVE.W  D0,(A0)                                480     nS
+210     06017A     0006    0006  supr data rd word                      520     nS
+211     06017C     0090    0090  supr data rd word                      480     nS
+212     000B00     5493  ADDQ.L  #2,(A3)                                520     nS
+213     060090     0067    0067  supr data wr word                      480     nS
+214     000B02     528A  ADDQ.L  #1,A2                                  520     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  trace  disassemble_from_line_number 200


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

Displaying in Absolute Format

The "absolute" trace display option allows you to display status information in absolute format (binary, hex, or mnemonic). The "absolute status mnemonic" display is the same as default mnemonic display, except that opcodes are not disassembled.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                 Absolute Status              time count
Base:     hex     hex                      binary                     relative
+200     06017C     0090  11101100                                    480     nS
+201     06017A     0006  11101100                                    520     nS
+202     000B06     66F0  10110110                                    480     nS
+203     000B16     6765  10101111                                      1.0   uS
+204     000B08     60E0  10110110                                    520     nS
+205     000AF8     1012  10110110                                    720     nS
+206     000AFA     4880  10110110                                    520     nS
+207     000B16     6765  10101111                                    480     nS
+208     000AFC     2053  10110110                                    520     nS
+209     000AFE     3080  10110110                                    480     nS
+210     06017A     0006  11101110                                    520     nS
+211     06017C     0090  11101110                                    480     nS
+212     000B00     5493  10110110                                    520     nS
+213     060090     0067  11101100                                    480     nS
+214     000B02     528A  10110110                                    520     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  trace  absolute  status  binary


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

Displaying in Mnemonic Format

The "mnemonic" trace display option allows you to display the trace information in mnemonic format (that is, opcodes and status). The default trace display is in mnemonic format.

```
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
+011     000AEA     247C  MOVEA.L #000000B0C,A2                         480     nS
+012     000AEC     0000    0000  supr prog                             520     nS
+013     000AEE     0B0C    0B0C  supr prog                             480     nS
+014     000AF0     26BC  MOVE.L  #00006007C,(A3)                       520     nS
+015     000AF2     0006    0006  supr prog                             480     nS
+016     000AF4     007C    007C  supr prog                             520     nS
+017     000AF6     600C  BRA.B   0000B04                               480     nS
+018     06017A     0006    0006  supr data wr word                     520     nS
+019     06017C     007C    007C  supr data wr word                     480     nS
+020     000AF8     1012  MOVE.B  (A2),D0                               520     nS
+021     000B04     4A12  TST.B   (A2)                                  720     nS
+022     000B06     66F0  BNE.B   0000AF8                               520     nS
+023     000B0C     5468    54    supr data rd byte                     480     nS
+024     000B08     60E0  BRA.B   0000AEA                               520     nS
+025     000AF8     1012  MOVE.B  (A2),D0                               760     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  trace  disassemble_from_line_number 11


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

Including High-Level Source Lines

To include high-level source lines in the trace display, you must use the "set" command. The "set" command allows you to include symbolic information in trace, memory, register, and software breakpoint displays. The "set" command affects all displays for the current window.

The "set source on/off/only" command allows you to include source file information in the trace list or memory mnemonic display. The "source only" option specifies that only the source file information will be displayed.

When source lines are included, comments that contain file and line information appear before the source lines.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                   mnemonic                      relative
      ##########cprg.c - line     6 thru    17 #################################
      char *message;

           for (;;)
           {
                message = "This message is to be written indefinitely. ";
+011     000AEA     247C  MOVEA.L #000000B0C,A2                       480     nS
+012     000AEC     0000    0000  supr prog                           520     nS
+013     000AEE     0B0C    0B0C  supr prog                           480     nS
      ##########cprg.c - line    18 ############################################
                dest_ptr = dest;
+014     000AF0     26BC  MOVE.L  #00006007C,(A3)                     520     nS
+015     000AF2     0006    0006  supr prog                           480     nS
+016     000AF4     007C    007C  supr prog                           520     nS
      ##########cprg.c - line    19 ############################################

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 set  source  on


pod_cmd    set    perfinit perfrun          perfend                    ---ETC--
```

Additional Options with Source On/Only

Also, when source lines are turned on, three additional options are available in the set command: inverse video, tabs are, and number of source lines.

Inverse Video. This option allows you to display source lines in inverse video.

Tabs Are. This option allows you to specify the number of spaces between tab stops so that the appropriate number of spaces can be inserted for source lines. The default value is eight. Values from two to 15 can be entered.

Number of Source Lines. Typically, there are lines in the source file that are not associated with actual instructions (declarations, comments, etc.). This option allows you to specify the number of these source lines to be displayed for every source line that is associated with an actual instruction. Only source lines up to the the previous source line that corresponds to actual code will be displayed. The default value is five. Values from one to 50 can be entered.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                   mnemonic                      relative
      ##########cprg.c - line     6 thru    17 ################################
      /*****************************************/
      /* This is a comment block              */
      /* to demonstrate the "number           */
      /* of source lines" trace               */
      /* display option.                      */
      /*****************************************/

      char *message;

           for (;;)
           {
                message = "This message is to be written indefinitely. ";
+011     000AEA     247C  MOVEA.L #000000B0C,A2                       480     nS
+012     000AEC     0000    0000  supr prog                           520     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 set  source  on  number_of_source_lines 12


pod_cmd    set    perfinit perfrun          perfend                   ---ETC--
```

Including Symbol Information

The "set symbols on/off" command allows you to specify that address information be displayed in terms of program symbols.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:      Address       Data            Opcode or Status           time count
Base:       symbols       hex            mnemonic w/symbols           relative
+011   |cprg.c:forLoop1     247C  MOVEA.L #000000B0C,A2               520     nS
+012   cprg:main+00000C     0000    0000  supr prog                   480     nS
+013   cprg:main+00000E     0B0C    0B0C  supr prog                   520     nS
+014   cprg:main+000010     26BC  MOVE.L  #00006007C,(A3)             480     nS
+015   cprg:main+000012     0006    0006  supr prog                   520     nS
+016   cprg:main+000014     007C    007C  supr prog                   480     nS
+017   cprg:main+000016     600C  BRA.B   cprg.c:continue2            520     nS
+018    DATA|dest_ptr       0006    0006  supr data wr word           480     nS
+019   D|cprg.c:+000100     007C    007C  supr data wr word           520     nS
+020   cprg.:whileLoop1     1012  MOVE.B  (A2),D0                     480     nS
+021   cprg.c:continue2     4A12  TST.B   (A2)                        760     nS
+022   cprg:main+000026     66F0  BNE.B   cprg.:whileLoop1            480     nS
+023   C|cprg.c:String1     5468    54    supr data rd byte           520     nS
+024   cprg.c:continue1     60E0  BRA.B   |cprg.c:forLoop1            480     nS
+025   cprg.:whileLoop1     1012  MOVE.B  (A2),D0                     760     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 set  source  off  symbols  on


pod_cmd    set    perfinit perfrun          perfend                    ---ETC--
```

Changing Column Widths

The "set width" command allows you to change the width of the address and mnemonic (or absolute) columns in the trace list. Values from one to 80 can be entered.

When address information is being displayed in terms of symbols (in other words, symbols on), you may wish to increase the width of the address column to display more of the symbol information.

When trace information is displayed in mnemonic format, you can additionally specify the width of symbols in the "Opcode or Status" column.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:                  Address                     Data            Opcode or Stat
Base:                   symbols                     hex            mnemonic w/symb
+011   PROG|/users/guest/dir68k/cprg.c:forLoop1     247C  MOVEA.L #000000B0C,A2
+012   PROG|use/guest/dir68k/cprg.c:main+00000C     0000    0000  supr prog
+013   PROG|use/guest/dir68k/cprg.c:main+00000E     0B0C    0B0C  supr prog
+014   PROG|use/guest/dir68k/cprg.c:main+000010     26BC  MOVE.L  #00006007C,(A3
+015   PROG|use/guest/dir68k/cprg.c:main+000012     0006    0006  supr prog
+016   PROG|use/guest/dir68k/cprg.c:main+000014     007C    007C  supr prog
+017   PROG|use/guest/dir68k/cprg.c:main+000016     600C  BRA.B   cprg.c:continu
+018                DATA|dest_ptr                   0006    0006  supr data wr w
+019    DATA|/users/guest/dir68k/cprg.c:+000100     007C    007C  supr data wr w
+020   PROG|user/guest/dir68k/cprg.c:whileLoop1     1012  MOVE.B  (A2),D0
+021   PROG|users/guest/dir68k/cprg.c:continue2     4A12  TST.B   (A2)
+022   PROG|use/guest/dir68k/cprg.c:main+000026     66F0  BNE.B   cprg.:whileLoo
+023   COMM|/users/guest/dir68k/cprg.c:String1      5468    54    supr data rd b
+024   PROG|users/guest/dir68k/cprg.c:continue1     60E0  BRA.B   |cprg.c:forLoo
+025   PROG|user/guest/dir68k/cprg.c:whileLoop1     1012  MOVE.B  (A2),D0

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 set  width  label 40


pod_cmd    set    perfinit perfrun          perfend                    ---ETC--
```

Displaying Count Absolute/Relative

Count information may be displayed two ways: relative (which is the default), or absolute. When relative is selected, count information is displayed relative to the previous state. When absolute is selected, count information is displayed relative to the trigger condition.

The "count absolute/relative" trace display option is not available when counting is turned off in the trace command.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data            Opcode or Status           time count    xbits
Base:     hex     hex                 mnemonic                absolute      hex
+011     000AEA     247C  MOVEA.L #000000B0C,A2            +   5.48  uS     000
+012     000AEC     0000    0000  supr prog                +   6.00  uS     000
+013     000AEE     0B0C    0B0C  supr prog                +   6.48  uS     000
+014     000AF0     26BC  MOVE.L  #00006007C,(A3)          +   7.00  uS     000
+015     000AF2     0006    0006  supr prog                +   7.48  uS     000
+016     000AF4     007C    007C  supr prog                +   8.00  uS     000
+017     000AF6     600C  BRA.B   0000B04                  +   8.48  uS     000
+018     06017A     0006    0006  supr data wr word        +   9.00  uS     000
+019     06017C     007C    007C  supr data wr word        +   9.48  uS     000
+020     000AF8     1012  MOVE.B  (A2),D0                  +  10.0   uS     000
+021     000B04     4A12  TST.B   (A2)                     +  10.7   uS     000
+022     000B06     66F0  BNE.B   0000AF8                  +  11.2   uS     000
+023     000B0C     5468    54    supr data rd byte        +  11.7   uS     000
+024     000B08     60E0  BRA.B   0000AEA                  +  12.2   uS     000
+025     000AF8     1012  MOVE.B  (A2),D0                  +  13.0   uS     000

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 set  symbols  off  width  label 8; display  trace  count  absolute


pod_cmd    set    perfinit perfrun          perfend                    ---ETC--
```

Offsetting Address Information

The "offset_by" trace display option allows you to cause the address information in the trace display to be offset by the amount specified. The offset value is subtracted from the instruction's physical address to yield the address that is displayed.

By specifying an offset, you can still display symbols and source lines (providing they're turned on) after a program gets relocated.

You can also specify an offset to cause the listed addresses to match the addresses in compiler or assembler listings.

```
Trace List                  Offset=A00    More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data            Opcode or Status           time count    xbits
Base:     hex     hex                 mnemonic                absolute     hex
+011     0000EA     247C  MOVEA.L #000000B0C,A2            +   5.48  uS     000
+012     0000EC     0000    0000  supr prog                +   6.00  uS     000
+013     0000EE     0B0C    0B0C  supr prog                +   6.48  uS     000
+014     0000F0     26BC  MOVE.L  #00006007C,(A3)          +   7.00  uS     000
+015     0000F2     0006    0006  supr prog                +   7.48  uS     000
+016     0000F4     007C    007C  supr prog                +   8.00  uS     000
+017     0000F6     600C  BRA.B   0000104                  +   8.48  uS     000
+018     05F77A     0006    0006  supr data wr word        +   9.00  uS     000
+019     05F77C     007C    007C  supr data wr word        +   9.48  uS     000
+020     0000F8     1012  MOVE.B  (A2),D0                  +  10.0   uS     000
+021     000104     4A12  TST.B   (A2)                     +  10.7   uS     000
+022     000106     66F0  BNE.B   00000F8                  +  11.2   uS     000
+023     00010C     5468    54    supr data rd byte        +  11.7   uS     000
+024     000108     60E0  BRA.B   00000EA                  +  12.2   uS     000
+025     0000F8     1012  MOVE.B  (A2),D0                  +  13.0   uS     000

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  trace  offset_by 0a00h


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

Returning to the Default Trace Display

The "set default" command allows you to return to the default 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
+011     000AEA     247C  MOVEA.L #000000B0C,A2                            480     nS
+012     000AEC     0000    0000  supr prog                                520     nS
+013     000AEE     0B0C    0B0C  supr prog                                480     nS
+014     000AF0     26BC  MOVE.L  #00006007C,(A3)                          520     nS
+015     000AF2     0006    0006  supr prog                                480     nS
+016     000AF4     007C    007C  supr prog                                520     nS
+017     000AF6     600C  BRA.B   0000B04                                  480     nS
+018     06017A     0006    0006  supr data wr word                        520     nS
+019     06017C     007C    007C  supr data wr word                        480     nS
+020     000AF8     1012  MOVE.B  (A2),D0                                  520     nS
+021     000B04     4A12  TST.B   (A2)                                     720     nS
+022     000B06     66F0  BNE.B   0000AF8                                  520     nS
+023     000B0C     5468    54    supr data rd byte                        480     nS
+024     000B08     60E0  BRA.B   0000AEA                                  520     nS
+025     000AF8     1012  MOVE.B  (A2),D0                                  760     nS

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 set  default


pod_cmd    set    perfinit perfrun          perfend                    ---ETC--
```

Displaying External Analyzer Information

The "external" trace display option allows you to include data from the external analyzer in the trace list. External bits are displayed by default if your emulator contains an external analyzer. If you do not wish to have the external bits information in the display, you can turn them off.

The bits associated with the external analyzer labels may be displayed in binary or hexadecimal format.

The following display shows three external labels (< CTRL> -F was entered to scroll the screen left). Labels must be defined in the external analyzer configuration (and prior to trace command entry) before they appear as softkey selections when displaying the trace. Refer to the "Defining External Labels" section in the "Using the External Analyzer" chapter.

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:     Opcode or Status               time count    xbits   hi_byte low_byte
Base:          mnemonic                    relative     hex     binary   binary
+011   #000000B0C,A2                       480     nS     0000 00000000 00000000
+012   supr prog                           520     nS     0000 00000000 00000000
+013   supr prog                           480     nS     0000 00000000 00000000
+014   #00006007C,(A3)                     520     nS     0000 00000000 00000000
+015   supr prog                           480     nS     0000 00000000 00000000
+016   supr prog                           520     nS     0000 00000000 00000000
+017   0000B04                             480     nS     0000 00000000 00000000
+018   supr data wr word                   520     nS     0000 00000000 00000000
+019   supr data wr word                   480     nS     0000 00000000 00000000
+020   (A2),D0                             520     nS     0000 00000000 00000000
+021   (A2)                                720     nS     0000 00000000 00000000
+022   0000AF8                             520     nS     0000 00000000 00000000
+023   supr data rd byte                   480     nS     0000 00000000 00000000
+024   0000AEA                             520     nS     0000 00000000 00000000
+025   (A2),D0                             760     nS     0000 00000000 00000000

STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  trace  external  xbits  hex  then  hi_byte  binary  then  low_byte  bi
nary

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

Trace Status Display

In addition to the trace display options mentioned in this chapter, you can display analyzer status with the command below.

display status <RETURN>

The trace status information displayed with this command is the same as displayed with the pod command "ts". Refer to the *Terminal Interface: Analyzer User's Guide* for a complete description of this status information.

```
Status

Emulator Status

  M68000--Running user program

Trace Status

  Emulation trace complete
  Arm ignored
  Trigger in memory
  Arm to trigger ?
  States 512 (512) -1..510
  Sequence term 2
  Occurrence left 1



STATUS:   M68000--Running user program    Emulation trace complete______...R....
 display  status


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

4

Making Software Performance Measurements

Overview

This chapter:

- Introduces you to the Software Performance Measurement Tool (SPMT) and describes the types of measurements you can make with it.
- Describes the process of using the SPMT.
- Shows you how to make SPMT measurements on the supplied demo program.

Introduction

The Software Performance Measurement Tool (SPMT) is a feature of the Softkey Interface that allows you to make software performance measurements on your programs.

The SPMT post-processes information from the analyzer trace list. When you end a performance measurement, the SPMT dumps the post-processed information to a binary file, which is then read using the **perf32** report generator utility.

Two types of software performance measurements can be made with the SPMT: activity measurements, and duration measurements.

Activity Measurements

Activity measurements are measurements of the number of accesses (reads or writes) within an address range. The SPMT shows you the percentage of analyzer trace states that are in the specified address range, as well as the percentage of time taken by those states. Two types of activity are measured: memory activity, and program activity.

Memory Activity

Memory activity is all activity that occurs within the address range.

Program Activity

Program activity is the activity caused by instruction execution in the address range. Program activity includes opcode fetches and the cycles that result from the execution of those instructions (reads and writes to memory, stack pushes, etc.).

For example, suppose an address range being measured for activity contains an opcode that causes a stack push, which results in multiple write operations to the stack area (outside the range). The memory activity measurement will count only the stack push opcode cycle. However, the program activity measurement will count the stack push opcode cycle and the write operations to the stack.

By comparing the program activity and the memory activity in an address range, you can get an idea of how much activity in other areas is caused by the code being measured. An activity measurement report of the code (prog), data, and stack sections of a program is shown in figure 4-1.

Duration Measurements

Duration measurements provide a best-case/worst-case characterization of code execution time. These measurements record execution times that fall within a set of specified time ranges. The analyzer trace command is set up to store only the entry and exit states of the module to be measured (for example, a C function or Pascal procedure). The SPMT provides two types of duration measurements: module duration, and module usage.

```
 Label

prog
      Address Range       ADEH thru     1261H


      Memory Activity
           State Percent  Rel =  57.77  Abs =  57.77
                         Mean = 295.80  Sdv =  26.77
           Time  Percent  Rel =  60.97  Abs =  60.97

      Program Activity
           State Percent  Rel =  99.82  Abs =  99.82
                         Mean = 511.10  Sdv =   0.88
           Time  Percent  Rel =  99.84  Abs =  99.84

data
      Address Range     6007AH thru    603A5H


      Memory Activity
           State Percent  Rel =  30.51  Abs =  30.51
                         Mean = 156.20  Sdv =  31.87
           Time  Percent  Rel =  28.09  Abs =  28.09

      Program Activity
           State Percent  Rel =   0.18  Abs =   0.18
                         Mean =   0.90  Sdv =   0.88
           Time  Percent  Rel =   0.16  Abs =   0.16

stack
      Address Range     40000H thru    43FFFH


      Memory Activity
           State Percent  Rel =  11.72  Abs =  11.72
                         Mean =  60.00  Sdv =  29.24
           Time  Percent  Rel =  10.94  Abs =  10.94

      Program Activity
           State Percent  Rel =   0.00  Abs =   0.00
                         Mean =   0.00  Sdv =   0.00
           Time  Percent  Rel =   0.00  Abs =   0.00



         Graph of Memory Activity relative state percents >= 1
prog                  57.77%  *****************************
data                  30.51%  ****************
stack                 11.72%  ******
```

Figure 4-1. Memory Activity and Program Activity

```
        Graph of Memory Activity relative time percents >= 1
prog                 60.97%  *******************************
data                 28.09%  **************
stack                10.94%  ******


        Graph of Program Activity relative state percents >= 1
prog                 99.82% **************************************************


        Graph of Program Activity relative time percents >= 1
prog                 99.84% **************************************************

      Summary Information for     10 traces


           Memory Activity
           State count
                Relative count     5120
                Mean sample      170.67
                Mean Standard Dv 29.30
                95% Confidence 12.28% Error tolerance
           Time  count
                Relative Time - Us 2221.20


           Program Activity
           State count
                Relative count     5120
                Mean sample      170.67
                Mean Standard Dv 0.58
                95% Confidence 0.24% Error tolerance
           Time  count
                Relative Time - Us 2221.20
      Absolute Totals
                Absolute count - state     5120
```

Figure 4-1. Memory and Program Activity (Cont'd)

Module Duration

Module duration measurements record how much time it takes to execute a particular code segment (for example, a function in the source file).

Module Usage

Module usage shows how much of the execution time is spent outside of the module (from exit to entry). This measurement gives an indication of how often the module is being used.

Using the Software Performance Measurement Tool

Activity and duration measurements are made with the SPMT in a five-step process, as follows:

1. Set up the trace command.
2. Initialize the performance measurement.
3. Run the performance measurement.
4. End the performance measurement.
5. Generate the performance measurement report.

These five steps are described in the following paragraphs.

Setting Up the Trace Command

Before you initialize and run performance measurements, the current trace command (in other words, the last trace command entered) must be properly set up.

In general, you want to give the SPMT as many trace states as possible to post-process, so you should increase the trace depth to the maximum number, as shown in the following command.

display trace depth 512 <RETURN>

Also it is important that "time" be counted by the analyzer; otherwise, the SPMT measurements will not be correct.

Activity Measurements

If you wish to measure activity as a percentage of all activity, the current trace command should be the default (in other words, "trace counting time < RETURN> "). The default trace command triggers on any state, and all captured states are stored. Also, since states are stored "after" the trigger state, the maximum number of captured states appears in each trace list.

Using Trace Commands Other than the Default. You can qualify trace commands any way you like to obtain specific information. However, when you qualify the states that get stored in the trace memory, your SPMT results will be biased by your qualifications; the percentages shown will be of only those states stored in the trace list.

Duration Measurements

For duration measurements, the trace command must be set up to store only the entry and exit points of the module of interest. Since the trigger state is always stored, you should trigger on the entry or exit points. For example:

> ***trace after*** `symbol_entry` ***or*** `symbol_exit` ***only*** `symbol_entry` ***or*** `symbol_exit` ***counting time*** `<RETURN>`

Or:

> ***trace after*** `module_name start` ***or*** `module_name end` ***only*** `module_name start` ***or*** `module_name end` ***counting time*** `<RETURN>`

Where "symbol_entry" and "symbol_exit" are symbols from the user program. Or, where "module_name" is the name of a C function or Pascal procedure (and is listed as a procedure symbol in the global symbol display).

Initializing the Performance Measurement

After you set up the trace command, you must tell the SPMT the address ranges on which you wish to make activity measurements or the time ranges to be used in the duration measurement. This is done by initializing the performance measurement. You can initialize the performance measurement in the following ways:

- Default initialization (activity measurement using global symbols if the symbols database is loaded).

- Initialize with user-defined files (activity or duration measurement).
- Initialize with global symbols (activity measurement).
- Initialize with local symbols (activity measurement).
- Restore a previous performance measurement (if the emulation system has been exited and reentered).

Default Initialization

Entering the "performance_measurement_initialize" command with no options specifies an activity measurement. If a valid symbolic database has been loaded, the addresses of all global procedures and static symbols will be used; otherwise, a default set of ranges that cover the entire processor address range will be used.

Initialization with User Defined Ranges

You can specifically give the SPMT address or time ranges to use by placing the information in a file and entering the file name in the "performance_measurement_initialize" command. The formats for the address range file (activity measurements) and time range file (duration measurements) are described below.

Address Range File Format. Address range files may contain program symbols (procedure name or static), user defined address ranges, and comments. An example address range file is shown below.

```
# Any line which starts with a # is a comment.
# All user's labels must be preceded by a "|".

|users_label  10H 1000H
program_symbol

# A program symbol can be a procedure name or a static.  In the case of a pro-
# cedure name the range of that procedure will be used.

|users_label2 program_symbol1 -> program_symbol2

# "->" means thru.  The above will define a range which starts with symbol1
# and goes thru symbol2.  If both symbols are procedures then the range will
# be defined as the start of symbol1 thru the end of symbol2.

dir1/dir2/source_file.s:local_symbol
```

```
# The above defines a range based on the address of local_symbol.
```

Time Range File Format. Time range files may contain comments and time ranges in units of microseconds (us), milliseconds (ms), or seconds (s). An example time range file is shown below.

```
# Any line which starts with a # is a comment.

1 us 20 us
10.1 ms 100.6 ms
3.55 s  6.77 s

# us microseconds
# ms milliseconds
#  s seconds
#
# The above are the only abbreviations allowed.  The space between the  number
# and the units abbreviation is required.
```

Selecting Duration Measurements

Activity measurements are selected when the "performance_measurement_initialize" command is entered with no options, with just a file name, or with the global or local symbol options. You must enter one of the following commands to select a duration measurement.

performance_measurement_initialize duration <RETURN>

performance_measurement_initialize <FILE> duration <RETURN>

When no user defined time range file is specified, the following set of default time ranges are used.

```
1 us 10 us
10.1 us 100 us
100.1 us 500 us
500.1 us 1 ms
1.001 ms 5 ms
5.001 ms 10 ms
10.1 ms 20 ms
20.1 ms 40 ms
40.1 ms 80 ms
80.1 ms 160 ms
160.1 ms 320 ms
320.1 ms 640 ms
640.1 ms 1.2 s
```

Initialization with Global Symbols

When the "performance_measurement_initialize" command is entered with no options or with the "global_symbols" option, the global symbols in the symbols database become the address ranges for which activity is measured. If the symbols database is not loaded, a default set of ranges that cover the entire processor address range will be used.

The global symbols database contains procedure symbols, which are associated with the address range from the beginning of the procedure to the end, and static symbols, which are associated with the address of the static variable.

Initialization with Local Symbols

When the "performance_measurement_initialize" command is entered with the "local_symbols_in" option and a source file name, the symbols associated with that source file become the address ranges for which activity is measured. If the symbols database is not loaded, an error message will occur telling you that the source filename symbol was not found.

You can also use the "local_symbols_in" option with procedure symbols; this allows you to measure activity related to the symbols defined in a single function or procedure.

Below are example commands showing performance measurement initialization with local symbols.

performance_measurement_initialize
local_symbols_in spmt_demo.C: <RETURN>

performance_measurement_initialize
local_symbols_in spmt_demo.C:math_library
<RETURN>

performance_measurement_initialize
local_symbols_in math_library <RETURN>

Restoring the Current Measurement

The "performance_measurement_initialize restore" command allows you to restore old performance measurement data from the **perf.out** file in the current directory.

If you have not exited and reentered emulation, you can add traces to a performance measurement simply by entering another "performance_measurement_run" command. However, if you exit and reenter the emulation system, you must enter the "performance_measurement_initialize restore" command before you can add traces to a performance measurement. When you restore a performance measurement, **make sure your current trace command is identical to the command used with the restored measurement**.

The "restore" option checks the emulator software version and will only work if the **perf.out** files you are restoring were made with the same software version as is presently running in the emulator. If you ran tests using a former software version and saved **perf.out** files, then updated your software to a new version number, you will not be able to restore old **perf.out** measurement files.

Running the Performance Measurement

The "performance_measurement_run" command processes analyzer trace data. When you end the performance measurement, this processed data is dumped to the binary "perf.out" file in the current directory. The **perf32** report generator utility is used to read the binary information in the "perf.out" file.

If the "performance_measurement_run" command is entered without a count, the current trace data is processed. If a count is specified, the current trace command is executed consecutively the number of times specified. The data that results from each trace command is processed and combined with the existing processed data. The STATUS line will say "Processing trace < NO.> " during the run so you will know how your measurement is progressing. The only way to stop this series of traces is by using **< CTRL> -C** (sig INT).

The more traces you include in your sample, the more accurate will be your results. At least four consecutive traces are required to obtain statistical interpretation of activity measurement results.

Ending the Performance Measurement

The "performance_measurement_end" command takes the data generated by the "performance_measurement_run" command and places it in a file named **perf.out** in the current directory. If a file named "perf.out" already exists in the current directory, it will be overwritten. Therefore, if you wish to save a performance measurement, you must rename the **perf.out** file before performing another measurement.

The "performance_measurement_end" command does not affect the current performance measurement data which exists within the emulation system. In other words, you can add more traces later to the existing performance measurement by entering another "performance_measurement_run" command.

Once you have entered the "performance_measurement_end" command, you can use the **perf32** report generator to look at the data saved in the **perf.out** file.

Note

The "perf.out" file is a binary file. Do not try to read it with the HP-UX **more** or **cat** commands. The **perf32** report generator utility (described in the following section) must be used to read the contents of the "perf.out" file.

Using the "perf32" Report Generator

The **perf32** report generator utility must be used to read the information in the "perf.out" file and other files dumped by the SPMT (in other words, renamed "perf.out" files). The **perf32** utility is run from the HP-UX shell. You can fork a shell while in the Softkey Interface and run **perf32**, or you can exit the Softkey Interface and run **perf32**.

Options to "perf32"

A default report, containing all performance measurement information, is generated when the **perf32** command is used without any options. The options available with **perf32** allow you to limit the information in the generated report. These options are described below.

-h Produce outputs limited to histograms.

-s Produce a summary limited to the statistical data.

-p Produce a summary limited to the program activity.

-m Produce a summary limited to the memory activity.

-f< file> Produce a report based on the information contained in < file> instead of the information contained in perf.out.

For example, the following commands save the current performance measurement information in a file called "perf1.out", and produce a histogram showing only the program activity occupied by the functions and variables.

```
mv perf.out perf1.out <RETURN>
perf32 -hpf perf1.out <RETURN>
```

Options **-h**, **-s**, **-p**, and **-m** affect the contents of reports generated for activity measurements. These options have no effect on the contents of reports generated for duration (time interval) measurements.

Interpreting Reports of Activity Measurements

Activity measurements are measurements of the number of accesses (reads or writes) within an address range. The reports generated for activity measurements show you the percentage of analyzer trace states that are in the specified address range, as well as the percentage of time taken by those states. The performance measurement must include four traces before statistics (mean and standard deviation) appear in the activity report. The information you will see in activity measurement reports is described below.

Memory Activity. All activity found within the address range.

Program Activity. All activity caused by instruction execution in the address range. Program activity includes opcode fetches and the cycles that result from the execution of those instructions (reads and writes to memory, stack pushes, etc.).

Relative. With respect to activity in all ranges defined in the performance measurement.

Absolute. With respect to all activity, not just activity in those ranges defined in the performance measurement.

Mean. Average number of states in the range per trace. The following equation is used to calculate the mean:

$$mean = \frac{states\ \ in\ \ range}{total\ \ states}$$

Standard Deviation.

$$std\ dev = \sqrt{\frac{1}{N-1} \times \sum_{i=1}^{N} S_{\mathrm{sumq}} - N\ (mean)^2}$$

Deviation from the mean of state count. The following equation is used to calculate standard deviation:

Where:

N	Number of traces in the measurement.
mean	Average number of states in the range per trace.
S_{sumq}	Sum of squares of states in the range per trace.

Symbols Within Range. Names of other symbols that identify addresses or ranges of addresses within the range of this symbol.

Additional Symbols for Address. Names of other symbols that also identify this address.

Note

Some compilers emit more than one symbol for certain addresses. For example, a compiler may emit "math_library" and "_math_library" for the first address in a routine named math_library. The analyzer will show the first symbol it finds to represent a range of addresses, or a single address point, and it will show the other symbols under either "Symbols within range" or "Additional symbols for address", as applicable. In the "math_library" example, it may show either "math_library" or "_math_library" to represent the range, depending on which symbol it finds first. The other symbol will be shown below "Symbols within range" in the report. These conditions appear particularly in default measurements that include all global and local symbols.

Relative and Absolute Counts. Relative count is the total number of states associated with the address ranges in the performance measurement. Relative time is the total amount of time associated with the address ranges in the performance measurement. The absolute counts are the number of states or amount of time associated with all the states in all the traces.

Error Tolerance and Confidence Level. An approximate error may exist in displayed information. Error tolerance for a level of confidence is calculated using the mean of the standard deviations and the mean of the means. Error tolerance gives an indication of the stability of the information. For example, if the error is 5% for a confidence level of 95%, then you can be 95% confident that the information has an error of 5% or less.

The Student's "T" distribution is used in these calculations because it improves the accuracy for small samples. As the size of the sample increases, the Student's "T" distribution approaches the normal distribution.

The following equation is used to calculate error tolerance:

$$error\ pct. = \frac{O_m \times t}{N \times P_m} \times 100$$

Where:

O_m	Mean of the standard deviations.
t	Table entry in Student's "T" table for a given confidence level.
N	Number of traces in the measurement.
P_m	Mean of the means (i.e., mean sample).

Interpreting Reports of Duration Measurements

Duration measurements provide a best-case/worst-case characterization of code execution time. These measurements record execution times that fall within a set of specified time ranges. The information you will see in duration measurement reports is described below.

Number of Intervals. Number of "from address" and "to address" pairs (after prefetch correction).

Maximum Time. The greatest amount of time between the "from address" to the "to address".

Minimum Time. The shortest amount of time between the "from address" to the "to address".

Average Time. Average time between the "from address" and the "to address". The following equation is used to calculate the average time:

$$mean = \frac{amount\ of\ time\ \text{for}\ all\ intervals}{number\ of\ intervals}$$

Standard Deviation. Deviation from the mean of time. The following equation is used to calculate standard deviation:

$$std\ dev = \sqrt{\frac{1}{N-1} \times \sum_{i=1}^{N} S_{sumq} - N\,(mean)^2}$$

Where:

N	Number of intervals.
mean	Average time.
S_{sumq}	Sum of squares of time in the intervals.

Error Tolerance and Confidence Level. An approximate error may exist in displayed information. Error tolerance for a level of confidence is calculated using the mean of the standard deviations and the mean of the means. Error tolerance gives an indication of the stability of the information. For example, if the error is 5% for a confidence level of 95%, then you can be 95% confident that the information has an error of 5% or less.

The Student's "T" distribution is used in these calculations because it improves the accuracy for small samples. As the size of the sample increases, the Student's "T" distribution approaches the normal distribution.

The following equation is used to calculate error tolerance:

$$error\ pct. = \frac{O_m \times t}{N \times P_m} \times 100$$

Where:

O_m	Mean of the standard deviations in each time range.
t	Table entry in Student's "T" table for a given confidence level.

N	Number of intervals.
P_m	Mean of the means (i.e., mean of the average times in each time range).

Examples

This section:

- Describes the SPMT demo program.
- Performs an example activity measurement on the demo program and describes the results.
- Performs an example duration measurement on the demo program and describes the results.

The SPMT Demo Program

The SPMT demo program is a C program that has been supplied with your Softkey Interface. This program is used in examples in this chapter to illustrate the SPMT. A diagram of the function calls in the demo program is shown in figure 4-2.

Refer to your compiler documentation for information on compiling the demo program and to your *Emulator Softkey Interface User's Guide* for information on configuring the emulator and loading and executing programs.

Generally, you perform the following steps before using the SPMT to make software performance measurements.

- Compile the demo program.
- Enter the emulation system and configure the emulator (map memory, restrict to real-time).
- Load and run the demo program.

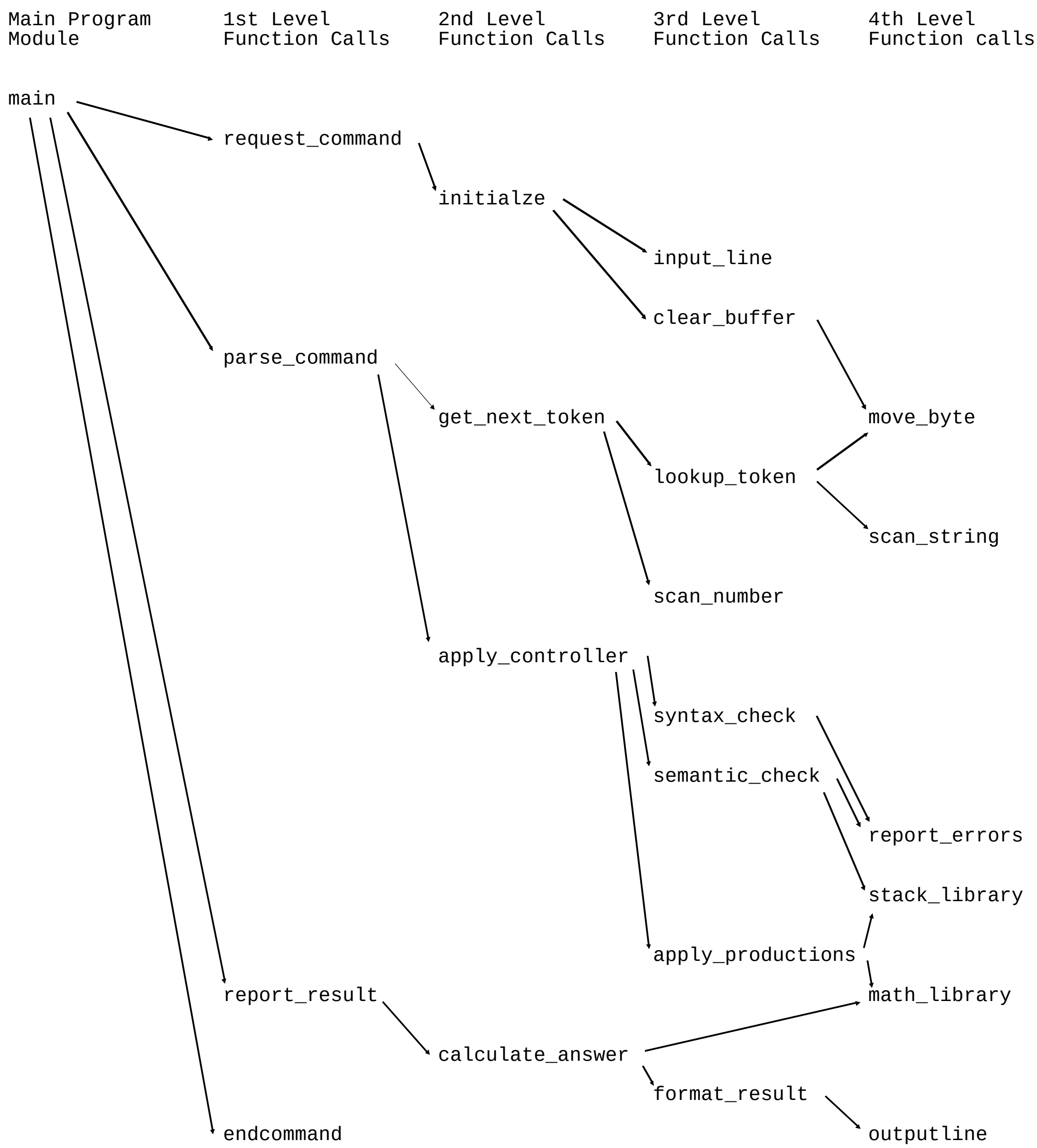


Figure 4-2. Demo Program Function Calls

Example of Compiling and Executing the Demo Program

If you wish to step through the examples in this chapter, you must have the appropriate C compiler for your particular emulator. You do not have to step through the examples to learn how the SPMT works; reading through the examples should be sufficient.

Note

The following procedure for compiling the SPMT demo program (with the HP 64902 C Cross Compiler), modifying the emulator configuration, and loading and running the sample program is for the HP 64742 68000 Emulator.

Most likely, there will be differences when compiling, configuring, and loading for other HP 64700 Series emulators.

Copying the Demo Program. The demo program can be copied with the following command.

```
$ cp
/usr/hp64000/demo/emul/hp64742/spmt_demo.c
spmt_demo.c <RETURN>
```

Compiling the Demo Program. The absolute file used to generate the SPMT examples shown later in this chapter was generated with the following HP 64902 68000 C Cross Compiler command:

```
$ cc68000 -hvOGNr hp64742 -o spmt_demo
spmt_demo.c <RETURN>
```

Copying the Default Emulator Configuration File. Since the HP 64902 68000 C Cross Compiler provides default configuration files for the HP 64742 68000 Emulator, copy the default emulator configuration file to the current directory before you enter the emulation system.

```
$ cp /usr/hp64000/env/hp64742/config.EA
config.EA <RETURN>
```

To configure the emulator to restrict to real-time runs, edit the default configuration file:

```
$ chmod 644 config.EA <RETURN>
$ vi config.EA <RETURN>
```

Add a line which reads "Restrict to real-time runs? yes" just before the memory map definition, save your changes, and exit out of the editor.

Entering the Emulation System. If you have installed your emulator and Softkey Interface software as directed in the *HP 64700-Series Emulators Softkey Interface Installation Notice*, you can enter the emulation system.

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

```
$ emul700 <emul_name> <RETURN>
```

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

Configuring the Emulator. Once you have entered the emulation system, you can load the default emulator configuration (copied and modified earlier) with the following command:

```
load configuration config <RETURN>
```

Loading the Demo Program. Enter the following command to load the demo program:

```
load spmt_demo <RETURN>
```

Running the Demo Program. Finally, to run the demo program, enter the following command:

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

Activity Measurement Example

The following examples assume that the SPMT demo program has been loaded into the emulator and is executing.

display trace depth 512 <RETURN>

trace counting time <RETURN>

performance_measurement_initialize addr_ranges <RETURN>

The "addr_ranges" file contains the names of all the functions in the demo program:

```
apply_controlle
apply_productio
calculate_answe
clear_buffer
endcommand
format_result
get_next_token
initialze
input_line
lookup_token
math_library
move_byte
outputline
parse_command
report_errors
report_result
request_command
scan_number
scan_string
semantic_check
stack_library
syntax_check
```

Since these labels are program symbols, you do not have to specify the address range associated with each label; the SPMT will search the symbol database for the addresses of each label.

An easy way to create the "addr_ranges" file is to use the "copy global_symbols" command to copy the global symbols to a file named "addr_ranges"; then, fork a shell to HP-UX (by entering "! < RETURN> " on the Softkey Interface command line) and edit the file so that it contains the procedure names shown in figure 4-2. Enter a **< CTRL> -D** at the HP-UX prompt to return to the Softkey Interface.

To run the performance measurement, enter the following command:

performance_measurement_run 20 <RETURN>

The command above causes 20 traces to occur. The SPMT processes the trace information after each trace, and the number of the trace being processed is shown on the status line.

Enter the following command to cause the processed trace information to be dumped to the "perf.out" file.

performance_measurement_end <RETURN>

Now, to generate a report from the "perf.out" file, type the following on the command line to fork a shell and run the **perf32** utility:

!perf32 | more

Information similar to the listing in figure 4-3 is scrolled onto your display.

```
 Label

math_library
      Address Range       C54H thru      CA6H


      Memory Activity
           State Percent  Rel =  41.31  Abs =  23.72
                         Mean = 121.45  Sdv = 105.82
           Time  Percent  Rel =  41.28  Abs =  25.00

      Program Activity
           State Percent  Rel =  46.60  Abs =  46.49
                         Mean = 238.05  Sdv = 206.72
           Time  Percent  Rel =  46.12  Abs =  46.00

apply_productio
      Address Range       E5CH thru      ED4H


      Memory Activity
           State Percent  Rel =  13.20  Abs =   7.58
                         Mean =  38.80  Sdv =  38.34
           Time  Percent  Rel =  14.03  Abs =   8.50

      Program Activity
           State Percent  Rel =  11.16  Abs =  11.13
                         Mean =  57.00  Sdv =  59.68
           Time  Percent  Rel =  11.89  Abs =  11.86

scan_string
      Address Range       B5CH thru      B98H


      Memory Activity
           State Percent  Rel =  10.54  Abs =   6.05
                         Mean =  31.00  Sdv =  72.87
           Time  Percent  Rel =  10.22  Abs =   6.19

      Program Activity
           State Percent  Rel =   9.83  Abs =   9.80
                         Mean =  50.20  Sdv = 117.57
           Time  Percent  Rel =   9.67  Abs =   9.65

move_byte
      Address Range       B1EH thru      B5AH


      Memory Activity
           State Percent  Rel =   9.13  Abs =   5.24
                         Mean =  26.85  Sdv =  62.77
           Time  Percent  Rel =   8.87  Abs =   5.37

      Program Activity
           State Percent  Rel =   8.49  Abs =   8.47
                         Mean =  43.35  Sdv = 101.60
           Time  Percent  Rel =   8.37  Abs =   8.35
```

Figure 4-3. Example Activity Measurement

```
stack_library
      Address Range       C16H thru      C52H


      Memory Activity
           State Percent  Rel =   8.45  Abs =   4.85
                         Mean =  24.85  Sdv =  42.28
           Time  Percent  Rel =   8.08  Abs =   4.89

      Program Activity
           State Percent  Rel =   8.13  Abs =   8.12
                         Mean =  41.55  Sdv =  70.29
           Time  Percent  Rel =   7.91  Abs =   7.89

initialze
      Address Range       F24H thru      F88H


      Memory Activity
           State Percent  Rel =   3.40  Abs =   1.95
                         Mean =  10.00  Sdv =  44.72
           Time  Percent  Rel =   3.40  Abs =   2.06

      Program Activity
           State Percent  Rel =   3.21  Abs =   3.20
                         Mean =  16.40  Sdv =  73.34
           Time  Percent  Rel =   3.22  Abs =   3.22

syntax_check
      Address Range       DA8H thru      DF4H


      Memory Activity
           State Percent  Rel =   3.37  Abs =   1.93
                         Mean =   9.90  Sdv =  44.04
           Time  Percent  Rel =   3.36  Abs =   2.03

      Program Activity
           State Percent  Rel =   3.17  Abs =   3.16
                         Mean =  16.20  Sdv =  71.75
           Time  Percent  Rel =   3.18  Abs =   3.17

report_errors
      Address Range       BD8H thru      C14H


      Memory Activity
           State Percent  Rel =   2.45  Abs =   1.41
                         Mean =   7.20  Sdv =  23.49
           Time  Percent  Rel =   2.35  Abs =   1.42


      Program Activity
           State Percent  Rel =   2.35  Abs =   2.34
                         Mean =  12.00  Sdv =  39.15
           Time  Percent  Rel =   2.30  Abs =   2.29
```

Figure 4-3. Example Activity Measurement (Cont'd)

```
lookup_token
      Address Range       D42H thru      DA6H


      Memory Activity
           State Percent  Rel =   2.07  Abs =   1.19
                         Mean =   6.10  Sdv =  14.29
           Time  Percent  Rel =   2.20  Abs =   1.33

      Program Activity
           State Percent  Rel =   1.55  Abs =   1.54
                         Mean =   7.90  Sdv =  19.78
           Time  Percent  Rel =   1.66  Abs =   1.66

semantic_check
      Address Range       DF6H thru      E5AH


      Memory Activity
           State Percent  Rel =   1.99  Abs =   1.14
                         Mean =   5.85  Sdv =  25.70
           Time  Percent  Rel =   2.04  Abs =   1.23

      Program Activity
           State Percent  Rel =   1.76  Abs =   1.76
                         Mean =   9.00  Sdv =  38.63
           Time  Percent  Rel =   1.81  Abs =   1.80

apply_controlle
      Address Range       FF0H thru     1072H


      Memory Activity
           State Percent  Rel =   1.80  Abs =   1.04
                         Mean =   5.30  Sdv =   7.73
           Time  Percent  Rel =   1.95  Abs =   1.18

      Program Activity
           State Percent  Rel =   1.16  Abs =   1.15
                         Mean =   5.90  Sdv =   8.52
           Time  Percent  Rel =   1.32  Abs =   1.32

request_command
      Address Range      10E6H thru     1132H


      Memory Activity
           State Percent  Rel =   0.61  Abs =   0.35
                         Mean =   1.80  Sdv =   8.05
           Time  Percent  Rel =   0.61  Abs =   0.37

      Program Activity
           State Percent  Rel =   0.59  Abs =   0.59
                         Mean =   3.00  Sdv =  13.42
           Time  Percent  Rel =   0.59  Abs =   0.59
```

Figure 4-3. Example Activity Measurement (Cont'd)

```
outputline
      Address Range       CA8H thru      CF2H


      Memory Activity
           State Percent  Rel =   0.37  Abs =   0.21
                          Mean =  1.10  Sdv =   0.97
           Time  Percent  Rel =   0.33  Abs =   0.20

      Program Activity
           State Percent  Rel =   0.71  Abs =   0.71
                          Mean =  3.65  Sdv =   3.23
           Time  Percent  Rel =   0.66  Abs =   0.66

format_result
      Address Range       ED6H thru      F22H


      Memory Activity
           State Percent  Rel =   0.37  Abs =   0.21
                          Mean =  1.10  Sdv =   3.77
           Time  Percent  Rel =   0.37  Abs =   0.22

      Program Activity
           State Percent  Rel =   0.42  Abs =   0.42
                          Mean =  2.15  Sdv =   6.02
           Time  Percent  Rel =   0.41  Abs =   0.41

calculate_answe
      Address Range      1074H thru     10E4H


      Memory Activity
           State Percent  Rel =   0.31  Abs =   0.18
                          Mean =  0.90  Sdv =   4.02
           Time  Percent  Rel =   0.31  Abs =   0.19

      Program Activity
           State Percent  Rel =   0.27  Abs =   0.27
                          Mean =  1.40  Sdv =   6.26
           Time  Percent  Rel =   0.28  Abs =   0.28

report_result
      Address Range      119AH thru     11ECH


      Memory Activity
           State Percent  Rel =   0.31  Abs =   0.18
                          Mean =  0.90  Sdv =   4.02
           Time  Percent  Rel =   0.32  Abs =   0.19

      Program Activity
           State Percent  Rel =   0.27  Abs =   0.27
                          Mean =  1.40  Sdv =   6.26
           Time  Percent  Rel =   0.29  Abs =   0.28
```

Figure 4-3. Example Activity Measurement (Cont'd)

```
get_next_token
      Address Range       F8AH thru      FEEH


      Memory Activity
           State Percent  Rel =   0.15  Abs =   0.09
                         Mean =   0.45  Sdv =   2.01
           Time  Percent  Rel =   0.17  Abs =   0.10

      Program Activity
           State Percent  Rel =   0.09  Abs =   0.09
                         Mean =   0.45  Sdv =   2.01
           Time  Percent  Rel =   0.10  Abs =   0.10

endcommand
      Address Range      11EEH thru     11F6H


      Memory Activity
           State Percent  Rel =   0.10  Abs =   0.06
                         Mean =   0.30  Sdv =   1.34
           Time  Percent  Rel =   0.09  Abs =   0.05

      Program Activity
           State Percent  Rel =   0.14  Abs =   0.14
                         Mean =   0.70  Sdv =   3.13
           Time  Percent  Rel =   0.13  Abs =   0.13

scan_number
      Address Range       B9AH thru      BD6H


      Memory Activity
           State Percent  Rel =   0.03  Abs =   0.02
                         Mean =   0.10  Sdv =   0.31
           Time  Percent  Rel =   0.03  Abs =   0.02

      Program Activity
           State Percent  Rel =   0.07  Abs =   0.07
                         Mean =   0.35  Sdv =   0.93
           Time  Percent  Rel =   0.06  Abs =   0.06

clear_buffer
      Address Range       CF4H thru      D40H


      Memory Activity
           State Percent  Rel =   0.02  Abs =   0.01
                         Mean =   0.05  Sdv =   0.22
           Time  Percent  Rel =   0.01  Abs =   0.01

      Program Activity
           State Percent  Rel =   0.03  Abs =   0.03
                         Mean =   0.15  Sdv =   0.67
           Time  Percent  Rel =   0.03  Abs =   0.03
```

Figure 4-3. Example Activity Measurement (Cont'd)

```
input_line
      Address Range       ADEH thru      B1CH


      Memory Activity
           State Percent  Rel =   0.00  Abs =   0.00
                         Mean =   0.00  Sdv =   0.00
           Time  Percent  Rel =   0.00  Abs =   0.00

      Program Activity
           State Percent  Rel =   0.00  Abs =   0.00
                         Mean =   0.00  Sdv =   0.00
           Time  Percent  Rel =   0.00  Abs =   0.00

parse_command
      Address Range      1134H thru     1198H


      Memory Activity
           State Percent  Rel =   0.00  Abs =   0.00
                         Mean =   0.00  Sdv =   0.00
           Time  Percent  Rel =   0.00  Abs =   0.00

      Program Activity
           State Percent  Rel =   0.00  Abs =   0.00
                         Mean =   0.00  Sdv =   0.00
           Time  Percent  Rel =   0.00  Abs =   0.00


         Graph of Memory Activity relative state percents >= 1
math_library          41.31%  *********************
apply_productio       13.20%  *******
scan_string           10.54%  ******
move_byte              9.13%  *****
stack_library          8.45%  ****
initialze              3.40%  **
syntax_check           3.37%  **
report_errors          2.45%  *
lookup_token           2.07%  *
semantic_check         1.99%  *
apply_controlle        1.80%  *


         Graph of Memory Activity relative time percents >= 1
math_library          41.28%  *********************
apply_productio       14.03%  *******
scan_string           10.22%  *****
move_byte              8.87%  *****
stack_library          8.08%  ****
initialze              3.40%  **
syntax_check           3.36%  **
report_errors          2.35%  *
lookup_token           2.20%  *
semantic_check         2.04%  *
apply_controlle        1.95%  *
```

Figure 4-3. Example Activity Measurement (Cont'd)

```
         Graph of Program Activity relative state percents >= 1
math_library          46.60%  ************************
apply_productio       11.16%  ******
scan_string            9.83%  *****
move_byte              8.49%  ****
stack_library          8.13%  ****
initialze              3.21%  **
syntax_check           3.17%  **
report_errors          2.35%  *
lookup_token           1.55%  *
semantic_check         1.76%  *
apply_controlle        õ.16%  *


         Graph of Program Activity relative time percents >= 1
math_library          46.12%  ***********************
apply_productio       11.89%  ******
scan_string            9.67%  *****
move_byte              8.37%  ****
stack_library          7.91%  ****
initialze              3.22%  **
syntax_check           3.18%  **
report_errors          2.30%  *
lookup_token           1.66%  *
semantic_check         1.81%  *
apply_controlle        1.32%  *

      Summary Information for     20 traces


           Memory Activity
           State count
                Relative count     5880
                Mean sample       13.36
                Mean Standard Dv 23.03
                95% Confidence 80.71% Error tolerance
           Time  count
                Relative Time - Us 2682.00


           Program Activity
           State count
                Relative count    10216
                Mean sample       23.22
                Mean Standard Dv 38.59
                95% Confidence 77.82% Error tolerance
           Time  count
                Relative Time - Us 4417.52
      Absolute Totals
                Absolute count - state    10240
                Absolute count - time - Us 4428.68
```

Figure 4-3. Example Activity Measurements (Cont'd)

The measurements for each label are printed in descending order according to the amount of activity. You can see that the "math_library" function has the most activity. Also, you can see that no activity is recorded for several of the functions. The histogram portion of the report compares the activity in the functions that account for at least 1% of the activity for all labels defined in the measurement.

Duration Measurement Examples

Before you perform duration measurements, you should be aware of the prefetch and recursion considerations associated with these measurements.

Prefetch and Recursion Considerations

When using the SPMT to perform duration measurements, there should be only two addresses stored in the trace memory: the entry address, and the exit address. Prefetches or recursion can place several entry addresses before the first exit address, and/or several exit addresses before the first entry address. Duration measurements are made between the last entry address in a series of entry addresses, and the last exit address in a series of exit addresses (see figure 4-4). All of the entry and exit addresses which precede these last addresses are assumed to be unused prefetches, and are ignored during time measurements.

```
START  - unused prefetch
START  - unused prefetch
START  - unused prefetch
START  - START actually taken -
END    - unused prefetch
END    - unused prefetch             Measure duration
END    - unused prefetch
END    - END actually taken   -
START  - unused prefetch
START  - unused prefetch             Measure duration
START  - unused prefetch
START  - START actually taken -
END    - unused prefetch
END    - unused prefetch
```

Figure 4-4. Prefetch Correction

If you are using the HP 64902 68000 C Cross Compiler, version number 2.00 or higher, prefetches will not be a problem because the debug option of the compiler inserts no-op padding ahead of the entry and exit events in the code.

If you are using any other compiler, including earlier versions of the HP 64902 68000 C Cross Compiler, the prefetches will be present. Even so, duration measurements will not be affected. The SPMT makes its duration measurements from the last start address in the series of start addresses, to the last end address in the series of end addresses. The other start and end addresses are unused prefetches and are ignored by the software of the SPMT.

Recursive procedures will still affect the accuracy of your measurements.

The prefetch correction has the following consequences:

- Prefetches are ignored. They do not affect the accuracy of the measurement in process.

- When measuring a recursive function, module duration will be measured between the last recursive call and the true end of the recursive execution. This will affect the accuracy of the measurement.

- If a module is entered at the normal point, and then exited by a point other than the defined exit point, the entry point will be ignored. It will be judged the same as any other unused prefetch, and no time-duration measurement will be made. Its time will be included in the measure of time spent outside the procedure or function.

- If a module is exited from the normal point, and then reentered from some other point, the exit will also be assumed to be an unused prefetch of the exit state.

Note

If you are making duration measurements on a function that is recursive, or one that has multiple entry and/or exit points, you may wind up with invalid information.

Example Duration Measurement

The following examples assume that the SPMT demo program has been loaded into the emulator and is executing.

display trace depth `512 <RETURN>`

trace after `math_library start` ***or*** `math_library end` ***only*** `math_library start` ***or*** `math_library end <RETURN>`

The "trace" specification sets up the analyzer to capture only the states that contain the start address of the "math_library" function or the end address of the "math_library" function. Since the trigger state is also stored, the analyzer is set up to trigger on the entry or exit address of the "math_library" function. With these states in memory, the analyzer will derive two measurements: time from start to end of math_library, and time from end to start of math_library.

Enter the following command to initialize the duration measurement:

performance_measurement_initialize `time_ranges` ***duration*** `<RETURN>`

The "time_ranges" file contains:

```
1 us 20 us
21 us 40 us
41 us 60 us
61 us 80 us
81 us 100 us
101 us 120 us
121 us 140 us
141 us 160 us
161 us 180 us
181 us 200 us
201 us 5 ms
```

You can fork a shell to HP-UX (by entering "! < RETURN> " on the Softkey Interface command line) and create the "time_ranges" file. Enter a **< CTRL> -D** at the HP-UX prompt to return to the Softkey Interface.

To run the performance measurement, enter the following command:

performance_measurement_run 10 <RETURN>

The command above causes 10 traces to occur. The SPMT processes the trace information after each trace, and the number of the trace being processed is shown on the status line.

Enter the following command to cause the processed trace information to be dumped to the "perf.out" file.

performance_measurement_end <RETURN>

Now, to generate a report from the "perf.out" file, type the following on the command line to fork a shell and run the **perf32** utility:

!perf32 | more

Information similar to the listing in figure 4-5 is scrolled onto your display.

Two sets of information are given in the duration measurement report: module duration and module usage. The first set is the "module duration" measurement. (You can tell because the "from address" is lower than the "to address".)

The module duration report in figure 4-5 shows that the average amount of time it takes for the "math_library" module to execute is roughly 82 microseconds.

Module usage measurements show how much time is spent outside the module of interest; they indicate how often the module is used. The report shown in the second part of figure 4-5 shows that there is heavy demand for the "math_library" function. In fact, the time between executions of the "math_library" module is generally less than the amount of time it takes for the "math_library" module to execute.

```
           Time Interval Profile



From Address       C54
      File /users/guest/dir68k/spmt_demo.c
      Symbolic Reference at math_library+0
To Address         CA6
      File /users/guest/dir68k/spmt_demo.c
      Symbolic Reference at math_library+52
Number of intervals   2132
Maximum Time 173.600 us
Minimum Time 22.800 us
Avg Time     81.748 us


      Statistical summary - for     10 traces
           Stdv  39.09
           95% Confidence 2.03% Error tolerance


         Graph of relative percents
1 us 20 us               0.00%
21 us 40 us             17.92%  *********
41 us 60 us             16.84%  *********
61 us 80 us             10.13%  *****
81 us 100 us            10.08%  *****
101 us 120 us           16.09%  ********
121 us 140 us           10.65%  ******
141 us 160 us            3.42%  **
161 us 180 us            3.42%  **
181 us 200 us            0.00%
201 us 5 ms              0.00%



From Address       CA6
      File /users/guest/dir68k/spmt_demo.c
      Symbolic Reference at math_library+52
To Address         C54
      File /users/guest/dir68k/spmt_demo.c
      Symbolic Reference at math_library+0
Number of intervals   2132
Maximum Time 12859.320 us
Minimum Time 11.400 us
Avg Time     82.540 us


      Statistical summary - for     10 traces
           Stdv 747.44
           95% Confidence 38.44% Error tolerance
```

Figure 4-5. Example Duration Measurement

```
         Graph of relative percents
1 us 20 us            88.27% ********************************************
21 us 40 us            0.00%
41 us 60 us            0.00%
61 us 80 us            0.00%
81 us 100 us           0.00%
101 us 120 us          0.00%
121 us 140 us          0.00%
141 us 160 us          0.00%
161 us 180 us          0.00%
181 us 200 us          0.00%
201 us 5 ms            1.13%  *
```

Figure 4-5. Example Duration Measurement (Cont'd)

Notes

5

Using the External Analyzer

Introduction

An HP 64700-Series emulator may be ordered with an external analyzer. The external analyzer provides 16 external trace channels. These trace channels allow you to capture activity on signals external to the emulator, typically other target system signals. The external analyzer may be configured as an extension to the emulation analyzer, as an independent state analyzer, or as an independent timing analyzer.

If your emulator contains an external analyzer, you can define up to eight labels for the 16 external data channels in the configuration. These external analyzer labels can be used in trace commands, and the data associated with these labels can be displayed in the trace list. One external analyzer label, "xbits", is defined by the default configuration and is included in the default trace list.

Before You Can Use the External Analyzer

There are several things to do before you can use the external analyzer; these things are listed below and explained in the following paragraphs.

- Assemble the analyzer probe.
- Connect the probe to the emulator.
- Connect the probe wires to the target system.

Assembling the Analyzer Probe

The analyzer probe is a two-piece assembly, consisting of a ribbon cable and 18 probe wires (16 data channels and the J and K clock inputs) attached to a connector. Either end of the ribbon cable may be connected to the 18-wire connector, and the connectors are keyed so that you can only attach them in one way. Align the key of the ribbon cable connector with the slot in the 18-wire connector, and firmly press the connectors together (see figure 5-1).

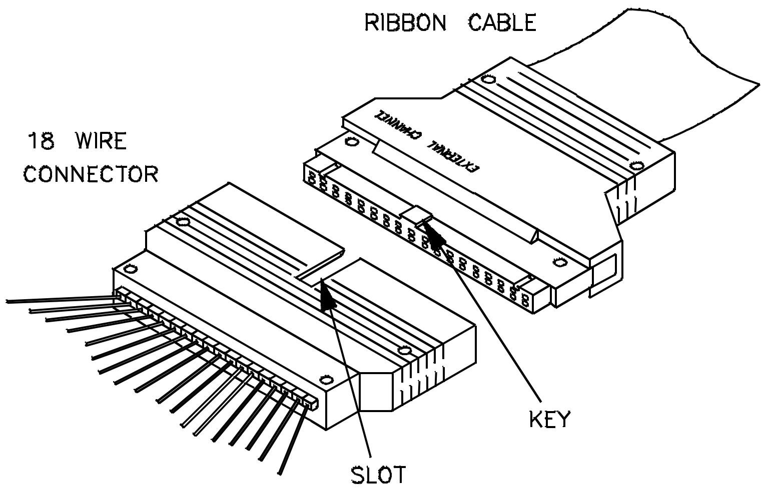


Figure 5-1. Assembling the Analyzer Probe

Each of the 18 probe wires has a signal and a ground connection. Each probe wire is labeled for easy identification. Thirty-six grabbers are provided for the signal and ground connections of each of the 18 probe wires. The signal and ground connections are attached to the pin in the grabber handle (see figure 5-2).

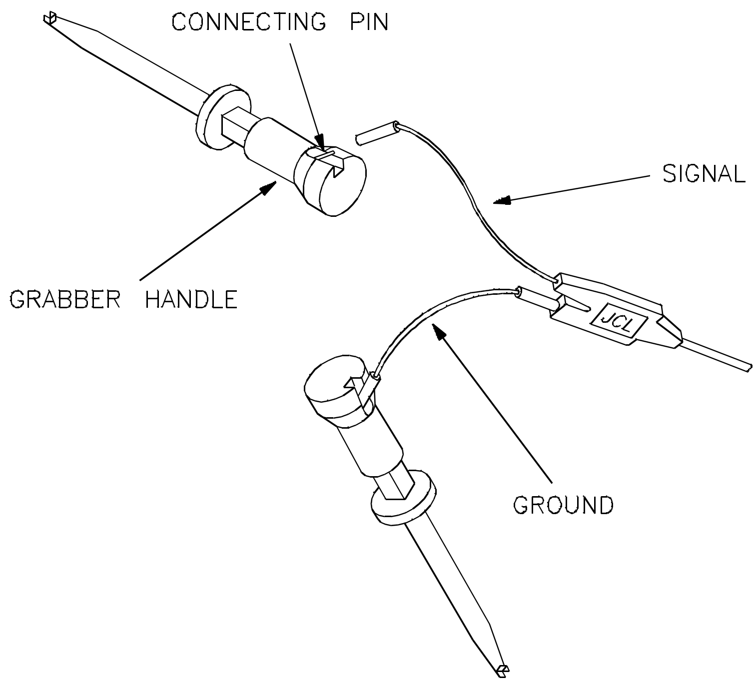


Figure 5-2. Attaching Grabbers to Probe Wires

Connecting the Probe to the Emulator

The external analyzer probe is attached to a connector under the snap-on cover in the front upper right corner of the emulator. Remove the snap-on cover by pressing the side tabs toward the center of the cover; then, pull the cover out (see figure 5-3).

Caution

Check for bent connector pins before connecting the analyzer probe to the emulator.

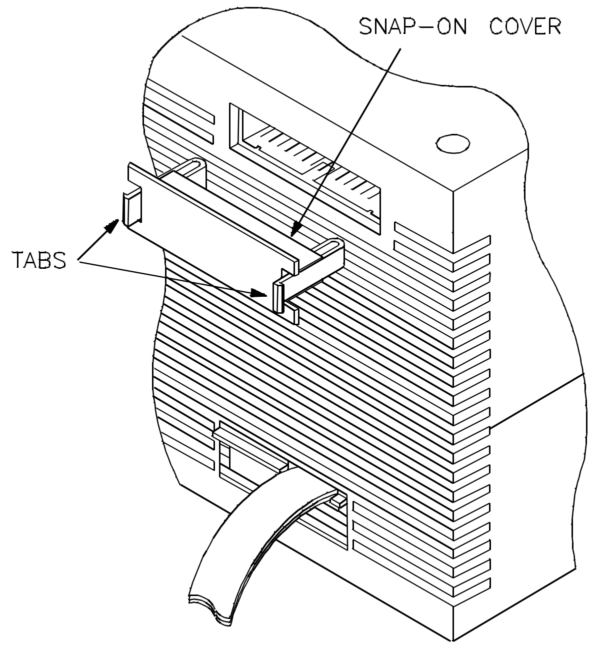


Figure 5-3. Removing Cover to Emulator Connector

Each end of the ribbon cable connector is keyed so that you can connect it to the emulator in only one way. Align the key of the ribbon cable connector with the slot in the emulator connector, and gently press the ribbon cable connector into the emulator connector (see figure 5-4).

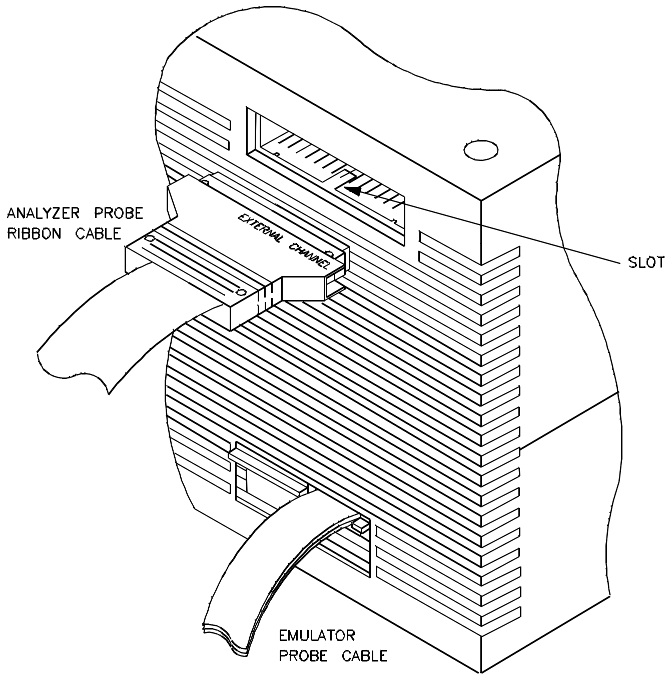


Figure 5-4. Connecting the Probe to the Emulator

Caution

Turn OFF target system power before connecting analyzer probe wires to the target system. The probe grabbers are difficult to handle with precision, and it is extremely easy to short the pins of a chip (or other connectors which are close together) with the probe wire while trying to connect it.

Connecting Probe Wires to the Target System

You can connect the grabbers to pins, connectors, wires, etc., in the target system. Pull the hilt of the grabber towards the back of the grabber handle to uncover the wire hook. When the wire hook is around the desired pin or connector, release the hilt to allow the tension of the grabber spring to hold the connection (see figure 5-5).

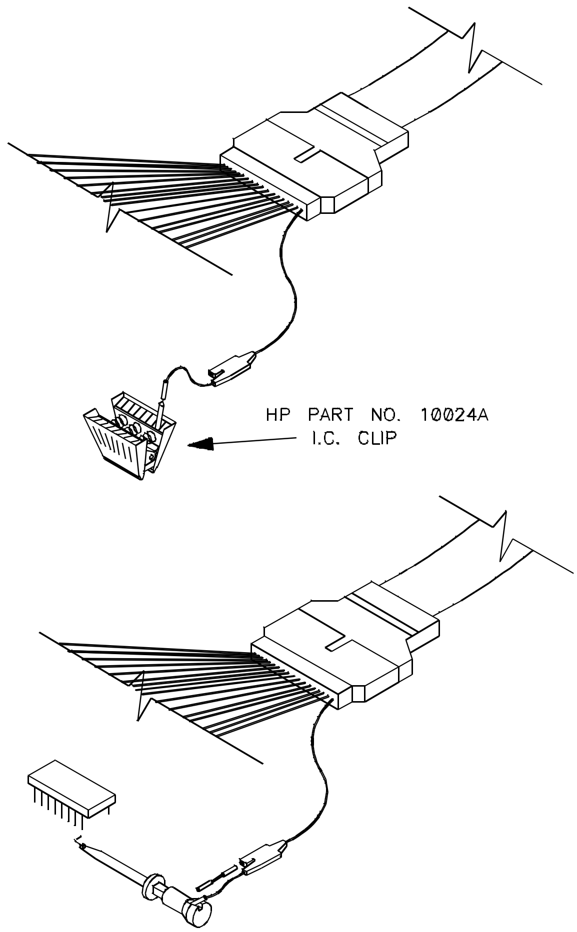


Figure 5-5. Connecting the Probe to the Target System

Configuring the External Analyzer

After you have assembled the external analyzer probe and connected it to the emulator and target system, the next step is to configure the external analyzer.

The external analyzer is a versatile instrument, and you can configure it to suit your needs. For example, you can specify threshold voltage levels on the external analyzer channels, and you can operate the external analyzer in several different modes. The external analyzer configuration options allow you to:

- Specify the threshold voltages for the external channels.
- Select the external analyzer mode.
- Specify the slave clock mode (only if the "state" external analyzer mode is selected).
- Define external analyzer labels.

The default configuration specifies that the external analyzer is aligned with the emulation analyzer. TTL level threshold voltages are defined, as well as an external label named "xbits" which contains all 16 channels.

To modify the external analyzer configuration, enter the following command:

modify configuration <RETURN>

Now, press the < RETURN> key until you see the following question. (If you pass this question, you can use the RECALL softkey to back up to it.)

Modify external analyzer configuration?

Pressing the "yes" softkey causes the external analyzer configuration questions to be asked. These questions are described in the paragraphs that follow.

Should Emulation Control the External Bits?

This configuration question allows you to specify whether the emulation interface should control the external analyzer.

yes — The default configuration selects "yes". You must answer "yes" to access the remaining external analyzer configuration questions.

At the end of the configuration process the external analyzer mode and threshold voltages will be set; existing labels will be deleted, and only the labels specified in response to the questions below will be defined.

no — If emulation does not control the external bits, the external analyzer configuration will not be modified in any way by the emulation interface.

Threshold Voltage?

The external analyzer probe signals are divided into two groups: the lower byte (channels 0 through 7 and the J clock), and the upper byte (channels 8 through 15 and the K clock). You can specify a threshold voltage for each of these groups with the configuration questions shown below.

Threshold voltage for bits 0-7 and J clock? TTL
Threshold voltage for bits 8-15 and K clock? TTL

The default threshold voltages are specified as **TTL** which translates to 1.40 volts.

Voltages may be in the range from -6.40 volts to 6.35 volts (with a 0.05V resolution). You may also specify **CMOS** (which translates to 2.5 volts), or **ECL** (which translates to -1.3 volts).

External Analyzer Mode?

This configuration question allows you to select the mode of the external analyzer.

The default configuration selects the "emulation" external analyzer mode. In this mode, you have 16 external trace signals on which data is captured synchronously with the emulation clock.

The external analyzer may also operate as an independent state analyzer, or it may operate as an independent timing analyzer if a host computer interface program is used.

emulation — Selects the emulation mode (which is the default). In this mode, the external analyzer becomes an extension of the emulation analyzer. In other words, they operate as one analyzer.

The external bits are clocked with the emulation clock. External labels may be used in trace commands to qualify trigger, storage, prestore, or count states. External labels may be viewed in the trace display.

state — Selects the independent state mode of the external analyzer. The external bits are not available for use from the emulation interface. You can, however, use pod commands to control the external state analyzer in its independent mode.

timing — Selects the timing mode of the external analyzer. The external bits are not available for use from the emulation interface. Because the pod commands for the timing analyzer dump information in binary format, you will need to use Timing Analyzer Softkey Interface, or other interface program, to capture the timing analyzer data.

Slave Clock Mode for External Bits? (State Mode Only)

There are two modes of demultiplexing that can be set for the 16 channels of the external analyzer: mixed clocks and true demultiplexing.

off — By default, the slave clocks are turned OFF. If the slave clock is "off", all 16 external bits are clocked with the emulation clock.

mixed

When the slave clock mode is "mixed", the lower eight external bits (0-7) are latched when the slave clock (as specified by your answers to the next four questions) is received. The upper eight bits and the latched lower eight are then clocked into the analyzer when the emulation clock is received (see figure 5-6).

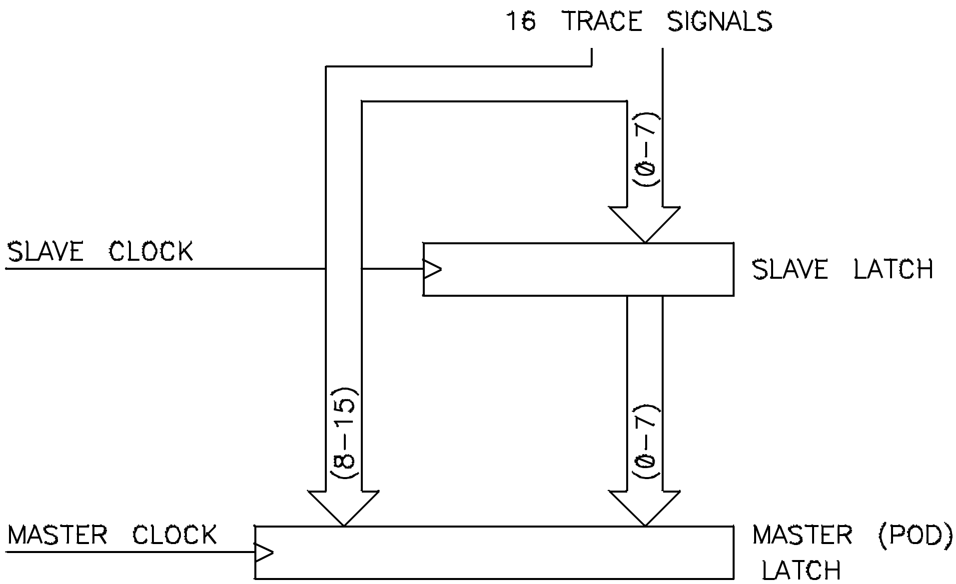


Figure 5-6. Mixed Clock Demultiplexing

If no slave clock has appeared since the last master clock, the data on the lower 8 bits of the pod will be latched at the same time as the upper 8 bits. If more than one slave clock has appeared since the last master clock, only the first slave data will be available to the analyzer (see figure 5-7).

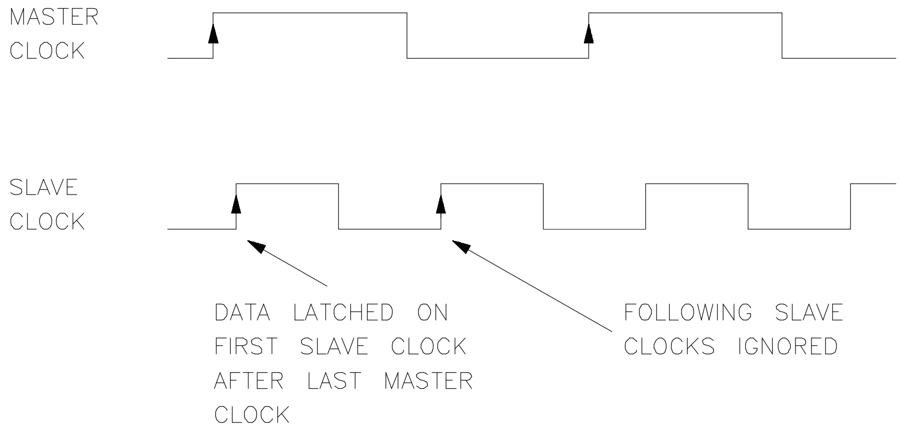


Figure 5-7. Slave Clocks

demux — When the slave clock mode is "demux", only the lower eight external channels (0-7) are used. The slave clock (as specified by your answers to the next four questions) latches these bits and the emulation clock samples the same channels again. The latched bits show up as bits 0-7 in the trace data, and the second sample shows up as bits 8-15 (see figure 5-8).

If no slave clock has appeared since the last master clock, the data on the lower 8 bits of the pod will be the same as the upper 8 bits. If more than one slave clock has appeared since the last master clock, only the first slave data will be available to the analyzer.

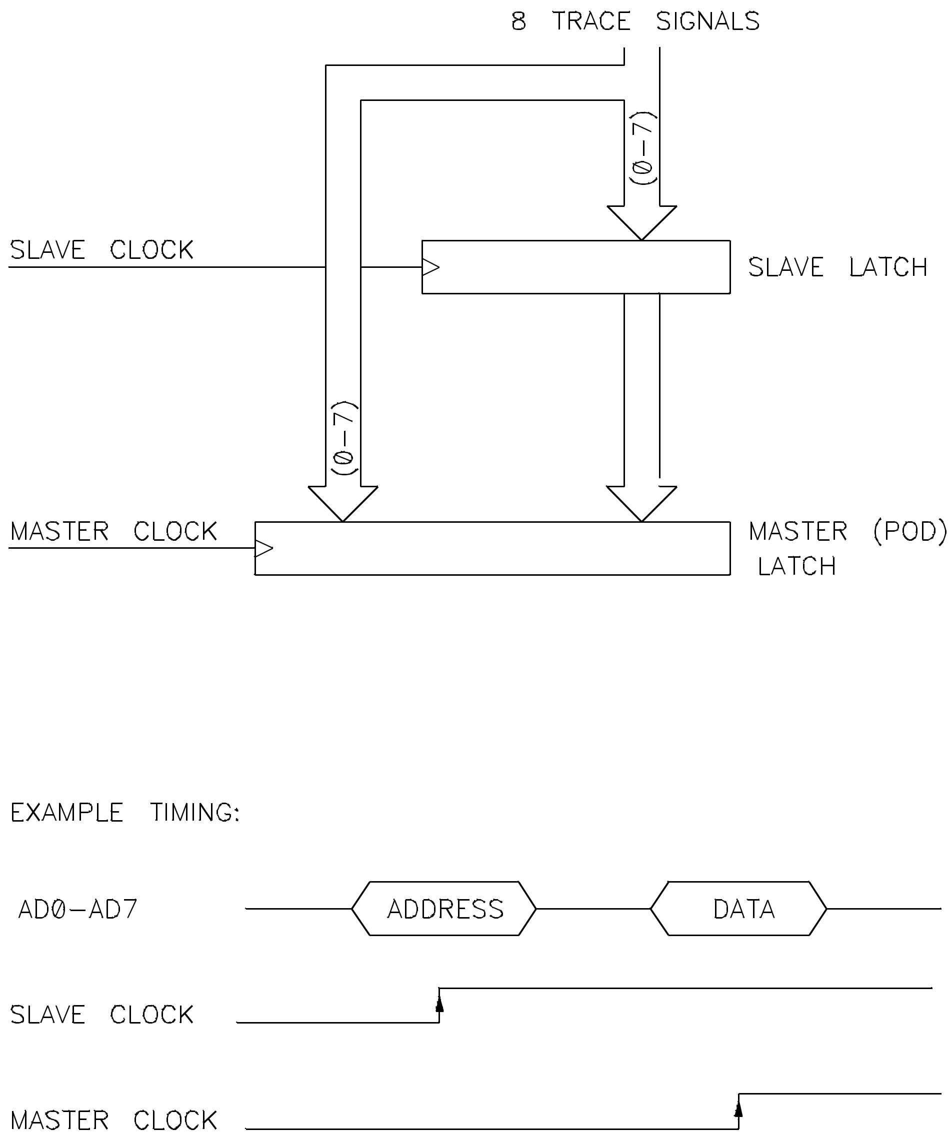


Figure 5-8. True Demultiplexing

Edges of J (K,L,M) clock used for slave clock?

These four questions are asked when you select either the "mixed" or "demux" slave clock mode. They allow you to define the slave clock. You can specify rising, falling, both, or neither (none) edges of the J, K, L, and M clocks. When several clock edges are specified, any one of the edges clocks the trace.

Clocks J and K are the external clock inputs of the external analyzer probe. The L and M clocks are generated by the emulator. Typically, the L clock is the emulation clock derived by the emulator and the M clock is not used.

Defining External Labels

The remaining external analyzer configuration questions allow you to define external labels.

Note

The Timing Analyzer Softkey Interface does not use then external labels from the configuration. You maintain labels for the timing analyzer software within the Timing Analyzer Softkey Interface itself.

First external label name?
First external label start bit?
First external label width?

External labels can be defined with bits in the range of 0 through 15. The start bit may be in the range 0 through 15, but the width of the label must not cause the label to extend past bit 15. Thus, the sum of the start bit number plus the width must not exceed 16.

Once external labels are defined, they may be used in trace commands to qualify events (if the emulation controls the external analyzer). Also, you can modify the trace display to include data for the various trace labels.

First external label polarity?

This configuration question allows you to specify positive or negative logic for the external bits. In other words, positive means high= 1, low= 0. Negative means low= 1, high= 0.

Define second external label?

Allows you to define additional labels. Up to eight external labels can be defined.

Configuring Interactive Measurements

When using the "state" or "timing" options for the external analyzer mode, you can configure the analyzer to trigger the external analyzer. This ensures that traces are returned only when the analyzer is running.

To configure the analyzer for interactive measurements, enter

`modify configuration` <RETURN>

in the emulator Softkey Interface.

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

`Modify Interactive Measurement Specification?`

Answer the prompt "yes" by pressing the **yes** softkey, or entering "yes" at the command line. You then see a display as depicted in figure 5-9.

```
                 Interactive Measurement Specification

             BNC <-??-> ---\                      BNC <-??-> ---\
                           |                                    |
            CMBT <-??-> ---|                     CMBT <-??-> ---|
                           | Trig1                              | Trig2
        Emulator <---------|                 Emulator <-??------|
                           |                                    |
        Analyzer ------> --/                  Analyzer <-??-> --|
                                                                |
                                     External Analyzer <-??-> --/

NOTES:
  1. The connections marked "??" are set up here in configuration.
  2. drive = ---->  receive = <----  (The display won't change, however.)
  3. The External Analyzer question is only asked when the External Analyzer
     mode is state or timing.


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


 drive   receive  neither    both                                        RECALL
```

Figure 5-9. Interactive Measurement Configuration

Note

The "External Analyzer" option for **Trig2** only appears if you have selected **state** or **timing** for the external analyzer mode.

Using the Analyzer Trigger to Drive the External Analyzer

The analyzer can be used to drive the external analyzer when it finds its trigger condition. This is done by setting up the analyzer to drive the *trig2* internal trigger and set up the external analyzer to receive it.

In this configuraton, the analyzer triggers the external analyzer to start a trace only after it finds its trigger condition. This allows you to coordinate timing measurements with the occurrence of a specific analyzer state.

Refer to the chapter on "Coordinated Measurements" in the *HP 64700-Series Emulators Softkey Interface Reference* for more information on setting up interactive measurements.

Saving the Configuration

Save the changes to the configuration file either in a system defined file or in a user file of your choice. The system defined file is:

/usr/hp64000/inst/emul/< product_number> /userconfig.EA

where < product_number> is the Hewlett-Packard product number, such as "64742A" for the HP 64742A Motorola 68000 Emulator.

The user defined file can be any file name you specify.

If a system defined file name is used, the emulator will used that configuration as the default when initializing the emulator.

6

Timing: Introduction

Overview

External timing commands are present in the firmware resident Terminal Interface. However, these commands output data in a binary format, and a host computer interface program is necessary to interpret and display the binary information. The Timing Analyzer Softkey Interface program is one such host computer interface.

Features of the Timing Analyzer

The Timing Analyzer Softkey Interface features include:

- 16 Channels at up to 100 MHz.
- Standard or glitch capture modes.
- Trace memory holding 1010 samples; 505 samples in glitch capture mode.
- Trigger when signals on the external probe match a specified pattern for greater than or less than a specified duration. Edge and glitch qualifiers may be included in the trigger specification.
- Trigger point at the start, center, or end of the trace to view signals after, about, or before the trigger.
- Display of data in graphic or list format.

- User-defined labels for the external probes signals.
- Store measurement data along with the system configuration.
- Comparison of stored and current measurements.
- Automatically mark user-specified events in trace data memory.
- Calculation of statistics on marked events.
- Support of graphics monitors as well as terminals (with an ASCII character waveform display).
- Support of screen dumps to graphics printers (for printing waveform displays).
- Support for cross triggering between the analyzer and the external analyzer.

Measurement Modes

You can use the external timing analyzer in either of two modes: standard (data acquisition) or glitch capture (data and glitch acquisition).

Standard Mode

In the standard mode, the timing analyzer samples data on the external analyzer probe at the selected sample rate. Up to 1010 samples can be stored, and the maximum sample rate is 100 MHz (10 ns intervals). See figure 6-1.

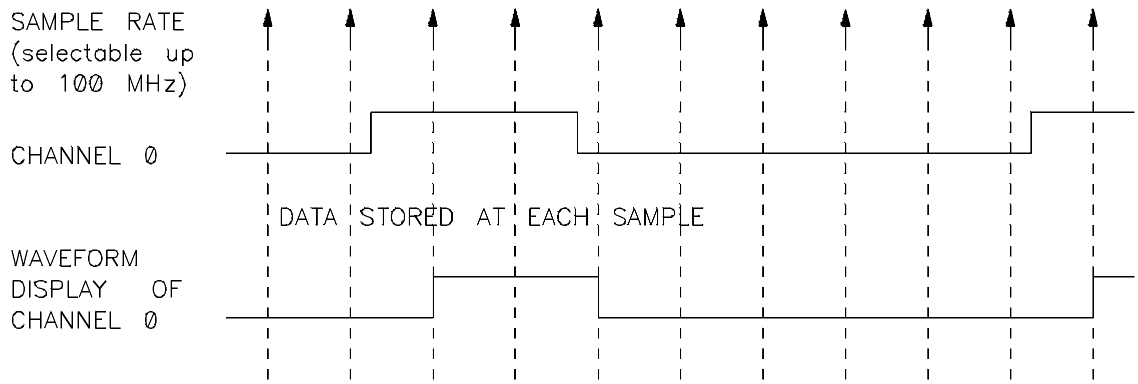


Figure 6-1. Standard Data Acquisition Mode

Glitch Capture Mode

This is the same as the standard mode except that glitch information is also stored for each sample. A glitch is detected when there are two or more transitions on a signal between samples. The storing of glitch information reduces the number of samples that can be stored to 505, and the maximum sample rate is 50 MHz (20 ns intervals). See figure 6-2.

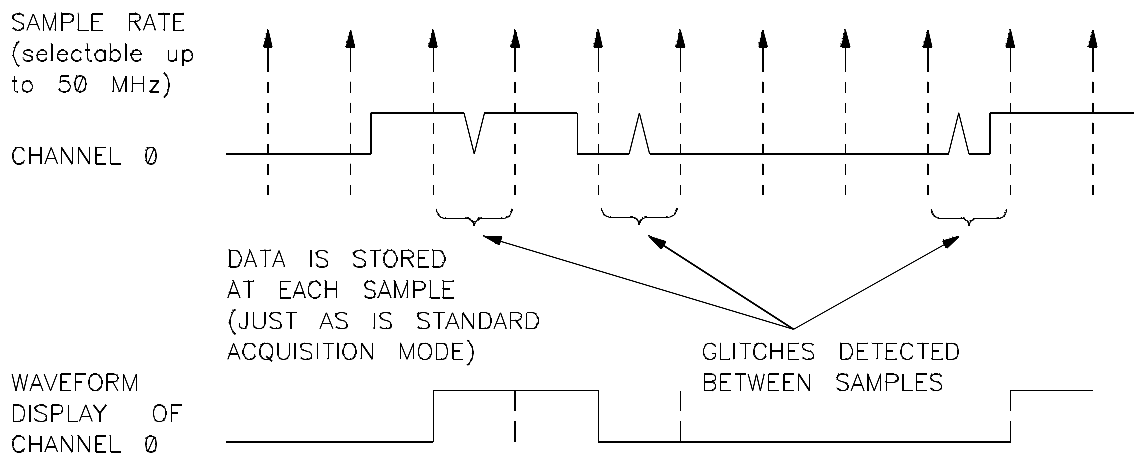


Figure 6-2. Glitch Capture Data Acquisition Mode

Trace Memory

Trace memory can store 1010 trace states if the standard mode is used or 505 states if the glitch capture mode is used.

The Trace Specification

The trace specification allows you to define the trigger condition about which trace states are captured. The trigger specification can be defined on transitions of signals, levels of signals, or glitches on signals. The trace memory can be positioned anywhere about the trigger condition or after the trigger condition by a specified amount of time.

The Format Specification

The format specification defines the labels which are to be associated with the probe signals along with the logical polarity of the signal. The format specification also defines the threshold values to be used during the capture of trace memory.

The Post-Process Specification

The post processing specification defines the analysis to be done on each set of sampled data. It enhances productivity by reducing the amount of time needed to make a range of computations about the trace memory. There are features to find, mark, compare, and gather statistics about events occurring in trace memory. In addition, a repetitive execution can be halted when a post processing event is found.

The Timing Diagram

The timing diagram presents the trace memory in a waveform display. The diagram will appear as a graphics diagram on high or medium resolution monitors, and as an ASCII character diagram on standard terminals. In either case, the diagram is easy to magnify, roll, and define so that the pertinent trace memory data is shown. In addition, marks can be added to the diagram to highlight events in trace memory.

The Trace List

The trace list displays the trace memory contents in list format. The trace memory data can be displayed in binary, octal, decimal, and hexadecimal formats, along with a time tag which indicates when the samples were captured in relationship to the trigger. The trace list data also can be marked to indicate events in trace memory. Finally, the trace list can be processed to show only samples which meet specified conditions.

Notes

7

Timing: Getting Started

Overview

This chapter describes:

- Prerequites for using the Timing Analyzer Softkey Interface.
- Entering the Timing Analyzer Softkey Interface.
- Making a simple measurement.
- Entering numerical values.

Prerequisites for Using the Timing Analyzer Softkey Interface

Before you can use the external timing analyzer, you must have already completed the following tasks:

- Verified that the emulator contains an external analyzer.
- Installed the Timing Analyzer Softkey Interface software.
- Assembled the analyzer probe and connected the analyzer probe grabbers to points which have signals of interest (refer to "Using the External Analyzer" earlier in this manual).

Installation

The Timing Analyzer Softkey Interface is included with the Softkey Interface software for you emulator. Refer to the *HP 64700-Series Emulators Softkey Interface Reference* for information on installing the software and hardware.

If you are using the Timing Analyzer with the X Window System, refer to *Installing X on the Series 300: Version 11*, or the equivalent manual for your system.

Entering the Timing Analyzer Softkey Interface

To enter the Timing Analyzer Softkey Interface, you must enter the Softkey Interface as you normally enter the Emulator Softkey Interface. For example, you would enter:

```
$ emul700 em68k  <RETURN>
```

at the HP-UX prompt. This initializes the emulator and loads the configuration. If you have not already modified the configuration for the external analyzer, you should do so here. Refer to chapter 5 "Using the External Analyzer" for information on configuring the external analyzer.

After the configuration for external timing has been loaded, exit out of the interface locked, and then enter the Timing Analyzer Softkey Interface with the following command:

```
$ emul700 -u sktiming em68k <RETURN>
```

Note

The Timing Analyzer Softkey Interface can also be entered within an X Window System window, or as a "measurement system" option. The commands listed earlier can be entered at the "$" prompt within a X Windows window. Refer to appendix B "Timing Output and Diagrams" for information on setting up the X Windows System. For information on adding the Timing Analyzer to your measurement system, refer to your *Emulation/Internal Analysis Operating Manual*.

Making a Simple Timing Measurement

A simple timing analyzer measurement can be made using the following sequence.

1. Default the analyzer to an initialized condition.

```
display format_specification  <RETURN>
default all_specifications  <RETURN>
```

2. Set the threshold for the logic family which you desire to analyze.

```
threshold xbits ecl  <RETURN>
```

3. Connect the probe grabbers to points which have signals you desire to analyze and define label names.

```
define My_label xbits_bit 0 <RETURN>
```

Caution

Remember to turn off the power to the target system before connecting the probe grabbers.

4. Determine the trigger point upon which you are desiring to trigger. To trigger on My_label being logically true, enter the commands:

```
display trace_specification  <RETURN>
trigger on pattern My_label = 1 <RETURN>
```

5. Match the sampling rate of the analyzer to about 10 times the rate of your system clock. Higher rates for more resolution and lower rates for a longer trace period.

```
sample period_is 20 nsec  <RETURN>
```

6. Display the desired information on the timing diagram.

```
display timing_diagram  <RETURN>
present My_label then X_upper  <RETURN>
```

7. Execute the analyzer measurement.

```
execute  <RETURN>
```

8. Move the cursor on the timing diagram by pressing the softkey CURSOR and then using the < leftarrow> and < rightarrow> to move the cursor.

Note

Use **< CTRL> -F** and **< CTRL> -G** along with **< PREV>** and **< NEXT>** to roll the timing diagram.

---

9. Magnify the diagram to see more detail.

magnify x10 <RETURN>

10. Mark two points on the diagram to measure the time between them.

mark x <RETURN>

Move the cursor.

mark o on_cursor <RETURN>

Move cursor to examine time interval between mark_x and mark_o.

Entering Numerical Values

As with the emulator Softkey Interface, you can enter numerical values in four standard bases: binary, decimal, octal, and hexadecimal. You must include the base letter after the number you enter, as follows:

B b	Binary (example: 10010110b).
Q q O o	Octal (example: 377o or 377q).
D d (default)	Decimal (example: 2048d or 2048).
H h	Hexadecimal (example: 0a7fh). You must precede any hexadecimal number that begins with an A, B, C, D, E, or F with a zero.

Don’t care digits may be included in binary, octal, or hexadecimal numbers and they are represented by the letters **X** or **x**. A zero must precede any numerical value that begins with an "X".

Notes

8

Timing: Using the Analyzer

Overview

This chapter shows you how to use the external timing analyzer. The main sections in this chapter describe how to:

- Move around the analyzer interface.
- Reference analyzer signals.
- Select measurement options.
- Specify the trigger condition.
- Start and stop a trace.
- Use the timing diagram.
- Use the trace list.
- Analyze trace memory.
- Compare current and store measurements.
- Copy analyzer data.
- End a session.

Moving Around the Analyzer Interface

The Timing Analyzer Softkey Interface uses eight screens to display analyzer information. Each screen has a set of associated functions. Some of the functions are common to all of the screens and some of the functions are specific to a particular screen.

The interface has three "specification" screens that are used to set up the measurement environment; two output screens, one graphic and one tabular; a screen to enter and display native analyzer ("pod") commands; and two screens to display logging information, one for errors and one for analyzer events.

You use the "display" command to change from one screen to the other. The "display" command has the following options:

Trace Specification	which controls the data acquisition mode, trigger condition definition, and sample rate.
Format Specification	which controls the probe, including the definition of labels that refer to probe signals.
Post Process Specification	which defines procedures to be executed after each trace measurement.
Timing Diagram	which displays the measurement data in a graphic form.
Trace List	which displays the measurement data in tabular form.
Pod Commands	which allows you to send some native analyzer specific commands and to display the responses.
Error Log	which displays a log of command line errors.
Event Log	which displays a log of analyzer events.

The command

display trace_specification <RETURN>

will display and set the Softkey labels for the trace specification options.

Referencing Analyzer Signals

This section will describe how to specify threshold voltages for the probe signals, test for activity on the probe, and manage labels for each of the signals.

Specifying Threshold Voltages

The external probe signals are divided into two groups named ***x_lower*** and ***x_upper***. The x_lower group refers to probe signals 0 through 7; the x_upper group refers to probe signals 8 through 15. In addition, an ***xbits*** group is used to refer to all of the probe signals (0 through 15). Threshold voltage levels can be specified for each group separately or for both by specifying the xbits group. TTL threshold voltage levels are specified by default.

You use the **threshold** command to change the threshold voltage specification. After you select the group for which you will be specifying the threshold voltage, the following options are available:

ttl sets the threshold at + 1.40V.

ecl sets the threshold at -1.30V.

cmos sets the threshold at + 2.50V.

< VOLTS> which prompts you for a voltage in the range of -6.40V to 6.35V (with a 50 mV resolution).

The commands

display format_specification <RETURN>

threshold xbits cmos <RETURN>

specifies a CMOS threshold voltage level for all signals.

Note

The threshold settings will not take effect until the next time you execute a trace or activity test.

Testing for Signal Activity

After you have connected the probe grabbers to the points which have signals of interest, you can test for activity on the probe. The command

activity_test <RETURN>

starts the sampling test.

A high or low status is indicated under each probe bit number on the display, and the word "Activity" is included in the list of labels, indicating the activity test is turned on. Enter **activity_test** < RETURN> again to turn off the activity test.

Managing Labels

The timing analyzer provides you a means of labeling probe signals, either individually or in groups. You can use the three default labels or define new ones. Any label can be modified, deleted, or renamed.

Defining Labels

The "define" command allows you to label the analyzer probe signals. Each label refers to one or more probe signals. Once a label is defined, you can then use this label name when specifying patterns, edges, or glitches on data signals. Label names can be up to eight characters long, must begin with a letter, and may be followed by up to seven alphanumeric characters.

Three labels are predefined: "XBITS", "X_lower", and "X_upper". To define these labels yourself you would use the commands:

define XBITS ***xbits_bit*** 0 ***width*** 16
logic_polarity positive_true <RETURN>

define X_lower ***xbits_bit*** 0 ***width*** 8
logic_polarity positive_true <RETURN>

define X_upper ***xbits_bit*** 8 ***width*** 8
logic_polarity positive_true <RETURN>

Note

You should not confuse the default labels XBITS, X_lower, and X_upper with the probe signal groups xbits, x_lower, and x_upper.

The command

```
define CLK xbits_bit 0 logic_polarity
positive_true <RETURN>
```

gives probe signal "0" the label CLK and uses a positive polarity as logical true.

Modifying Label Definitions

The "modify" command retrieves the command used to define a particular label. Once the definition is returned to the command line, you can use the command line editing features to modify the label definition.

Deleting Labels

The "delete" command can be used to delete all labels, or individual labels, that are not used in any other specification.

The command

```
delete X_upper  <RETURN>
```

deletes the default label "X_upper" from the format specification.

Renaming Labels

The "rename" command allows you to change a label name.

The command

```
rename CLK to clock <RETURN>
```

renames the label "CLK" to "clock".

Selecting Measurement Options

The section describes how to select one of the two analyzer modes and the sample period or rate.

Selecting the Timing Analyzer Mode

The Timing Analyzer Softkey Interface has two modes: standard and glitch capture.

Standard Mode

The standard mode is the usual data acquisition mode. In this mode, all sixteen probe signals can be analyzed and 1010 samples can be taken. This is the default mode for the analyzer.

The commands

display trace_specification <RETURN>
mode_is standard <RETURN>

change the mode to standard.

Glitch Capture Mode

The glitch capture mode is used to detect and display multiple signal transitions between data samples. If more than one transition is detected between samples, the information is stored in a portion of trace memory reserved for glitch information and is displayed on the screen. A glitch is displayed as a broken vertical bar or a series of broken vertical bars depending on the magnification. (In ASCII diagrams, the default symbol is a colon ":".) A vertical bar also indicates the occurrence of multiple data transitions too close together to be displayed at the selected horizontal magnification.

The command

mode_is glitch_capture <RETURN>

sets the analyzer mode to glitch capture.

Selecting the Sample Period or Rate

You can select the sample period (or rate, if it's more convenient) with the "sample" command. Valid periods in the standard mode are between 10 ns (100 MHz) and 50 ms (20 Hz). Valid periods in the glitch capture mode are between 20 ns (50 MHz) and 50 ms (20 Hz). The accuracy of the sample rate is that of the crystal oscillator, approximately + /- 0.01%.

The command

sample period_is `10` ***nsec*** `<RETURN>`

sets the sample rate to once every 10 nanoseconds.

Specifying the Trigger Condition

The "mode" and "sample" commands control the acquisition of data, but they do not tell the analyzer *when* to acquire data. This is done by using the "trigger" command to specifying a trigger condition.

The trigger condition is the combined specifications of all labels (that is, the pattern, edge, or glitch specifications for each label ORed together to form the complete specification for the 16 external data signals).

The "trigger" command is part of the trace specification.

Trigger on Anything

The default condition is to trigger on any activity on the probe signals.

The command

trigger on anything `<RETURN>`

sets the analyzer to its default condition.

Trigger on Pattern

In this type of trigger condition, a specific set of signal activities, a pattern, must be present in order for the trigger condition to be met.

You specify a data pattern consisting of 1s, 0s, or X's (don't cares) on one or more of the labels or label signals. The most significant bit is probe signal 15, or the highest bit number for the label specified, and the least significant bit is probe signal 0, or the least significant bit of the label specified. Refer to "Entering Numerical Values" in the previous chapter for valid pattern value options.

The command

trigger on pattern X_lower = `3AH  <RETURN>`

sets analyzer to test for the pattern of bits 00111010B on the lower eight signals of the probe and, if found, trigger the filling of trace memory.

Note

The "hold" time for the trigger on a pattern is 30 nanoseconds. Therefore the "trigger on pattern..." command is equivalent to "trigger on greater_than 20 nsec_of ....". Refer to "Trigger on Pattern Duration" below.

Trigger on Pattern Duration

You can specify, as part of the trigger condition, a pattern duration. The duration specifies that a pattern will exceed ("greater_than") or not exceed ("less_than") the amount of time specified in order for the trigger condition to be met.

If the pattern is valid but the duration is not met, there is a 20 ns reset time before looking for a pattern again.

Greater Than Duration

The trigger occurs after the pattern is present on the probe for at least the specified time duration. You can select a duration from 30 ns to 10 ms in 10 ns increments.

The command

trigger on greater_than `100` ***nsec_of XBITS*** `.2 =0 <RETURN>`

sets the analyzer to trigger a trace if probe bit 2 is logically false for more than 100 nanoseconds.

Less Than Duration

The trigger occurs when the specified pattern is present on the probe signals for greater than 20 ns but less than the specified time duration. Durations from 40ns to 10 ms in 10 ns increments can be selected.

The command

```
trigger on less_than 60 nsec_of XBITS .2 =0
<RETURN>
```

sets the analyzer to trigger a trace if probe bit 2 is logically false for less than 60 nanoseconds and more than 20 nanoseconds.

Trigger on Any Glitch

This type of trigger condition can only occur in the glitch capture mode. In order for this trigger condition to become true, a glitch must be found on the specified signals.

The command

```
trigger on any_glitch  <RETURN>
```

sets the analyzer to only trigger if a glitch is found on any signal.

Qualifying Patterns

When you specify this "trigger on" option, the trigger is further qualified by a selected change in another signal or set of signals. The change can be a "positive_edge", a "negative_edge", or any edge condition ("positive_or_negative_edge") on a specific bit or bits.

In glitch capture mode, the qualifiers are glitches on one or more signals.

This option is useful as a means to locate the effects of an occasional event such as a reset or interrupt.

The command

```
trigger on anything qualified_with
positive_edge XBITS .10 or_on positive_edge
XBITS .2  <RETURN>
```

sets the analyzer to trigger a trace on any activity after a positive edge is detected on probe signal 10 or probe signal 2.

Trigger Delay

Trigger delay is the amount of time to delay the trigger after a valid trigger condition. Trigger delay can be anywhere between 0 and 10 ms in 10 ns increments.

The command

trigger `100` ***nsec_after pattern*** `. . . <RETURN>`

sets the analyzer to trigger a trace 100 nanoseconds after a selected pattern condition is met.

Trigger Positioning

The "trigger position_is" command allows you to place the trigger at the start, center, or end of the trace. Therefore, if you want to look at events before the trigger, you place the trigger at the end of the trace; if you want to look at events after the trigger, you place the trigger at the start of the trace. If you want to look at events near the trigger, you place the trigger at the center of the trace.

The command

trigger position_is start_of_trace `<RETURN>`

positions the trigger at the start of the trace so that most of trace memory is available to store data samples after the trigger.

Modifying the Trigger Condition

A trigger condition can be changed by entering a new condition or by using the

trigger modify `<RETURN>`

command to return the trigger condition to the command line for editing.

Starting and Stopping a Trace

The purpose of the Timing Analyzer software is to acquire and process data from the external analyzer. You initiate the data acquisition by executing a trace. After a trace is started, the data from the external analyzer is scanned until the trigger condition is met or you stop the trace with the "halt" command. Traces can also be "repetitive" so that statistical analysis can be performed on the acquired data.

Note

Refer to appendix D "Accurate Timing Measurements" for information on the statistical analysis of timing data.

Execute

The "execute" command starts a single trace. The trace is normally completed when the analyzer encounters the trigger condition. At that point trace memory is filled, the data is processed, and, finally, it is displayed, either in the timing diagram or the trace list form.

The command

execute <RETURN>

starts a trace.

Execute Repetitively

You use the "execute repetitively" command to acquire data for statistical analysis. The analyzer software continuously acquires traces until one of four condition are met:

- The execution is stopped by a "halt" command.
- The execution is stopped by a post process "halt_repetitive_execution" condition being met.
- A total of 9999 trace runs have been completed.
- The system is stopped.

The command

execute repetitively <RETURN>

starts a repetitive trace.

Halt

The "halt" command stops a trace that is waiting for a trigger or is executing repetitively.

The command

halt restore_last_trace <RETURN>

allows you to restore the last completed trace when executing a repetitive analysis.

Using the Timing Diagram

This section describes the timing diagram and the features that allow you to change the display format. The timing diagram displays the currently in trace memory. Unless you are currently displaying the trace list, when you start a timing trace ("execute"), the timing diagram is automatically displayed.

Timing Diagram Organization

The organization of the timing diagram is shown in figure 8-1.

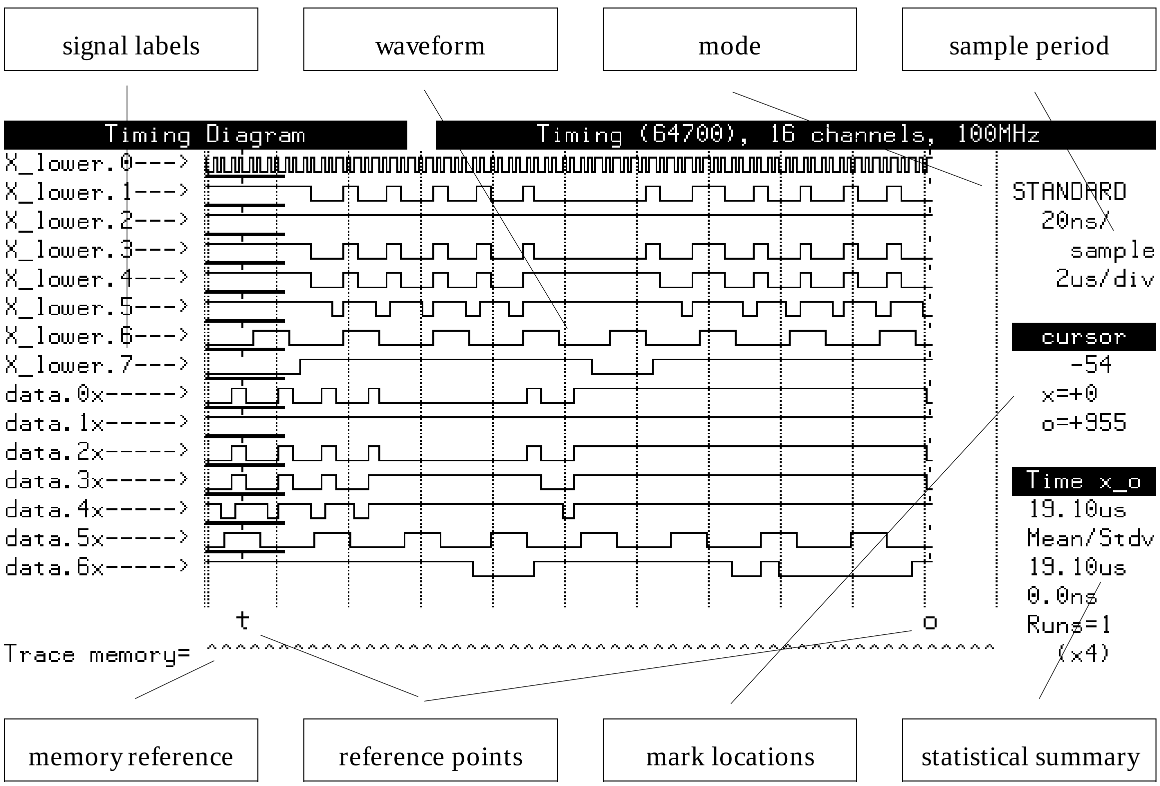


Figure 8-1. Timing Diagram Organization

Signal Labels. The labels for the signals are displayed here. If the label refers to more than one probe signal, the logical bit number is added to the label display. If the signal refers to a label in a compare file, the signal reference is followed by an "x".

You can change the signals displayed and the order they are displayed at any time. You can also choose to display the signal level at the cursor ("L" or "H" for low or high, respectively). If this option is selected, the signal level at the cursor is displayed in the place of the "> " on the right of the signal label.

Waveform. The waveform depicts the trace data using graphics characters, or ASCII characters if graphics are not supported. The waveform is segmented horizontally into divisions. The relative width of each division is controlled by the sample period/rate and the magnification. The size of each division is displayed below the sample period. The relative vertical size of the waveform can be set to small, medium, or large.

Mode. The mode for the current trace is displayed here.

Sample Period. The sample period for the current trace is displayed here.

Mark Locations. The locations of the cursor, mark_x and mark_o are displayed here. The number displayed is the relative sample number in trace memory.

Statistical Summary. The statistical summary displays the currently selected information. You can choose to display ("indicate") amount of time between the mark_x and mark_o points (interval statistics) or the number of occurrence marks (abcd) between the mark_x and mark_o points (occurrence statistics). For each of the two types of analysis, you can choose to display the range (maximum and minimum), or the mean and standard deviation. The number of samples (runs) is also displayed.

Reference Points. Four reference points are displayed: the cursor, the trigger point, and the mark_x and mark_o points. If the reference point is in the current waveform display, a vertical dashed line depicts the point and is labeled below. If the reference point is not in the portion of trace memory currently in the waveform display, the label drops down into the memory reference line in its relative location in trace memory. If more than one reference point is in the same location, an "m" is used to denote multiple references.

The cursor is always in the display area, but is not labeled. In the lower magnifications, the cursor contains an optional horizontal set of bars. These "magnify indicators" depict the amount of trace memory that will be displayed if the magnification is increased by a factor of ten.

The multiple occurrence mark (abcd) positions are also depicted with vertical dashed lines but are not labeled.

Memory Reference. The "depth" of trace memory is depicted in a memory reference line. The relative portion of trace memory currently in the waveform display is depict with a caret ("^ "). Any portion of memory not in the display is depicted with a dash ("-"). Reference points not currently in the waveform display are labeled in the memory reference line. The current magnification is displayed to the right of the memory reference line.

Presenting Signals

By default, the timing diagram presents all of the defined signals, initially XBITS, X_lower, and X_upper. You can choose any of the signals to be presented, as well as the order in which they are presented.

Signals are presented by selecting the labels defined in the format specification. If a label refers to more than one signal, you can choose to present all of the signals, or select one using the logical bit number.

The command

present X_lower <RETURN>

presents all of the signals referred to by the default label X_lower.

The command

present X_lower then blank then X_upper .0 <RETURN>

presents all of the signals referred to by the default label X_lower, a blank line, and then the first signal in label X_upper (probe signal "8").

The command

present <RETURN>

toggles between user-defined labels and the corresponding probe signal labels. Therefore, if the user-defined label "CLK" refers to probe signal "0", the display would toggle between "CLK" and "xbits_bit_0".

Refer to the section on *Comparing Current and Stored Measurements* later in this chapter for information on presenting compare file signals.

Moving the Cursor

The cursor is an arbitrary reference point. On the timing diagram, it is used primarily to identify a particular event on the waveform. The cursor can be directly moved in three ways: using control keys, using the "CURSOR" softkey toggle, or entering the trace memory sample number.

The control keys **< CTRL> -F** and **< CTRL> -G** move the cursor to the right and left, respectively, without effecting command line editing. The incremental shift using this method is relatively large.

If you use the "CURSOR" softkey, the right and left arrows can be used to move the cursor. The "CURSOR" softkey serves as a toggle to switch the function of the arrow keys from use on the command line to use on the waveform, and back again. When the arrow keys are used to move the cursor in the waveform, the label on the softkey appears as "CURSOR*". The CURSOR/arrow key method of moving the cursor allows for more detailed control.

The cursor can also be positioned by entering the relative sample number on the command line. The command

100 <RETURN>

moves the cursor to sample number one hundred, while the command

-50 <RETURN>

positions the cursor at sample number minus fifty. In all cases, if the corresponding sample is not currently on the display, it will be centered in the display.

Refer "Locating Events in Trace Memory" later in this section for more information on positioning the cursor.

Showing Levels at the Cursor

The level for each of the signals is depicted in the waveform. You can also choose to display the signal level, high ("H"), low ("L"), or glitch ("G"), at the current cursor position by enabling that feature.

The command

indicate levels_at_cursor on <RETURN>

enables the display of signals levels at the cursor. The levels are indicated to the right of the labels.

Magnifying the Diagram

The default magnification of the trace memory data is x4. With this magnification, a standard mode sample fills the waveform. A higher magnification indicates that more detail is displayed (less trace memory is depicted in the waveform) and a lower magnification indicates that less detail is displayed.

The command

magnify x100 <RETURN>

changes the "magnification" on the trace to 100 times.

Changing the Waveform Size

The relative vertical size of the waveform can be changed to make visualization easier. You can set the waveform to small, medium, or large. Small is the default waveform size.

The command

waveform_size large <RETURN>

sets the waveform size to large.

Scrolling the Diagram

If you "magnify" the signal resolution or change the waveform size, all of the information may not fit in the waveform display area. The display area can then be thought of as a window on the whole waveform. The window can be scrolled (rolled) left and right, or up and down.

You can move the display area left and right using the cursor, as described earlier, or you can move it using the < **Prev**> and < **Next**> keys. The < **Prev**> key moves the display area to the left, earlier in trace memory, and < **Next**> move the display area to the right, later in trace memory.

If the timing diagram cannot display all of the signals on one screen, you can scroll the display area up and down. The ^ (< **uparrow**>) and the **v** (< **downarrow**>) keys scroll the display up and down one signal for each keystroke. The **Shift-^** (**Shift-< uparrow**>) and **Shift-v** (**Shift-< downarrow**>) keys move the display up and down one screen display for each keystroke.

If all of trace memory is displayed, either left and right, or up and down, the keys have no effect.

Using the Trace List

The trace list displays the trace memory data in a columnar (table) format.

Trace List Organization

The organization of the trace list is shown in figure 8-2.

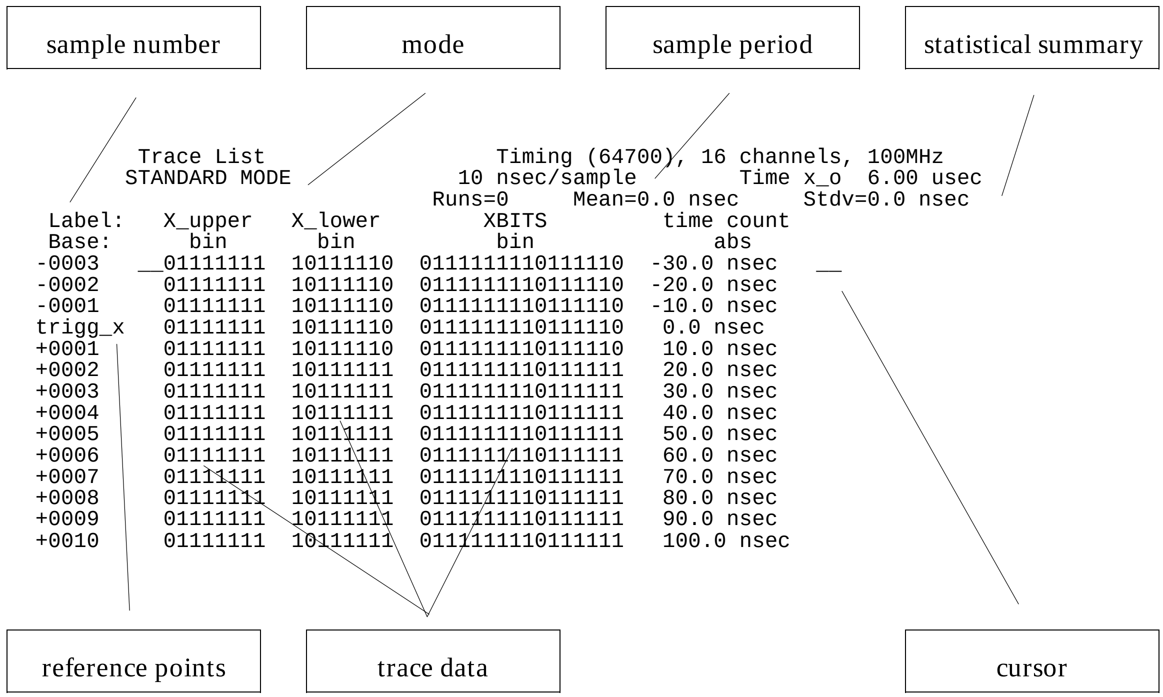


Figure 8-2. Trace List Organization

Sample Number. The relative sample number is displayed in this column. The trigger point, labeled "trigger" is relative sample number zero. If reference points appear at the trigger, the sample number appears as "trigg_".

Reference Points. The reference points mark_x, mark_o, and the four multiple occurrence points (abcd). The reference points appear in the column immediately to the right of the sample number. If more than one reference point is in the same location, an "m" is used to denote multiple references.

Mode. The mode for the current trace is displayed here.

Sample Period. The sample period for the current trace is displayed here.

Statistical Summary. The statistical summary displays the currently selected information. You can choose to display ("indicate") amount of time between the mark_x and mark_o points (interval statistics) or the number of occurrence marks (abcd) between the mark_x and mark_o points (occurrence statistics). For each of the two types of analysis, you can choose to display the range (maximum and minimum), or the mean and standard deviation. The number of samples (runs) is also displayed.

Cursor. The cursor has the same function as the cursor in the timing diagram.

Trace Data. The trace data includes one or more of the following: trace memory data samples, time count, mark names, and compare file memory samples. Memory samples are displayed by label name and can be presented in binary, octal, decimal, or hexadecimal formats. The time count can be displayed as "absolute", the accumulated time before or since the trigger, or "relative", the amount of time since the last sample.

Displaying Trace Data

As with the timing diagram, you have control of the information included on the trace list display. However, unlike the timing diagram, the trace list *cannot* be scrolled left and right to display extra information. Therefore, if you specify more information to be displayed than will fit on the screen, the extra information will be truncated.

The default condition is to display all of the labels included in the format specification in the order they are listed there. If enough space is left in the display area, the absolute time count is also included.

Using default label names, the command

present X_lower then X_upper .0 <RETURN>

displays the probe signals "0" through "8" in the display, grouped eight and one, and displayed as binary number, while the command

present X_lower in_hex then X_upper in_hex then time_count absolute <RETURN>

displays all of the probe signals in two eight bit groups in hexadecimal format along with the absolute time count.

The command

present X_lower then X_upper then time_count absolute then mark_names <RETURN>

again displays all of the probe signals, this time in binary format, the absolute time count, and mark names, where included with mark definitions in the format specification.

Moving the Cursor

The cursor is an arbitrary reference point. Like the timing diagram, it is used primarily to identify a particular event in trace memory. The cursor can be directly moved using cursor movement keys or entering the trace memory sample number.

The cursor can be moved by using the "^ " (< uparrow>) or "v" (< downarrow>) keys to move the cursor up and down through the trace samples.

The cursor can also be positioned by entering the relative sample number on the command line. The command

100 <RETURN>

moves the cursor to sample number one hundred, while the command

-50 <RETURN>

positions the cursor at sample number minus fifty. In all cases, if the corresponding sample is not currently on the display, it will be centered in the display.

Refer "Locating Events in Trace Memory" later in this section for more information on positioning the cursor.

Scrolling the Trace List

The trace list display area can then be thought of as a window on trace memory. The window can be scrolled up and down.

The **Shift-^** (**Shift-< uparrow>**) and **Shift-v** (**Shift-< downarrow>**) keys move the display up and down one line for each keystroke. Use the **< Prev>** and **< Next>** keys to move the display area up one screen full, earlier in trace memory, and < Next> to move the display area to down on screen full, later in trace memory.

Using the ^ (**< uparrow>**) at the sample at the top of the screen has the same effect as **Shift-^** ; using the **v** (**< downarrow>**) at the sample at the bottom of the screen has the same effect as **Shift-v**.

Analyzing Trace Memory

Trace memory can be analyzed interactively, that is by commands entered at the command line, or automatically by using post processing definitions. The "find" command is an interactive command, while the "mark" and "process_for_data" commands work both interactively and in post processing. All of these commands work for both the timing diagram and the trace list.

Locating Events in Trace Memory

The "find" command locates specified events in trace memory and positions the cursor at that point.

The command

find trigger <RETURN>

moves the cursor to the trigger reference point and attempts to center that point in the middle of the display.

You can search for a specific signal pattern with a command like

find entering X_lower =01010111 <RETURN>

that looks for specific combination of true an false signals on probe bits "0" through "7", or for a transition specific signal with a command like

find entering X_lower .2 = 1 <RETURN>

which attempts to find where bit 2 of X_lower (probe signal "2") becomes true. In both cases, the search begins at the cursor and, if the pattern is found, the sample is positioned in the middle of the display. The "entering" syntax denotes that a signal or signals are changing to the indicated logical value for the pattern specified.

Note

The **trigger** command does not have a corresponding "entering" function. It simply matches the pattern specified.

The command

find entering X_lower .2 = 1 all <RETURN>

is has the same effect as the last search, but starts from the beginning of trace memory instead of at the cursor.

You can also look for any change on a signal. The command

find any_transition on X_lower .2 = 1 thru end <RETURN>

searches for any change on bit 2 of X_lower from the cursor to the end of trace memory. Other options can terminate the search: trigger, mark_x, mark_o, a specific sample number, the start of trace memory, as well as the end of the trace. If the specified "thru" location is before the cursor, the search starts at that location and ends at the cursor.

You can designate a search pattern must last for a specified duration; it can be less than or greater than a selected amount of time. The command

find greater_than 100 nsec_of X_lower .2 = 1 thru 600 <RETURN>

searches for at least one hundred nanoseconds of a high signal on bit 2 of X_lower from the cursor to sample number 600. A duration of less than 20 nanoseconds cannot be specified because of hardware reset cycles.

The events you search for can be simple, as described to this point, or complex combinations of signals. The command

find entering X_lower .2 = 1 and X_lower .4 = 0 <RETURN>

specifies a search of trace memory from the cursor to the end to find a sample where bit 2 of X_lower is going high (true) and bit 4 of X_lower is going low (false). Any combination of signals and labels, as well as ranges, can be specified.

You can also search for marked events, both single occurrence (mark_x and mark_o) and multiple occurrence (abcd) marks.

Marking Events

The "find" command locates events in trace memory for interactive analysis. The Timing Analyzer Softkey Interface has the added feature of assigning identifiers (marks) to events. Marks can be assigned from the command line for events in current trace memory, or automatically after each trace measurement in the post processing specification. Any marks specified at the command line are also store for subsequent post processing.

The two single occurrence marks, mark_x and mark_o, are used to identify the first occurrence of a specified event. These two marks always exist. Their primary purpose is to define a range of samples. The multiple occurrence marks (mark_a, mark_b, mark_c, and mark_d) are used to identify all occurrences of a specified event.

The syntax for the "mark" command is very much like that for the "find" command. The command

mark x on_first_occurrence_of entering
X_lower =01010111 <RETURN>

places mark_x on the first occurrence of the pattern 01010111B on X_lower in trace memory while the command

mark x on_first_occurrence_of entering
X_lower .2 = 1 <RETURN>

places mark_x on the first occurrence of bit 2 of X_lower being high.

In addition, the "mark" command can automatically mark an event before or after another condition: the trigger, another mark a specific sample number, the cursor, or the end of memory. As an example, the command

mark o on_first_occurrence_of entering
X_lower .2 = 1 after mark_x <RETURN>

places mark_o on the first occurrence after the mark_x point of bit 2 of X_lower being high. The mark_x and mark_o points locate and determine specific intervals.

The multiple occurrence marks (abcd) mark all occurrences of the specified event. A range can be used to limit the marking process. The marks can be used to count events and are an effective means of performing statistical analyses on trace measurements. The command

mark a on_all_occurrences_of entering
X_lower .2 = 1 after trigger <RETURN>

places a mark_a on all occurrences of bit 2 of X_lower occurring after the trigger.

You can assign names to any of the marks. These names can then be displayed in the trace list to facilitate analysis. In addition, the multiple occurrence marks can be turned off or on without losing the marking condition.

Processing for Data

The "process_for_data" command is a way to limit the samples displayed on the trace list. As with the "find" and "mark" commands, a simple or complex set of conditions can be specified. You can display only samples that have been previously marked, specified new conditions for the display, or display samples a set number of samples after a new condition. The "process_for_data" command definition is also retained, like the "mark" command definitions, for post processing.

The command

process_for_data marked <RETURN>

displays all previously marked data on the trace list. The command

process_for_data sampled 10 ***samples_after pos_transition_on X_lower .2*** <RETURN>

displays only trace samples that are ten samples after any positive (from low to high) transition on bit 2 of X_lower. The amount of time represented by the ten samples is dependent on the sample rate. Therefore, if the sample period is 10 nanoseconds, the samples displayed will be 100 nanoseconds after the positive transition.

Determining Intervals

Both the timing diagram and the trace list display the time interval between the mark_x and mark_o points. Therefore, the simplest way to determine an interval between two events is to mark the two events of interest with mark_x and mark_o, and then read the interval from the display.

If the events are not already marked, simply move the cursor to the event of interest and position one of the marks at that point using a command like

mark x <RETURN>

The command

mark x on_cursor <RETURN>

"attaches" the mark to the cursor. You can then move the cursor to the desired location, and use

mark x <RETURN>

to position the mark at its current location.

Statistics

The Timing Analysis Softkey Interface can perform statistical analyses on repetitive executions traces. Two types of analysis are available: interval statistics and occurrence statistics. For both types of analysis, you can calculate the minimum and maximum values, or the mean and standard deviations.

Choosing Statistics Types

Interval Statistics Interval statistics are based on the time interval between mark_x and mark_o samples. The interval between the mark_x and mark_o points is calculated and includes as a statistical sample.

Occurrence Statistics Occurrence statistics are based on the number of multiple occurrence marks (abcd) assigned during post processing of each trace measurement. The number of marks between the mark_x and mark_o points are totaled included as a statistical sample.

The type of statistical analysis can be specified on either the timing diagram or trace list screen. The commands

display timing_diagram <RETURN>
indicate time_interval_x_o
mean_and_standard_deviation <RETURN>

specifies interval statistics calculating the mean and standard deviation for the samples collected.

You can disable the calculation of statistic by specifying a statistics type without a calculation, as in

indicate number_of_marks_x_o <RETURN>

which will only display the number of marks between mark_x and mark_o.

Excluding Samples

The "statistics" post process command allows you to exclude samples from the statistical calculations. The default condition is to *always* include traces in the calculation. You exclude traces when they have a interval from mark_x to mark_o of greater than (or less than) a specified amount of time, or when the number of marks from mark_x to mark_o is greater than (or less than) a specified number of marks.

Logging Statistics

You can log the detail from each sample to a file for subsequent review. The "statistics"' command in the post processing specification allows you to specify a file name to write the information. For logging to occur, one of the statistical analysis calculation options must have been previously chosen.

The commands

display timing_diagram <RETURN>

indicate time_interval_x_o mean_and_standard_deviation <RETURN>

post statistics log_to_file STATISTICS <RETURN>

post halt_repetitive_execution when_runs_equal 8 <RETURN>

execute repetitively <RETURN>

start a eight run statistical trace. After execution halts, enter the command

post statistics log_to_file none <RETURN>

to disable statistics logging. The generated output logs will look like:

```
64700 Timing Analyzer                   Mon Jun 19 14:47:05 1989

  Time x_o  1.66 usec      x=-54    o=+112   Runs=1     Mon Jun 19 14:47:20 1989
        Max=1.66 usec     Min=1.66 usec      Mean=1.66 usec      Stdv=0.0 nsec
  Time x_o  5.77 usec      x=-54    o=+523   Runs=2     Mon Jun 19 14:47:30 1989
        Max=5.77 usec     Min=1.66 usec      Mean=3.71 usec      Stdv=2.91 usec
  Time x_o  7.82 usec      x=-54    o=+728   Runs=3     Mon Jun 19 14:47:40 1989
        Max=7.82 usec     Min=1.66 usec      Mean=5.08 usec      Stdv=3.14 usec
  Time x_o  3.47 usec      x=-54    o=+293   Runs=4     Mon Jun 19 14:47:50 1989
        Max=7.82 usec     Min=1.66 usec      Mean=4.68 usec      Stdv=2.69 usec
  Time x_o  1.62 usec      x=-54    o=+108   Runs=5     Mon Jun 19 14:48:00 1989
        Max=7.82 usec     Min=1.62 usec      Mean=4.07 usec      Stdv=2.70 usec
  Time x_o  7.10 usec      x=-54    o=+656   Runs=6     Mon Jun 19 14:48:11 1989
        Max=7.82 usec     Min=1.62 usec      Mean=4.57 usec      Stdv=2.71 usec
  Time x_o  2.32 usec      x=-54    o=+178   Runs=7     Mon Jun 19 14:48:21 1989
        Max=7.82 usec     Min=1.62 usec      Mean=4.25 usec      Stdv=2.62 usec
  Time x_o  2.32 usec      x=-54    o=+178   Runs=8     Mon Jun 19 14:48:34 1989
        Max=7.82 usec     Min=1.62 usec      Mean=4.01 usec      Stdv=2.52 usec
```

Comparing Current and Stored Measurements

The Timing Analyzer Softkey Interface allows you to store trace memory measurements in a file and then compare the stored data to a subsequent trace measurement.

Storing Measurements

In order to properly compare a saved trace measurement, its data is stored with all the specification (configuration) information. A configuration save file that is saved with the data can then be referred to as a "compare file".

The command

configuration save_in COMPARE ***with_data write_protect*** <RETURN>

creates a file "COMPARE.TR" that contains specification information along with the current trace memory data. The file is write protected so that you cannot write to the file with a subsequent "configuration save_in" command.

Selecting a Compare File

In order to compare current trace data with previously stored trace data, you must first select a compare file.

The command

compare file_is COMPARE <RETURN>

from the post processing specification screen will select the previously generated "COMPARE.TR" as the current compare file. You can also enter

post compare file_is COMPARE <RETURN>

from any screen to select "COMPARE.TR".

Compare files must have:

- All labels referenced in the compare file must have equivalent definitions in the current format specification, and

- The trigger position must be the same

or a "Compare file spec does not agree with hardware" error will occur.

Presenting Stored Signals

Once a compare file is selected, the signals from the previous trace can be presented, just as the current labels can be presented, on either the timing diagram or the trace list. If a compare file has been selected the "compare" Softkey label appears in place of the "< COMPAR> " Softkey label. This Softkey allow you to select one of the "external" (compare file) labels.

The command

present compare_file X_lower then X_lower
<RETURN>

on the timing diagram will present the probe signals "0" through "7" from the store data and then the current trace data. The display will look like figure 8-3.

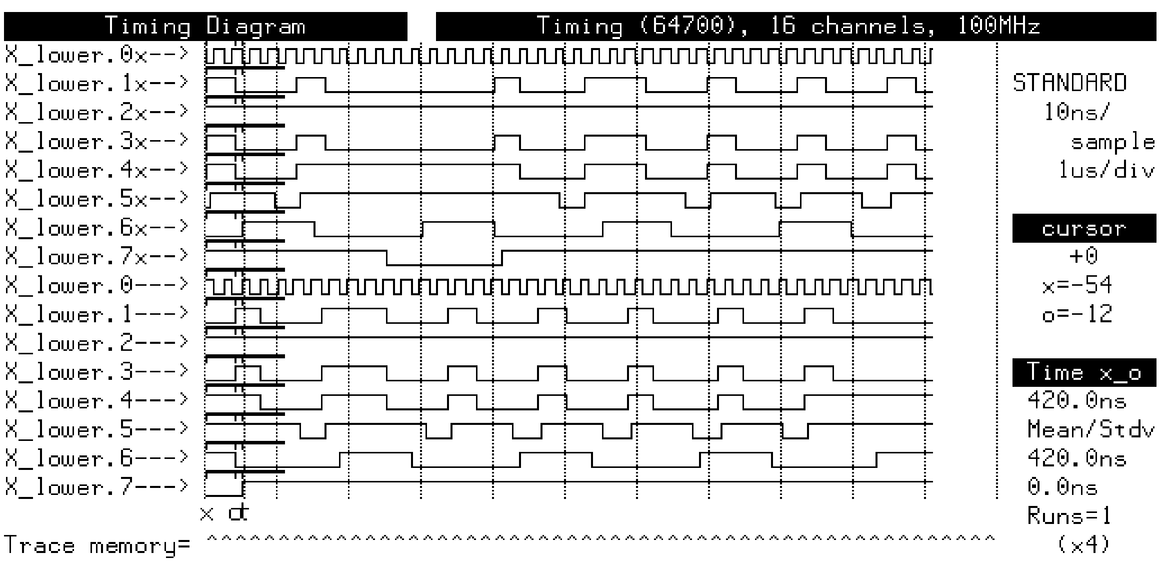


Figure 8-3. Comparing Stored and Current Traces

Copy Analyzer Data

The Timing Analyzer Softkey Interface can copy all or part of its information to a file or to a printer, or "pipe" it to a HP-UX process. Refer to appendix B "Timing Output and Diagrams" for information on setting up the system printer environment.

Copying Specifications

You can copy one of the three specifications, or all of the specifications, to the selected output. The command

copy all_specifications to printer <RETURN>

to copy all specification information to the printer. The "default" specifications will look like:

```
64700 Timing Analyzer                   Tue Jun 16 09:13:31 1989

     Trace Specification            Timing (64700), 16 channels, 100MHz

     STANDARD MODE

    TRIGGER
         on
            anything
         position_is start_of_trace

    SAMPLE
         period_is 10 nsec
         rate_is 100 MHz


    Format Specification            Timing (64700), 16 channels, 100MHz

     STANDARD MODE

                   Xbits_upper               Xbits_lower
 Threshold:         ttl +1.40V                ttl +1.40V
  Label      15 14 13 12  11 10  9  8   7  6  5  4   3  2  1  0   polarity
 XBITS        *  *  *  *   *  *  *  *   *  *  *  *   *  *  *  *   pos_true
 X_lower                                *  *  *  *   *  *  *  *   pos_true
X_upper      *  *  *  *   *  *  *  *                             pos_true


 Post_process Specification          Timing (64700), 16 channels, 100MHz
        STANDARD MODE             10 nsec/sample

 MARK  STATUS  on_first_occurrence_of                               NAME
   x     on    default (start_of_trace)
   o     on    default (end_of_trace)
```

When selecting a file as output, data is appended to an existing file by default. You can choose to overwrite an existing file with a command like

copy trace_specification to `TRACESPEC` ***noappend*** `<RETURN>`

which copies the current trace specification to a file TRACESPEC, replacing any data in the file if it already exists.

Copying Trace Data

You can copy the trace memory data in either the timing diagram or trace list format. Use

copy timing_diagram to printer `<RETURN>`

to print the current timing diagram on a graphics printer.

Note

The output of a "copy timing_diagram", or "copy display" when displaying the timing diagram, is a PCL file and should only be used with devices or programs capable of handling that format.

These commands only work on the HP 9000 Series 300/400 computer. If you are using an HP 9000 Series 700 or Sun SPARCsystem computer running the X Window System, you can use the UNIX **xwd** and **xpr** commands to print the contents of the timing analyzer window.

The "copy trace_list" command allows you to print the entire listing or to choose a range of trace memory samples. When choosing a range, the cursor position is one end and a selected reference point is the other. Reference points can be the trigger, mark_x or mark_o, a selected sample number, or the start or end of trace memory. If the cursor is at sample number zero, the command

copy trace_list thru `100` ***to*** `TraceRange` `<RETURN>`

will copy trace memory samples zero through 100 to the file "TraceRange" in trace list format.

Copying Measurement Data

The Timing Analyzer can also create a file containing "raw" trace memory data in hexadecimal format. This file can be used for other analysis. The command

copy measurement_data_in_hex to TIMING_DUMP <RETURN>

copies the current trace memory data to a file named *TIMING_DUMP* in hexadecimal format.

Note Refer to appendix B "Timing Output and Diagrams" for information on the format of the file.

Ending a Session

The Softkey system maintains the current status of the analyzer for all users. The method you use to end a timing, or emulator, session determines how the status information is used to coordinate interactions if more than process is using the same analyzer.

There are four options when ending a timing analysis session: exiting and releasing the emulator/analyzer session, exiting with the intent of continuing but not effecting other users, exiting with the intent of continuing and blocking access to the emulator/analyzer, and exiting and selecting another measurement system. The latter option is available only if you are using the timing interface under *pmon* or *MEAS_SYS* environment.

Releasing the System

If you use the command

end release_system <RETURN>

the Timing Analyzer coordinates the normal shutdown of all processes using the analyzer. The sessions of all users, whether on separate machines, on one machine, or running in multiple windows on one monitor, are ended and the analyzer status information is discarded.

In order to use the system again, you must reinitialize the analyzer configuration.

Ending to Continue Later

You can continue later without effecting other users by simply using the command

end <RETURN>

which ends the current session, but preserves the configuration status. Other users are not effected.

Ending and Blocking other Access

You can also end a session with the intention to continue later and block other users access to the analyzer by using the command

end locked <RETURN>

which ends the current session, preserves the configuration status, and ends other users sessions. You can then reexecute the Timing Analysis Softkey Interface later, assuring that you return to the system as you left it.

Selecting the Measurement System or Another Module

In the *pmon* or *MEAS_SYS* environment, if you have configured the system with more than one measurement system, you can select one of the other measurement systems when ending the Timing Analysis interface.

Notes

9

Timing : Commands

Overview

This chapter includes:

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

Softkey Interface Features

Softkeys

Softkey Interface commands are entered 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 have entered a command, but want to make a change or correction, you can 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, **< CTRL> -U** is a quick way to erase the current command line. 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 on the command line 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.

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 one of the editors available on your host computer. See the chapter on command files in the *HP 64700-Series Emulators Softkey Interface Reference* 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:

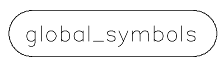
enter: **help**	then press a softkey that appears
enter: **?**	then press a softkey that appears
enter: **pod_commands "help emul"**	this will return help from the emulator (include quotation marks)

Syntax Conventions

Conventions used in the command syntax diagrams are defined below.

Oval-shaped Symbols

Oval-shaped symbols indicate 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:

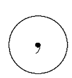


Rectangular-shaped Symbols

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.

Circles

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



Summary of Commands

Softkey Interface commands for the external timing analyzer are summarized in table 9-1.

The commands available on each screen are outlined in table 9-2. The following commands can be entered at the command line on all screens: !HP-UX_COMMAND, cd, diagram, format, help, list, log_commands, pod, trace, and wait.

Table 9-1. Summary of Commands

!HP-UX_COMMAND[1]	log_commands[1]
activity_test	magnify
cd[1]	mark
compare	mode_is
configuration	modify
copy	pod[1,2]
CURSOR	pod_command
default	post[1,2]
define	present (timing_diagram)
delete	present (trace_list)
diagram[1,2]	process_for_data
display	rename
end	< ROLL>
execute	sample
find	statistics
format[1,2]	threshold
halt	trace[1,2]
halt_repetitive_execution	trigger
help[1]	wait[1]
indicate	waveform_size
list[1,2]	

[1] These commands are not displayed on softkeys.

[2] These commands invoke the corresponding softkey commands.

Table 9-2. Command Assignments

Commands	trace	format	post	diagram	list	pod	error	event
activity_test		x						
compare			x					
configuration	x	x	x	x	x	x	x	x
copy	x	x	x	x	x	x	x	x
CURSOR				x				
default	x	x	x			x	x	x
define		x						
delete		x						
display	x	x	x	x	x	x	x	x
end	x	x	x	x	x	x	x	x
execute	x	x	x	x	x	x	x	x
find				x	x			
halt	x	x	x	x	x	x	x	x
halt_repetitive_execution			x					
indicate				x	x			
magnify				x				
mark			x	x	x			
mode_is	x	x						
modify		x						
pod_command						x		
present				x	x			
process_for_data			x		x			
rename		x						
< ROLL>				x	x			
sample	x			x	x			
statistics			x					
threshold		x						
trigger	x							
waveform_size				x				

Command Syntax

The syntax for the HP 64700 external timing analyzer varies considerably from that used in the HP 64700-Series emulators. Therefore, the complete timing analyzer syntax is presented here. In certain cases, you may want to refer to your *Emulator Softkey Interface User's Guide* or the *HP 64700-Series Emulators Softkey Interface Reference* for details about how your emulator operates.

activity_test

This command toggles the display of probe signal activity.

Syntax

Function

This is a format specification command that displays the activity on the external analyzer probe. If the activity_test is currently being displayed, this command turns the activity_test display off. The activity line appears at the top of the label list.

Default Values

Probe signal activity is not displayed. If you display another screen, the activity_test is set to its default value, toggled off.

Parameters

none

Examples

display format_specification <RETURN>
activity_test <RETURN>

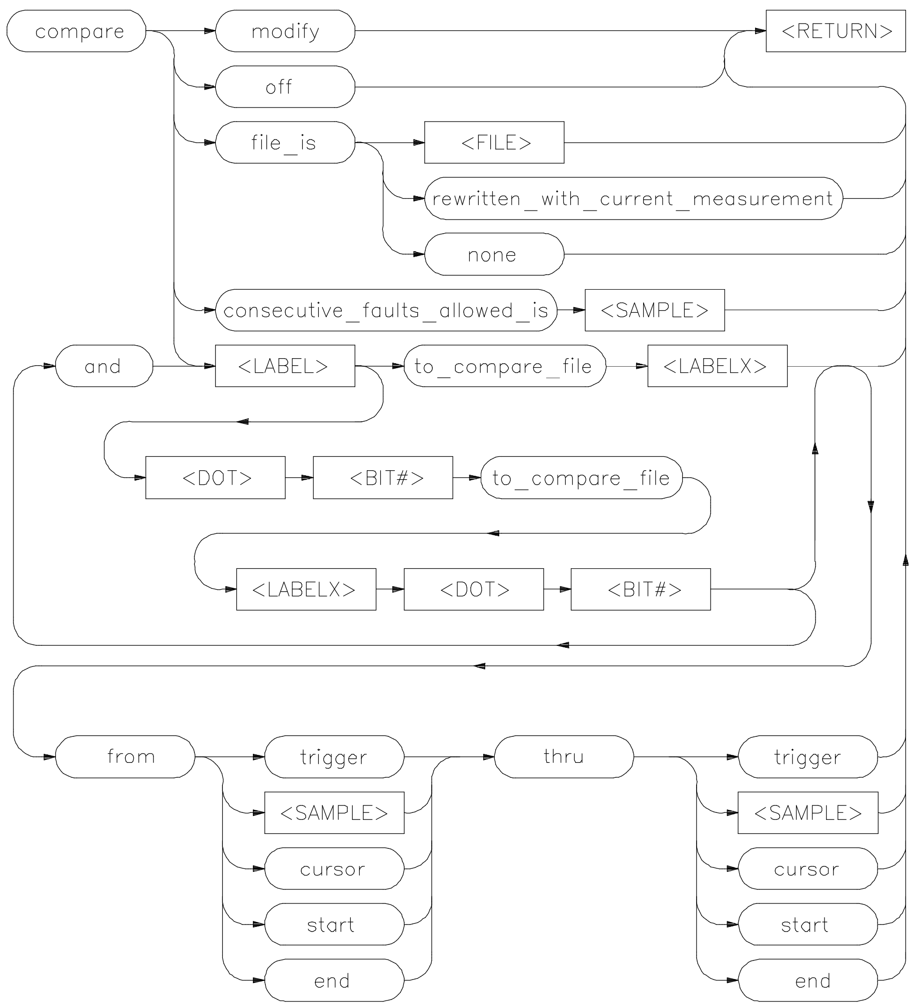
Related Commands

none

compare

This command specifies a post processing compare definition.

Syntax



* available only when compare file is selected

Function

This post process specification command is used to select a compare file or to specify a comparison to be performed after each measurement.

Default Values

none

Parameters

< BIT# >	This prompts you to enter the integer bit number to be used.
consecutive_faults_ allowed_is	This option allows you to define the number of consecutive faults that will be processed.
< SAMPLE>	This prompts you to enter the integer number of faults.
< DOT>	This prompts you to enter the literal "." which, if used, indicates that a specific bit number will be designated.
file_is	This option allows you to select a compare file compare file for compare and display (refer to "present"). Compare files are configuration files saved using the "with_data" option.
< FILE>	This prompts you to enter the name of the compare file.
from	This option specifies the trace memory location point from which the comparison will begin.
trigger	This parameter specifies the trigger point as the beginning point of the comparison.
< SAMPLE>	This parameter specifies the sample number entered as beginning point of the comparison.

cursor	This parameter specifies the cursor location as the beginning point of the comparison.
start	This parameter specifies the start of trace memory as the beginning point of the comparison.
end	This parameter specifies the end of trace memory as the beginning point of the comparison.
< LABEL>	This prompts you to enter a label name or to select a label from one of the softkeys.
< LABELX>	This prompts you to enter a label from the compare file or to select a compare file label from one of the softkeys.
modify	This option returns the current compare command to the command line for editing.
off	This option defaults the comparison definition.
rewritten_with_ current_ measurement	This option allows you to rewrite the current trace measurements to the currently selected compare file.
to_compare_file	This option allows you to compare measurement data from a previously selected compare file with data from "current" measurements.
thru	This specifies the end comparison range point. The valid parameters are the same as those for the **from** option.

Examples

display post_process_specification `<RETURN>`
compare file_is `FIRST_TRACE <RETURN>`

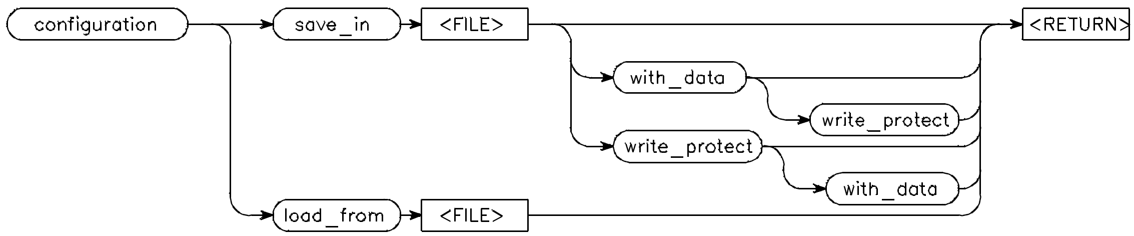
Related Commands

configuration
present

configuration

This command creates or retrieves a configuration file.

Syntax



Function

The configuration command allow you to save configurations or to load a previously saved file to reset specifications. Measurement data may also be saved with the configuration information. In that case, the configuration can be used as a compare file.

Default Values

none

Parameters

load_from	This option allows you to specify a configuration file to load. The saved configuration specifications are retrieved and reset the current specifications.
save_in	This option allows you to specify a file for saving the current specification settings.
< FILE>	This prompts you for a file name. A trace (.TR) file is created.
with_data	This parameter specifies the inclusion of the current measurement data in the file along with the specifications; use this option to

create a file for use with the compare command.

write_protect — This parameter sets the write protect variable in the file descriptor.

Examples

configuration save_in FIRST_TRACE ***with_data*** <RETURN>

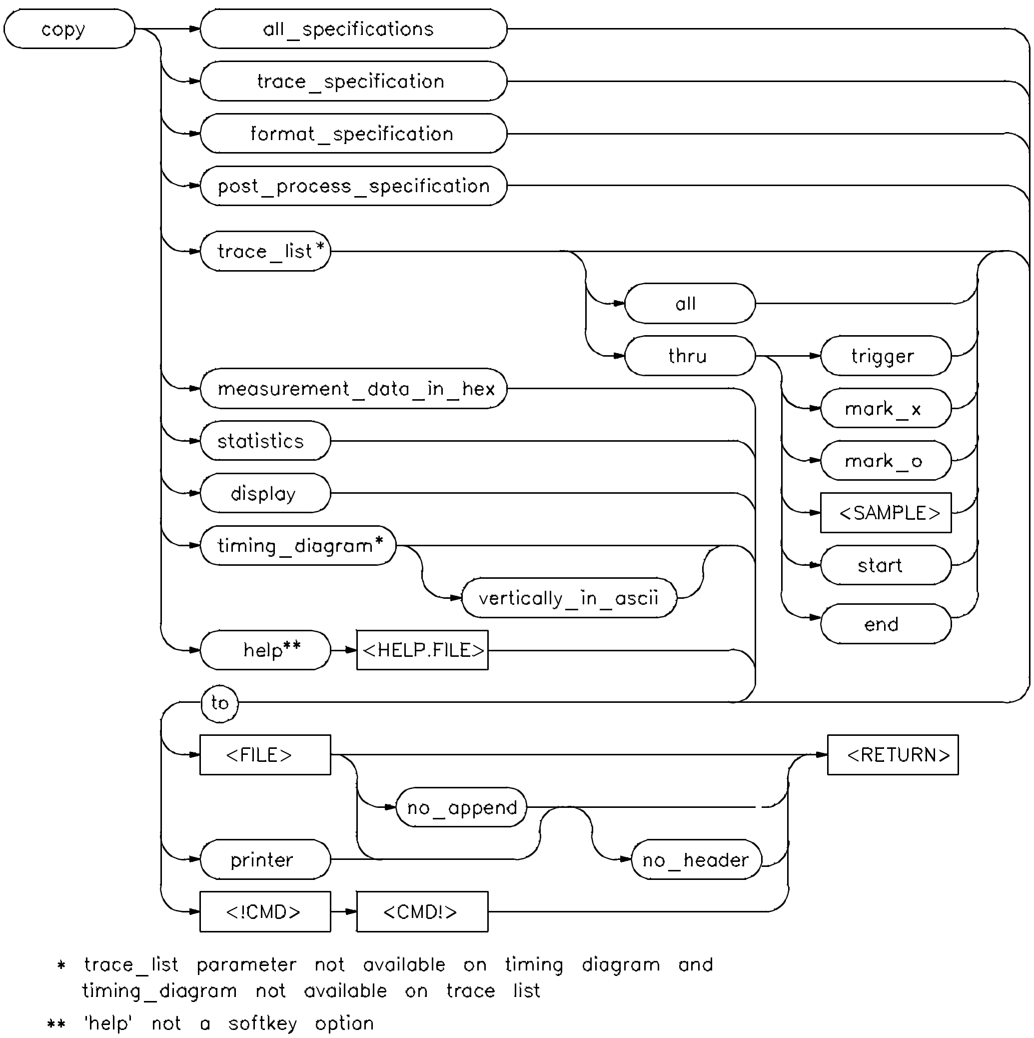
Related Commands

compare

copy

This command copies specifications, displays or measurement data to selected output.

Syntax



Function

The copy command allows you to copy the all or selected specifications, displays or lists, statistics, help, or measurement data to a file, printer or HP-UX command.

Note

< CTRL> -C (SIGINT) will interrupt a copy command.

Default Values

none

Parameters

all_specifications	This option copies all of the specifications (trace, format, and post process) to the selected output in their entirety.
display	This option copies the current screen to the selected output.
format_specification	This option copies only the format specification information to the selected output.
measurement_data_in_hex	This option copies the captured timing measurement data to the selected output in hexadecimal form. This allows for other analysis of the raw data. The format for this information is described in appendix B later in this manual.
noappend	This parameter forces the overwrite of a file's information if the selected file name already exists.
noheader	This parameter suppresses the report header from the output. The header contains the source ("64700 Timing Analyzer") and the day, date and time of the output. This parameter is useful when building a listing

file from multiple executions of the command.

post_process_ specification	This option copies only the post process specification information to the selected output.
timing_diagram	This option copies the timing diagram to the selected output. The default format is in "raw" HP PCL format.
vertically_in_ascii	This parameter specifies the timing diagram output to be in ASCII characters oriented vertically on the page. That is rotated clockwise ninety degrees from its normal orientation.
trace_list	This option copies the trace list to the selected output. This option is available in all display forms except trace_specification, format_specification and timing_diagram.
all	This parameter specifies that all of the trace memory data will be copied.
thru	This parameter selects a range in trace memory for the copy. The first point is the cursor position. The second point is a selected parameter.
trigger	This parameter specifies that trace memory data between the cursor and the trigger will be copied.
mark_x	This parameter specifies that trace memory data between the cursor and mark_x will be copied.
mark_o	This parameter specifies that trace memory data between the cursor and mark_o will be copied.

< SAMPLE>	This prompts for you to enter a sample number. Trace memory data between the cursor and the entered sample number will be copied.
start	This parameter specifies that trace memory data between the start of memory and the cursor will be copied.
end	This parameter specifies that trace memory data between the cursor and the end of memory will be copied.
trace_specification	This option copies only the trace specification information to the selected output.
help	This is a "hidden" option that copies the help files to the selected output.
< FILE>	This prompts you for a file name to contain the output information.
printer	The option selects the printer as output. If text output is being generated the output is piped to the program in the PRINTER environment variable. If the output is graphics, the GPRINTER environment variable is used. See "Timing Output and Diagrams" in appendix B later in this manual.
< !CMD!>	This prompts you to enter the literal "!" to indicate that a program name will be entered. In this case, the output is piped to standard output.
< CMD!>	This prompts you to enter the program name to receive the piped output from the copy command.

to — This option initiates the selection of a destination for the specified data.

Examples

copy measurement_data_in_hex to HEXDUMP <RETURN>

copy display to printer <RETURN>

copy all_specifications to specfile<RETURN>

Related Commands none

CURSOR

This command toggles the use of the cursor keys from the command line to the timing diagram.

Syntax None

Function When the CURSOR softkey appears without the asterisk, the right and left arrow keys are available for command line editing. When the CURSOR softkey appears with an asterisk (CURSOR*), you can move the cursor in the timing diagram display with the right and left arrow. The sample number of the cursor position is shown on the right of the display. This command is only available from the softkeys.

Default Values The right and left arrow keys are set up for command line editing.

Parameters None.

Examples

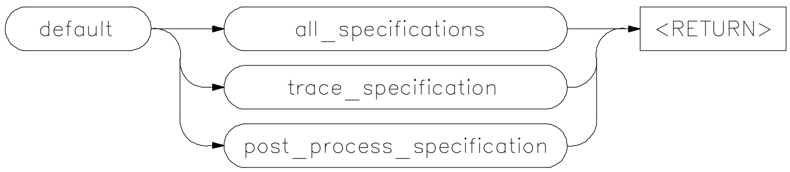
display timing_diagram <RETURN>
CURSOR <RETURN>
CURSOR* <RETURN>

Related Commands **magnify**

default

This command sets specifications to their default values.

Syntax



Function

The default specifications can be set for the trace or post process specifications or for all specifications.

Default Values

none

Parameters

all_specifications	This option sets the trace, format, and post process specifications to their default values.
post_process_ specification	This option sets the post process specification to its default values.
trace_specification	This option sets the trace specification to its default values.

Examples

default all_specifications <RETURN>

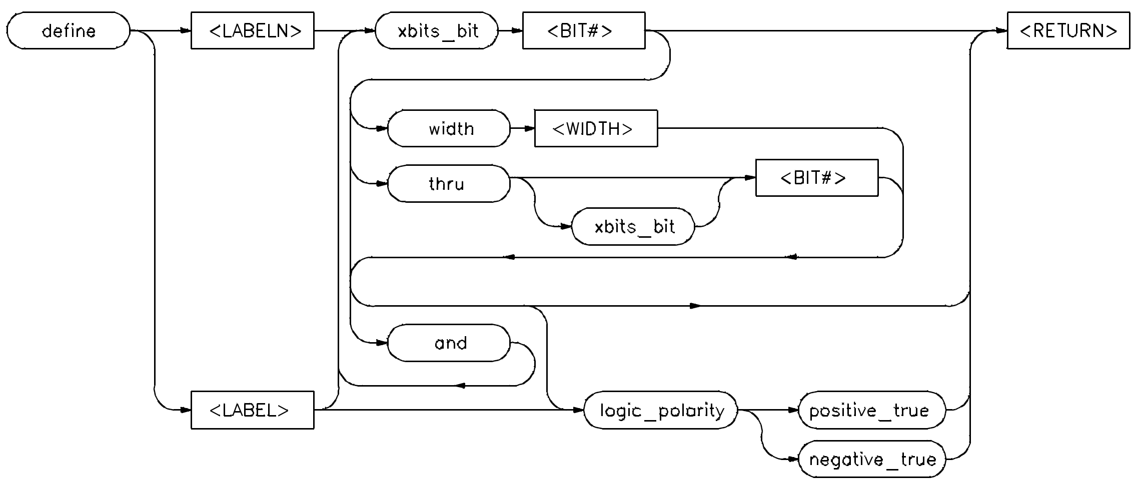
Related Commands

configuration

define

This command creates a label which is used to refer to probe signals.

Syntax



Function

This is a format specification command that creates a new label and associates one or more probe signals to the label. Logic polarity can also be designated. An existing label definition can be modified using this command.

Default Values

The default polarity is "positive true".

Parameters

< LABELN> This represents the prompt < LABEL> for a new label. The label name is entered in the command line. Labels can be up to eight characters long and must begin with a letter.

< LABEL>	This prompts you for the name of an existing label either entered or selected from a softkey.
xbits_bit	This is a syntactical item referring to the external probe signal bits.
< BIT# >	This prompts you to enter the starting bit number.
width	This option selects the signals from the previously selected < BIT# > for an integer number (< WIDTH>) of signals.
< WIDTH>	This prompts you for the integer number of consecutive signals to associate with the label.
and	This parameter allows the logical "and" of the previously selected signals with those to follow.
logic_polarity	This parameter is used to set the sense of a "1" as more positive or more negative than the probe threshold voltage. This allows trigger definitions to be made in terms of 1s and 0s, independent of the voltage sense of the lines being measured. The trace list values will reflect these definitions.
positive_true	This sets the logic_polarity to true if the voltage is positive.
negative_true	This sets the logic_polarity to true if the voltage is negative.

Examples

```
define CLOCK xbits 0 logic_polarity
positive_true <RETURN>
```

```
define DATA xbits 8 width 8  <RETURN>
```

Related Commands

delete
rename
threshold

delete

This command deletes one or all labels.

Syntax

Function

This format specification command deletes one or more labels. If a label is used in another specification, an error message will inform you it cannot be deleted. If a deleted label has been reference in the timing diagram or trace list, it is automatically removed from the display.

Default Values

none

Parameters

all_labels	This option deletes all of the defined labels. It is normally used to clear the default labels before defining new ones.
< LABEL>	This prompts you to enter the name of the label to delete or to select one from a soktkey.

Examples

delete X_lower <RETURN>

Related Commands

rename
define

diagram

This command initiates the timing diagram softkeys.

Syntax none

Function This command invokes the "diagram" softkeys from whatever "display" mode you are currently in. This allows you to enter timing_diagram specific commands without the necessity of using "display" to change specification modes.

Default Values none

Parameters Any of the available timing diagram commands.

Note

Some of the timing diagram softkeys are not activated from this command.

Examples

diagram mark x on_trigger <RETURN>

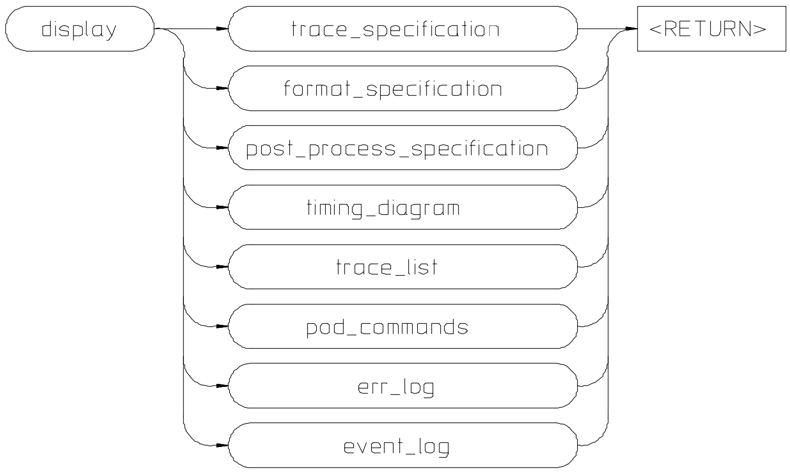
Related Commands

format
list
pod
post
trace

display

This command selects the screen to display.

Syntax



Function The display commands moves between the three specification screens and the two output screens. Three special function screens can also be displayed: pod commands, error log, and event log.

Entering any of the words trace, format, pod, post, diagram, and list will allow the commands from that specification or output screen to be accessible. As an example, from the timing diagram display you can change the "mode" by entering: "trace mode_is standard".

Default Values The default screen is the trace specification.

Parameters

trace_specification — This option selects the trace specification where you can specify an event that will trigger the timing analyzer to begin acquiring data.

format_specification	This option selects the format specification where you can define labels and set thresholds which the analyzer will use.
post_process_ specification	This option selects the post processing specification where you can define events to be completed after every execution.
timing_diagram	This option selects the timing diagram where you can observe the measurement data in a graphic representation.
trace_list	This option selects the trace list where you can observe the measurement data in a columnar report format.
pod_commands	This option select the pod commansd screen where you can enter native pod commands and display text from the analyzer firmware.
err_log	This option selects the error log screen which displays a roster of command errors. The information displayed includes time, the erroneous command line, and the text of the error message.
event_log	This option select the event log screen which displays a roster of recent events such as changes of status of the analyzer. The information displayed includes time, type of event, and an event message.

Examples

display timing_diagram <RETURN>

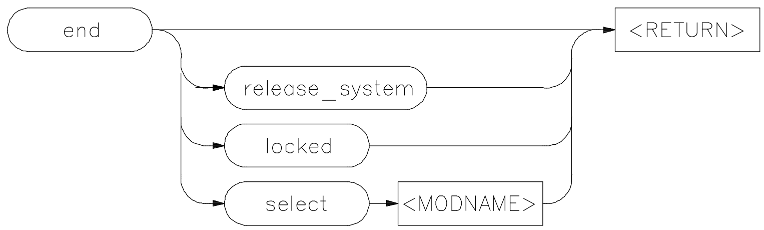
Related Commands

none

end

This command terminates the analyzer session.

Syntax



Function

This command terminates the current timing session. If you end or end locked, you keep the analyzer in a locked state. In addition, end locked terminates the any other sessions running on the analyzer in other windows or on other terminals. This saves the current emulation configuration so that on reentry of the analyzer, you can continue the analyzer session. If you are in the Measurement System, you can select another measurement system when ending the analyzer. You can also release the system when ending the session so that others may access and use the analyzer.

The options available for the "end" command depend upon how this emulation session was started:

emul700:	This command allows you to start multiple instances of the interface controlling the same emulator/analyzer from one or more windows and/or terminals. You can "end" just one instance or all instances at once.
Measurement System:	Only a single instance of the user interface is allowed. The "end" command ends that instance, and optionally allows you to select another module defined in the same Measurement System, or the "measurement_system" user interface itself.

You return to the HP-UX shell or PMON depending on how you entered emulation. Unless you choose "end release_system", the current emulation configuration is stored so that on reentry to the emulation module, you can resume the emulation session.

Note

Entering **< CTRL> -D** performs the same operation as entering "end < RETURN> ".

Entering **< CTRL> -**\ or **< CTRL> -|** (in other words, sending SIGQUIT to the user interface process) is the same as entering "end release_system < RETURN> ".

Default Values

When the analyzer session is ended, control is returned to the environment (HP-UX or PMON) you were in when the Softkey Interface was entered without releasing the analyzer. The analyzer is locked to the current user so that the session may be continued later. Other instances of the user interface are not affected.

Parameters

locked — This option closes all active instances of the user interface in any combination of windows and terminals. Each closed instance will return to the environment in which the Softkey Interface was entered. Thus, "end locked" is the same as entering "end" at all of the instances. The configuration of the ending instance becomes the configuration used when you start the analyzer later. This option is not available when operating the analyzer in the measurement system.

release_system — This option closes all active instances of the user interface in any combination of windows and terminals. Each closed instance will return to the environment in which the Softkey Interface was entered. In addition, the emulator is unlocked so that it may be used by other users on your HP-UX system. The information needed to continue your session is also removed. (If you do not release the system, no other users may access it.)

select — This measurement system option allows you to select another module or to enter the measurement system.

< MODNAME> — This prompts you for the name of another module in the measurement system. The analyzer ends and the named module immediately starts. This option will only appear if other modules are in configured in your Measurement System. The current configuration is saved so that you can return to this module later.

measurement_system — This measurement system option ends the analyzer and enter into the measurement system module.

Examples

end <RETURN>

end release_system <RETURN>

end select measurement_system <RETURN>

Related Commands

none

execute

This command starts one or more timing measurements.

Syntax

Function

The timing analyzer can make both single module or multiple module measurements by using the execute and halt commands. The execute command will start the timing analyzer alone (single module measurement) if the timing analyzer is not connected to the intermodule bus (IMB).

The execute command will start the timing analyzer and all other modules connected to the IMB if the timing analyzer is connected (trigger enable, trigger received, etc).

Default Values

An execute command with no parameters will start an execution of this module or if this module is connected to the IMB (trigger enable, trigger received, etc.) the execution will start all modules connected to the IMB.

Parameters

repetitively	This option allows you to repetitively execute measurements. This allows you to accumulate measurements for statistical analysis. As soon as one execution completes another will be started until a "halt" is executed from the command line or the "halt_repetitive_execution" conditions are met.

Examples

execute repetitively <RETURN>

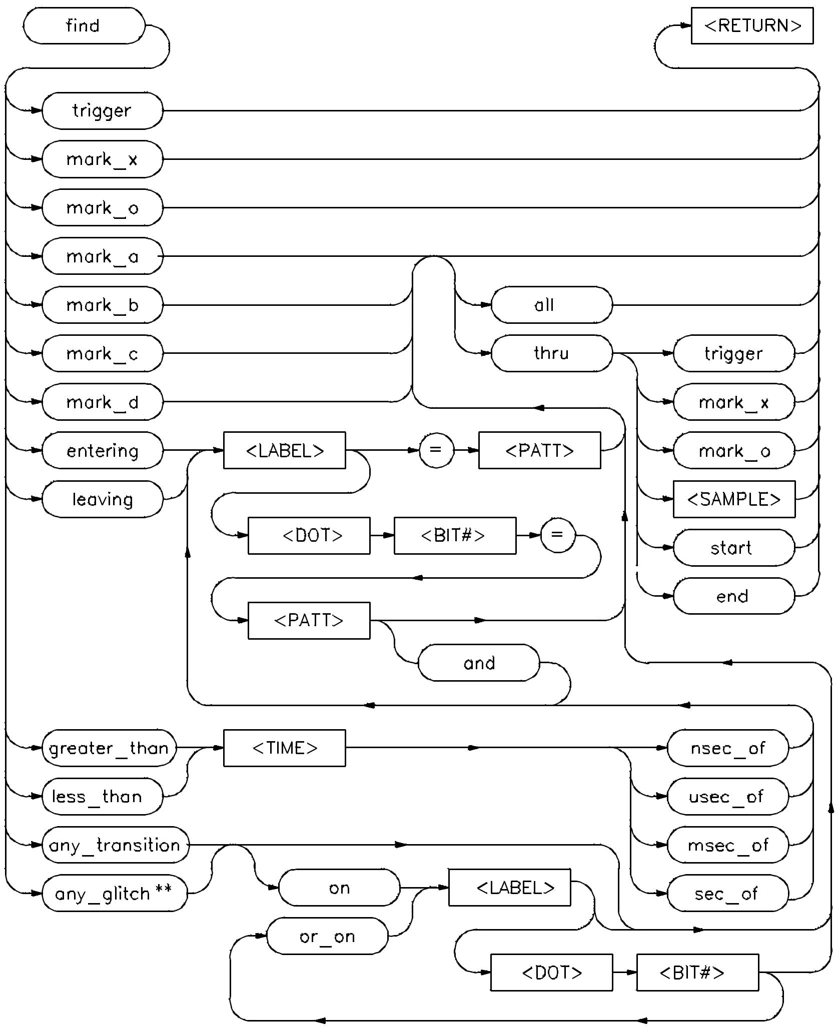
Related Commands

halt
halt_repetitive_execution

find

This command finds a trigger-like event in trace memory.

Syntax



** available only on glitch_capture mode

Function This command locates the specified event in trace memory and brings it into the display, centers it, if possible, and locates the cursor at that point. You can use this command to locate the trigger, any of the marked samples, or specified signal conditions on a label, multiple labels or a combination of label bits.

Default Values none.

Parameters

all	This parameter specifies that all of the trace memory data will be searched.
any_glitch	This option locates the object label, labels, or combination of label bits that have a "glitch" within the range specified.This option is available only in glitch_capture mode.
any_transition	This option locates the object label, labels, or combination of label bits that have a "transition" within the range specified .
< BIT# >	This prompts you to enter the integer bit number.
< DOT>	This prompts you to enter the literal "." to designate a specific bit number for a label.
entering	This option locates the object label, labels, or combinations of label bits entering the specified pattern within the specified range.
=	This designates the assignment of a specific pattern entered as a numerical value for comparison.
greater_than	This option locates the object label, labels, or combinations of label bits with the

specified pattern for more than the selected duration within the specified range.

< LABEL>	This prompts you to enter a label name or select the label name from the softkeys.
leaving	This option locates the object label, labels, or combinations of label bits leaving the specified pattern within the specified range.
less_than	This option locates the object label, labels, or combinations of label bits with the specified pattern for less than the selected duration within the specified range.
mark_x	This option locates and displays the mark_x event.
mark_o	This option locates and displays the mark_o event.
mark_a	This option locates a "a" marked event within the specified range.
mark_b	This option locates a "b" marked event within the specified range.
mark_c	This option locates a "c" marked event within the specified range.
mark_d	This option locates a "d" marked event within the specified range.
< PATT>	This prompts you to enter a pattern of signals. The pattern is a numerical value, the significance of which is dependent on the number of signals being tested. Refer to the section on *Entering Numerical Values* in this manual for options on entering patterns.

thru	This parameter selects a range in trace memory to search. The first point is the cursor position. The second point is a selected parameter.
trigger	This parameter specifies that trace memory data between the cursor and the trigger will be searched.
mark_x	This parameter specifies that trace memory data between the cursor and mark_x will be searched.
mark_o	This parameter specifies that trace memory data between the cursor and mark_o will be searched.
< SAMPLE>	This prompts for you to enter a sample number. Trace memory data between the cursor and the entered sample number will be searched.
start	This parameter specifies that trace memory data between the start of memory and the cursor will be searched.
end	This parameter specifies that trace memory data between the cursor and the end of memory will be searched.
< TIME>	This prompts you to enter the amount of time a pattern should exceed (greater_than) or not exceed (less_than).
nsec_of, usec_of, msec_of, sec_of	These parameters specify the units of measurement of < TIME> in nanoseconds, microseconds, milliseconds, or seconds, respectively.
trigger	This option locates and displays the trigger.

on	This option specifies the label, labels, or combination of label bits in the "any_transition" and "any_glitch" options.
or_on	This option further specifies the objects referred to in the "on" option.
and	This option further specifies the objects referred to in the *entering*, *leaving*, *greater_than*, or *less_than* options.

Examples

find entering XBITS .3 ***=*** 0 <RETURN>
find mark_a thru mark_o <RETURN>

Related Commands

mark

format

This command initiates the format specification softkeys.

Syntax

none

Function

This command invokes the "format" softkeys from whatever "display" mode you are currently in. This allows you to enter format specification specific commands without the necessity of using "display" to change specification modes.

Default Values

none

Parameters

Any of the available format specification commands.

Note

Some of the format_specification softkeys are not activated from this command.

Examples

format mode_is glitch_capture <RETURN>

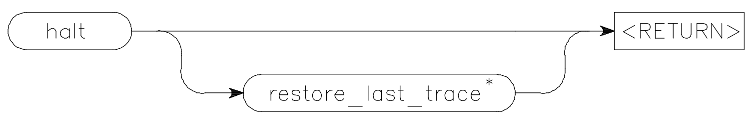
Related Commands

diagram
list
pod
post
trace

halt

This command terminates the execution in process.

Syntax



* available only during repetitive execution

Function

The halt command will terminate the current measurement and display the last segment of captured data before the halt command was entered. This command is only available while executing measurements and is most useful to halt a repetitive execution.

Default Values

The available measurement information is retrieved. If trace memory was only partially filled, only that information will be available for display.

Parameters

restore_last_trace — This option will restore the last completed trace. This is available when using repetitive execution and is most useful when the trigger event does not happen very often.

Examples

halt restore_last_trace <RETURN>

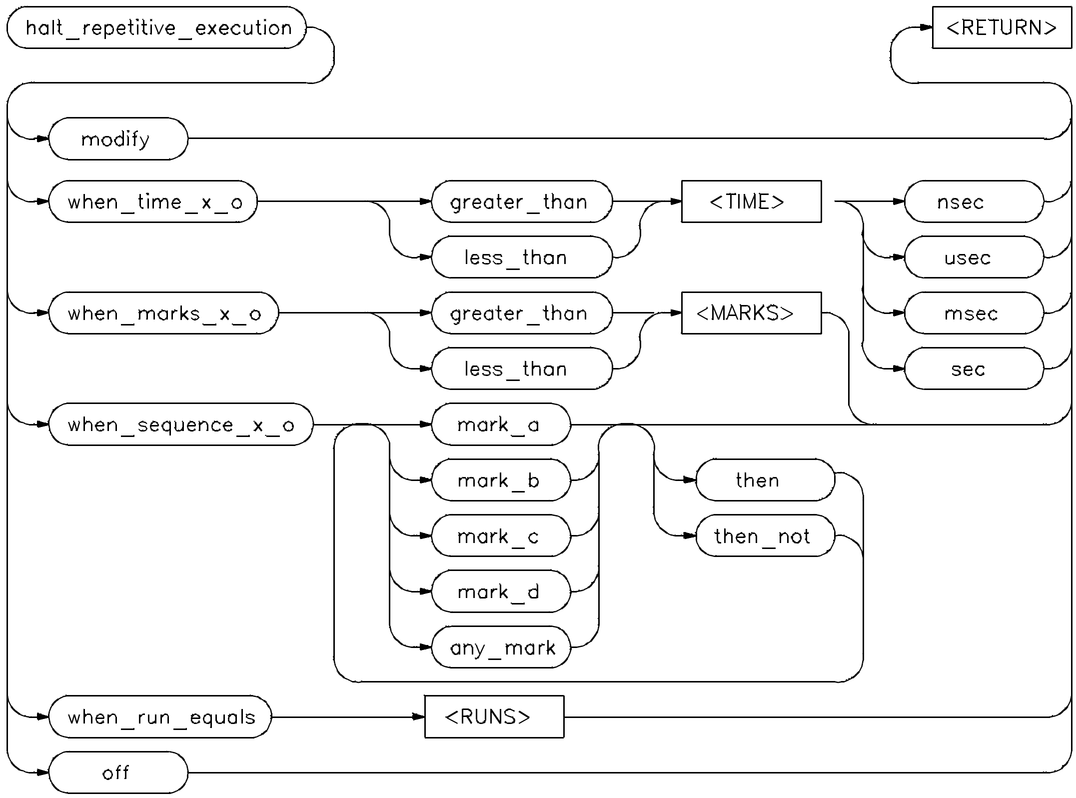
Related Commands

execute
halt_repetitive_execution

halt_repetitive_execution

This command conditionally halts a repetitive execution.

Syntax



Function

This post processing command terminates a repetitive execution if the conditions specified in the command are met. Conditions include numbers of runs, specified sequences of marks, specified mark counts, or when the time between mark_x and mark_o is more or less than a certain value.

Default Values

A repetitive execution will continue until halted by command or post processing conditions are specified.

Parameters

modify	This option returns the current halt_repetitive_execution command to the command line for editing.
when_marks_x_o	This option specifies a count of the marks (a,b,c,d) between the mark_x and mark_o as the condition to halt a repetitive execution. The count can be more or less than a specified amount.
greater_than	This parameter specifies that the count must be more than the selected value.
less_than	This parameter specifies that the count must be less than the selected value.
< MARKS>	This prompts you to enter the count of marks.
when_runs_equal	This option specifies the halt condition to be a prescribed number of measurements. This is useful when gathering statistical information.
< RUNS>	This prompts you to enter the integer number of runs.
when_sequence_x_o	This option specifies a sequence of marks, or an absence of marks, between mark_x and mark_o as the condition to halt a repetitive execution.

mark_a, mark_b, mark_c, mark_d, any_mark	One of these parameters is selected as a condition.
then	This parameter allows you to include an additional condition in a sequence.
then_not	This parameter allows you to exclude a condition from a sequence.
when_time_x_o	This option specifies a mark_x to mark_o duration as the condition for halting a repetitive execution. If the interval is *greater_than* or *less_than* the interval between mark_x and mark_o then execution is halted.
greater_than	This parameter specifies the duration must be more than the selected time.
less_than	This parameter specifies the duration must be less than the selected time.
< TIME>	This prompts you to enter the amount of time for the duration.
nsec, usec, msec, sec	These parameters specify the units of measurement of < TIME> in nanoseconds, microseconds, milliseconds, or seconds, respectively.
off	

Examples

```
halt_repetitive_execution when_time_x_o
greater_than 100 nsec <RETURN>
```

```
halt_repetitive_execution when_sequence_x_o
mark_a then mark_b then_not mark_c <RETURN>
```

Related Commands

execute
halt

help

This command allows you to display information about the system and analyzer features during an analyzer session.

Syntax

Function

Typing "help" or "?" causes the available help options to be displayed on the softkey labels. The system will display the help information on the screen when you select an option.

The help command is not displayed on a softkey. You must enter it into the command line from the keyboard. A question mark may be used in place of "help" to access the help information.

Default Values

none

Parameters

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

Examples

help system_commands <RETURN>

? format_specification_commands <RETURN>

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

system_commands
simple_measurement_commands
trace_specification_commands
format_specification_commands
post_process_specification_commands
display_commands
execute_commands
diagram_commands
list_commands
configuration_commands
copy_commands
graphic_diagrams
ascii_diagrams
diagram_outputs
end_commands

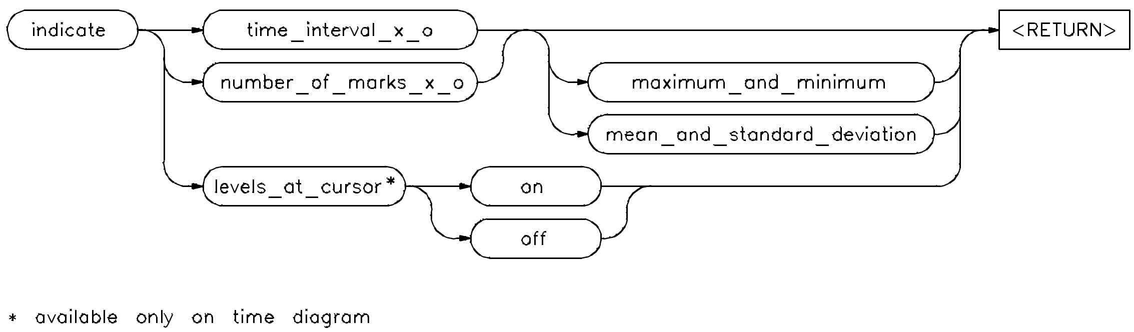
Related Commands

none

indicate

This command selects the statistical information displayed.

Syntax



Function

This command allows you to select display information. The display information can be either the time between mark_x and mark_o or the number of marks (a, b, c, d) between mark_x and mark_o. For each of the two measures, the maximum and minimum values or the mean and standard deviation values are displayed. On the timing diagram, you can also set the display of signal levels at the cursor on or off.

Default Values

The default display uses the time interval between mark_x and mark_o. The signal levels at cursor are off.

Parameters

time_interval_x_o	This option specifies the mark_x to mark_o time interval is to be displayed.
number_of_marks_ x_o	This option specifies the mark count between mark_x and mark_o is to be displayed.

maximum_and_ minimum	This parameter specifies the maximum and minimum values for the selected display option are to be displayed.
mean_and_ standard_deviation	This parameter specifies the mean and standard deviation for the selected display option are to be displayed.
levels_at_cursor	This option selects the display of signal levels at the diagram cursor on the timing diagram. The level indicator can be turned on or off.
on	This parameter sets the level indicators for each of the displayed signals on.
off	This parameter sets the level indicators for each of the displayed signals off.

Examples

indicator time_interval_x_o maximum_and_minimum <RETURN>

indicator levels_at_cursor on <RETURN>

Related Commands

statistics

list

This command initiates the trace list softkeys.

Syntax

None

Function

This command invokes the "list" softkeys from whatever "display" mode you are currently in. This allows you to enter trace list specific commands without the necessity of using "display" to change specification modes.

Default Values

none

Parameters

Any of the available trace list commands.

Note

Some of the trace list softkeys are not activated from this command.

Examples

list process_for_data off <RETURN>

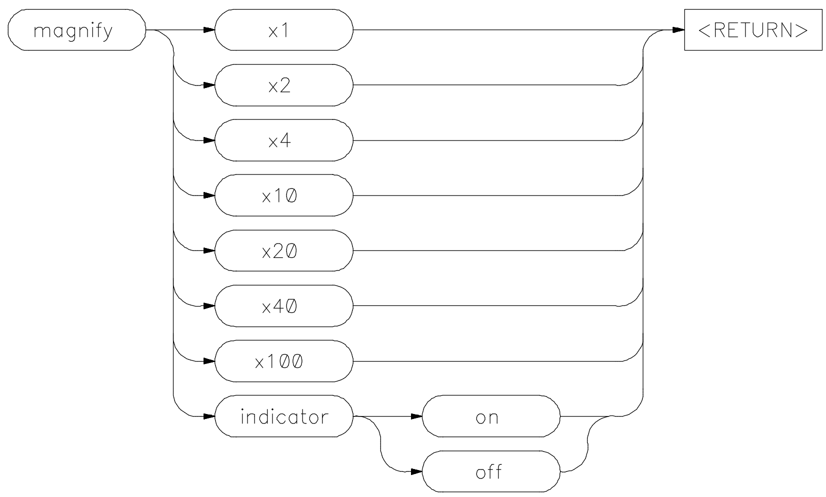
Related Commands

diagram
format
pod
post
trace

magnify

This command changes the timing diagram display resolution.

Syntax



Function

This timing diagram command allows you to change the time per division specification in the timing diagram display. This allows you to observe the signals in greater or lesser detail.

It also allows you to turn the indicator bar on or off. The indicator bar depicts the sample width that will be displayed after increasing the magnification by a factor of ten.

Default Values

The default magnification is x4 with the magnify indicator on.

Parameters

x1, x2, x4, x10, x20, x40, x100	Choose one of these options to set the magnification to a factor of 1, 2, 4, 10, 20, 40, or 100, respectively. The time per division information on the screen changes accordingly, along with the magnification value field. In ASCII diagrams, the time per character is indicated.
indicator	This option turns the magnify indicator on or off. The indicator shows the area of the display that will appear during the next x10 level of magnification. In x4 magnification, the magnify indicator shows the area of the display that will appear in the x40 magnification display. The indicator does not appear for magnifications over x10.
on	This parameter turns the indicator bar on.
off	This parameter turns the indicator off.

Examples

magnify x100 <RETURN>

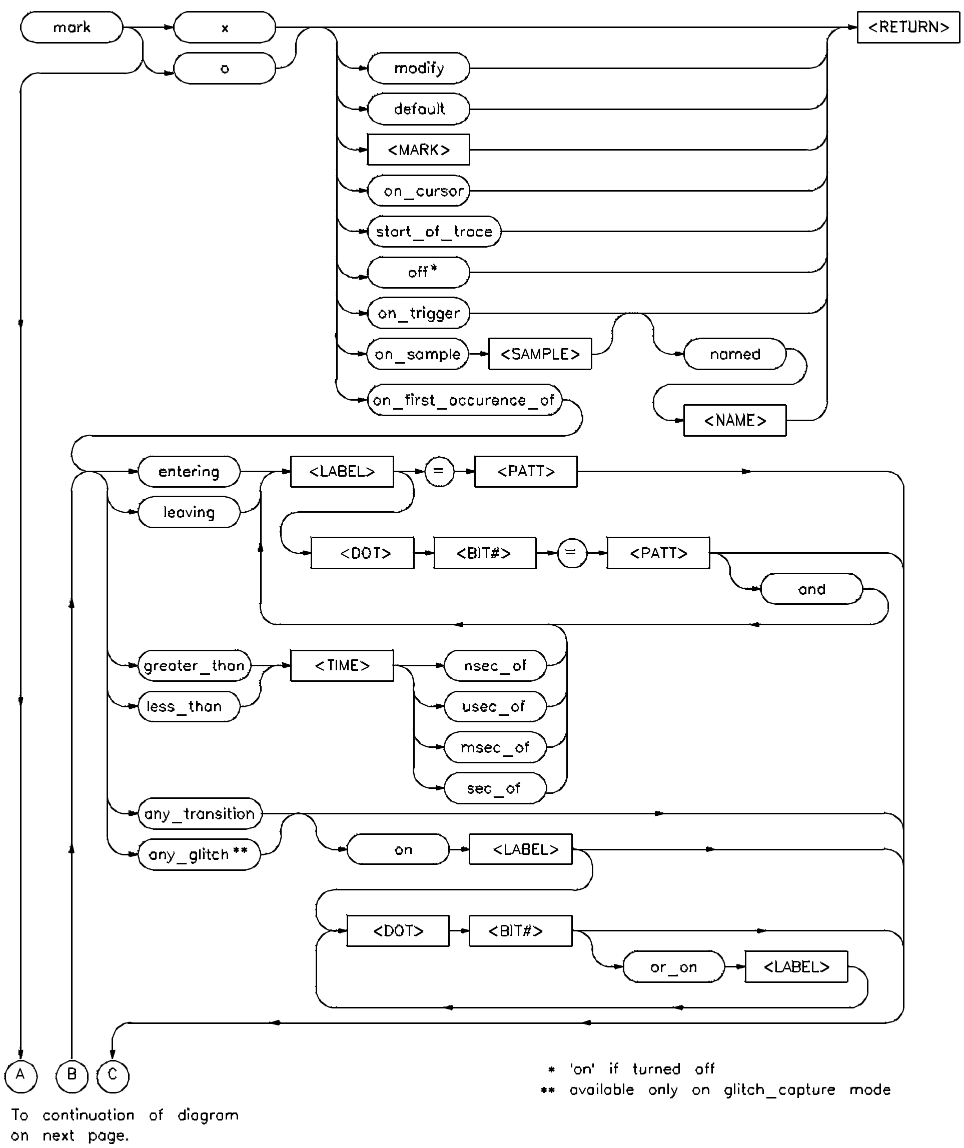
Related Commands

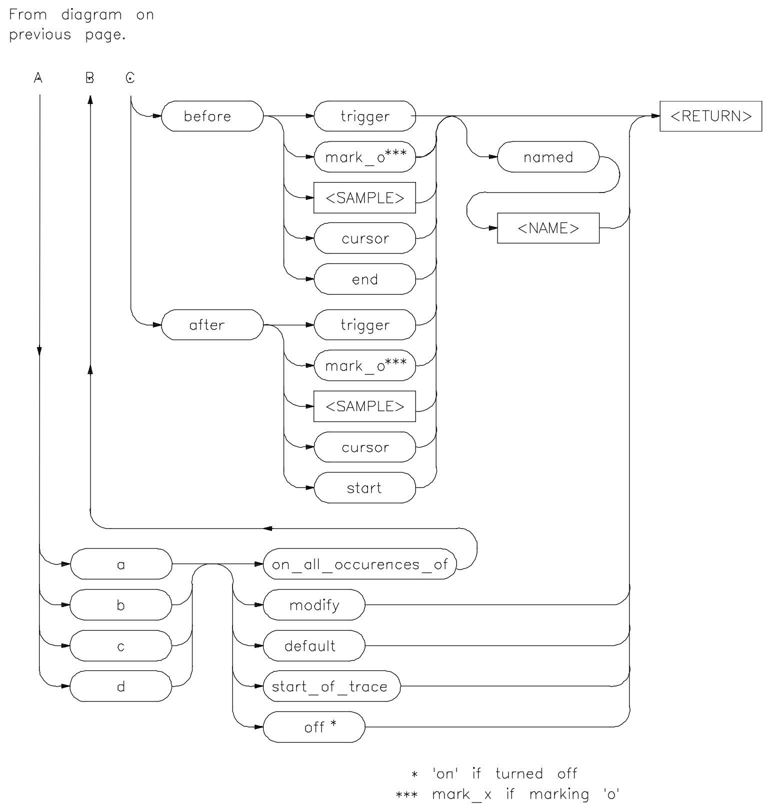
CURSOR

mark

This command marks specified conditions in trace memory.

Syntax





Function Marks are identifiers assigned to samples in trace memory. They can be assigned to any "event" you specify. Marks can be assigned from the command line for events in current trace memory, or as a post processing function. Marks specified at the command line are also store for subsequent post processing.

Marks can be used to:

- Define statistical ranges.
- Calculate time intervals.
- Select samples for display.

- Specify conditions for halting repetitive executions.

There six mark names recognized: mark_x, mark_o, mark_a, mark_b, mark_c, and mark_d. The first two marks, mark_x and mark_o, are single occurrence marks and are used to identify the first occurrence of a specified event. These two marks always exist and are used to define a range of samples. The others are multiple occurrence marks and are used to identify all occurrences of a specified event.

You can assign names to any of the marks. These names can be displayed in the trace list. Marks can also be turned off or on to facilitate processing.

Default Values

The defaults for mark_x and mark_o are "start of trace" and "end of trace", respectively. The other marks have no defaults.

Parameters

x, o, a, b, c, d	Choose one of these options to select the mark to be defined or modified.
after	This option qualifies the positioning of a mark by specifying it will be after certain condition.
and	This option allows you to specify an additional label or label bit to be added to the conditions.
any_glitch	This option specifies glitch samples be marked
on	This parameter qualifies the glitch condition by specifying a label name or label bit number.

or_on	This parameter allow you to add an additional glitch condition label name or label bit number.
any_transition	
on	This parameter qualifies the transition condition by specifying a label name or label bit number.
or_on	This parameter allow you to add an additional transition condition label name or label bit number.
before	This option qualifies the positioning of a mark by specifying it will be before certain condition.
< BIT# >	This prompts you to enter the label integer bit number.
cursor	This parameter specifies the position of the mark to be before or after the trigger.
default	This option resets the mark to its default value.
< DOT>	This prompts you to enter the literal "." to designate a specific bit number for a label.
end	This parameter specifies the position of the mark is to be before the end of trace memory.
entering	This option specifies a pattern condition will be entered.
=	This designates the assignment of a specific pattern entered as a numerical value for comparison.

greater_than	This option specifies a pattern will exist for more than an specified duration.
< LABEL>	This prompts you to enter a label name or select the label name from the softkeys.
leaving	This option specifies a pattern condition will be exited.
less_than	This option specifies a pattern will exist for less than an specified duration.
< MARK>	This prompts you to enter < RETURN> to place the mark at the current cursor location.
mark_o	This parameter specifies the position of mark_x before or after mark_o.
mark_x	This parameter specifies the position of mark_o before or after mark_x.
modify	This option returns the current mark command to the command line for editing.
named	This option specifies an name will be assigned to the mark. The names can be displayed in the trace list.
< NAME>	This prompts you to enter the mark name.
on_all_ occurrences_of	This option specifies a signal pattern as the condition for marking events for mark_a, mark_b, mark_c, and mark_d.
on_cursor	This option specifies the mark is to be moved to the current cursor location.
on_first_ occurrence_of	This option specifies a signal pattern as the condition for marking an event for mark_x and mark_o.

on_sample	This option specifies the mark is to be moved to the specified trace memory sample number.
on_trigger	This option specifies the mark is to be moved to the current trigger location.
< PATT>	This prompts you to enter a pattern of signals. The pattern is a numerical value, the significance of which is dependent on the number of signals being tested. Refer to the section on *Entering Numerical Values* in this manual for options on entering patterns.
< SAMPLE>	This prompts you to enter an integer sample number.
start	This parameter specifies the position of the mark is to be after the start of trace memory.
< TIME>	This prompts you for the time period to look for a specified pattern.
nsec_of, usec_of, msec_of, sec_of	These parameters specify the units of measurement of < TIME> in nanoseconds, microseconds, milliseconds, or seconds, respectively.
trigger	This parameter specifies the position of the mark to be before or after the trigger.

Examples

mark x on_first_occurrence_of entering XBITS
= 0B0H <RETURN>

mark a on_all_occurrences_of any_transition
after mark_x <RETURN>

mark d on <RETURN>

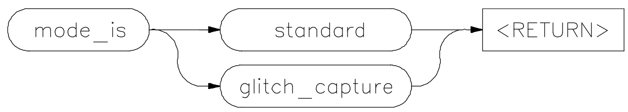
mark c on_all_occurrences_of entering XBITS
.1 ***=*** 0 ***named*** EVENT <RETURN>

Related Commands **find**

mode_is

This command sets the data acquisition mode.

Syntax



Function

This command sets mode to either "standard" or "glitch_capture". In the standard mode, trace memory can hold 1024 samples taken at the rate specified. In the glitch_capture mode, trace memory can hold 512 samples and glitchs are noted.

Default Values

The analyzer defaults to standard mode.

Parameters

standard	This option allows full analyzer depth to be used.
glitch_capture	This option will set the analyzer to detect glitches using half the analyzer depth.

Examples

mode_is standard <RETURN>

Related Commands

none

modify

This command returns the define label command for editing.

Syntax

Function

This format specification command allows you to modify the definition of any existing label.

Default Values

none

Parameters

< LABEL>	This prompts you to enter a label name or select the label name from the softkeys.

Examples

modify CLOCK <RETURN>

Related Commands

none

pod

This command initiates the pod commands softkeys.

Syntax None

Function This command invokes the "pod_commands" softkeys from whatever "display" mode you are currently in. This allows you to enter pod_command specific commands without the necessity of using "display" to change specification modes.

Default Values none

Parameters Any of the available pod_commands commands.

Note

Some of the pod_commands softkeys are not activated from this command.

Examples

pod pod_command "ver" <RETURN>

Related Commands
diagram
format
list
post
trace

pod_command

This command sends native commands to the analyzer.

Syntax

Function

This command sends native commands to the analyzer terminal interface. You should be very careful to avoid command which will effect the interface or the configuration. Those commands would "confuse" the protocol, and more likely than not, cause the analyzer to hang.

The Softkey Interface provides the following warning when displaying the pod_commands softkeys:

```
                             --- 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 effect 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":

    sttty, 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 process
    init, pv      - will reset pod and force end release_system
    t             - do not use, will confuse trace status polling and unload
```

Default Values

none

Parameters

< CMD> This prompts you to enter a quoted sting literal to send to the analyzer firmware. The contents of the quoted string are sent and the results are displayed on the screen.

Examples

pod_command "ver" <RETURN>

Related Commands

none

post

This command initiates post_process specification softkeys.

Syntax none

Function This command invokes the "post_process" softkeys from whatever "display" mode you are currently in. This allows you to enter post process specific commands without the necessity of using "display" to change specification modes.

Default Values none

Parameters Any of the available post_process specification commands.

Note

Some of the post_process softkeys are not activated from this command.

Examples

post compare file_is COMP_FILE <RETURN>

Related Commands

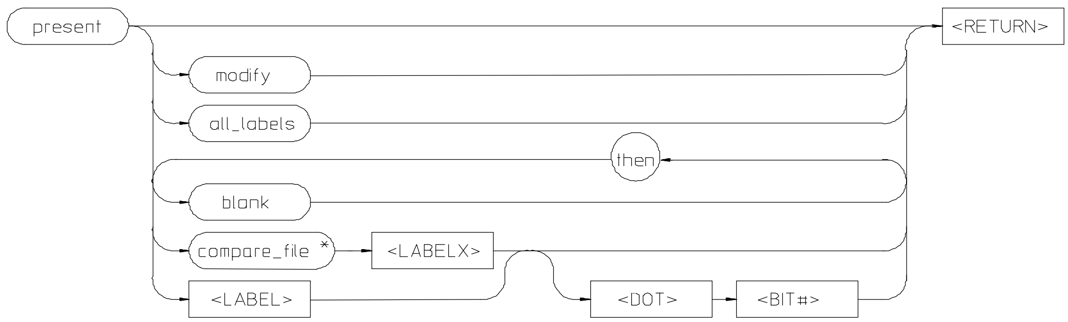
diagram
format
list
pod
trace

present

This command specifies the objects to be presented.

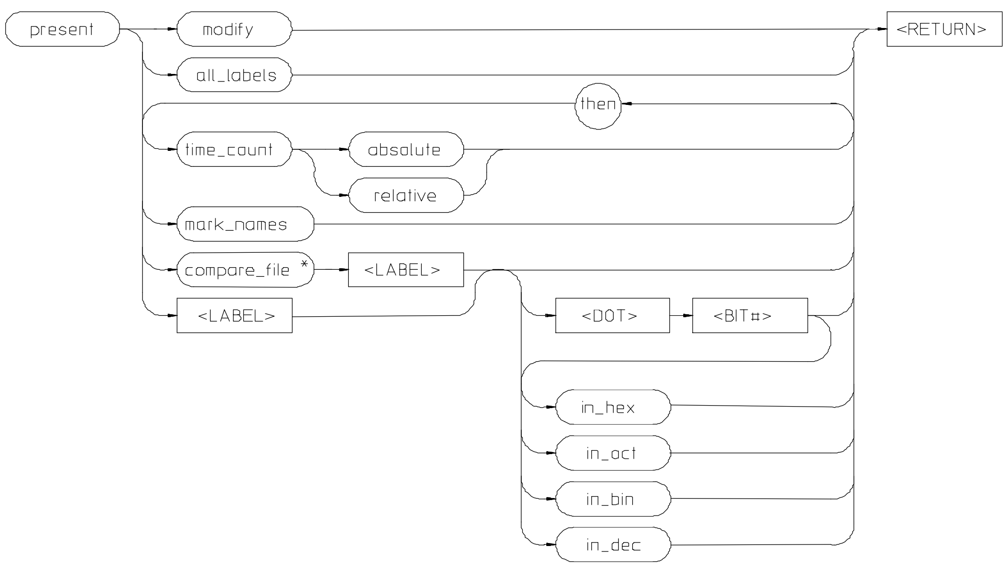
Syntax

The timing diagram syntax is:



* available only when compare_file is selected, otherwise <COMPAR>

The trace list syntax is:



* available only when compare_file is selected, otherwise <COMPAR>

Function

This command controls the display format for the timing diagram and trace list. The timing diagram display can include all defined labels or any combination of blank lines, labels or label bits, or compare file labels or label bits (if the compare file has been selected). The trace list display can include all labels or any combination of time counts, mark names, labels or label bits, or compare file labels or label bits (if the compare file has been selected).

In the timing diagram, entering the display command with no options allows you to toggle the between the default labels referring to probe signals and the user-defined labels.

Labels display from a compare file have a trailing "x" appended to indicate an external reference.

Default Values

All defined labels are displayed by default.

Parameters

all_labels	This option displays all of the labels currently defined in the format specification.
blank	This timing diagram option inserts a blank line in the display.
< BIT# >	This prompts you to enter the integer bit number.
compare_file	This option selects the compare file labels for display. This is available only when a compare file has been defined in the post process specification. The softkey label appears as < COMPAR> if a compare file has not been selected. User-defined labels from the compare file are displayed with a trailing "x" to designate an external reference.
< DISPLY>	This timing diagram softkey label is used to remind the you that the display command without parameters is used to toggle

	between default and user-defined labels on the timing diagram.
< DOT>	This prompts you to enter the literal "." to designate a specific bit number for a label.
in_hex, in_oct, in_bin, in_dec	Specifies the number base when the trace list is being shown. The bases are hexadecimal, octal, binary, and decimal, respectively.
mark_names	This trace list option creates a column for user-specified mark names.
modify	This option returns the current display command to the command line for editing.
then	This option selects another object for display.
time_count	This trace list option creates a column for the time count. Counts are absolute or relative.
absolute	This parameter specifies an absolute count for the display. The time displayed is the cumulative amount of time before or after the trigger.
relative	This parameter specifies a relative count for the display. The time displayed is then the amount of time between samples.

Examples

present TIMER then DATA .0 ***thru*** 3 ***then blank then DATA*** .4 ***thru*** 7 <RETURN>

present DATA then blank then compare_file DATA <RETURN>

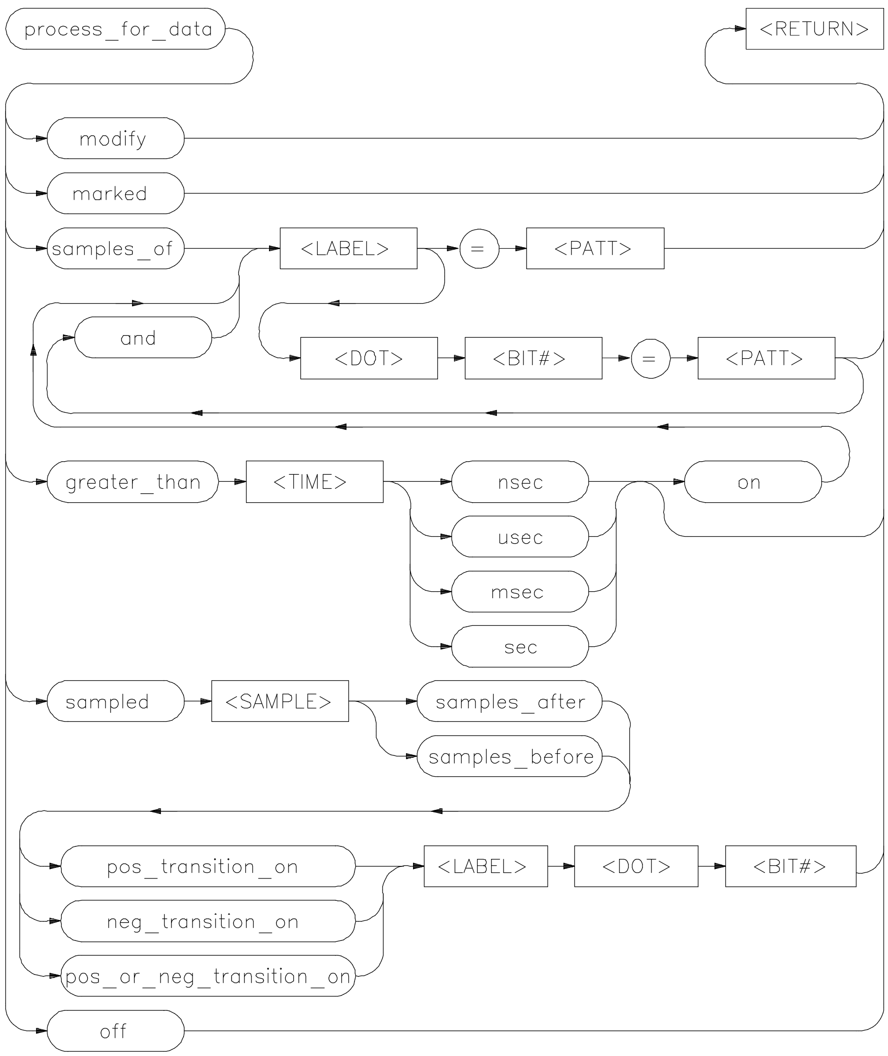
present mark_names then TIMER in_bin then DATA in_hex then time_count relative <RETURN>

Related Commands **define**

process_for_data

This command limits the trace list display to specified samples.

Syntax



Function This trace list command limits the samples displayed to those from trace memory that meet the specified conditions. Conditions can include samples with a specified pattern, samples a fixed count from specified transition, only marked samples, or samples with a specified pattern lasting more than a specific time period.

Default Values The trace list displays all of the samples in trace memory.

Parameters

and	This option allows you to specify an additional label or label bit to be added to ***entering*** or ***leaving***.
< BIT# >	This prompts you to enter the label integer bit number.
default	This option clear any conditions and set it to its default value.
< DOT>	This prompts you to enter the literal "." to designate a specific bit number for a label.
=	This designates the assignment of a specific pattern entered as a numerical value for comparison.
greater_than	This option specifies a pattern will exist for more than an specified duration.
< LABEL>	This prompts you to enter a label name or select the label name from the softkeys.
marked	This option specifies that only marked samples are to be displayed.
modify	This option returns the current process for data command to the command line for editing.

off	This option turns the process for data condition off. When there is no process for data condition, all of the samples in trace memory are displayed in the trace list.
on	This parameter qualifies the transition condition by specifying a label name or label bit number.
< PATT>	This prompts you to enter a pattern of signals. The pattern is a numerical value, the significance of which is dependent on the number of signals being tested. Refer to the section on *Entering Numerical Values* in this manual for options on entering patterns.
samples_of	This option specifies that only samples matching a specified pattern will be displayed.
sampled	This option specifies that only samples a user-selected number of samples before or after a specific condition will be displayed. The condition can only be a transition (positive, negative, or positive or negative) on a specific signal.
< SAMPLE>	This prompts you to enter a number of samples before or after a specified event that will be displayed.
samples_after	This parameter specifies the displayed samples are after the event specified.
samples_before	This parameter specifies the displayed samples are before the event specified.
neg_transition_on	This parameter specifies a negative transition on the target signal.

pos_transition_on	This parameter specifies a positive transition on the target signal.
pos_or_neg_ transition_on	This parameter specifies any transition on the target signal.
< TIME>	This prompts you for the time period to look for a specified pattern.
nsec, usec, msec, sec	These parameters specify the units of measurement of < TIME> in nanoseconds, microseconds, milliseconds, or seconds, respectively.

Examples

process_for_data samples_of X_lower .1 ***=*** 1 <RETURN>

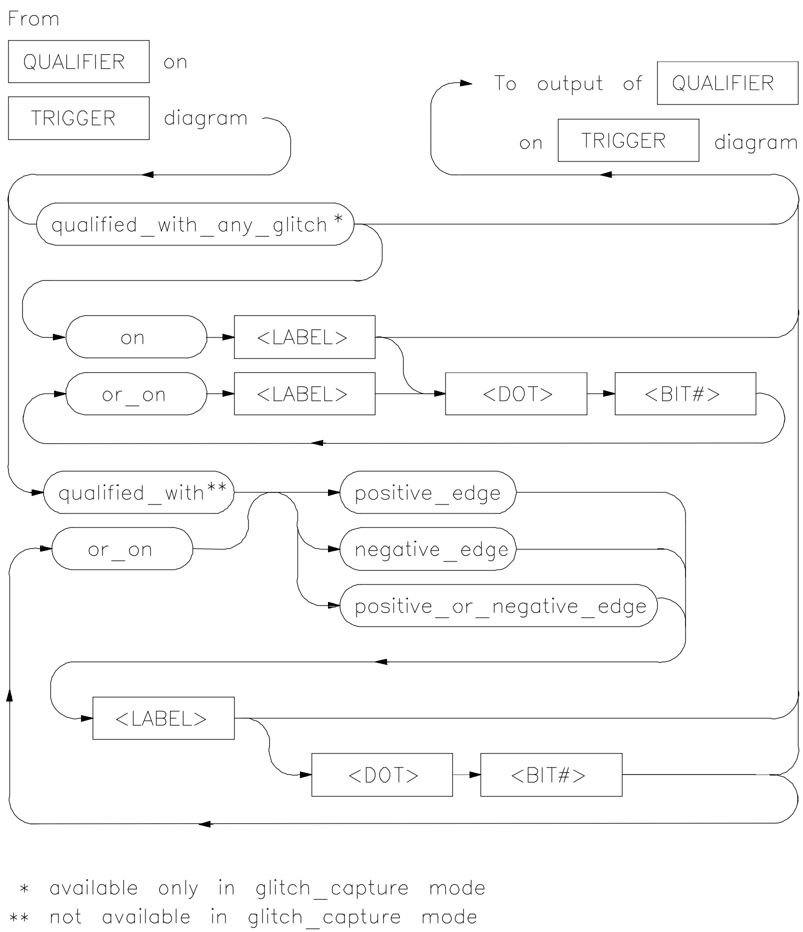
Related Commands

none

QUALIFIER

The **QUALIFIER** parameter is used with the "trigger" command to specify a conditions after which a trigger condition will be tested.

Syntax



Function You can specify a condition or set of conditions that must be met before the trigger condition will be evaluated. In standard mode, qualifiers on "edges", or changes in signal values, while in glitch capture mode, qualifiers are glitches.

Default Values There is no default for qualifying standard mode triggers. Glitch capture mode triggers can be qualified with a glitch on any signal.

Parameters

< BIT# >	This prompts you to enter bit number for the glitch or edge test.
< DOT>	This prompts you to enter the literal "." to designate a specific bit number of a label.
< LABEL>	This prompts you to enter a label name or select the label name from the softkeys.
on	You use this option to qualify the signals on which a glitch may trigger the analyzer.
or_on	This option used to add additional signals to the qualifier.
qualified_with	This option qualifies the trigger condition by allowing you to specify "edge conditions" on labels or label bits.
negative_edge	This parameter qualifies the trigger by looking for a "negative edge" (transition).
positive_or_ negative_edge	This parameter qualifies the trigger by looking for any edge (transition).
positive_edge	This parameter qualifies the trigger by looking for
qualified_with_ any_glitch	This option qualifies the trigger condition in glitch capture mode by specifying the trigger condition can only be met after a glitch has

occurred. If you specify one or more signals, the trigger will be qualified by glitches only on those lines. Otherwise, glitches on any line will cause the qualification to be true.

Examples

trigger on pattern XBITS `.0` ***=*** `0`
qualified_with positive_edge XBITS `.1`
`<RETURN>`

trigger position_is center_of_trace `<RETURN>`

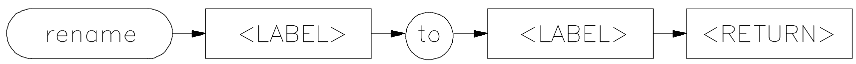
Related Commands

none

rename

This command renames a defined label.

Syntax



Function

This format specification command renames any previously defined to another name.

Default Values

none

Parameters

to	This option initiates the entry of the new name for the label.
< LABEL>	This prompts you to enter a label name or select the label name from the softkeys. The new label name can only be entered.

Examples

rename CLOCK to CLOCK1 <RETURN>

Related Commands

define
delete

< ROLL>

This softkey describes how to move around the timing diagram.

Syntax

none

Function

The < ROLL> softkey reminds you that you can enter a sample number, or use the **< CTRL> -F** and **< CTRL> -G** (or **< NEXT>** and **< PREV>**) keys, to roll the timing diagram. The cursor position, as indicated to the right of the waveforms, changes accordingly.

The timing diagram can also be scrolled up and down if the waveform size or the number of signals displayed exceeds the size of the screen.

Default Values

none

Parameters

< SAMPLE>	By entering a sample number on the command line, you reposition the cursor to that sample number on the display.
< CTRL> -F	Causes the cursor to move to the right on the waveform (to higher samples numbers in trace memory).
< CTRL> -G	Causes the cursor to move to the left on the waveform (to lower sample numbers in trace memory).
< NEXT>	Causes the next page of waveform data to be displayed (higher samples numbers).
< PREV>	Causes the previous page of waveform data to be displayed (lower sample numbers).
^ **< uparrow>**	Scrolls the diagram up by one line if all of the displayed signals do not fit on one screen.

v < **downarrow**>	Scrolls the diagram down by one line if all of the displayed signals do not fit on one screen.
< **Shift**> **-^** < **Shift**> **-< uparrow**>	Scrolls the diagram up by one screen if all of the displayed signals do not fit on one screen.
< **Shift**> **-v** < **Shift**> **-< downarrow**>	Scrolls the diagram down by one screen if all of the displayed signals do not fit on one screen.

Examples

```
436 <RETURN>
```

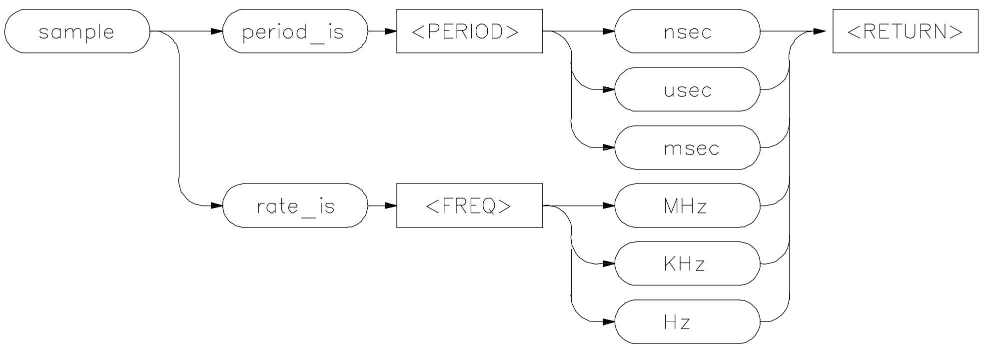
Related Commands

none

sample

This command specifies the sampling period or rate.

Syntax



Function

The sample command sets the sampling period or rate for subsequent measurements. Data is acquired by the analyzer and stored in trace memory at that effective rate.

Default Values

The default is the fastest sample rate in each mode.

Parameters

period_is	This option specifies the sample rate based on the period.
< PERIOD>	This prompts you to enter the length of the sample period. The allowed range is 10 nsec (20 nsec in glitch_capture mode) to 500 msec.
nsec, usec, msec	These parameters specify the units of measurement of < PERIOD> in nanoseconds, microseconds, or milliseconds, respectively.

rate_is	This option specifies the sample rate based on the frequency if sampling.
< FREQ>	This prompts you to enter the sample frequency. The allowed range is 100 megahertz (50 megahertz in glitch_capture mode) to 2 hertz.
MHz, kHz, Hz	These parameters specify the units of measurement of < FREQ> in megahertz, kilohertz, and hertz, respectively.

Examples

sample rate_is 100 ***MHz*** <RETURN>

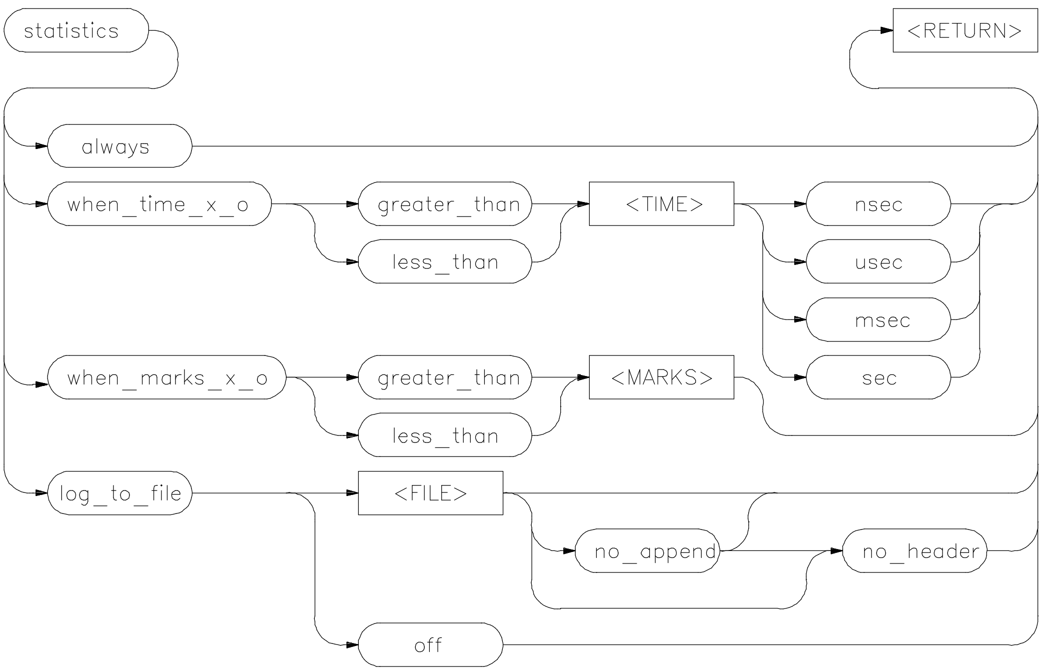
Related Commands

none

statistics

This command selects samples to be included in statistical analysis.

Syntax



Function

This command specifies samples on which statistics are to be calculated. Samples are qualified with mark counts or time intervals from mark_x to mark_o.

This command can also log the statistics after every execution to a specified file. In order for logging to take place, the sampling type must be selected with the "indicate" command.

Default Values

All measurements are included in a statistical sample unless excluded by this command.

Parameters

always	This option resets this option to its default condition.
when_marks_x_o	This option specifies that only traces with a number of marks between the mark_x and mark_o points (inclusive) will be included in the statistical sample.
greater_than	This parameter specifies the mark count must be greater than the number entered.
less_than	This parameter specifies the mark count must be less than the number entered.
< MARKS>	This prompts you to enter the number of marks below or above which the trace will be included in the sample.
when_time_x_o	This option specifies that only traces with a specified time interval between the mark_x and mark_o points will be included in the statistical sample.
greater_than	This parameter specifies the duration must be more than the time entered.
less_than	This parameter specifies the duration must be less than the time entered.
< TIME>	This prompts you to enter the time interval below or above which the trace will be include in the sample.
nsec, usec, msec, sec	These parameters specify the units of measurement of < TIME> in nanoseconds, microseconds, milliseconds, or seconds, respectively.

log_to_file	The option selects a file name to receive statistical information.
< FILE>	This prompts you for a file name to contain the output information.
noappend	This parameter forces the overwrite of a file's information if the selected file name already exists.
noheader	This parameter suppresses the report header from the output. The header contains the source ("64700 Timing Analyzer") and the day, date and time of the output. This parameter is useful when building a listing file from multiple executions of the command.
off	This option turns off the logging of statistics.

Examples

statistics log_to_file testcounts <RETURN>

statistics when_time_x_o greater_than 100 ***nsec*** <RETURN>

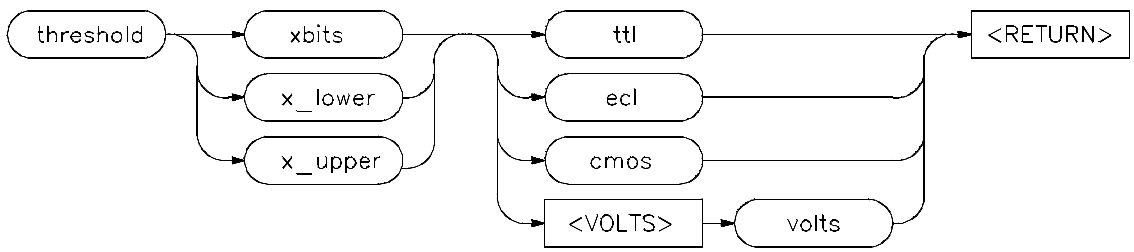
Related Commands

indicate

threshold

This command sets the point at which a signal voltage is considered a logical true.

Syntax



Function

This command sets the voltage level at which a signal is considered true (a logical "1") for each of the two signal groups, or for all signals. Threshold voltages can be in the range of + 6.35V to -6.40V in 50mV increments. The defined levels for TTL, ECL and CMOS can be selected.

Default Values

The threshold defaults are TTL for all signals.

Parameters

xbits	This option is a system default reference to all 16 of the probe signals.
x_lower	This option is a system default reference to the first eight probe signals (bits 0 through 7).
x_upper	This option is a system default reference to the second eight probe signals (bits 8 through 15).

ttl	This parameter sets the threshold voltage to + 1.40V.
ecl	This parameter sets the threshold voltage to -1.30V.
cmos	This parameter sets the threshold voltage to + 2.50V.
< VOLTS>	This prompts you to enter the voltage if it is non-standard.
volts	This is a syntactic element used when entering the voltage.

Examples

threshold x_lower ecl <RETURN>

Related Commands

define

trace

This command initiates the trace specification softkeys.

Syntax none

Function This command invokes the "trace" softkeys from whatever "display" mode you are currently in. This allows you to enter trace specification specific commands without the necessity of using "display" to change specification modes.

Default Values none

Parameters Any of the available trace specification commands.

Note

Some of the trace specification softkeys are not activated from this command.

Examples

trace mode_is glitch_capture <RETURN>

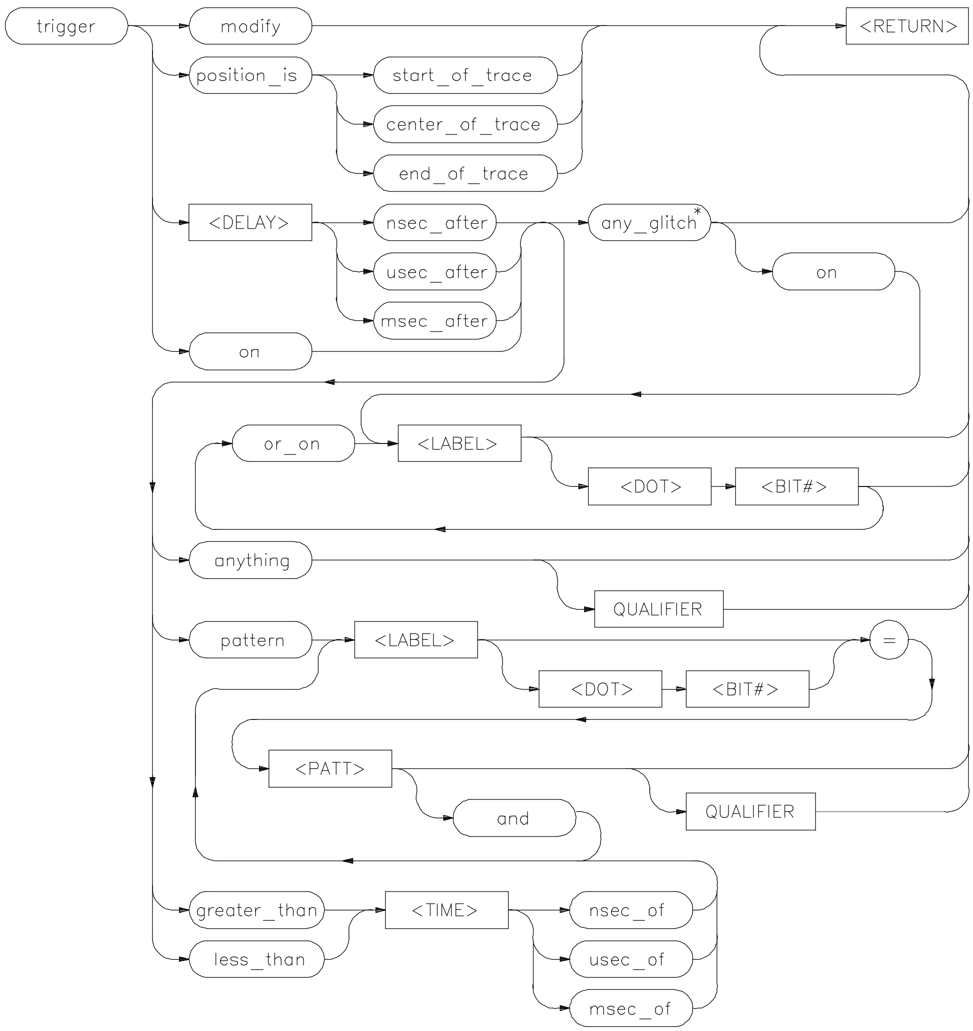
Related Commands

diagram
format
list
pod
post

trigger

This command specifies trigger conditions.

Syntax



* available only in the glitch_capture mode

Function The trigger is an event on the probe which causes the analyzer to begin acquiring data and filling trace memory.

The trigger command can also position the trigger at the start, center, or end of trace memory.

Default Values The default is a "don't care" trigger positioned at the start of the trace.

Parameters

and	This option allows you to specify an additional label or label bit to be added to the conditions.
anything	This option specifies the trigger to be any signal change. The change can be qualified with an edge condition.
any_glitch	This option specifies the trigger to be a glitch. The glitch can be on any signal, or qualified by label or label bit. This option is available only in glitch capture mode.
on	You use this option to qualify the signals on which a glitch may trigger the analyzer.
or_on	This option used to add additional signals to the glitch condition qualifier.
< BIT# >	This prompts you to enter bit number for the pattern or edge.
< DELAY>	This prompts you to enter a delay. The analyzer waits the duration of the delay after the trigger before beginning to sample the signals.
nsec_after, usec_after, msec_after	These parameters specify the units of measurement of < DELAY> in

nanoseconds, microseconds, or milliseconds, respectively.

< DOT> — This prompts you to enter the literal "." to designate a specific bit number of a label.

greater_than — This option specifies a duration a pattern should last for the specified signals. If the pattern is detected but does not last at least the specified amount of time, the trigger condition is not met.

less_than — This option specifies a duration a pattern should not exceed for the specified signals. If the pattern is detected an lasted longer than the specified amount of time, the trigger condition is not met.

modify — This option returns the current trigger command to the command line for editing.

pattern — This option specifies the trigger condition will be a signal pattern.

on — This option specifies the trigger point of a measurement or set of measurements.

< PATT> — This prompts you to enter a pattern of signals. The pattern is a numerical value, the significance of which is dependent on the number of signals being tested. Refer to the section on *Entering Numerical Values* in this manual for options on entering patterns.

position_is — This option specifies the positioning of the trigger in trace memory. If the trigger is positioned at the start of trace memory, the majority of samples will be after the trigger; if it is positioned at the end of trace memory, the majority of samples will be before the trigger; and if it is positioned at the center, the samples will be evenly distributed on either side of the trigger.

start_of_trace	This parameter positions the trigger near the start of trace memory.
center_of_trace	This parameter positions the trigger near the middle of trace memory.
end_of_trace	This parameter positions the trigger near the end of trace memory.
QUALIFIER	This qualifies the trigger condition by allowing you to specify "edge conditions" on labels or label bits which must occur before the trigger condition will be evaluated. In glitch capture mode, you specify signals on which glitches will occur before the trigger condition will be evaluated.
with	Specifies that a glitch must occur on the previously named channels while the following condition is true.
< TIME>	This prompts you to enter the time a signal pattern should be evaluated.
nsec_of, usec_of, msec_of	These parameters specify the units of measurement of < TIME> in nanoseconds, microseconds, or milliseconds, respectively.

Examples

trigger on pattern XBITS `.0` ***=*** `0` ***qualified_with positive_edge XBITS*** `.1` `<RETURN>`

trigger position_is center_of_trace `<RETURN>`

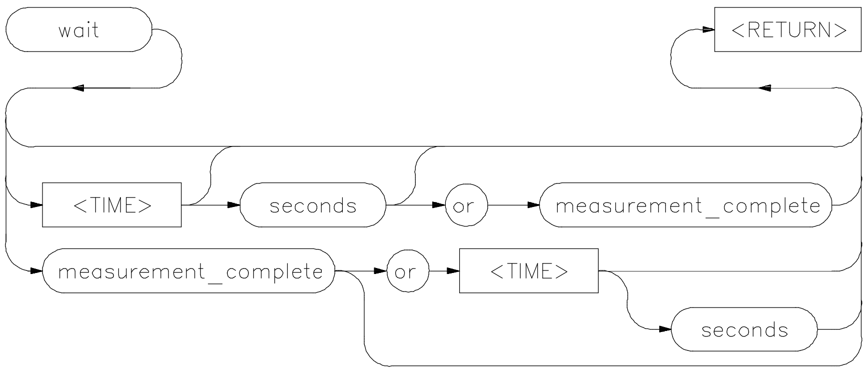
Related Commands

none

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 analyzer level. The usefulness of delays is to allow the analyzer system and external analyzer time to reach a certain condition or state before executing the next command.

The wait command does not appear on the softkey labels. You must type the wait command on the command line. After you type "wait", the command parameters will be accessible on the softkeys.

Default Values

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

Note

If **set intr < CTRL> -C** has not been executed on your system, **< CTRL> -C** may be defined as the backspace key.

Parameters

measurement_ complete	This option causes the system to pause until a pending measurement completes, or until a **< CTRL> -C** signal is received. If a measurement is not in progress, the wait command will be completed 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.
< TIME>	This prompts you to enter the number of seconds to insert for the delay.
seconds	This sets the unit of measure for the delay to seconds. Seconds is the only option.

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** was previously defined as the interrupt signal. Subsequent commands in the command file will not execute while the command file is paused.

You can verify whether or not the interrupt signal is defined as **< CTRL> -C** by typing "set" at the system prompt.

Examples

wait <RETURN>

wait 10 <RETURN>

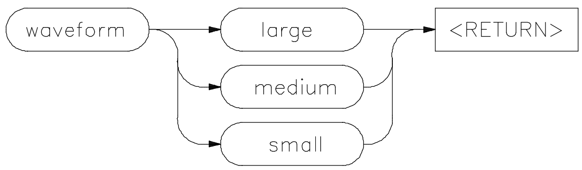
Related Commands

none

waveform_size

This command adjust the waveform size on the display.

Syntax



Function

This timing diagram command allows you to adjust the size of the displayed waveforms. A small waveform size allows you to compare waveforms of a greater number of signals. A large waveform size allows you to view the signal in more detail.

If you are displaying more channels than can be shown on the screen, you can use the < **Shift**> -< **up arrow**> and < **Shift**> -< **down arrow**> keys to page down or page up through the channels. The up arrow and down arrow keys shift the display down or up a waveform at a time.

Default Values

Small waveforms are displayed by default.

Parameters

large	This option displays the waveforms with a maximum vertical height.
medium	This option displays the waveforms with a moderate vertical height.
small	This option displays the waveforms with a minimum vertical height.

Examples

display timing_diagram <RETURN>
waveform_size medium <RETURN>

Related Commands **present**

Notes

A

External Analyzer Specifications

General Specifications

- Threshold Accuracy = + /- 50 mV.
- Threshold Voltage Range = -6.40V to 6.35V.
- Dynamic Range = + /- 10 V about threshold setting.
- Minimum Input Swing = 600 mV pp.
- Minimum Input Overdrive = 250 mV or 30% of threshold setting, whichever is greater.
- Absolute Maximum Input Voltage = + /- 40 V.
- Probe Input Resistance = 100K ohms + /- 2%.
- Probe Input Capacitance = approximately 8 pF.
- Maximum + 5 V Probe Current = 0.650 A.
- + 5 V Probe Voltage Accuracy = + 5.0 + /- 5%.

State Analyzer

- Data Setup Time = 10 ns min.
- Data Hold Time = 0 ns min.
- Qualifier Setup Time = 20 ns min.
- Qualifier Hold Time = 5 ns min.

- Minimum Clock Width = 10 ns.
- Minimum Clock Period:
  - No Tagging Mode = 40 ns (25 Mhz clock).
  - Event Tagging Mode = 50 ns (20 MHz clock).
  - Time Tagging Mode = 60 ns (16 MHz clock).
- Minimum Time from Slave Clock to Master Clock = 10 ns.
- Minimum Time from Master Clock to Slave Clock = 50 ns.

Timing Analyzer

- Sample Rate Accuracy = 0.01%.
- Asychronous Pattern - Trigger on pattern is less than or than specified duration. Pattern is logical AND of specified low, high or don't care for each channel. If the pattern is true then false for less than the duration there is a 20 ns reset time before looking for the pattern again.
- Greater Than Duration = Range 30 ns to 10 ms. Resolution is + /-10 ns or 0.01% whichever is greater. Accuracy is + /-10 ns + 0.01% + 20/-0 ns.
- Less Than Duration = Range 40 ns to 10 ms. Resolution is + /-10 ns or 0.01% whichever is greater. Pattern must be valid for at least 20 ns. Accuracy is + /-10 ns + 0.01% + 20/-0 ns.
- Delay Accuracy = 0.01% + /-10 ns.
- Minimum Detectable Glitch = 5 ns at threshold.
- Skew = 4 ns (typical).

B

Timing Output and Diagrams

Overview

The Timing Analyzer Softkey Interface provides you a mechanism to generate output for a variety of environments and hardware. Measurements can be displayed in diagrammatic or list form. Diagrams are created using ASCII characters, the default, for ASCII terminals and non-graphics printers, or in graphics format for terminals and printers with graphics support.

Timing Diagram Outputs

The timing analyzer will produce a graphics diagram output if you are using the graphics diagram, and an ASCII diagram output if you are using an ASCII character diagram. The ASCII diagram output is suitable for including in ASCII files and printing on standard ASCII printers.

The graphics timing diagram output, on the other hand, is raw raster output and can not be sent to the printer like standard ASCII data. In fact, it is best not to mix ASCII outputs and graphics even in the same files because a print out of the file will yield only ASCII or only graphics and will NOT yield desirable results. The graphics data must be sent to the printer character device and the ASCII must be sent to the printer block device.

Graphics Timing Diagrams

The graphic timing diagram is available when using a high or medium resolution bit-mapped monitor on the host SPU which controls a timing analyzer. Bit-mapped displays are listed at boot up as ITE (Internal Terminal Emulator).

Graphic timing diagrams are available when using the X Window System on a bit-mapped display, but are not available when running HP Windows/9000 on a bit-mapped display, or on an ASCII terminal.

Graphic timing diagrams are available on a remote SPU only with the X Window System. For information on this option, refer to "Using the Timing Analyzer Under the X Window System" later.

TERM Shell Variable

To access the timing analyzer graphic diagrams make sure that your TERM shell variable is set properly. It should be one of the following values: hp300h, hp300l, hp98548, hp98549, hp98550, hp98700, or hp98720w.

The value for TERM must match the monitor video board or graphics display system installed in your host; refer to the list below.

hp300h	HP 98544A High-resolution Monochromatic Video Board.
hp300h	HP 98545A / HP 98547A High-resolution Color Video Boards.
hp300l	HP 98542A Medium-resolution Monochromatic Video Board.
hp300l	HP 98543A Medium-resolution Color Video Board.
hp98548	HP 98548A Super High-resolution Video Board.
hp98549	HP 98549A High-resolution Color Video Board.

hp98550	HP 98550A Super High-resolution Color Video Board.
hp98700	HP 98700H Graphics Display Station.
hp98720w	HP 98720H / HP 98721 Graphics Display Stations.

WMSCRN Shell Variable

Now when you enter the timing analyzer and press "execute" a graphics timing diagram should appear. If graphics do not appear then perhaps the timing analyzer software can not find your monitor. The timing analyzer assumes that the monitor is "/dev/crt" but if your monitor is something different you can inform the timing analyzer software of that fact by setting the shell variable WMSCRN to your monitor with a command like:

```
WMSCRN=/dev/crt1 <RETURN>
export WMSCRN <RETURN>
```

You may want to set the WMSCRN shell variable in your ".profile" file so that you do not have to redefine WMSCRN upon each login.

WMBASEFONT Shell Variable

In the timing analyzer we can also accept different font sizes which yield different size graphic diagrams. The font size can be set to any of the available fonts with the exception that a resultant 24 line x80 column display must be available. If this display is not possible with the font which you select then the timing analyzer will have a display initialization failure.

To change the font size (in timing analyzer only), set the shell variable WMBASEFONT with a command like:

```
WMBASEFONT=/usr/lib/raster/12x20/cour.b.0U
<RETURN>
 export WMBASEFONT <RETURN>
```

The value 12x20/cour.b.0U specifies that you want to use a pixel cell 12 dots wide by 20 high in a courier bold font. By examining the files in the sub-directories of "/usr/lib/raster" you can determine the font sizes available for your host and pick an appropriate one. A larger font cell size will yield a larger diagram and a smaller font cell size will yield a smaller diagram.

You may also want to set the WMBASEFONT shell variable in your ".profile" file so that you do not have to redefine WMBASEFONT upon each login.

Required Filesets

The fileset AFA_FM, which contains raster fonts for graphics and text, is required for the timing analyzer to produce graphic displays. See the *HP-UX System Administrator Manual, Part 2* for information on loading the required filesets.

Using the Timing Analyzer Under the X Window System

The graphic timing diagram is available under the X Window System Version 11. X Windows refers to applications running on a bit-mapped display with the X Window System managing the display. To access the timing analyzer graphic diagram within the X Window System, simply start up the timing analyzer in a general purpose terminal emulator window, such as "hpterm(1)".

As long as the appropriate shell variables are defined, the timing analyzer will detect that its running under X Windows and will create a special subwindow within the terminal emulator window for displaying the graphic timing diagram. The terminal emulator window is still used for displaying ASCII text. If the appropriate shell variables are not defined, or the timing analyzer is unable to create the graphics subwindow, an ASCII timing diagram will be displayed.

DISPLAY Shell Variable

The shell variable DISPLAY is used by X Window applications to specify the host, display number and screen number to receive bit-mapped output. It is automatically defined by the X Window System during startup if does not already exist. This shell variable must be defined or the timing analyzer will not attempt to initialize the graphic timing diagram for X Windows. Enter a command like:

```
DISPLAY="mynode:0.0" <RETURN>
export DISPLAY <RETURN>
```

to specify the DISPLAY variable, where "mynode" is the local host name.

For information on using the Timing Analyzer connected to a remote host, refer to "Remote Connections" later in this section.

WINDOWID Shell Variable

The timing analyzer looks for the shell variable WINDOWID to determine the windowid of the terminal emulator window which the timing analyzer is running in. This information is required by the timing analyzer to create the graphics subwindow for displaying the graphic timing diagram. This shell variable is automatically defined by the "hpterm" general purpose terminal emulator.

LINES and COLUMNS Shell Variables

The timing analyzer looks for the shell variables LINES and COLUMNS to determine the size and location to place the graphics subwindow within the terminal emulator window. These variables are defined automatically when the "hpterm" terminal emulator window is created.

Note

If the terminal emulator window is resized and/or the shell variables LINES and COLUMNS do not reflect the correct size of the terminal emulator window, the timing analyzer will position it's graphics subwindow incorrectly. If this occurs, you should exit the timing analyzer and run the command

```
eval '/usr/bin/X11/resize' <RETURN>
```

to reset these shell variables to the correct values.

X Defaults

The timing analyzer recognizes the following X defaults, which when placed in your $HOME/.Xdefaults file, will change the appearance of the timing analyzer's graphics subwindow.

foreground	Specifies the foreground color (for drawing the waveforms) of the timing analyzer's graphics subwindow. The default is black.
background	Specifies the background color of the timing analyzer's graphics subwindow. The default is white.

reverseVideo	Specifies that the foreground and background colors should be reversed.
timingGeometry	Specifies the location and size to create the timing analyzer's graphics subwindow. This X default should not be used unless the timing analyzer is positioning and/or sizing it's graphics subwindow incorrectly. This can occur if the terminal emulator window is created with an unusually large internal border or a left-justified scroll bar is being used.

The geometry is specified as "= WxH+ X+ Y" indicating that the window should have a width "W" and height "H" in pixels and the upper left corner "X" pixels to the right and "Y" pixels below the upper left corner of the terminal emulator window. "W" should be set to 57 times the width of the terminal emulator's font cell and "H" should be equal to 7 less than the total number of rows in the terminal emulator window times the height of the font cell. "X" should be set to 13 times the width of the font cell plus the width of any internal border and left-justified scroll bar. "Y" should be equal to the height of the font cell plus the width of any internal border. Either "WxH" or "+ X+ Y"can be omitted to obtain the default window size or window position as calculated by the timing analyzer.

The timing analyzer attempts to match the foreground and background colors used in the terminal emulator window by recognizing the same X defaults used by "hpterm". The timing analyzer will use these X defaults when they are not qualified with the client name "hpterm" or when they are qualified with the timing analyzer's client name "timing". For example, if your .Xdefaults file contains the following lines:

```
*foreground:   green
*background:   blue
```

both the terminal emulator window and the timing analyzer's graphics subwindow will use a blue background with green text (waveforms). However, if you have specified these X defaults for "hpterm" only:

```
hpterm*foreground:   green
hpterm*background:   blue
```

the timing analyzer will use the default black on white because these X defaults were specified for the "hpterm" terminal emulator only and not the timing analyzer. By adding two more X defaults specifically for the timing analyzer, the graphics subwindow will once again match the colors used in terminal emulator window:

```
hpterm*foreground:   green
hpterm*background:   blue

timing*foreground:   green
timing*background:   blue
```

Remote Connections

It is possible in the X Window environment to run the timing analyzer from a remote host. To gain access to a remote host, the following criteria must first be observed:

- You must have the internet address and hostname of the remote host in your system's "/etc/hosts" file.
- You must have a valid login on the remote host.
- You must have the remote host listed in the "/etc/X0.hosts" file.
- You must have the remote host listed in a ".rhost" file in your home directory on your local system, and your local system listed in a ".rhost" file on the remote host.

You should refer to the "Customizing Your Local X Environment" in *Using the X Window System: HP 9000 Series 300/800 Computers* for more information on using remote hosts.

Once this environment has been configured, an "hpterm" window can be opened for the timing analyzer on the remote host with the command:

```
remsh timingnode -n "DISPLAY=mynode:0.0
/usr/bin/X11/hpterm" & <RETURN>
```

where:

timingnode is the remote host name, and

mynode is the local host name.

The remote host will open an "hpterm" window with the shell variable, DISPLAY indicating to send the bit-mapped output back to your local machine. You should now be able to run the timing analyzer in this "hpterm" terminal emulator window.

ASCII Timing Diagrams

The ASCII timing diagram is a user definable diagram. All of the characters which form the diagram characters can be user defined by the shell variable TIMING_ASCII. In addition the locations of the cursor and the marks can be defined by this shell variable.

Default ASCII Diagram

If the TIMING_ASCII shell variable is not found a default diagram is put up which is equivalent to using this value for the shell variable.

```
TIMING_ASCII="  _ ,  * ,  : , _, *, :,_,*,::1,4,2,4,2,4:0,4,0,4,0,4:v,|"
```

The TIMING_ASCII shell variable expects four types of information:

- ASCII characters to represent low, high, and glitch conditions.
- Starting and interval rows for the marks to be located.
- Starting and interval rows for the cursors to be located.
- Initial and subsequent characters for the cursor.

These four types of input are loosely delimited by the colon character ":".

Note

The first three colon ":" characters in the example above are ASCII characters used to represent the glitch capture mode in large, medium and small waveforms, respectively.

Customizing the ASCII Diagram

Waveform Sizes and ASCII Characters

The first part of the shell variable determines the ASCII diagram characters which are to define the three waveforms sizes.

```
              <----large---><medium-><small>
              low ,high,glch,  l,  h,  g,l,h,g,
TIMING_ASCII="abcd,abcd,abcd,ef,ef,ef,g,g,g:..."
```

Where "abcd" represent the four characters to be displayed for a low, high, and glitch level in the large waveform. As examples, the default diagram is:

For a low "abcd" = "   _ ".

For a high "abcd" = "   * ".

For a glitch "abcd" = "   : ".

Where "ef" represent the two characters to be displayed for a low, high, and glitch level in the medium waveform. As examples, the default diagram is:

For a low "ef" = " _".

For a high "ef" = " *".

For a glitch "ef" = " :".

Where "g" represents the one character to be displayed for a low, high, glitch, and middle level in the small waveform. As examples, the default diagram is:

For a low	"g" = "_".
For a high	"g" = "*".
For a glitch	"g" = ":".

Row Locations for Mark Indicators

The second part of the shell variable determines the row locations for the mark indicators.

```
TIMING_ASCII="...:m_st_l,m_int_l,m_st_s,m_int_s:..."
```

where:

m_st_l	is the mark starting row for large waveform
m_int_l	is the mark interval row for large waveform
m_st_s	is the mark starting row for small waveform
m_int_s	is the mark interval row for small waveform.

Acceptable values for all variables are integers between "0" and "15".

A "m_st_l" value of "1" implies that in the large waveform marks are to be indicated starting with the second row of data ("0" = first row).

A "m_int_l" value of "4" implies that in the large waveform marks are to located on every four rows after the starting row.

Row Locations for Cursor Indicators

The third part of the shell variable determines the row locations for the cursor indicators.

```
TIMING_ASCII="...:c_st_l,c_int_l,c_st_s,c_int_s:..."
```

where:

c_st_l is the cursor starting row for large waveform

c_int_l is the cursor interval row for large waveform

c_st_s is the cursor starting row for small waveform

c_int_s is the cursor interval row for small waveform

Acceptable values for all variables are integers between "0" and "15".

A "c_st_l" value of "0" implies that in the large waveform the cursor is to be indicated starting with the first row of data.

A "c_int_l" value of "4" implies that in the large waveform the cursor is to located on every four rows after the starting row.

Characters to Define the Cursor

The fourth part of the shell variable determines the start character and subsequent character which is to define the cursor.

```
TIMING_ASCII="...:cursor_start_character,cursor_subsequent_character"
```

A cursor_start_character of "v" implies that "v" is to be the first character displayed in the cursor.

A cursor_subsequent_character of "| " implies that "| " is to be the character displayed in all of the other cursor locations.

Assigning the TIMING_ASCII Shell Variable

To set the TIMING_ASCII shell variable from the shell, enter:

```
TIMING_ASCII="... " <RETURN>
export TIMING_ASCII <RETURN>
```

To view what was entered as the shell variable, enter:

```
set <RETURN>
```

You may also want to set the TIMING_ASCII shell variable in your ".profile" file so that you do not have to redefine TIMING_ASCII after each login.

The TIMING_ASCII shell variable is read until an error in format is found. Therefore if an error is indicated you should be able to determine approximately where the error occurred by looking at the diagram and noting which of the displays are as you expected them to be.

Printer Requirements

The graphic timing diagram outputs can be saved in files or sent to a laser (HP LaserJet) printer, or an equivalent, as well as a dot-matrix printer which supports raw raster dumps.

The timing analyzer needs two printer shell variables setup to function properly PRINTER and GPRINTER. The PRINTER shell variable should be set to "lp -s" with the commands

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

or an equivalent command. The PRINTER variable determines where all the ASCII displays of the timing analyzer are "piped" when a "copy < specification> to printer" command is entered.

The GPRINTER variable should be setup to "lp -or -s" with the command

```
GPRINTER="lp -or -s" <RETURN>
export GPRINTER <RETURN>
```

or an equivalent command. The GPRINTER variable determines where the graphics display of the timing analyzer is "piped" when a "copy timing_diagram to printer" command is entered.

Finally graphics dumps tend to create rather large files. Therefore, if you keep your graphics outputs in files you may want to limit the number that you keep.

Your system administrator may need to adjust the model for your printer to include the raw option (r, raw). A complete example of an appropriate printer model can be found in the manual.

Using Measurement Data in Hexadecimal

You can store raw measurement data in a ASCII file in hexidecimal format by using the "copy measurement_data_in_hex to < FILE> " command. This file can be used for further analysis or comparison.

Understanding the Measurement Data Output

The measurement data output is a list of all the data samples in trace memory stored from first to last. In all cases, the first data sample in the file corresponds to the first sample in trace memory regardless of the starting sample number. Each data sample is in the form

BBAA

where:

BB	represents data from the upper eight signals (x_upper)
AA	represents data from the lower eight signals (x_lower).

Standard Mode Data Format

In the standard mode, each of the data sample bits represents a signal bit value.

Bit0 = xbit0 data
Bit1 = xbit1 data
Bit2 = xbit2 data
Bit3 = xbit3 data
Bit4 = xbit4 data
Bit5 = xbit5 data
Bit6 = xbit6 data
Bit7 = xbit7 data

A "1" in a bit indicates the data sample is high.
A "0" in a bit indicates the data sample is low.

Glitch Capture Mode Data Format

In the glitch capture mode, the data sample represent the values:

Bit0 = xbit0 data
Bit1 = xbit1 data
Bit2 = xbit2 data
Bit3 = xbit3 data
Bit4 = xbit0 glitch
Bit5 = xbit1 glitch
Bit6 = xbit2 glitch
Bit7 = xbit3 glitch

A "1" in a data bit indicates the data sample is high.
A "0" in a data bit indicates the data sample is low.
A "1" in a glitch bit indicates that a glitch occurred.
A "0" in a glitch bit indicates that a glitch did not occur.

Comparing Measurement Data to a Trace List

To illustrate the relationship between the raw measurement data file and the trace memory samples, review at the two listings below. The first listing a the first portion of a measurement data file.

```
64700 Timing Analyzer                  Fri Jun  9 11:00:59 1989

7EB4
7EB4
7EB4
7EB4
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
7EB5
```

The second is a portion of a trace list corresponding to the same set of data.

```
64700 Timing Analyzer                 Fri Jun  9 11:05:25 1989

         Trace List                  Timing (64700), 16 channels, 100MHz
        STANDARD MODE             10 nsec/sample        Time x_o  10.09 usec

  Label:   X_upper  X_lower   time count
  Base:      hex      hex         abs
 -0013_x __  7E       B4     -130.0 nsec  __
 -0012       7E       B4     -120.0 nsec
 -0011       7E       B4     -110.0 nsec
 -0010       7E       B4     -100.0 nsec
 -0009       7E       B5     -90.0 nsec
 -0008       7E       B5     -80.0 nsec
 -0007       7E       B5     -70.0 nsec
 -0006       7E       B5     -60.0 nsec
 -0005       7E       B5     -50.0 nsec
 -0004       7E       B5     -40.0 nsec
 -0003       7E       B5     -30.0 nsec
 -0002       7E       B5     -20.0 nsec
 -0001       7E       B5     -10.0 nsec
 trigger     7E       B5      0.0 nsec
 +0001       7E       B5      10.0 nsec
 +0002       7E       B5      20.0 nsec
 +0003       7E       B5      30.0 nsec
```

Notes

C

Timing Messages

Overview

Three types of messages appear in the analyzer message line: status messages, informational messages, and error messages. The message text along with an explanation and possible responses are detailed later.

A measurement status message also appears on the right hand side of the status line. This message will be "External trace complete". It will be one of the "trace" status messages if in an execution phase.

Status Messages

STATUS: xxxxxxx--Running in monitor

This message indicates the type of emulator/analyzer monitor used. The "xxxxxxx" may be M68000, Z80, or other any other emulator type that is running with an analyzer installed.

STATUS: Connecting to xxxxxx

This start up message indicates the timing analysis software is beginning the process of communicating with the external analyzer. The "xxxxxxx" is the emulator name entered at the command line.

STATUS: Initializing user interface

This start up message indicates the timing analysis software is beginning the diagram initialization process. If a graphics interface is not found, the message "ERROR: Timing graphics initialization failed" is displayed, and an ASCII diagram format is used.

STATUS: External analysis not configured for timing

This message appears when the configuration file loaded into the emulator/analyzer has not been configured for external timing analysis. The following message also appears:

```
The configuration file loaded has not configured the external
analyzer for Timing measurements. This can be accomplished by
entering the emulation interface and modifying the configuration.

 The configuration questions that need to be answered are as follows:

    Modify external analyzer configuration: yes
    Should emulation control the external bits: yes
    External analyzer mode? timing

NOTE: To make this the default powerup configuration, save the
configuration file as userconfig.EA in the associated product directory.

/usr/hp64000/inst/emul/<product_number>/userconfig.
```

The emulation software is used to modify and load the new configuration, as indicated by the message. See "Configuring the External Analyzer" in "Using the External Analyzer" in this manual and the "Emulation Configuration" chapter in the *HP64700-Series Emulators Softkey Interface Reference* for more information.

STATUS: Indicated max_min/mean_stdv forced halt at 9999 runs

This message indicates that a repetitive execution has been halted to avoid an overflow on the counter. Only 9999 runs can be executed.

STATUS: Mark found on sample number < sample>

This message is displayed when the find command locates on of the four marks (abcd).

STATUS: HP64700 I/O error; communications timed out

This message indicates that the communications connection has been lost or interrupted. No further communications will be made between the Timing Analyzer software and the emulator. The usual response is to end the session, which must release the system, and restart the emulator. Analysis of current trace data may be possible, however. Check for loose cables before restarting the Timing Analyzer.

STATUS: Marking complete, found xxx marks (abcd)

This message indicates marking has been completed in the post processing. The xxx represents the total number of marks found for all mark names (mark_a, mark_b, mark_c, and mark_d).

STATUS: Marking complete, marking limit of 511 exceeded

This message indicates marking has been completed in the post processing and more than 511 events meet the conditions for marking (abcd).

STATUS: Sample period is now < period> nsec/usec/msec

This message is displayed after a new sample period is entered. Note that the entered values are rounded to values appropriate for the analyzer.

STATUS: Sample rate is now < freqency> MHz/kHz/Hz

This message is displayed after a new sample rate is entered. Note that the enterd values are rounded to values appropriate for the analyzer.

Informational Messages

Enter: 'Return' to toggle diagram labels between defined and default

This message appears when entering the softkey labelled < ***DISPLY***> for the display option in the timing_diagaram. It is used as a reminder on the toggle function.

< Execution messages>

The following messages indicate the status of an ***execute*** cycle:

"Waiting for trigger" Trigger event is not being found or not enabled (trigger enable false, master enable false).

"External trace running" Trigger event has been found and trace is being made.

"External trace complete" Trigger event was found and trace is complete.

"External trace halted" The last trace was halted.

ENTER: 'Return' to place the mark at the current cursor location

This message appears when entering the < MARK> softkey under the "mark x" or "mark o" commands. The specified mark will be moved to the cursor location if a < RETURN> is entered. The result is the same as entering "mark x on_cursor" or "mark o on_cursor", respectively.

ENTER: Sample number, ctrl-f/g or Next/Prev to roll the diagram

This message appears when using the < ROLL> softkey in the timing diagram. The graphic can be rolled left and right, up and down, in order to display all of the information on the diagram. Refer to < ROLL> in the "Timing: Commands" chapter for more details.

Error Messages

ERROR: 'and' is not possible, all bit are already specified

The message appears when you try to enter a trigger specification which uses bits that have already been spcified. You need to reenter the trigger specification.

ERROR: Cannot save configuration without data into compare file

This message is displayed when you try to save a configuration without data into the currently assigned compare file. Use the "with_data" parameter, or turn off the compare file.

ERROR: Cannot start; Emulator not initialized

This message indicates that the emulator/analyzer has not been initialized. Start the emulator Softkey Interface and load a timing configuration file set up for timing. Then "end" the emulator session and restart the Timing Analysis Softkey Interface.

ERROR: Compare definition is invalid

This message is displayed then you redefine labels in such a way that the compare definition is no longer valid. The compare definition will need to be recreated.

ERROR: Compare file is invalid, compare file is removed

This message indicates that a compare file has been modified and is no longer recognized as a compare file. You will need to reassign a compare file.

ERROR: Compare file is invalid, data does not exist in file

You have tried to use a compare file which has not been saved "with_data". Resave the configuration with data before assigning a compare file or use a valid file.

ERROR: Compare file is invalid, data is halted

This message is displayed when you try to define a halted measurement as a compare file. You need to use a file with the correct format.

ERROR: Compare file spec does not agree with hardware

This message indicates that the current measurement was acquired with a configuration that does not match that used to acquire the compare file data; the sample rate, trigger position, or mode used does not match. You need to correct the conditions before a compare will succeed.

ERROR: Compare label < LABEL> specified bit does not exist

This message indicates your compare specification includes a label-bit that was not defined in the format specification of the compare coinfiguration file. You need to correct the compare specification to match the configuration file.

ERROR: Compare label < LABEL> with does not match data label

This message appears when the compare label you enter does not have the same number of bits as the data label you have entered. You need to correct the reference.

ERROR: Compare not possible on data which is halted

This message appears after you halt a measurement and a compare file is assigned. Only a completed measurement can be compared to the compare file data.

ERROR: Configuration file version is not compatible with software

This message indicates that you are trying to use a timing analysis configuration file with a later data (future version). You need to check the dates on the system.

ERROR: Configuration load failure

This message appears when the configuration file could not be loaded due to an interrupt or other HP-UX system problem. You identify the problem and try to reload the configuration file.

ERROR: Data is not present in hardware

This message appears when you try to save a configuration file "with_data" but there is no data in the analyzer. You should execute a measurement before saving the configuration file.

ERROR: Data label < LABEL> bits are already specified

This message is displayed when your trigger specification includes the same bit or bits in two or more labels. You need to correct the references.

ERROR: Data label < LABEL> is not a valid entry

This message indicates your trigger specification includes a label which is not valid for the current mode. You need to correct the reference.

ERROR: Data label "< LABEL> .xx" specifed bit does not exist

This message is associated with the "process_for_data" command and appears when you enter a label bit reference that is not one of the logical bits in that label. < LABEL> is the label name and xx is the bit number entered. You should choose a correct bit number reference.

ERROR: Duration is greater than trace memory

This message indicates you have entered a duration that is greater than the duration of the entire trace memory. You need to enter a valid duration or change the sample period to extend the trace memory duration length.

ERROR: Duration is less than one sample period

This message is associated with the "process_for_data" command and appears when you enter a duration less than the system sample period. You should shorten the sample period or choose an longer duration.

ERROR: Emul700dmn attach error, verify product version compatibility

This message appears when an error occurs starting multiple processes. This occurs most often while trying to start an emulator and timing interface session in multiple X Window System windows at the same time. Verify product version compatibility, and reexecute the programs.

ERROR: Expression too long, shorten mark and trigger expressions

This message is displayed when the combination of all the trigger and mark definitions is too long. You need to shorten or remove unused definitions.

ERROR: < FILE> is not a timing configuration file

This message is displayed when you try to save or load a configuration file that already is used to store a configuration of some other instrument, such as a state/software analyzer. You need to remove to the or use a different file name.

ERROR: File name must be less than or equal to 1024 characters

This message indicates a file name reference is too long. You need to correct the entry.

ERROR: < FILE> No such file or directory

This message is displayed when the specified file could not be found. You need to correct the entry.

ERROR: Halt_repetitive_execution definition is invalid

This message is displayed when you change the mode or redefine the sample period is such a way that the halt_repetitive_execution condition is not longer valid. You need to correct the halt_repetitive_execution condition.

ERROR: Help keys are not defined properly

This message indicates the timing analyzer help keys have been incorrectly defined. You need to contact Hewlett-Packard Customer Support.

ERROR: Label table is full

This message appears when you try to define more than 32 labels. You need to remove unused labels before continuing.

ERROR: Label width exceeds instrument's capabilities

This message is displayed when you try to create a label having more probe bits than are available in the analyzer. You need to limit your entry.

ERROR: Labels used in any other specification cannot be deleted

This message appears when trying to delete a label that is used in another specification. A label cannot be deleted if it is reference in a specification. In order to delete the label, remove the reference to it in all specifications.

ERROR: Mark not found

This message indicats that a find could not locate the mark in the segment of memory searched. You can try broadening the search conditions.

ERROR: Mark x or o not found

This message indicates the mark command could not locate the conditions for one or both of the marks x and o. You should correct the conditions.

ERROR: Mode is not glitch capture

This message appears when you try to enter a trigger specification requiring glitch detection and the analyzer is not in glitch capture mode. You need to correct the trigger specification or change modes.

ERROR: New label already exists

This message appears when trying to rename a format specification label to a new name. The new name already exits and cannot be reused without removing the existing name. The existing name can itself be renamed or deleted.

ERROR: "or_on" is not possible, all bits are already specified

This message is displayed when you try to enter a trigger specification which uses bits which have already been specified. You need to correct the specification.

ERROR: Pattern not found

This message indicates a find command could not locate the specified pattern and condition in the segment of memory searched. You can try broadening the search conditions.

ERROR: Permission denied

This message indicates the read or write permissions inhibit the analyzer from reading or modifying the specified file. You correct the permissions or use a different file.

ERROR: Process_for_data definition is invalid

This message appears when you change mode or redefine the labels in such a way that the "process_for_data" definition is no longer valid. You need to correct the "process_for_data" definition.

ERROR: Range must be greater than one sample

This message appears when you try to find an event and the range you specify is only one sample wide. You need to expand the search range specification.

ERROR: Sample exceeds memory depth

This message appears when you enter a "process_for_data" command with a number of samples "before" or "after" a transition and the number of samples is greater than the memory depth of the analyzer. You need to correct the "process_for_data" condition.

ERROR: Sample rate is too slow for sampled trigger duration

This message is displayed when you try to enter a combinational glitch trigger definition where the sample period is longer than the trigger duration. Glitch triggering requires all data to be sampled before trigger duration detection circuitry, and thus requires the sample period to be less than one-half the specified duration.

ERROR: Shell variable TIMING_ASCII is incorrect format

This message is displayed when the timing analyzer is using an ASCII diagram and the TIMING_ASCII shell variable is incorrectly defined. You need to correct the shell variable or remove it to use the default values. Refer to "Appendix C: Timing Output and Diagrams" for information on assiging the TIMING_ASCII shell variable.

ERROR: Single bit label must be entered

This message is displayed when you try to "process_for_data" relative to a transition on a label, and the label is defined as more than one bit wide. You should use a different label or change the "process_for_data" condition.

ERROR: Specification does not agree with captured data

This message appears when you attempt to save the configuration "with_data" after you have changed the specification from that used to capture the data. Change the specification to match that used to capture the data or execute a measurement to update the captured data.

ERROR: Specified compare labels do not exist in compare file

This message appears when loading a new compare file and one or more of the specifications or displays contain references to compare file labels that do not exist in the new file. You correct the invalid references.

ERROR: Statistics definition is invalid

This message is displayed when you change the mode or redefine the sample period in such a way that the statistics definition is no longer valid. You need to correct the statistic definition.

ERROR: Timing graphics initialization failed

This message appears when the timing analyzer is unable to use graphics for displaying the timing diagram; the ASCII diagram format will be used. Refer to appendix C "Timing Output and Diagrams" for information on setting up a graphics environment.

Notes

D

Accurate Timing Measurements

Introduction

This appendix provides information for making accurate time interval measurements with the timing analyzer.

Accurate time interval measurements depend primarily on the time interval resolution of the timing analyzer. If you have a good time interval resolution, using statistics can improve the accuracy of a measured time interval by the amount described in the equations which you will find in this appendix. Accurate statistical measurements can be made only if the input intervals are a uniform distribution of the interval to be measured. The following information describes important aspects in making accurate time interval measurements.

Time Interval Resolution

Resolution is a measurement of the precision of a timing trace and depends on the following factors:

- Sample Period.
- Interchannel Skew.
- Memory Depth.

Factors

Sample Period

The sample period is the amount of time between samples.

Interchannel Skew

Interchannel skew is the difference in delays of the probe channels, including delay differences from one channel to another, and delay differences in recognizing negative and positive transitions.

Skew is a function of several input variables as follows:

- Input signal slew rate in volts/ns (low slew rate increases the skew).
- Signal overdrive above the threshold as a percent of skew (low overdrive increases skew).
- Threshold value selected (high threshold settings increase skew).

The skew specifications of 4 ns for all signals within the probe is measured according to the following conditions:

- A 0.25-volt per nanosecond slew rate.
- A 0.6-volt amplitude signal with equal swings on either side of the threshold.
- A minus 1.3-volt threshold.

Memory Depth

The depth of trace memory is also important in time interval resolution. The memory depth sets the maximum time interval that can be measured with any sample period. In the 100 MHz standard mode of operation, a total interval of 10.24 us can be measured with full accuracy:

10.24 us = 1024 samples X 10 ns per sample

To measure longer intervals, the sample period must be increased.

Calculation

Resolution is formally defined as:

resolution = + /- (sample period + skew).

A high sample rate (100 MHz), coupled with low skew (plus or minus 4 ns for both opposite and same direction transitions), gives the timing analyzer very good resolution.

Improving the Accuracy of Time Interval Measurements

You can improve the accuracy of an interval measurement by making the measurement with a series of repetitive executions. When a single execution is made, the resolution is equal to + /-(sample period + skew). When measuring a stable interval using a series of repetitive executions, the accuracy of the measurement improves by the following:

accuracy = + /-(((sample period)/sqrt(n)) + skew)

where:

sample period	is the sample period specified in the timing analyzer trace specifications.
sqrt(n)	is the square root of the number of executions included in the measurement.
skew	is the delay differences between input channels.

A tenfold improvement (x10) is obtained in the accuracy of your measurement when using one hundred repetitive executions for the measurement.

Note

The time interval being measured must not be synchronous to the sampling clock of the timing analyzer. Typically, this is not a problem in the timing analyzer unless the sample rate is extremely low. The timing analyzer halts its interval sample clock between each measurement; therefore, the probability is low that the time interval being measured is synchronous with the timing interval sample clock.

Improving the Accuracy of Mean Value Measurements

The accuracy of the displayed mean value of a single interval depends on the number of executions in the series used to determine the mean value.

Assume the timing analyzer is measuring a stable, repetitive time interval approximately 100 us long. Using a 20 MHz sample rate (50 ns sample period), you capture 51.2 us of timing data, calculated as follows:

51.2 us = 1024 samples X 50 ns sample period.

A single measurement will have an accuracy of plus or minus the sum of the sample period plus the skew specification, or:

+ /-(50 ns + 4 ns)
= + /- 54 ns

By making one hundred measurements in a repetitive series, the accuracy of the mean value displayed will be improved by a factor of the square root of the number of traces included in the series, as shown in the following:

+ /-((50 ns/sqrt(100)) + 4 ns)
= + /- (5 ns + 4 ns)
= + /- 9 ns

Accuracy of Standard Deviation Measurements

An interval that does not vary still can be shown to have a large standard deviation due to the sampling process. The error in the displayed standard deviation depends on the size of two elements:

- the portion of the interval that exceeds the multiple of the sample periods; and
- the portion of the interval that includes complete sample periods.

Example 1

Assume the timing analyzer is measuring a time interval of exactly 25.0 ns. It makes 10 executions using a 10 ns sample period. Five of the executions show the interval to last 30 ns (three sample clocks), and five of the executions show the interval to last 20 ns (two sample clocks). Even though the input signal has a true standard deviation of 0.0 ns, the timing analyzer will calculate the standard deviation of this signal to be 5.28 ns and display this standard deviation on-screen. When the sampled standard deviation is less than one sample period, its value is determined mainly by the sampling process.

Example 2

Assume the timing analyzer is measuring a time interval that varies from 5 to 8 us. The timing analyzer is operating with a 10 ns sample period. After a series of repetitive measurements, the timing analyzer shows a standard deviation of 1.5 us for the interval being measured. This dispersion is determined by variations in the time interval itself, and not by the sampling process. In this case, the standard deviation is much larger than one sample period.

Statistical Errors Caused by Sampling Process

The timing analyzer calculates statistics on the sampled data in its memory, not on all of the data generated by the system under test. The data in memory may misrepresent the actual data. Misleading data can be captured when you trigger your trace on some occurrence that causes the timing analyzer to capture samples at misleading points in the data flow of a system under test.

Use of the "trigger on anything" specification may not overcome all measurement bias problems. Consider the case where the timing analyzer is measuring an interval of time between positive edges occurring on a probe line. Suppose there are two intervals on that line, and they are occurring alternately (one is 10 us long and the other is 20 us long). Interval measurements are made by marking "x" on the first positive edge of the selected label and marking "o" on the next positive edge after "x". The random beginning of a new trace will probably occur twice as often during the 20 us interval as during th 10 us interval. Because of this, the timing analyzer will appear to be finding twice as many 10 us intervals as 20 us intervals, but in the system under test there are equal numbers of 10 us and 20 us intervals.

One possible approach to solving the problem of misleading data in the above example is to find another line with a uniform squarewave operating at twice the frequency of the combined intervals. Such a squarewave will have as many positive edges preceding 10 us intervals as 20 us intervals. By triggering the interval measurements on positive edges in that squarewave, and marking "x" and "o" on the first interval after each trigger, the timing analyzer will measure as many 20 us intervals as 10 us intervals.

Index