HEWLETT PACKARD

User's Guide

HP 64700 Series Analyzer PC Interface

HP 64700 Series Emulators

Analyzer

PC Interface User's Guide

HEWLETT PACKARD

HP Part No. 64740-97007
Printed in U.S.A.
August 1990

Edition 4

Notice

Hewlett-Packard Company
Logic Systems Division
8245 North Union Boulevard
Colorado Springs, CO 80920, U.S.A.

Printing History

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

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

Edition 1	64740-90911 E1187	November 1987
Edition 2	64740-90911 E0988	September 1988
Edition 3	64740-97006	May 1990
Edition 4	**64740-97007**	**August 1990**

Using this Manual

This manual will show you how to use the HP 64700 series analyzer with the host computer PC 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 connect the external analyzer probe to target system signals and how to use the external analyzer as an extension of the emulation analyzer.
- Show you how to use the external timing analyzer via the Timing Interface.
- 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.

This manual will not:

- Show you how to use every PC Interface command and option; the PC Interface is described in the *HP 64700 Emulators PC Interface: User's Reference.*

Organization

Chapter 1 **Introducing the HP 64700 Series Analyzer**. This chapter lists the basic features of the analyzer. The following chapters show you how to use these features.

Chapter 2 **Using the Internal Analyzer**. This chapter shows you how to use the internal emulation analyzer, and it describes the basic steps performed when using the analyzer.

Chapter 3 **Internal Analyzer Examples**. This chapter contains examples of using the analyzer. It describes a sample program on which trace measurements are made, and then it shows you how to specify a simple trigger condition, how to use storage, prestore, and count qualifiers. This chapter also shows you how to use the sequencer and how to trace "windows" of program execution.

Chapter 4 **Using the External Analyzer**. This chapter shows you how to connect the external analyzer probe to the target system and how to configure the external analyzer (it may be aligned with the internal analyzer, configured as an external state analyzer, or configured as an external timing analyzer). This chapter shows you how to use the external analyzer when it is aligned with the internal analyzer and when it is configured as an external state analyzer.

Chapter 5 **Using the External Timing Analyzer**. This chapter shows you how to use the external timing analyzer. The external timing analyzer allows you to perform timing measurements and view the captured data in waveform displays.

Chapter 6 **Making Coordinated Measurements**. This chapter shows you how to allow other HP 64700 emulators to start trace measurements, how to use the analyzer trigger condition to break the emulator, how to use the emulation analyzer trigger to drive external trigger signals, and how to allow the analyzers to be armed (turned on) by external trigger signals.

Contents

Illustrations

1

Introducing the HP 64700 Series Analyzer

Overview

This manual describes the HP 64700 Series analyzer. Each HP 64700 Series emulator contains an internal emulation analyzer. Some emulators may optionally contain an external analyzer.

The *emulation analyzer*, also known as the *internal 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 PC Interface. A block diagram of the analyzer is shown in figure 1-1.

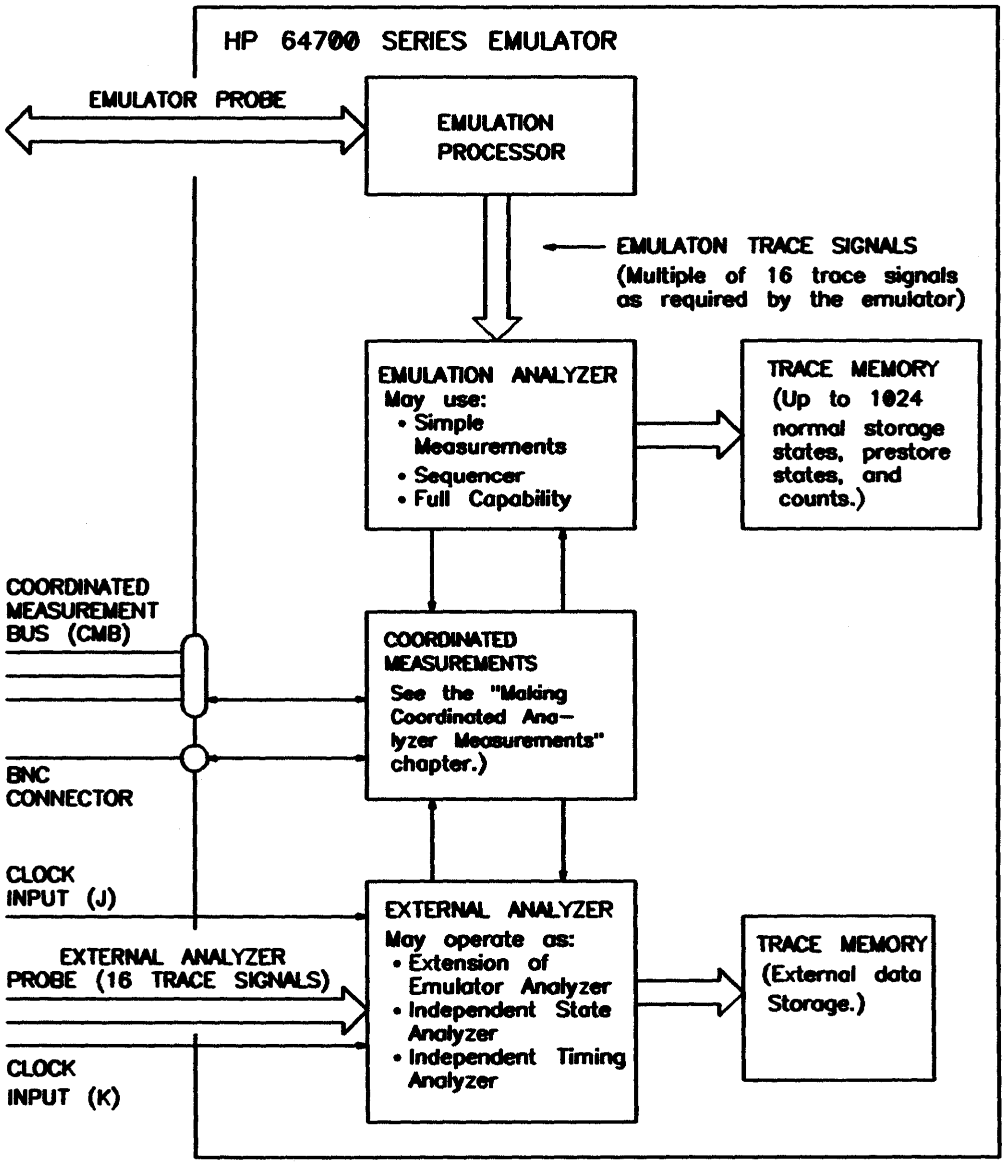

HP 64700 SERIES EMULATOR
EMULATOR PROBE
EMULATION PROCESSOR
EMULATON TRACE SIGNALS (Multiple of 16 trace signals as required by the emulator)
EMULATION ANALYZER May use: • Simple Measurements • Sequencer • Full Capability
TRACE MEMORY (Up to 1024 normal storage states, prestore states, and counts.)
COORDINATED MEASUREMENT BUS (CMB)
COORDINATED MEASUREMENTS See the "Making Coordinated Analyzer Measurements" chapter.)
BNC CONNECTOR
CLOCK INPUT (J)
EXTERNAL ANALYZER PROBE (16 TRACE SIGNALS)
CLOCK INPUT (K)
EXTERNAL ANALYZER May operate as: • Extension of Emulator Analyzer • Independent State Analyzer • Independent Timing Analyzer
TRACE MEMORY (External data Storage.)

Figure 1-1. Block Diagram of HP 64700 Series Analyzer

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 single "analysis begin" command. You can enter additional trace commands to qualify 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 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

You can use the analyzer to search for a particular sequence of states. The sequencer, which makes this possible, has several levels. Each level of the sequencer searches for a "find" or "trigger" state. When the find/trigger state is captured, the analyzer goes on searching at the following sequence level. If branches are turned on, a "branch" state is also searched for in each sequence level. If the branch state is found before the find/trigger state, the analyzer can jump to any sequence level where it continues searching for states.

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.

External Timing Analysis

If your emulator contains an external analyzer, you can configure the external analyzer to perform timing measurements. Some of the features of the external timing analyzer are listed below.

- Standard data acquisition mode with a maximum sample rate of 100 MHz (10 ns).
- Transitional data acquisition mode extends the storage time interval by storing only information changes and the time between them.
- Glitch data acquisition mode on all channels without reducing the number of channels.
- Trigger when signals on the external probe match a specified pattern for greater than or less than a specified time (duration). Edge and glitch qualifiers may be included in the trigger specification.
- You can place the trigger point at the start, center, or end of the trace to view signals after, about, or before the trigger.
- Timing Interface supports screen dumps to graphics printers (for printing waveform displays).

Note

The PC Interface requires EGA or VGA monitors and adapters to display timing waveforms.

Note

Two versions of the PC Interface are shipped with your emulator; one contains interface software for the timing analyzer and the other does not. The version that contains the timing interface software is larger and requires most of the PC's 640K bytes of RAM. The version that does not contain the timing interface software requires approximately 480K bytes of RAM.

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, the analyzer may be triggered by other emulators and instruments.

Also, if your emulator contains an external analyzer being used as an independent analyzer, coordinated measurements may take place between the internal emulation analyzer and the external analyzer.

Notes

Using the Internal Analyzer

Introduction

This chapter describes how to use the emulation analyzer from within the PC Interface. The steps performed when using the analyzer are:

- Modifying the trace specification (or using the default).
- Starting the trace.
- Displaying the trace.

These steps are described in the main sections of this chapter.

Modifying the Trace Specification

The PC Interface's "Analysis Trace Modify" command provides one screen (with one subscreen) from which you make a complete trace specifications. This section describes the options available on those screens. To access the trace specification screen, select:

```
Analysis, Trace, Modify
```

Note

If your emulator contains an external analyzer (which can operate as an independent analyzer), you will have to include another option, "Internal", to specify which analyzer the command is for.

The trace specification screen (see figure 2-1) shows the sequence levels that are being used, the qualifiers associated with those levels, the secondary branch mode, the count and prestore qualifiers, and the trigger position selection. When you initially enter the PC Interface, the default analyzer configuration specifies a trigger on any state, and that all captured states are stored.

Sequence Levels

The trace specification screen in figure 2-1 shows two sequence levels. At least two sequence levels will always be used because a trigger level is required and there must always be a level after the trigger level.

Each sequence level in the trace specification screen, except the last, has four associated fields; the sequence level field, the storage qualifier field, the find/trigger state field, and the occurrence count field. Only the sequence level field and the storage qualifier field are available in the last sequence level.

When you begin (start) a trace, the analyzer searches for the find/trigger state of the first sequence level; when that state is

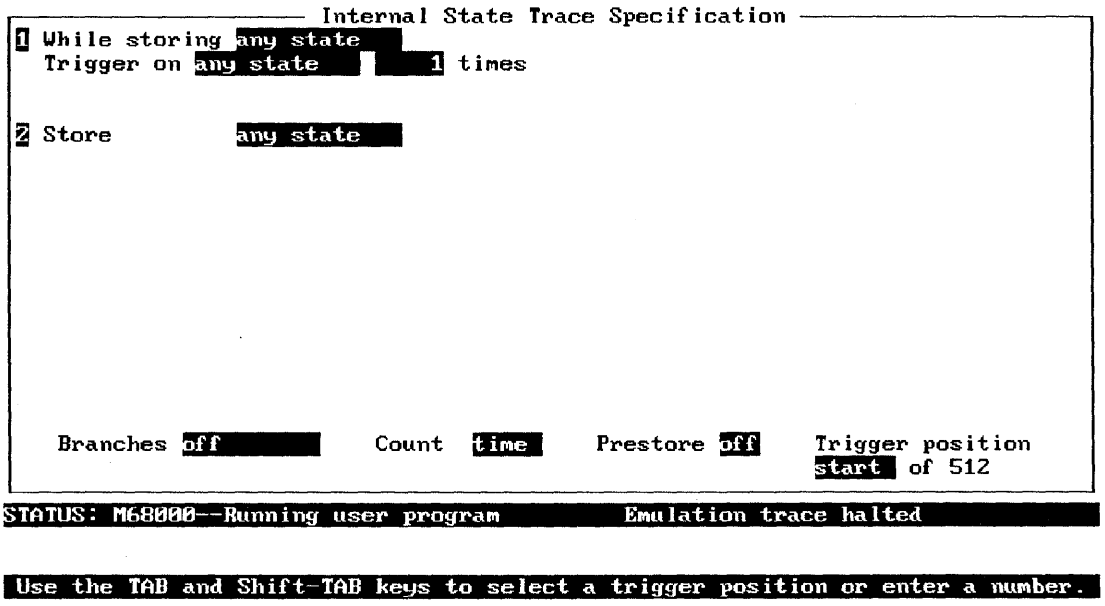

```
                  Internal State Trace Specification
1 While storing any state
  Trigger on any state        1 times

2 Store         any state




  Branches off          Count time      Prestore off     Trigger position
                                                         start  of 512
STATUS: M68000--Running user program        Emulation trace halted

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 2-1. Trace Specification Screen

found, the analyzer searches for the find/trigger state of the second sequence level, and so on.

One of the sequence levels is specified as the trigger level. The analyzer triggers when it finds the state associated with this level. The trigger is the reference point in the resulting trace. States are stored before, after, or about the trigger state. The trigger state is always on line 0 of the trace.

Sequence Level Field

The sequence level field shows the level (1 through 8) of the sequence term. This field also allows you to insert or delete sequence terms, or to specify the trigger level.

Move the cursor to the first sequence level field in the upper left corner of the screen, and use the **Tab** key to view the options associated with sequence levels. These options are:

I	Insert level.
D	Delete level.
T	Set the trigger level.

Select the "I" option and enter **Enter** to insert a new sequence level. Using the arrow cursor keys, reposition the cursor and repeat this process until eight sequence levels are used. This is the maximum number of sequence levels.

Note

When moving the cursor in the trace specification screen, use the arrow cursor keys. Using **Enter** (carriage return) in the state qualification fields causes the "patterns and expressions" screen to appear.

Move the cursor to level 5 and make this the trigger level by entering "T" and a **Enter**. Notice that level 5 now shows "Trigger on" instead of "Then find" while the previous trigger level now shows "Then find".

Move the cursor to the last sequence level and enter "D" and **Enter**. Notice that the previous level is deleted instead.

Now try to delete the trigger level. Notice that the following level is made the trigger level until it is the second to last; then, the previous level is made the trigger level.

Storage Qualifier Field

The storage qualifier field allows you to specify which states are stored while the analyzer searches for that sequence level's find or trigger state.

Move the cursor to a storage qualifier field and use the **Tab** key to scroll through the choices. You will see the following options:

any state	All states are stored.
no state	No states are stored.
r, !r	Only states in the range (r) or states not in the range (!r) are stored. Ranges are defined in the patterns and expressions screen described below.
a, b, c, d, e, f, g, h	Only states that match the pattern are stored. Up to eight different patterns can be defined. Patterns are defined in the patterns and expressions screen described below.
arm	Store only the state executing when the analyzer is armed (enabled) by some condition external to the analyzer. Conditions that can be used to arm the analyzer are described in the "Making Coordinated Measurements" chapter.

These same options are available in any state qualifier field.

Find State or Trigger on State Field

The "find state" or "trigger on" state field allows you to define the state that causes the analyzer to start searching for the following sequence level or that causes the analyzer to trigger.

The options available in this field are the same as for the storage qualifier field.

Occurrence Count Field

The occurrence count field specifies the number of times the state specified in the find/trigger field must be found before the analyzer goes on to the following sequence level.

The default base for an occurrence count is decimal. You may specify occurrence counts from 1 to 65535.

Branches

Branches alter the analyzer's search sequence if the specified branch state occurs before the find/trigger state.

When branches are turned off, the analyzer searches for the first sequence level's find/trigger state; when this state is found, the analyzer searches for the second sequence level's find/trigger state, and so on.

When branches are turned on, the analyzer searches for find/trigger states associated with each sequence level; if the find/trigger state is found before the branch state, the analyzer still continues its search at the following sequence level. However, if the branch state is found before the find/trigger state, the analyzer can continue its search at a different sequence level.

off	Branches are turned off. The analyzer searches only for the states defined in the find/trigger fields, starting with the first sequence level, then the second, and so on.
restart on	Branches are turned on. You are given another field in which to specify the restart state qualifier. This restart qualifier is used for all sequence levels, and the branch is always back to the first sequence level. This is also known as "global restart" because the same restart condition applies to all sequence levels.
per level	Branches are turned on. Two additional fields are added to each sequence level except the last:

one for the branch state qualifier, and one to specify the sequence level the analyzer branches to if the branch state is found before the find/trigger state.

No branch state qualifier field is added to the last sequence level, but you are given a field in which to specify the sequence level the analyzer branches to when the "then find" state is found.

Count

For each stored state, the analyzer can either count time or the number of occurrences of some state.

time	Time counts are recorded for each stored state.
state	The number of occurrences of some state are recorded for each stored state. You are given another field in which to specify the state count qualifier.
off	Counting can be turned off to allow more states to be stored. When the analyzer counts states or time, 512 states can be stored in a trace. When counting is turned off, 1024 states can be stored in a trace.

Counts can be displayed in relative format (relative to the previous state) or absolute format (relative to the trigger state). Refer to the "Changing the Trace Format" section that appears later in this chapter.

Prestore

Prestore allows you to store up to two states which match the prestore qualifier state before each normal store state. Prestore is useful, for example, in identifying the callers of a function or subroutine that is called from many places in the program.

off	Prestore is turned off.
on	Prestore is turned on. You are given another field in which to specify the prestore qualifier.

Trigger Position

The trigger position specifies where the trigger state appears in the trace.

center	Use "center" when you are interested in states that occur before and after the trigger.
end	Use "end" when you are only interested in states that occur before the trigger.
start	Use "start" when you are interested only in states that occur after the trigger.
<number>	In addition to start, center, and end, you can type in a number to specify the trigger position. For example, if you specify the trigger position as "10 of 512", nine states before the trigger will be stored, and 502 states will be stored after the trigger (assuming nine states must match the storage qualifiers before the trigger and 502 states must match the storage qualifiers after the trigger).

You will sometimes notice that the trigger position is not exact. The actual trigger position will be within +/- 1 state of the position specified if counting states or time; if the count qualifier is turned off, the actual trigger position will be within +/- 3 states of the number specified.

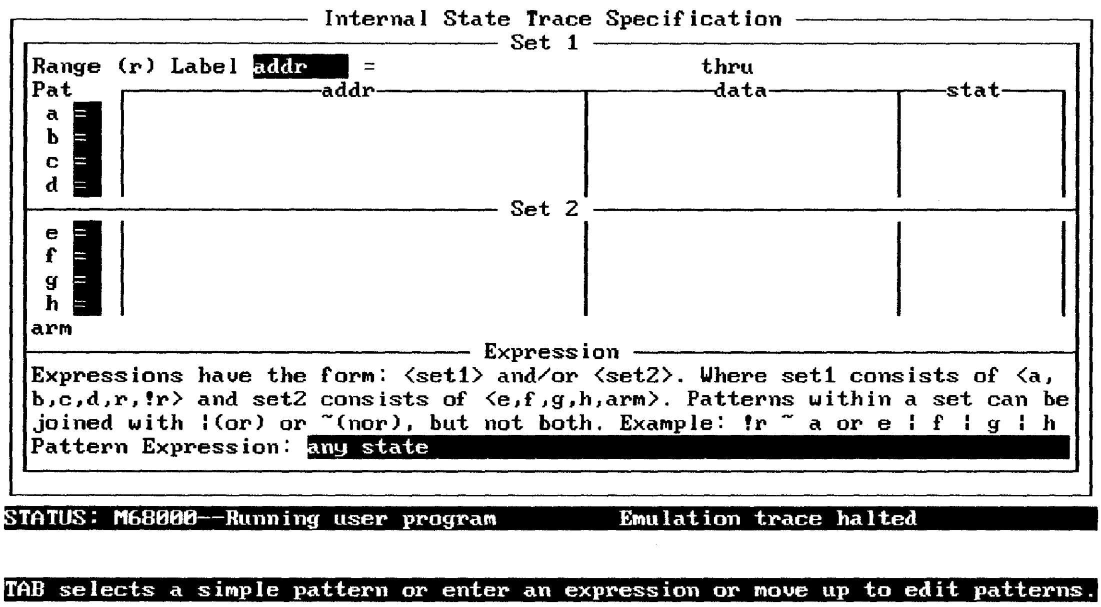
Specifying State Patterns and Expressions

Any time the cursor is in a state qualifier field, entering a **Enter** will cause the patterns and expressions screen to appear (see figure 2-2).

In the patterns and expressions screen, you assign values to the range and pattern resources and you combine these resources to create the pattern expression.

Values

When assigning values to the range and pattern resources, you can use symbols (in the address field only) or numeric constants in several number bases (including predefined equates in the status



```
                    Internal State Trace Specification
                                  Set 1
Range (r) Label addr    =                        thru
Pat              addr                       data                stat
 a =
 b =
 c =
 d =
                                  Set 2
 e =
 f =
 g =
 h =
arm
                                Expression
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h
Pattern Expression: any state

STATUS: M68000--Running user program         Emulation trace halted

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 2-2. Ranges, Patterns, and Expressions

field only). You can also use operators to combine symbols and numeric constants.

Constants. Values may be specified as numeric constants in hexadecimal, decimal, octal, or binary. The number bases are specified by the following characters:

Y y	Binary (example: 10010110y).
Q q O o	Octal (example: 377o or 377q).
T t	Decimal (example: 2048t).
H h (or no base)	Hexadecimal (example: 0a7fh or 0a7f). (The leading digit of a hexadecimal constant must be 0-9.)

Don't care digits may be included in binary, octal, or hexadecimal numbers and they are represented by the characters **X** or **x**. A value of all don't cares may be represented by a question mark (?) or a blank field.

Operators. Values can also be symbols and numeric constants combined with 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 descending 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.
<<, <<<, >>, >>>	Shift left, rotate left, shift right, rotate right.
&	Bitwise AND.
^	Bitwise exclusive or, XOR.
|	Bitwise inclusive OR.
&&	Logical AND/bit-wise merge. When bits are different, the first value overrides the second; e.g., 10xxy && 11x1y == 10x1y.

Refer to the *PC Interface: User's Reference* description of **expr** for operator truth tables.

Assigning Values to the Range Resource

The range resource defines a range of values associated with a particular trace label. Move the cursor to the range label field and use the **Tab** key to view the predefined trace labels. Notice that these labels are also used when assigning values to the pattern resources. Trace labels are described in the "Assigning Values to the Pattern Resources" section that follows.

Pressing the right arrow key moves the cursor to the starting value field. Notice that you are given a line in the bottom part of the display in which to enter the value. The value you enter on this line may be longer than the space available in the starting value field; when this happens, as much of the value as possible is shown in the starting value field. Pressing the right arrow key again moves the cursor to the ending value field.

Assigning Values to the Pattern Resources

You can assign address, data, status, and xbits (if an external analyzer is present) values to each of the pattern resources.

Move the cursor to the field to the immediate right of a pattern resource and press the **Tab** key a couple times. When "=" is selected, the pattern is equal to the state described by the following values; when "!=" is selected, the pattern is not equal to the state described.

The addr, data, stat, and xbits (if an external analyzer is present) columns allow you to assign values to each of the trace labels. When a pattern has values assigned to more than one trace label, these values are ANDed. In other words, if you assign an address value of 500H and a data value of 1234H to a pattern, a captured state must have an address of 5FFH AND data equal to 1234H in order to match the pattern.

Trace Labels. The following trace labels are predefined in most of the HP 64700 Series emulators:

addr	Represents the trace signals which monitor the emulation processor's address pins (trace signals 0 through 23 in the 68000 emulator).

data	Represents the trace signals which monitor the emulation processor's data pins (trace signals 32 through 47 in the 68000 emulator).
stat	Represents the trace signals which monitor other emulation processor signals (trace signals 24 through 31 in the 68000 emulator).
xbits	Represents the external trace signals. This trace label is only available if an external analyzer is present and aligned with the internal analyzer.

These trace labels are predefined. You can define additional trace labels in the screen presented with the "Analysis Format" command. Refer to the "Changing the Display Format" section that appears later in this chapter.

The trace signals associated with the addr, data, and stat trace labels are different for each emulator. Other emulators may provide additional trace labels to choose from.

Predefined Values for 68000 Emulator Status. Move the cursor to the "stat" field of a pattern resource. Notice that a line on which you can enter status values appears on the bottom part of the screen. Use the **Tab** key to view the predefined values. The predefined values for the 68000 emulator are:

Qualifier	Status Bits (31..24)	Description
byte	0xxxx xxx1Y	Byte cycle.
cyc6800	0xxxx x0xxY	6800 peripheral cycle.
data	0xxx0 1xxxY	Data cycle.
dma	0xx01 1xxxY	DMA cycle.
grd	00xxx xxxxY	Guarded memory access.
intack	0xx11 1xxxY	Interrupt acknowledge cycle.
prog	0xxx1 0xxxY	Program cycle.
read	0xxxx xx1xY	Read cycle.
super	0xx1x xxxxY	Supervisor cycle.
supdata	0xx10 1xxxY	Supervisor data cycle.
supprog	0xx11 0xxxY	Supervisor program cycle.
user	0xx0x xxxxY	User cycle.
userdata	0xx00 1xxxY	User data cycle.
userprog	0xx01 0xxxY	User program cycle.
word	0xxxx xxx0Y	Word cycle.
write	0xxxx xx0xY	Write cycle.
wrrom	0x0xx xx0xY	Write to ROM.

You can also select these predefined values when specifying status values for the pattern resources.

The predefined values for emulator status are different for each emulator. Refer to your *PC Interface: Emulator User's Guide* for the definitions of the status values predefined for your emulator.

Specifying Pattern Expressions

The range and pattern resources and the arm condition are split into two sets.

Set 1: **r**, **!r**, **a**, **b**, **c**, and **d**.

Set 2: **e**, **f**, **g**, **h**, and **arm**.

Interset Operators. Resources from the two sets can be combined with the **and** and **or** interset (between set) operators.

- You cannot use these operators to combine resources from the same set.

Intraset Operators. Resources within a set may be combined using one of the intraset operators, | (OR) or ~ (NOR).

- You cannot use both | (OR) and ~ (NOR) operators within the same set.
- You cannot combine resources from different sets with the | (OR) or ~ (NOR) operators.
- Note that "a ~ a" is allowed; this type of expression may occasionally be useful if you are running out of pattern resources and wish to specify a logical NOT of some existing pattern.

The intraset (within a set) operators (~, |) are evaluated first; then, the interset operators are evaluated.

Set Operator Limitations. Only the OR (|) and NOR (~) logical operators are available as intraset operators. However, you can create the AND and NAND operators by applying DeMorgan's law:

$$a \text{ AND } b = \bar{a} \sim \bar{b}$$
$$a \text{ NAND } b = \bar{a} \mid \bar{b}$$

To create a logical NOT of a pattern resource, invert the equality used when assigning values to that resource.

Resetting the Trace Specification

You can reset the analyzer to its default specification by selecting:

```
Analysis, Trace, Reset
```

The resulting trace specification is shown in figure 2-1.

Starting the Trace

After the trace specification has been made, you can start the trace by selecting:

```
Analysis, Begin
```

The emulator must be running before the analyzer can capture program execution.

When the trigger state has been found and trace memory has been filled with states, a message on the status line will show you that the trace is complete. When the status line shows you that the trace is complete, the next step is to display the trace.

If the status line continues to show that the trace is "running", two things are possible:

1. The trigger has been found, but there haven't been enough storage states captured to fill the trace memory buffer.

2. The trigger has not been found.

To find out which of these situations is the case, enter the "Analysis Display" command (described later in this chapter). The number of states that can be displayed is shown at the bottom of the screen. If there are states to display, then situation 1 above has occurred.

If there are no states to display, then situation 2 above has occurred. If the trigger state has not been found and you think it should have been found, you may want to halt the trace and review your trace specification.

Halting a Trace Measurement

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. To do this, select:

```
   Analysis, Halt
```

The trace halt command is also useful to deactivate signals which are driven when the trigger is found (refer to the "Making Coordinated Measurements" chapter).

Displaying the Trace

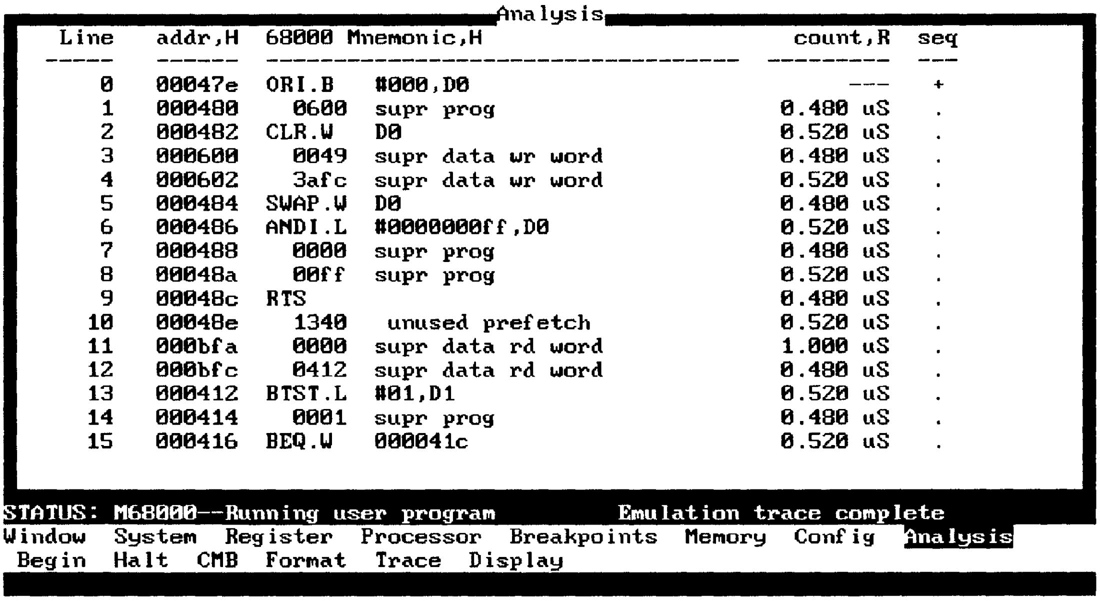
To display the trace, select:

```
   Analysis, Display
```

If states have been stored, you are now given two fields in which to specify the states to display. Type the number of the first state you want to display in the "Starting state to display" field, and press **Enter**. Use the right arrow key to move the cursor to the "Ending state to display" field. Type in the number of the last state you want to display, and press **Enter**.

If your emulator supports symbol storage, you see an additional field that gives you control of the display of addresses and symbols.

An example trace display is shown below.



```
                                     Analysis
  Line    addr,H   68000 Mnemonic,H                            count,R   seq
  -----   ------   ----------------------------------        ---------   ---
      0   00047e   ORI.B    #000,D0                                ---     +
      1   000480     0600   supr prog                         0.480 uS     .
      2   000482   CLR.W    D0                                0.520 uS     .
      3   000600     0049   supr data wr word                 0.480 uS     .
      4   000602     3afc   supr data wr word                 0.520 uS     .
      5   000484   SWAP.W   D0                                0.480 uS     .
      6   000486   ANDI.L   #000000ff,D0                      0.520 uS     .
      7   000488     0000   supr prog                         0.480 uS     .
      8   00048a     00ff   supr prog                         0.520 uS     .
      9   00048c   RTS                                        0.480 uS     .
     10   00048e     1340    unused prefetch                  0.520 uS     .
     11   000bfa     0000   supr data rd word                 1.000 uS     .
     12   000bfc     0412   supr data rd word                 0.480 uS     .
     13   000412   BTST.L   #01,D1                            0.520 uS     .
     14   000414     0001   supr prog                         0.480 uS     .
     15   000416   BEQ.W    000041c                           0.520 uS     .

STATUS: M68000--Running user program          Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

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 shows mnemonic information about the emulation bus cycle. The disassembled instruction mnemonic is shown for opcode fetch cycles. The data and mnemonic status ("0600 supr prog", for example) are shown for operand cycles.

If your emulator contains an external analyzer (the model number has an "L" suffix), the next column, labeled "xbits", shows the data captured on the external channels.

The next column shows the count information (time is counted by default). The "R" indicates that each count is relative to the previous state.

The last column contains information about the sequencer. The "+" on line 0 indicates the state caused a sequencer branch (in this case, the trigger on any state).

Sometimes, the trace will show opcode fetches for instructions which are not executed because of a transfer of execution to other addresses. 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.

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 few captured states in this situation; you must halt the trace to display the last few captured states.

Changing the Trace Format

To change the trace format, select:

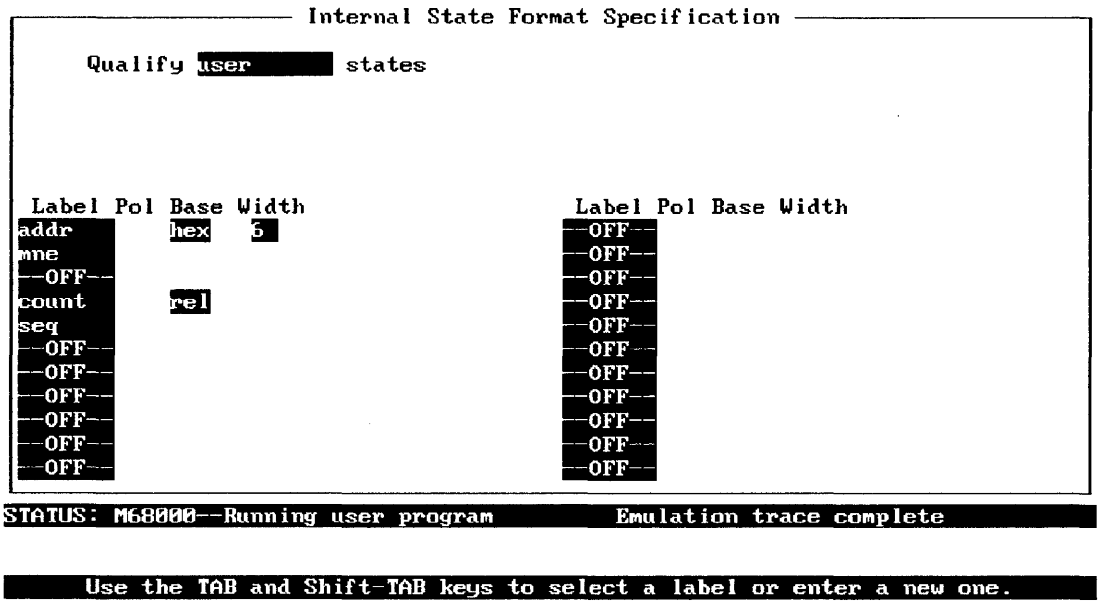
```
Analysis, Format
```

The analysis format screen is shown in figure 2-3.

The "Anlysis Format" command primarily allows you to arrange the columns of trace information in a different manner. However, notice that you can include any trace label in the trace. Also, notice that the trace label information can be displayed in various number bases, and that counts can be displayed relative or absolute.

Tracing Foreground/Background Execution

The first field in the "Analysis Format" display specifies whether the analyzer should capture user (foreground) states, background states, or all states (foreground and background). When background states appear in the trace, a string indicating that it is a background state typically appears at the end of the mnemonic column (for example, "BGM" appears when using the 68000 emulator).



```
                       Internal State Format Specification

     Qualify user       states



  Label  Pol  Base  Width                     Label  Pol  Base  Width
addr          hex    5                       --OFF--
mne                                          --OFF--
--OFF--                                      --OFF--
count         rel                            --OFF--
seq                                          --OFF--
--OFF--                                      --OFF--
--OFF--                                      --OFF--
--OFF--                                      --OFF--
--OFF--                                      --OFF--
--OFF--                                      --OFF--
--OFF--                                      --OFF--

STATUS: M68000--Running user program          Emulation trace complete

      Use the TAB and Shift-TAB keys to select a label or enter a new one.
```

Figure 2-3. Analysis Format Screen

Trace Labels

The bottom part of the "Analysis Format" display is used to specify which trace labels appear in which column of the display. Several trace labels are predefined for the internal emulation analyzer.

You can specify the number base (in hexadecimal, binary, octal, decimal, and ASCII) for several of the predefined trace labels.

addr	Values captured on the emulator's address bus. If your emulator supports symbol storage, you can change the width of the address column so that more symbol information is shown.
data	Values captured on the emulator's data bus.
stat	Values captured on the trace signals used for the emulator's status.

mne	Captured information disassembled and presented in mnemonic format.
count	State or time count information presented in absolute format (relative to the trigger state) or relative format (relative to the previous state in the trace).
seq	Shows a "+" if the captured state is one the analyzer was searching for and found (in other words, a find state, a trigger state, or an branch state).
DELETE	Deletes a trace label.
--OFF--	Turns a column off so that it is not displayed in the trace.

This concludes the "Using the Analyzer" chapter. You have learned about the analysis commands and all the trace specification fields except those that appear when you emulator contains an external analyzer. The additional fields are described in the "Using the External Analyzer" chapter.

3

Internal Analyzer Examples

Introduction

This chapter shows you how to use the emulation analyzer from within the PC Interface. It shows you how to make simple measurements as well as how to search for a sequence of states.

This chapter describes:

- The sample program on which example measurements are made.

This chapter provides the following examples of using the analyzer:

- Specifying a simple trigger.
- Using storage qualifiers.
- Using trace prestore.
- Changing the count qualifier.
- Trace a sequence of events using global restart.
- Trace a sequence of events while specifying "else" branches per level.
- Storing a "window" of program execution.
- Using multiple trigger levels.

Prerequisites

Before reading this chapter you should already know how the emulator operates. You should know how to use the PC Interface, and how to control the emulator from within the PC Interface. Refer to the appropriate *PC Interface: Emulator 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 examples in this chapter have been generated using an 68000 (HP 64742) emulator. The sample program is written in 68000 assembly language.

It is not important that you know the 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.

What the Sample Program Does

A pseudo-code algorithm of the sample program is shown in figure 3-1.

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 will show you how to use the analyzer to determine where a routine is called from.

```
              Initialize the stack pointer.
              Set up number counter.
      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.
              Decrement number counter.
              If the number counter is not zero
                  Then goto AGAIN (repeat program).
              Else
                  Reset number counter.
                  Push parameters for QSORT.
                  Call QSORT.

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

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.

      QSORT:  Quick-sort values in 256 byte data area.
```

Figure 3-1. Pseudo-Code Algorithm of Sample Program

An assembler listing of the sample program is shown in figure 3-2. 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. 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 the locations for the stack is located at 600H.

Before you can use the analyzer to perform measurements on the sample program, you must map memory and load the sample program.

```
HEWLETT-PACKARD: 68000 Assembler
FILE: C:\MNL\ODY\ANLY\PCI\SRC\68K\SRND.S
LOCATION OBJECT CODE LINE     SOURCE LINE

                           1 "68000"
                           2 **************************************************
                           3 * The "srnd.S" program runs in an infinite loop,
                           4 * writing random numbers to an output area and
                           5 * sorting the output area after 4FFH random
                           6 * numbers have been written.
                           7 **************************************************
                           8                 ORG     400H
                           9 ;-------------------------------------------------
                          10 ; Initialize the stack, and set up the counter
                          11 ; for the number of random numbers before sort.
                          12 ;-------------------------------------------------
  000400 2E7C             13 START           MOVE.L  #STACK,A7
  000402 00000BFE
  000406 343C 04FF        14                 MOVE.W  #4FFH,D2
                          15 ;-------------------------------------------------
                          16 ; The next two instructions move the second
                          17 ; previous random number into A1 (offset to
                          18 ; RESULTS area, and the previous random number
                          19 ; into D1.
                          20 ;-------------------------------------------------
  00040A 2241             21 AGAIN           MOVE.L  D1,A1
  00040C 2200             22                 MOVE.L  D0,D1
                          23 ;-------------------------------------------------
                          24 ; RAND returns random number in D0.
                          25 ;-------------------------------------------------
  00040E 6100 005A        26                 BSR     RAND
                          27 ;-------------------------------------------------
                          28 ; The following instructions determine which
                          29 ; caller calls WRITE_NUMBER (depends on last
                          30 ; two bits of the previous random number).
                          31 ;-------------------------------------------------
  000412 0801 0001        32                 BTST    #1,D1
  000416 6700 0004        33                 BEQ     ZERO_ONE
  00041A 600A             34                 BRA.B   TWO_THREE
  00041C 0801 0000        35 ZERO_ONE        BTST    #0,D1
  000420 6700 000E        36                 BEQ     CALLER_0
  000424 6010             37                 BRA.B   CALLER_1
  000426 0801 0000        38 TWO_THREE       BTST    #0,D1
  00042A 6700 0010        39                 BEQ     CALLER_2
  00042E 6012             40                 BRA.B   CALLER_3
                          41 ;-------------------------------------------------
                          42 ; The WRITE_NUMBER routine is called from four
                          43 ; different places.  The program is repeated
                          44 ; after the subroutine return.
                          45 ;-------------------------------------------------
  000430 6100 005C        46 CALLER_0        BSR     WRITE_NUMBER
  000434 6010             47                 BRA.B   TEST
  000436 6100 0056        48 CALLER_1        BSR     WRITE_NUMBER
  00043A 600A             49                 BRA.B   TEST
  00043C 6100 0050        50 CALLER_2        BSR     WRITE_NUMBER
  000440 6004             51                 BRA.B   TEST
  000442 6100 004A        52 CALLER_3        BSR     WRITE_NUMBER
```

Figure 3-2. Sample Program Listing

```
000446 0C42 0000     TEST            CMPI     #0,D2
                     ;---------------------------------------------------
                     ; If the counter is not zero, continue to write
                     ; random numbers.
                     ;---------------------------------------------------
00044A 57CA FFBE                     DBEQ     D2,AGAIN
                     ;---------------------------------------------------
                     ; The counter is zero.  Sort the random numbers
                     ; in the RESULTS area.
                     ;---------------------------------------------------
00044E 343C 04FF                     MOVE.W   #4FFH,D2    ; Reset counter.
                     ;---------------------------------------------------
                     ; Push the "high address" and "low address"
                     ; parameters expected by the QSORT routine.
                     ;---------------------------------------------------
000452 2F3C                          MOVE.L   #RESULTS+0FFH,-[A7]
000454 000005FF
000458 2F3C                          MOVE.L   #RESULTS,-[A7]
00045A 00000500
                     ;---------------------------------------------------
                     ; Call the QSORT routine.  Increment the
                     ; stack pointer after the call.
                     ;---------------------------------------------------
00045E 6100 0034                     BSR.W    QSORT
000462 DFFC                          ADDA.L   #8,A7
000464 00000008
000468 60A0                          BRA.B    AGAIN      ; Repeat program.

                     ***************************************************
                     * The RAND routine generates a pseudo-random
                     * number from 0-0FFH, and leaves the result
                     * in D0.
                     ***************************************************
00046A 2039 0000     RAND            MOVE.L   RAND_SEED,D0
00046E 0600
000470 C1FC 4E6D                     MULS.W   #4E6DH,D0
000474 2040                          MOVEA.L  D0,A0
000476 41E8 0339                     LEA      339H[A0],A0
00047A 2008                          MOVE.L   A0,D0
00047C 23C0 0000                     MOVE.L   D0,RAND_SEED
000480 0600
000482 4240                          CLR.W    D0
000484 4840                          SWAP     D0
000486 0280                          ANDI.L   #000000FFH,D0
000488 000000FF
00048C 4E75                          RTS

                     ***************************************************
                     * The WRITE_NUMBER routine writes the random
                     * number to the RESULTS area.  The second
                     * previous number is the offset in this area.
                     ***************************************************
00048E 1340 0500     WRITE_NUMBER    MOVE.B   D0,RESULTS[A1]
000492 4E75                          RTS

                     ***************************************************
```

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

```
                    * The QSORT subroutine is passed (on the stack)
                    * the high and low addresses of an area of bytes
                    * to be sorted.
                    ************************************************
000494 226F 0008    QSORT           MOVE.L  8[A7],A1   ; A1 = high index.
000498 206F 0004                    MOVE.L  4[A7],A0   ; A0 = low index.
                    ;-----------------------------------------------
                    ; The following section splits the area to be sorted
                    ; into two areas.  QSORT will be called to sort each
                    ; of these smaller areas.
                    ;
                    ; If high index is less than low index, then sort
                    ; is done.
                    ;-----------------------------------------------
00049C B3C8         OVER            CMPA.L  A0,A1
00049E 6F00 0054                    BLE     DONE
                    ;-----------------------------------------------
                    ; D2 = dividing value (from low index).
                    ;-----------------------------------------------
0004A2 1410                         MOVE.B  [A0],D2
                    ;-----------------------------------------------
                    ; (Increment allows DEC_HIGH loop to work first
                    ; time through.)
                    ;-----------------------------------------------
0004A4 5289                         ADDQ.L  #1,A1
                    ;-----------------------------------------------
                    ; Move low index up until it points to a value
                    ; greater than the dividing value.
                    ;-----------------------------------------------
0004A6 5288         INC_LOW         ADDQ.L  #1,A0
0004A8 B410                         CMP.B   [A0],D2
0004AA 6F00 000A                    BLE     DEC_HIGH
0004AE B3C8                         CMPA.L  A0,A1
0004B0 6D00 0018                    BLT     OUT
0004B4 60F0                         BRA.B   INC_LOW
                    ;-----------------------------------------------
                    ; Move high index down until it points to a value
                    ; less than or equal to the dividing value.
                    ;-----------------------------------------------
0004B6 5389         DEC_HIGH        SUBQ.L  #1,A1
0004B8 B411                         CMP.B   [A1],D2
0004BA 6DFA                         BLT     DEC_HIGH
                    ;-----------------------------------------------
                    ; If high index is less than or equal to low index,
                    ; the area is split; do not swap values.
                    ;-----------------------------------------------
0004BC B3C8                         CMPA.L  A0,A1
0004BE 6D00 000A                    BLT     OUT
                    ;-----------------------------------------------
                    ; If high index is greater than low index, swap
                    ; values and move indexes again.
                    ;-----------------------------------------------
0004C2 1610                         MOVE.B  [A0],D3
0004C4 1091                         MOVE.B  [A1],[A0]
0004C6 1283                         MOVE.B  D3,[A1]
0004C8 60DC                         BRA.B   INC_LOW
```

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

```
                        159 ;-----------------------------------------------
                        160 ; A0 = low address (needed to swap dividing value).
                        161 ;-----------------------------------------------
 0004CA 206F 0004       162 OUT             MOVE.L  4[A7],A0)
                        163 ;-----------------------------------------------
                        164 ; Swap dividing value and high index value.
                        165 ;-----------------------------------------------
 0004CE 1091            166                 MOVE.B  [A1],[A0]
 0004D0 1282            167                 MOVE.B  D2,[A1]
                        168 ;-----------------------------------------------
                        169 ; The area is now split into two smaller areas.
                        170 ; The last high index value is the middle of the
                        171 ; two areas.  The high and low addresses for the
                        172 ; second QSORT call are pushed first.
                        173 ;-----------------------------------------------
 0004D2 2F2F 0008       174                 MOVE.L  8[A7],-[A7]  ; Push high.
 0004D6 5289            175                 ADDQ.L  #1,A1
 0004D8 2F09            176                 MOVE.L  A1,-[A7]  ; Push middle + 1.
 0004DA 5589            177                 SUBQ.L  #2,A1
 0004DC 2F09            178                 MOVE.L  A1,-[A7]  ; Push middle - 1.
 0004DE 2F08            179                 MOVE.L  A0,-[A7]  ; Push low.
 0004E0 6100 FFB2       180                 BSR.W   QSORT
                        181 ;-----------------------------------------------
                        182 ; Increment stack pointer after call.
                        183 ;-----------------------------------------------
 0004E4 DFFC            184                 ADDA.L  #8,A7
 0004E6 00000008
 0004EA 6100 FFA8       185                 BSR.W   QSORT
 0004EE DFFC            186                 ADDA.L  #8,A7
 0004F0 00000008
 0004F4 4E75            187 DONE            RTS
                        188
                        189 ***********************************************
                        190 * Random numbers written to this area.
                        191 ***********************************************
                        192                 ORG     500H
 000500                 193 RESULTS         DS.B    100H
                        194
                        195 ***********************************************
                        196 * Variable used in RAND subroutine and stack area.
                        197 ***********************************************
                        198                 ORG     600H
 000600 0000 0001       199 RAND_SEED       DC.L    1
 000604                 200                 DS.W    2FDH
 000BFE                 201 STACK           DS.W    1
            É   202                 END     START

Errors=    0


CROSS REFERENCE TABLE FILE: C:\MNL\ODY\ANLY\PCI\SRC\68K\SRND.S
LINE#   SYMBOL         TYPE     REFERENCES

   21   AGAIN           A      58,    76
   46   CALLER_0        A      36
   48   CALLER_1        A      37
```

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

```
  50   CALLER_2          A      39
  52   CALLER_3          A      40
 142   DEC_HIGH          A     134,    144
 187   DONE              A     118
 132   INC_LOW           A     137,    158
 162   OUT               A     136,    150
 117   OVER              A
 107   QSORT             A      74,    180,    185
  83   RAND              A      26
 199   RAND_SEED         A      83,     88
 193   RESULTS           A      68,     69,     99
 201   STACK             A      13
  13   START             A     202
  53   TEST              A      47,     49,     51
  38   TWO_THREE         A      34
  99   WRITE_NUMBER      A      46,     48,     50,     52
  35   ZERO_ONE          A      33
```

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

Mapping Memory for the Sample Program

The program, destination, and stack areas of the sample program were ORGed at addresses 400H, 500H, and 600H, respectively. Therefore, map the range from 400H through 0BFFH to emulation memory before loading the program. Select the memory mapper configuration by either by using the left and right arrow keys to highlight command options and pressing **Enter**, or by typing the first letter of the command options.

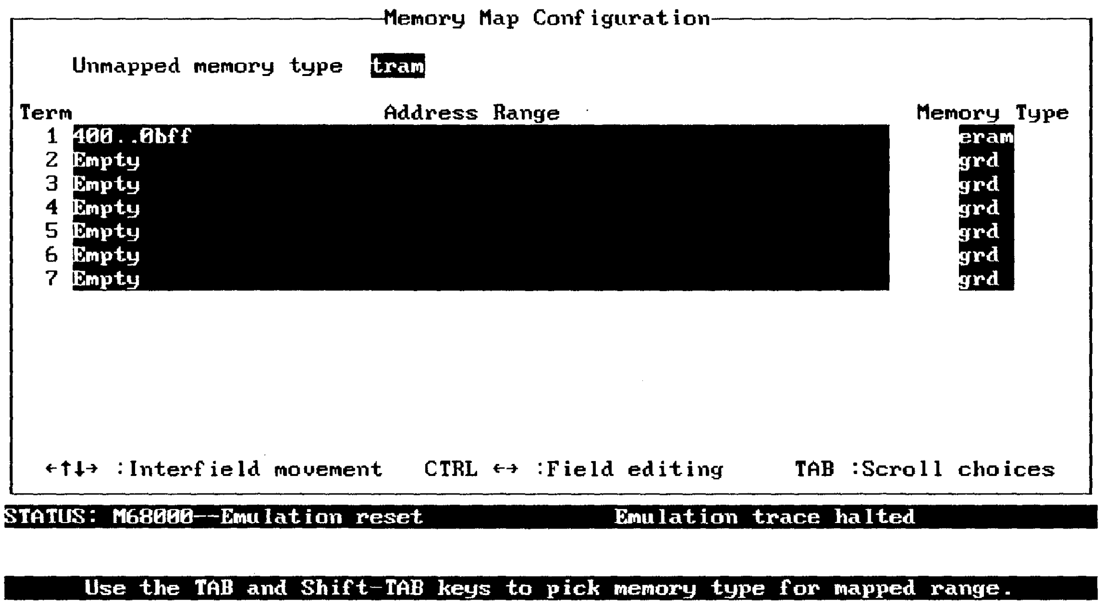
```
Config, Map, Modify
```

Using the arrow keys, move the cursor to the "address range" field of term 1. Enter:

```
400..0bff
```

Move the cursor to the "memory type" field of term 1, and press the TAB key to select the **eram** (emulation RAM) type. To save your memory map, use the **Enter** key to exit the field in the lower right corner. (The **End** key on Vectra keyboards moves the cursor directly to the last field.) The memory configuration display is shown in figure 3-3.

Memory mapping is described in more detail in your *PC Interface: Emulator User's Guide*.



```
                         Memory Map Configuration

   Unmapped memory type  tram

Term                     Address Range                                Memory Type
  1 400..0bff                                                            eram
  2 Empty                                                                grd
  3 Empty                                                                grd
  4 Empty                                                                grd
  5 Empty                                                                grd
  6 Empty                                                                grd
  7 Empty                                                                grd

  ←↑↓→ :Interfield movement   CTRL ←→ :Field editing     TAB :Scroll choices
STATUS: M68000--Emulation reset                 Emulation trace halted

     Use the TAB and Shift-TAB keys to pick memory type for mapped range.
```

Figure 3-3. Memory Map Configuration Display

Loading the Sample Program

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

```
Memory, Load
```

Move the cursor to the "Format" field and select the appropriate format (HP64000 in this example). Select "yes" in the field that forces the absolute file to be read. Now move the cursor to the "Absolute filename" field and enter the name of your symbol file (SRND.L in this example). The options available when loading absolute files into memory are described in more detail in the *PC Interface: User's Reference* manual.

Set Up the Stack Pointer

The 68000 emulator requires you to set up the system stack pointer before you can run the program. To set up the system stack pointer, you must first break into the monitor.

```
Processor, Break
```

To modify the system stack pointer, select:

```
Register, Modify
```

Use the **Tab** key to select register **ssp**, press **Enter**, type in the address 0BFEH, and press **Enter** once again.

Running the Sample Program

To start the emulator executing the sample program, select:

```
Processor, Go, Address
```

Type in the module and symbol names associated with the sample program's starting address, SRND.S:START. The status line will show that the emulator is "Running user program".

When entering the name of a local symbol for the first time, you need to include the module name. This causes that module to become the default module, and the next time you refer to a symbol from that module, you do not have to include the module name.

Activating the Trace Window

The examples which follow show results in the "zoomed" trace window. To activate this window, select:

```
Window, Zoom
```

Then either type in, or use the **Tab** key to select "Analysis", and press **Enter**.

Specifying a Simple Trigger

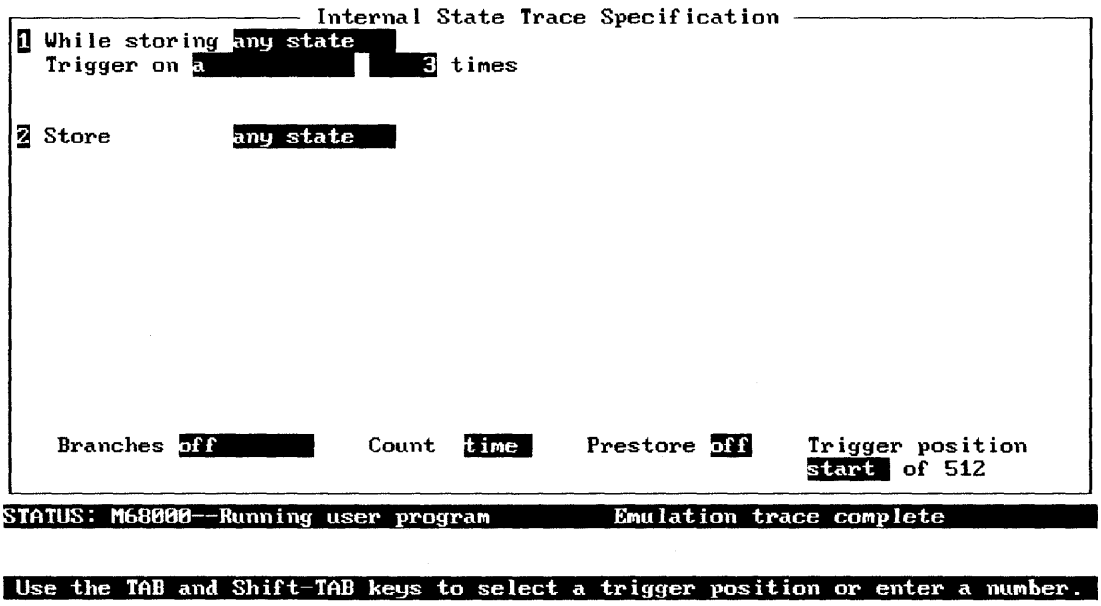
Suppose you want to look at the execution of the sample program after the AGAIN address, but only after it has occurred three times (in other words, after the program has executed its complete loop three times). To do this, select:

```
Analysis, Trace, Modify
```

Modify the trace specification as shown in figure 3-4. (To access the "patterns and expressions" screen, press the **Enter** key while the cursor is in the "trigger on" field.)

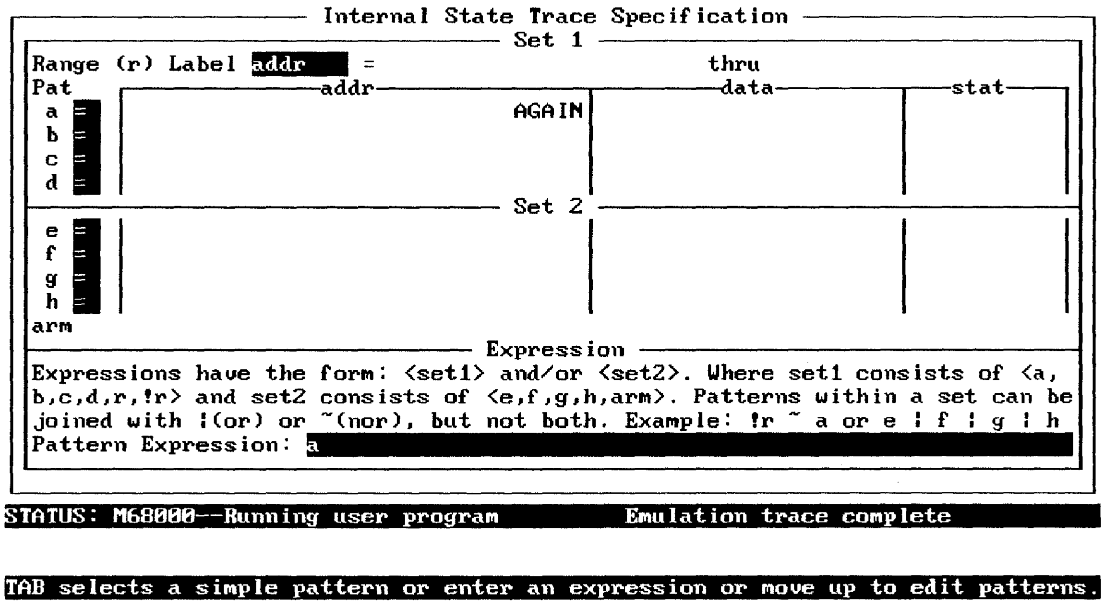
Notice that the address symbol AGAIN is assigned to pattern "a" and an occurrence count of "3" is specified.

To save the new specification, use the **Enter** key to exit out of the field in the lower right corner.



```
──────────────── Internal State Trace Specification ────────────────
1 While storing any state
  Trigger on a                3 times

2 Store         any state









   Branches off         Count time     Prestore off     Trigger position
                                                        start  of 512
STATUS: M68000--Running user program         Emulation trace complete

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 3-4. Simple Trigger Specification



```
──────────────── Internal State Trace Specification ────────────────
──────────────────────────── Set 1 ─────────────────────────────
Range (r) Label addr    =                          thru
Pat      ┌──────────────addr──────────────┬─────────data─────────┬───stat───┐
 a =     │                           AGAIN│                      │          │
 b =     │                                │                      │          │
 c =     │                                │                      │          │
 d =     │                                │                      │          │
──────────────────────────── Set 2 ─────────────────────────────
 e =     │                                │                      │          │
 f =     │                                │                      │          │
 g =     │                                │                      │          │
 h =     │                                │                      │          │
arm
────────────────────────── Expression ──────────────────────────
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h
Pattern Expression: a

STATUS: M68000--Running user program         Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-4. Simple Trigger Specification (Cont'd)

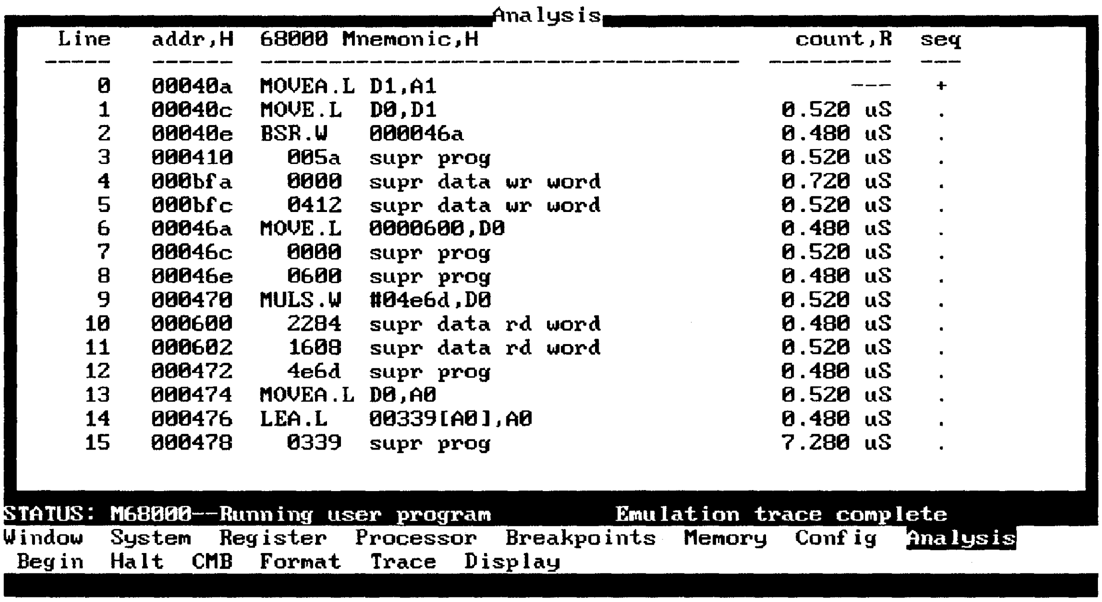
To begin the trace, select:

`Analysis, Begin`

After the "Trace complete" message is shown on the status line, display the trace by selecting:

`Analysis, Display`

Type the first state that can be displayed into the starting state field and press **Enter**. Move the cursor to the ending state field, enter the starting state plus 15, and press **Enter**. The resulting trace is similar to the trace shown in the following display.

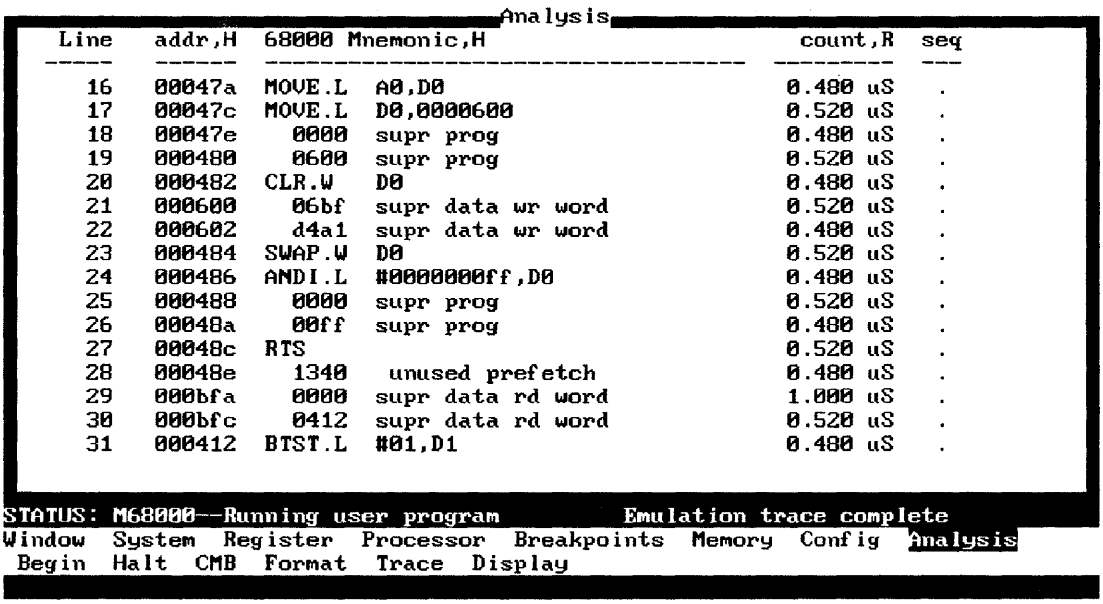

```
                                        Analysis
  Line    addr,H   68000 Mnemonic,H                            count,R   seq
 -----   ------   -----------------------------------    ---------   ---
     0   00040a   MOVEA.L D1,A1                                ---     +
     1   00040c   MOVE.L  D0,D1                          0.520 uS     .
     2   00040e   BSR.W   00046a                          0.480 uS     .
     3   000410     005a  supr prog                       0.520 uS     .
     4   000bfa     0000  supr data wr word               0.720 uS     .
     5   000bfc     0412  supr data wr word               0.520 uS     .
     6   00046a   MOVE.L  0000600,D0                      0.480 uS     .
     7   00046c     0000  supr prog                       0.520 uS     .
     8   00046e     0600  supr prog                       0.480 uS     .
     9   000470   MULS.W  #04e6d,D0                       0.520 uS     .
    10   000600     2284  supr data rd word               0.480 uS     .
    11   000602     1608  supr data rd word               0.520 uS     .
    12   000472     4e6d  supr prog                       0.480 uS     .
    13   000474   MOVEA.L D0,A0                           0.520 uS     .
    14   000476   LEA.L   00339[A0],A0                    0.480 uS     .
    15   000478     0339  supr prog                       7.280 uS     .

STATUS: M68000--Running user program          Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

In the trace list above, line 0 shows the beginning of the program loop and line 2 shows the call of the RAND subroutine. The disassembled mnemonics on line 6 shows the first instruction executed in the RAND subroutine.

To display more lines of the trace, select:

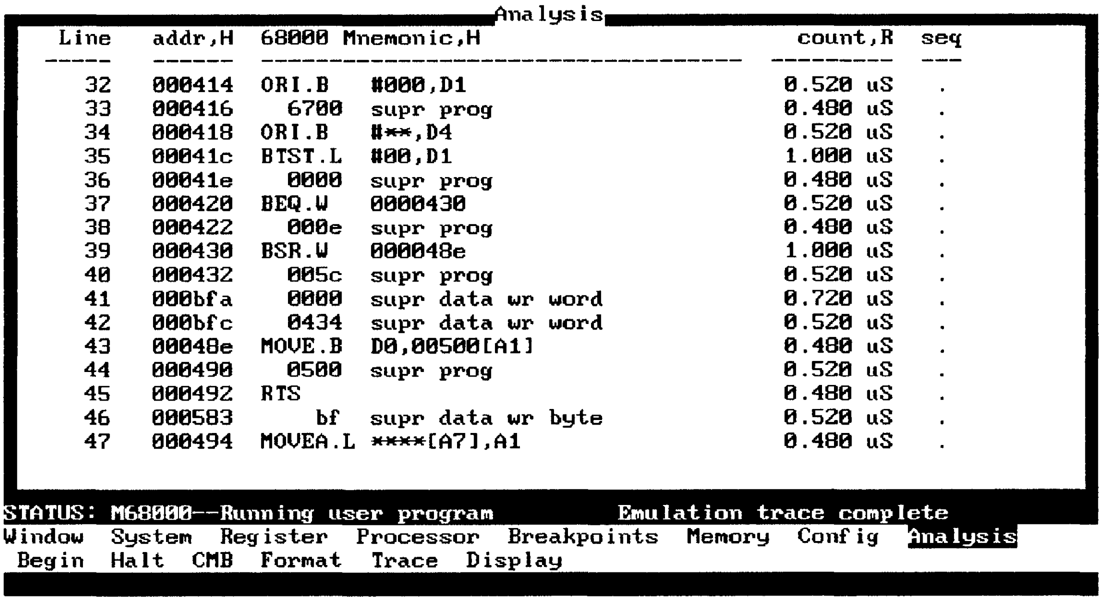
```
Analysis Display
```

Notice that the starting and ending lines are incremented by the number of lines given in the last display command. Press **Enter** two times to select these values.



```
                                    Analysis
 Line   addr,H  68000 Mnemonic,H                          count,R  seq
 -----  ------  --------------------------------   ---------  ---
   16   00047a  MOVE.L  A0,D0                              0.480 uS   .
   17   00047c  MOVE.L  D0,0000600                         0.520 uS   .
   18   00047e    0000  supr prog                          0.480 uS   .
   19   000480    0600  supr prog                          0.520 uS   .
   20   000482  CLR.W   D0                                 0.480 uS   .
   21   000600    06bf  supr data wr word                  0.520 uS   .
   22   000602    d4a1  supr data wr word                  0.480 uS   .
   23   000484  SWAP.W  D0                                 0.520 uS   .
   24   000486  ANDI.L  #000000ff,D0                       0.480 uS   .
   25   000488    0000  supr prog                          0.520 uS   .
   26   00048a    00ff  supr prog                          0.480 uS   .
   27   00048c  RTS                                        0.520 uS   .
   28   00048e    1340   unused prefetch                   0.480 uS   .
   29   000bfa    0000  supr data rd word                  1.000 uS   .
   30   000bfc    0412  supr data rd word                  0.520 uS   .
   31   000412  BTST.L  #01,D1                             0.480 uS   .

STATUS: M68000--Running user program        Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

In the display above, you see remaining execution of the instructions in the RAND subroutine. To display more lines in the trace, press **CTRL-R** to repeat the previous command. Since you pressed **Enter** in the previous command to select the automatically incremented starting and ending line numbers, the automatically incremented line numbers are selected again, and the next 16 lines of the trace are displayed.



```
                                   Analysis
  Line    addr,H   68000 Mnemonic,H                               count,R   seq
  -----   ------   -----------------------------------------   ---------   ---
    32    000414   ORI.B    #000,D1                              0.520 uS    .
    33    000416     6700   supr prog                            0.480 uS    .
    34    000418   ORI.B    #**,D4                               0.520 uS    .
    35    00041c   BTST.L   #00,D1                               1.000 uS    .
    36    00041e     0000   supr prog                            0.480 uS    .
    37    000420   BEQ.W    0000430                              0.520 uS    .
    38    000422     000e   supr prog                            0.480 uS    .
    39    000430   BSR.W    000048e                              1.000 uS    .
    40    000432     005c   supr prog                            0.520 uS    .
    41    000bfa     0000   supr data wr word                    0.720 uS    .
    42    000bfc     0434   supr data wr word                    0.520 uS    .
    43    00048e   MOVE.B   D0,00500[A1]                         0.480 uS    .
    44    000490     0500   supr prog                            0.520 uS    .
    45    000492   RTS                                           0.480 uS    .
    46    000583       bf   supr data wr byte                    0.520 uS    .
    47    000494   MOVEA.L  ****[A7],A1                          0.480 uS    .

STATUS: M68000--Running user program          Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

In the trace list above you see the instructions that write the random number to the RESULTS area.

Using Storage Qualifiers

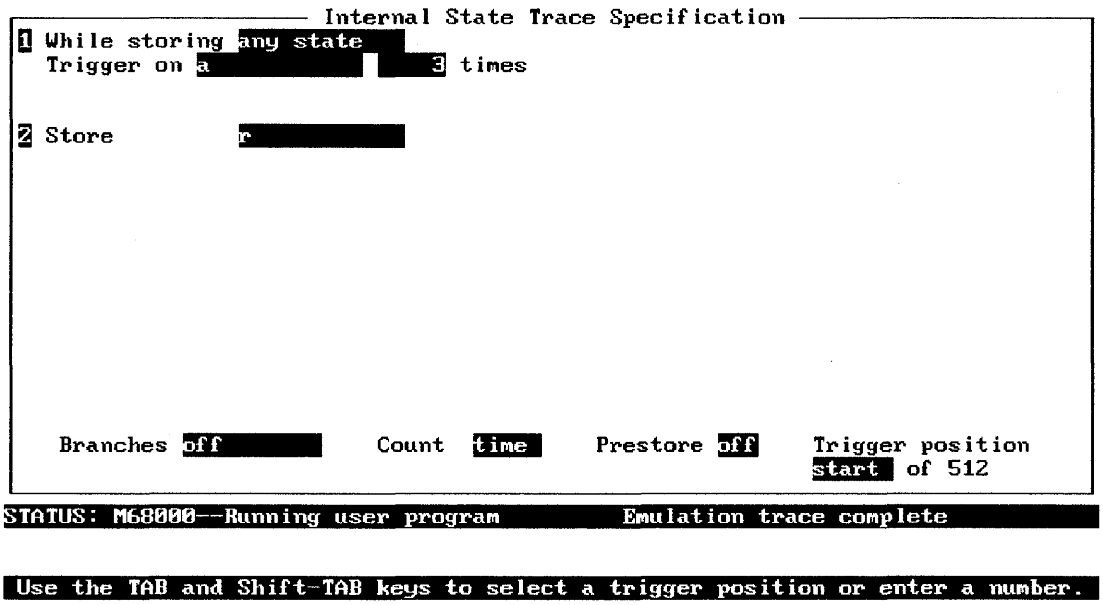
In the last example, all captured states were stored. To modify the trace specification of the previous example so that only the states which write random numbers to the RESULTS area are stored, select:

Analysis, Trace, Modify

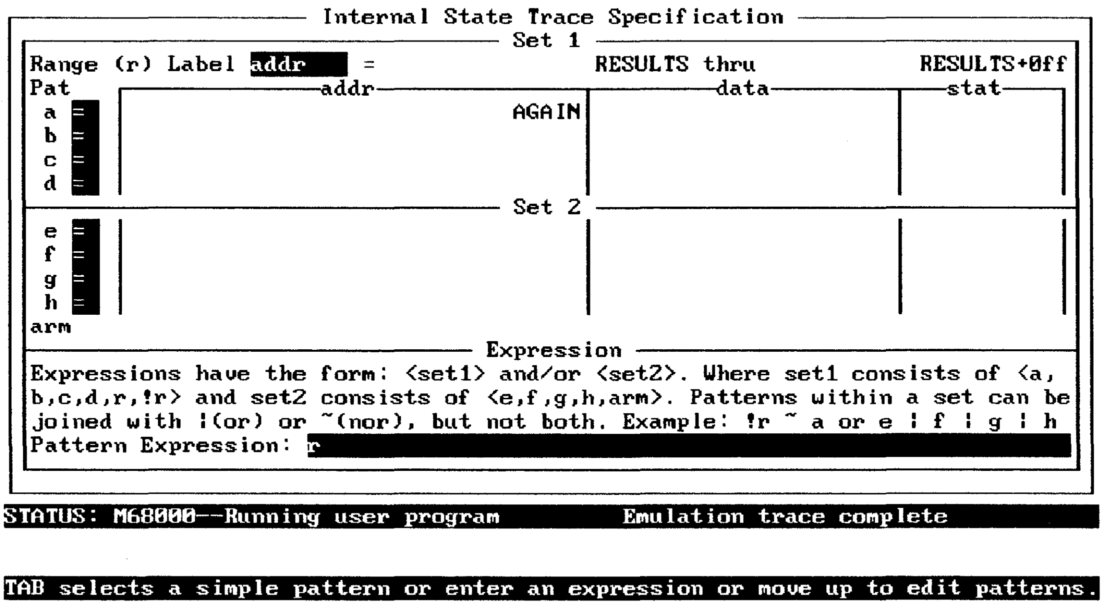
Modify the trace specification as shown in figure 3-5.

To begin the trace, select:

Analysis, Begin



```
                  Internal State Trace Specification
1 While storing any state
  Trigger on a                  3 times


2 Store         r










  Branches off         Count time     Prestore off    Trigger position
                                                      start  of 512
STATUS: M68000--Running user program        Emulation trace complete


 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```



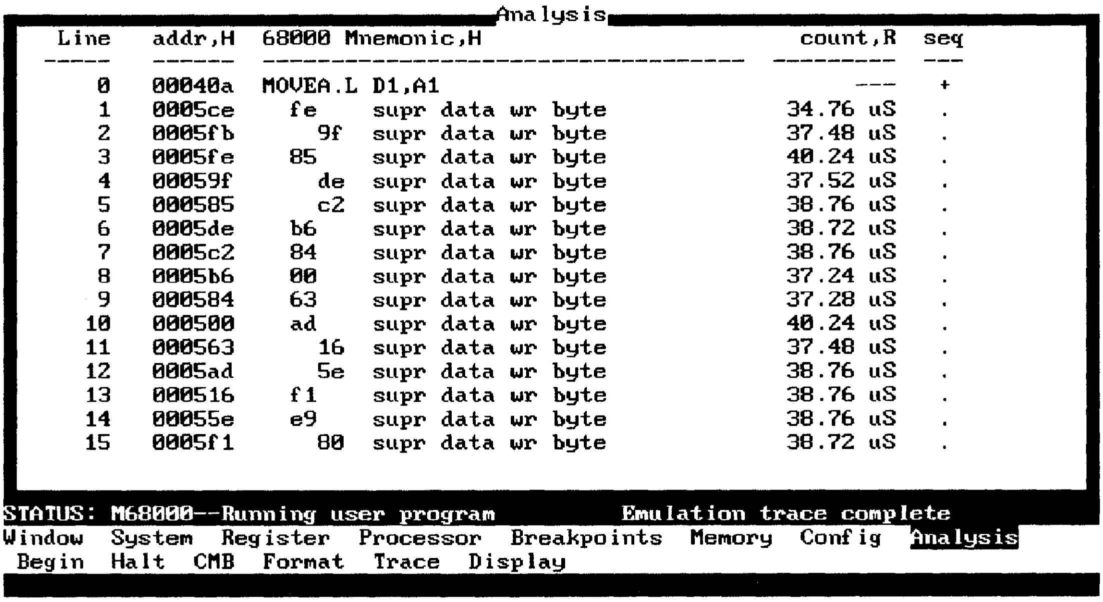
```
                  Internal State Trace Specification
                                  Set 1
Range (r) Label addr   =              RESULTS thru            RESULTS+0ff
Pat                  addr                     data              stat
 a =                          AGAIN
 b =
 c =
 d =
                                  Set 2
 e =
 f =
 g =
 h =
arm
                                Expression
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with |(or) or ~(nor), but not both. Example: !r ~ a or e | f | g | h
Pattern Expression: r
STATUS: M68000--Running user program        Emulation trace complete


TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-5. Storage Qualifier Specification

After the "Trace complete" message is shown on the status line, display the trace by selecting:

```
Analysis, Display
```

Type the number of the starting state plus 15 into the ending state field, and press **Enter**. The resulting trace will be similar to the display shown below.



```
                                   Analysis
  Line    addr,H   68000 Mnemonic,H                             count,R   seq
  -----   ------   ----------------------------------          ---------  ---
     0    00040a   MOVEA.L D1,A1                                     ---    +
     1    0005ce    fe     supr data wr byte                    34.76 uS    .
     2    0005fb      9f   supr data wr byte                    37.48 uS    .
     3    0005fe   85      supr data wr byte                    40.24 uS    .
     4    00059f      de   supr data wr byte                    37.52 uS    .
     5    000585      c2   supr data wr byte                    38.76 uS    .
     6    0005de   b6      supr data wr byte                    38.72 uS    .
     7    0005c2   84      supr data wr byte                    38.76 uS    .
     8    0005b6   00      supr data wr byte                    37.24 uS    .
     9    000584   63      supr data wr byte                    37.28 uS    .
    10    000500   ad      supr data wr byte                    40.24 uS    .
    11    000563      16   supr data wr byte                    37.48 uS    .
    12    0005ad      5e   supr data wr byte                    38.76 uS    .
    13    000516   f1      supr data wr byte                    38.76 uS    .
    14    00055e   e9      supr data wr byte                    38.76 uS    .
    15    0005f1      80   supr data wr byte                    38.72 uS    .

STATUS: M68000--Running user program          Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

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

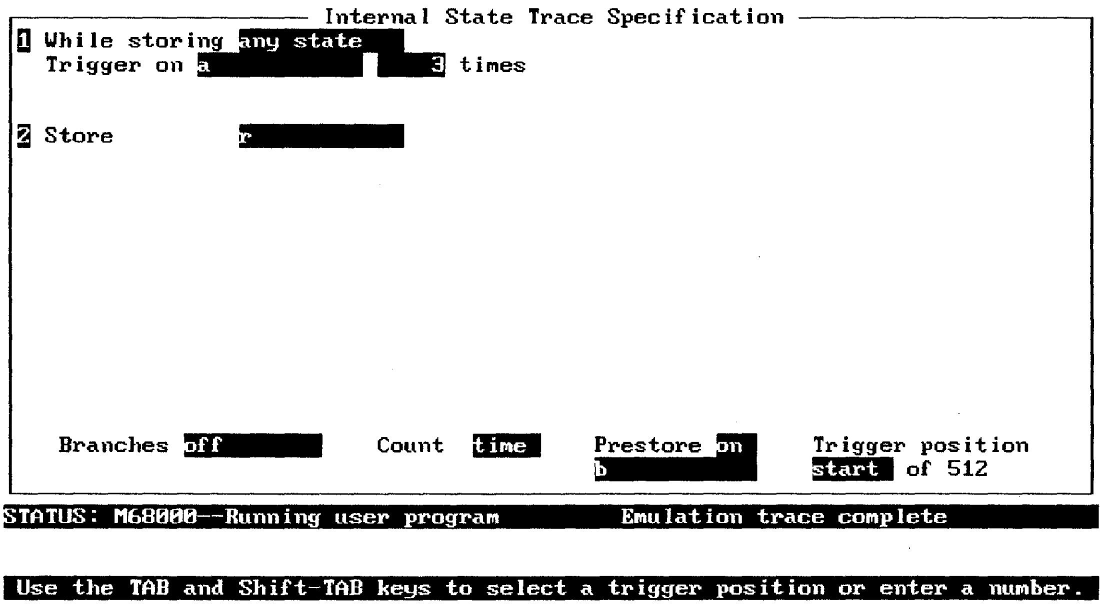
This trace shows that the last two hex digits of the address in the RESULTS area are the same as the random number which gets written two states earlier (see the data in the "mnemonic" column of the trace list). This is expected because the sample program writes the current random number using the second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

Using Trace Prestore

Prestore lets you to save up to two states which precede a normal store state. The following example uses a prestore qualifier to show which caller of WRITE_NUMBER corresponds to each value written to the RESULTS area. Because you know 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). First of all, select:

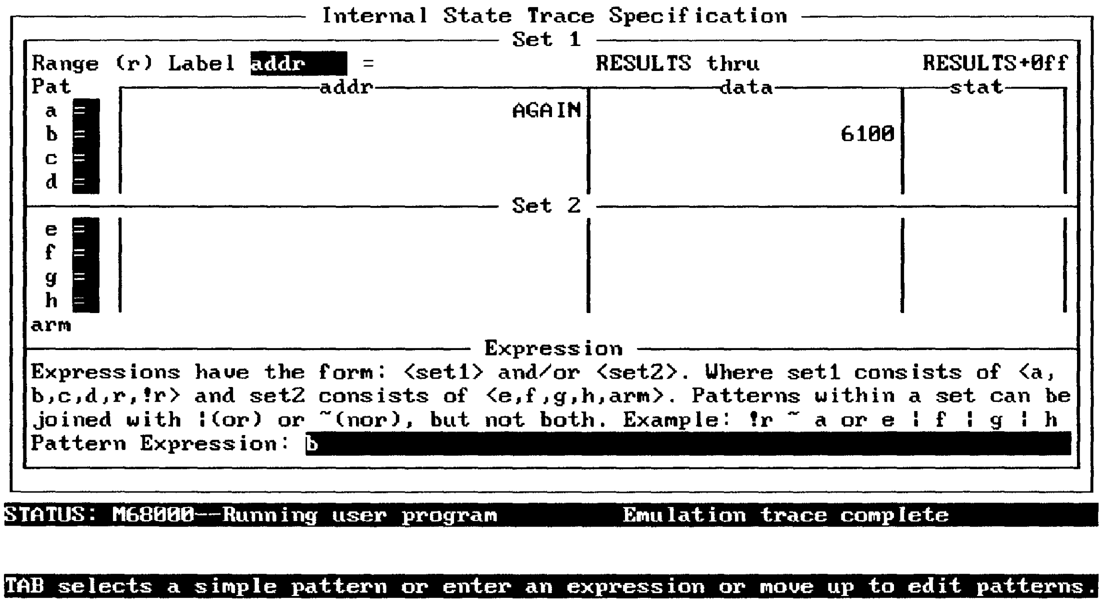
```
Analysis, Trace, Modify
```

Modify the trace specification as shown in figure 3-6.



```
                     Internal State Trace Specification
1 While storing any state
  Trigger on a                3 times

2 Store         r





   Branches off         Count  time       Prestore on       Trigger position
                                          b                 start  of 512
STATUS: M68000--Running user program          Emulation trace complete

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 3-6. Prestore Qualifier Specification



```
                    Internal State Trace Specification
                                Set 1
Range (r) Label addr    =                   RESULTS thru         RESULTS+0ff
Pat             addr                          data               stat
 a =                              AGAIN
 b =                                                  6100
 c =
 d =
                                Set 2
 e =
 f =
 g =
 h =
arm
                              Expression
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h
Pattern Expression: b

STATUS: M68000--Running user program          Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-6. Prestore Qualifier Specification (Cont'd)

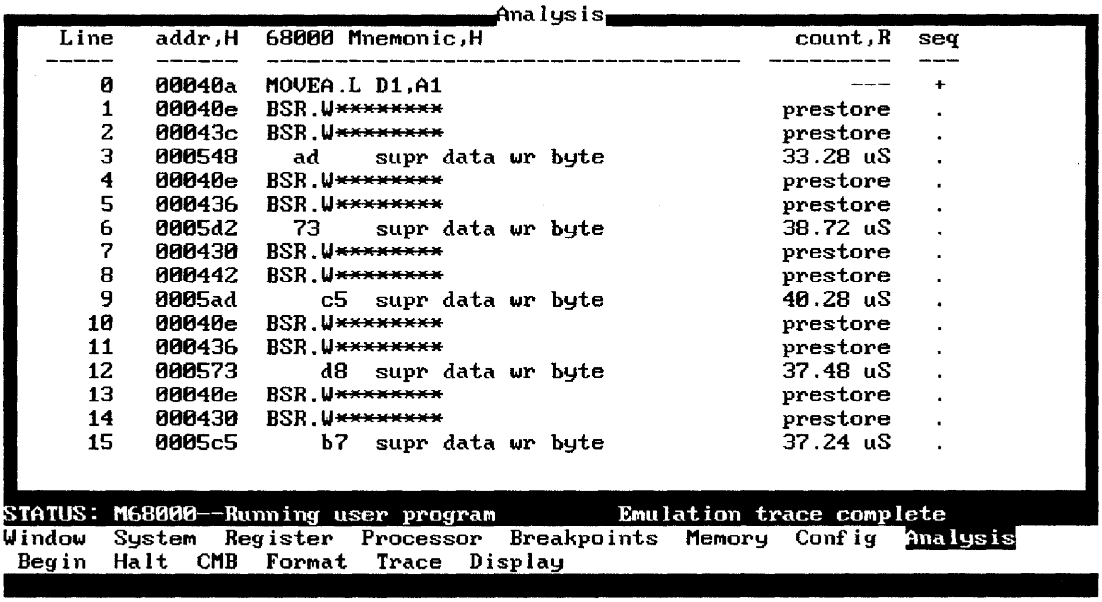
To begin the trace, select:

`Analysis, Begin`

After the "Trace complete" message is shown on the status line, display the trace by selecting:

`Analysis, Display`

Type the number of the starting state plus 15 into the ending state field, and press **Enter**. The resulting trace will be similar to the display which follows.



```
                                         Analysis
  Line    addr,H   68000 Mnemonic,H                           count,R   seq
 -----    ------   ----------------------------------         ---------  ---
    0     00040a   MOVEA.L D1,A1                                    ---    +
    1     00040e   BSR.W*********                             prestore    .
    2     00043c   BSR.W*********                             prestore    .
    3     000548     ad     supr data wr byte                 33.28 uS    .
    4     00040e   BSR.W*********                             prestore    .
    5     000436   BSR.W*********                             prestore    .
    6     0005d2     73     supr data wr byte                 38.72 uS    .
    7     000430   BSR.W*********                             prestore    .
    8     000442   BSR.W*********                             prestore    .
    9     0005ad       c5   supr data wr byte                 40.28 uS    .
   10     00040e   BSR.W*********                             prestore    .
   11     000436   BSR.W*********                             prestore    .
   12     000573       d8   supr data wr byte                 37.48 uS    .
   13     00040e   BSR.W*********                             prestore    .
   14     000430   BSR.W*********                             prestore    .
   15     0005c5       b7   supr data wr byte                 37.24 uS    .

STATUS: M68000--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

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, the "prestore" string is shown in the "count" column of the trace list. Note that the time counts are relative to the previous normal storage state. Turning off the count qualifier does not turn off prestore; however, the "prestore" string cannot be seen in the "count" column of the trace list.

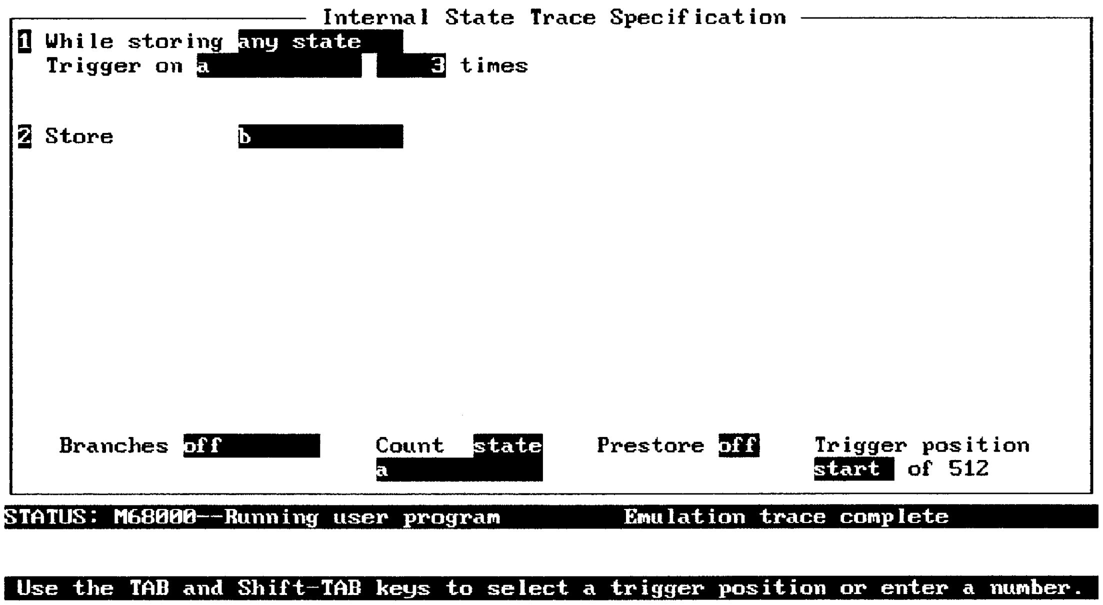
States which satisfy the prestore qualifier and the storage qualifier at the same time are stored as normal states.

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 set up the trace specification so that only writes to address 5C2H are stored; then, you can specify as a count qualifier a state which occurs once on each loop of the program. First of all, select:

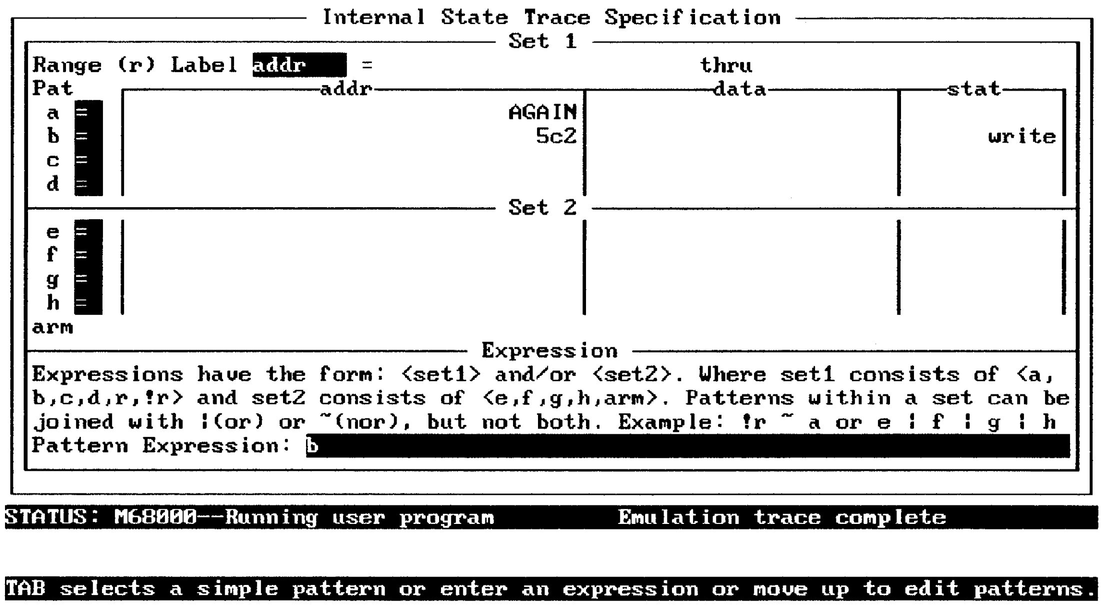
```
    Analysis, Trace, Modify
```

Modify the trace specification as shown in figure 3-7.



```
                    Internal State Trace Specification
1 While storing any state
  Trigger on a             3 times

2 Store         b




  Branches off         Count   state      Prestore off      Trigger position
                       a                                    start   of 512
STATUS: M68000--Running user program         Emulation trace complete

Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 3-7. Count Qualifier Specification



```
──────────────── Internal State Trace Specification ────────────────
┌──────────────────────────── Set 1 ──────────────────────────────┐
│Range (r) Label addr    =                   thru                 │
│Pat     ┌──────────addr──────────┬────────data────────┬──stat──┐ │
│ a =    │                   AGAIN│                    │        │ │
│ b =    │                     5c2│                    │   write│ │
│ c =    │                        │                    │        │ │
│ d =    │                        │                    │        │ │
├──────────────────────────── Set 2 ──────────────────────────────┤
│ e =    │                        │                    │        │ │
│ f =    │                        │                    │        │ │
│ g =    │                        │                    │        │ │
│ h =    │                        │                    │        │ │
│arm                                                              │
├────────────────────────── Expression ───────────────────────────┤
│Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
│b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
│joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h
│Pattern Expression: b                                            │
└─────────────────────────────────────────────────────────────────┘

STATUS: M68000--Running user program          Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-7. Count Qualifier Specification (Cont'd)

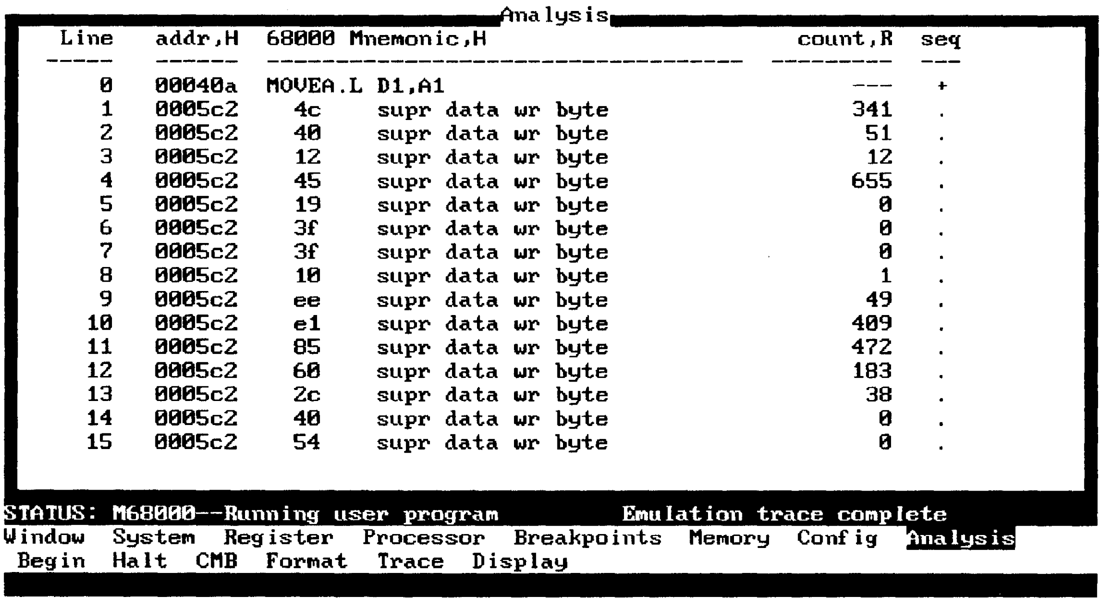
To begin the trace, select:

`Analysis, Begin`

After the "Trace complete" message is shown on the status line, display the trace by selecting:

`Analysis, Display`

Type the number of the starting state plus 15 into the ending state field, and press **Enter**. The resulting trace display will be similar to the display which follows.



```
                                        Analysis
  Line    addr,H   68000 Mnemonic,H                            count,R   seq
 -----    ------   --------------------------------          ---------   ---
     0    00040a   MOVEA.L D1,A1                                   ---    +
     1    0005c2    4c     supr data wr byte                       341    .
     2    0005c2    40     supr data wr byte                        51    .
     3    0005c2    12     supr data wr byte                        12    .
     4    0005c2    45     supr data wr byte                       655    .
     5    0005c2    19     supr data wr byte                         0    .
     6    0005c2    3f     supr data wr byte                         0    .
     7    0005c2    3f     supr data wr byte                         0    .
     8    0005c2    10     supr data wr byte                         1    .
     9    0005c2    ee     supr data wr byte                        49    .
    10    0005c2    e1     supr data wr byte                       409    .
    11    0005c2    85     supr data wr byte                       472    .
    12    0005c2    60     supr data wr byte                       183    .
    13    0005c2    2c     supr data wr byte                        38    .
    14    0005c2    40     supr data wr byte                         0    .
    15    0005c2    54     supr data wr byte                         0    .

STATUS: M68000--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

The trace listing above shows that the program executes a variable number of times for each time that a random number is written to 5C2H. Where counts of 0 are seen, the sample program is executing in the QSORT routine which sorts the values written to the RESULTS area.

Using "Restart On" Branches

Selecting "restart on" branches is useful in some situations to trace a specific combination of events. For example, CALLER_3 can be used to write any random number, but suppose you want to trace only the situation where CALLER_3 is used to write a random number to address 5C2H. You can set up the sequence levels so that the analyzer first searches for CALLER_3.

After CALLER_3 is found, the sequencer should then search for the write to address 5C2H.

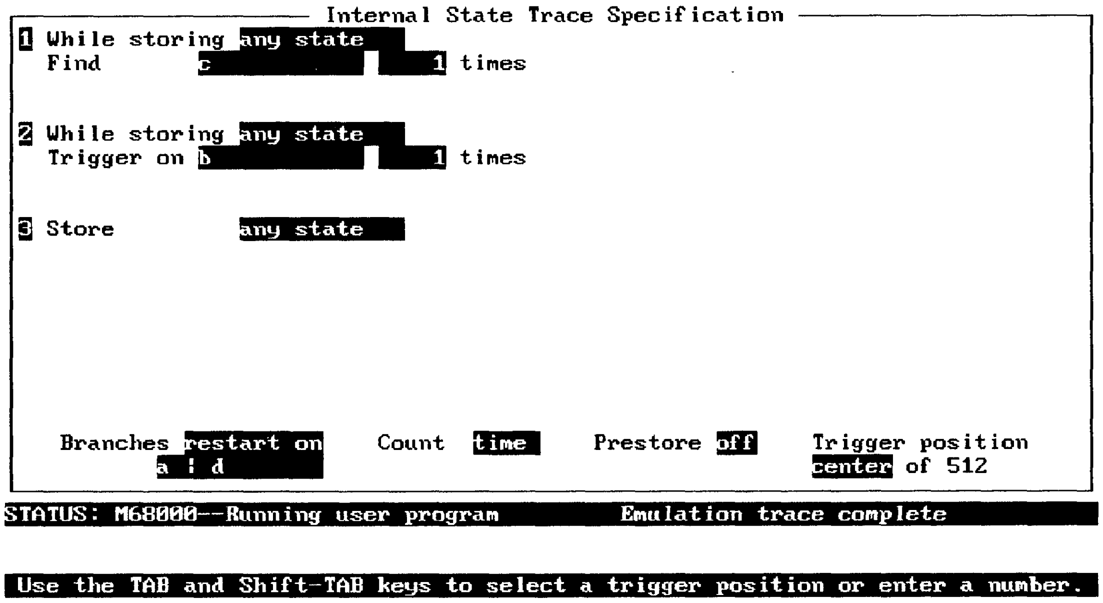
However, if the program jumps to AGAIN or calls QSORT before the write to 5C2H, you know that CALLER_3 is not used to write the random number this time, and the analyzer should start searching again from the beginning.

If the write to address 5C2H occurs before the program executes the instruction at AGAIN or some instruction in the QSORT routine, the analyzer will trigger.

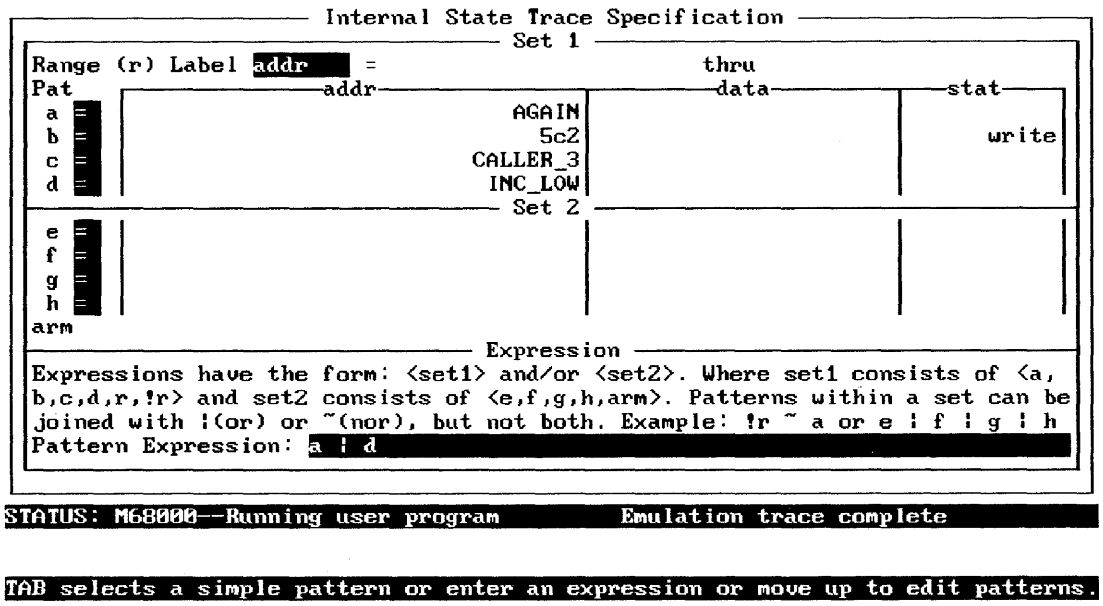
To make this specification, select:

```
Analysis, Trace, Modify
```

Modify the trace specification as shown in figure 3-8.



```
──────────────── Internal State Trace Specification ────────────────
1 While storing any state
  Find         c              1 times

2 While storing any state
  Trigger on   b              1 times

3 Store        any state




   Branches restart on     Count time     Prestore off     Trigger position
            a ! d                                           center of 512
STATUS: M68000--Running user program        Emulation trace complete

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```



```
──────────────── Internal State Trace Specification ────────────────
──────────────────────────── Set 1 ──────────────────────────────
Range (r) Label addr   =                    thru
Pat   ─────────addr─────────────┬──────────data──────────┬───stat───
 a =                     AGAIN  │                        │
 b =                       5c2  │                        │    write
 c =                  CALLER_3  │                        │
 d =                   INC_LOW  │                        │
──────────────────────────── Set 2 ──────────────────────────────
 e =                            │                        │
 f =                            │                        │
 g =                            │                        │
 h =                            │                        │
arm
────────────────────────── Expression ───────────────────────────
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h
Pattern Expression: a ! d
STATUS: M68000--Running user program        Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-8. Branches "Restart On" Specification

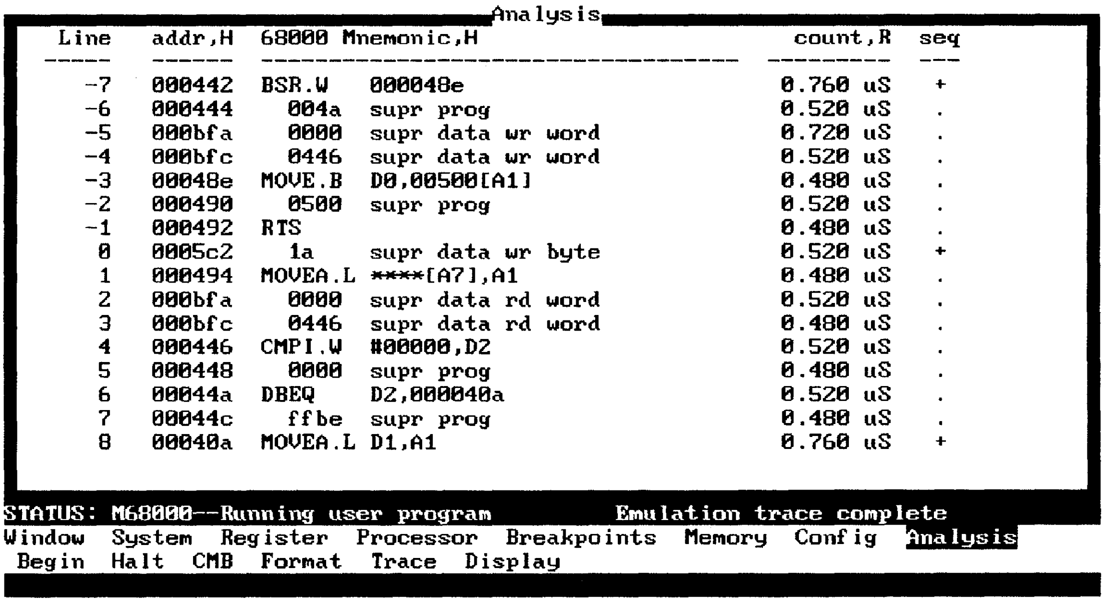
To begin the trace, select:

```
Analysis, Begin
```

After the "Trace complete" message is shown on the status line, display the trace by selecting:

```
Analysis, Display
```

Type -7 into the starting state field and 8 into the ending state field, and press **Enter**. The resulting trace will be similar to the display which follows.



```
                                   Analysis
 Line    addr,H   68000 Mnemonic,H                              count,R   seq
 -----   ------   ---------------------------------------      ---------  ---
   -7    000442   BSR.W    00048e                              0.760 uS   +
   -6    000444     004a   supr prog                           0.520 uS   .
   -5    000bfa     0000   supr data wr word                   0.720 uS   .
   -4    000bfc     0446   supr data wr word                   0.520 uS   .
   -3    00048e   MOVE.B   D0,00500[A1]                        0.480 uS   .
   -2    000490     0500   supr prog                           0.520 uS   .
   -1    000492   RTS                                          0.480 uS   .
    0    0005c2     1a     supr data wr byte                   0.520 uS   +
    1    000494   MOVEA.L  ****[A7],A1                         0.480 uS   .
    2    000bfa     0000   supr data rd word                   0.520 uS   .
    3    000bfc     0446   supr data rd word                   0.480 uS   .
    4    000446   CMPI.W   #00000,D2                           0.520 uS   .
    5    000448     0000   supr prog                           0.480 uS   .
    6    00044a   DBEQ     D2,00040a                           0.520 uS   .
    7    00044c     ffbe   supr prog                           0.480 uS   .
    8    00040a   MOVEA.L  D1,A1                               0.760 uS   +

STATUS: M68000--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

Using Branches "Per Level"

Selecting branches "per level" in the trace specification gives you access to the full power and flexibility of the analyzer. Branches per level are used when you need to trace more complicated conditions.

There is a "bug" in this chapter's sample program. Occasionally, after the 256 bytes of the RESULTS area have been sorted by the QSORT subroutine, you will see a byte out of order in the last eight or so bytes of the area. You can see what happens by setting software breakpoints before and after the QSORT routine is executed, running the program, and displaying memory.

First of all, break to the monitor by selecting:

```
Processor, Break
```

Now, define a keystroke sequence macro which will:

- Set a breakpoint at the beginning of the QSORT subroutine.
- Run the program until that breakpoint is hit (so you know the contents in the RESULTS area are about to be sorted).
- Set another breakpoint at the AGAIN address.
- Run the program until the AGAIN address is hit (the contents of the RESULTS area should be sorted at this point).
- Display the contents of the results area.

To enter a keystroke macro, select:

```
Config, Key_Macro
```

The cursor is now in the field which defines the key that executes the keystroke macro. Use the **Tab** key to select F3. Leave **ESC** as the key used to quit the keystroke sequence, and position the cursor to the keystroke sequence field and type:

```
bssQSORT
```

and press **Enter**. Now type:

```
pgpbssAGAIN
```

and press **Enter**. Continue typing:

```
pgpmdbRESULTS..RESULTS+0ffh
```

and press **Enter**. Terminate the kestroke macro by pressing **ESC**.

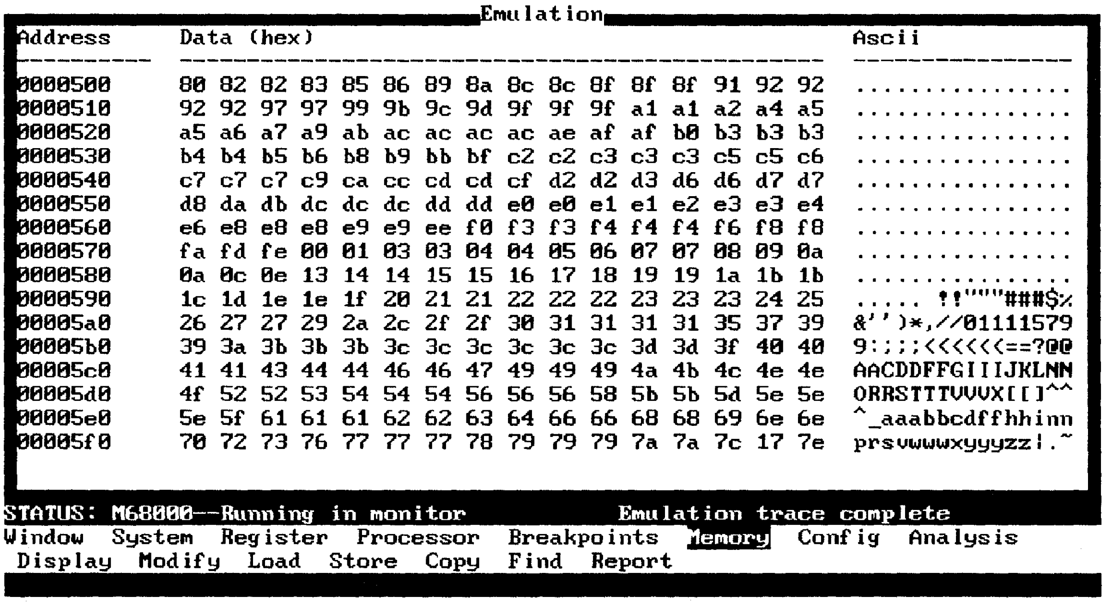
Before executing the F3 keystroke macro, define breakpoints at QSORT and AGAIN by selecting:

```
Breakpoints, Add
```

Type in "QSORT;AGAIN" into the breakpoint addresses field, and press **Enter**. When you add breakpoints, they are also set. Because breakpoints should be cleared when you execute the keystroke macro, select:

```
Breakpoints, Clear, All
```

Now, execute the keystroke macro by pressing F3, and press CTRL-Z to zoom the emulation memory display window.



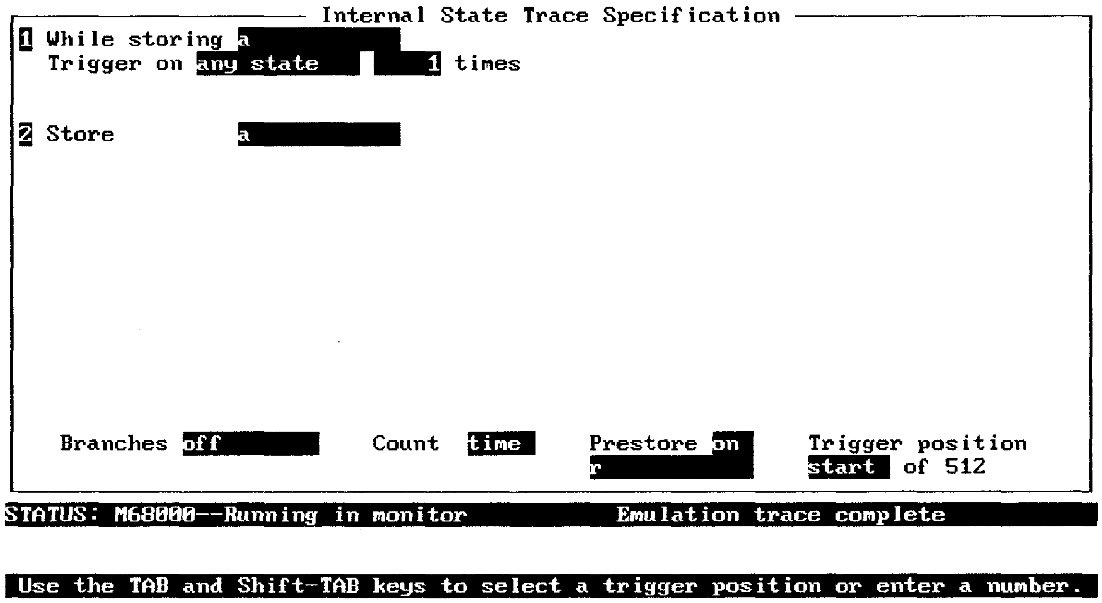
```
                                  Emulation
Address        Data (hex)                                         Ascii
---------      ----------------------------------------------     ----------------
00000500       80 82 82 83 85 86 89 8a 8c 8c 8f 8f 8f 91 92 92    ................
00000510       92 92 97 97 99 9b 9c 9d 9f 9f 9f a1 a1 a2 a4 a5    ................
00000520       a5 a6 a7 a9 ab ac ac ac ac ae af af b0 b3 b3 b3    ................
00000530       b4 b4 b5 b6 b8 b9 bb bf c2 c2 c3 c3 c3 c5 c5 c6    ................
00000540       c7 c7 c7 c9 ca cc cd cd cf d2 d2 d3 d6 d6 d7 d7    ................
00000550       d8 da db dc dc dc dd dd e0 e0 e1 e1 e2 e3 e3 e4    ................
00000560       e6 e8 e8 e8 e9 e9 ee f0 f3 f3 f4 f4 f4 f6 f8 f8    ................
00000570       fa fd fe 00 01 03 03 04 04 05 06 07 07 08 09 0a    ................
00000580       0a 0c 0e 13 14 14 15 15 16 17 18 19 19 1a 1b 1b    ................
00000590       1c 1d 1e 1e 1f 20 21 21 22 22 22 23 23 23 24 25    ..... !!"""###$%
000005a0       26 27 27 29 2a 2c 2f 2f 30 31 31 31 31 35 37 39    &'')*,//01111579
000005b0       39 3a 3b 3b 3b 3c 3c 3c 3c 3c 3c 3d 3d 3f 40 40    9:;;;<<<<<<==?@@
000005c0       41 41 43 44 44 46 46 47 49 49 49 4a 4b 4c 4e 4e    AACDDFFGIIIJKLNN
000005d0       4f 52 52 53 54 54 54 56 56 56 58 5b 5b 5d 5e 5e    ORRSTTTVVVX[[]^^
000005e0       5e 5f 61 61 61 62 62 63 64 66 66 68 68 69 6e 6e    ^_aaabbcdffhhinn
000005f0       70 72 73 76 77 77 77 78 79 79 79 7a 7a 7c 17 7e    prsvwwwxyyyzz|.~

STATUS: M68000--Running in monitor              Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Display  Modify  Load  Store  Copy  Find  Report
```

Look carefully at the last several bytes of the RESULTS area. You may notice that a byte is out of order. If not, press F3 and look at the display again. Sometimes, the program seems to work correctly; other times, you will see a byte out of order.

The memory display shows that the QSORT routine works for the most part, which makes it look like the problem occurs on the final write to the RESULTS area. To verify this, set up the sequencer to trigger on any event, store only the address following the return from QSORT (to the main program), and prestore writes to the last eight bytes of the RESULTS area. To do this, modify the trace specification by selecting:

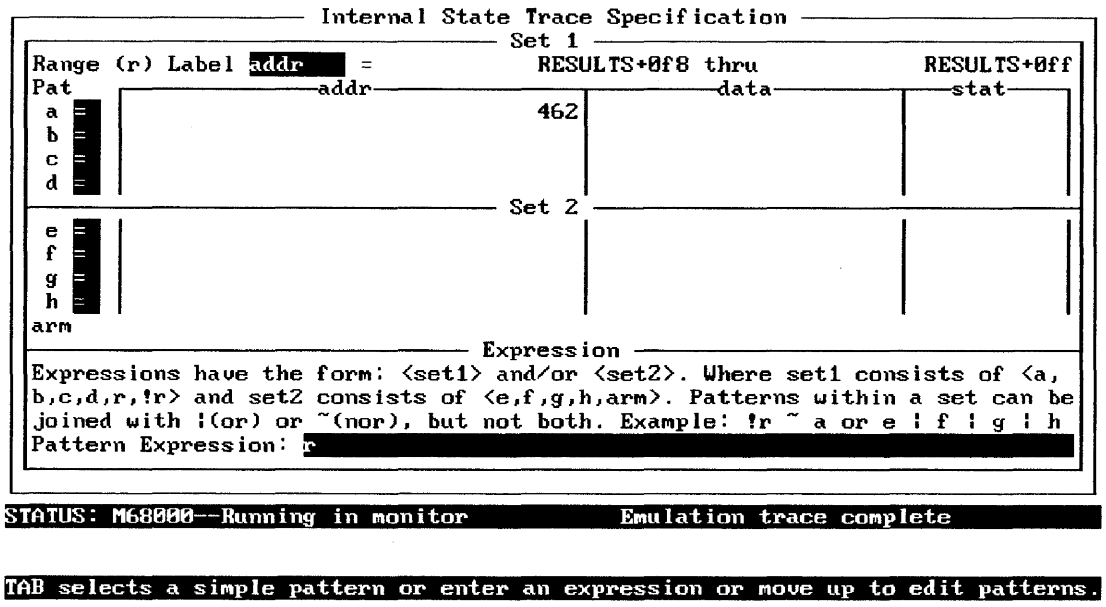
```
Analysis, Trace, Modify
```

Modify the trace specification as shown in figure 3-9.



```
──────────────── Internal State Trace Specification ────────────────
1 While storing a
  Trigger on any state   1 times

2 Store         a




  Branches off          Count time      Prestore on     Trigger position
                                         r               start  of 512
STATUS: M68000--Running in monitor          Emulation trace complete

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 3-9. Tracing Last Write to RESULTS Area



```
                    Internal State Trace Specification
                                Set 1
Range (r) Label addr    =          RESULTS+0f8 thru            RESULTS+0ff
Pat              addr                        data                stat
 a =                              462
 b =
 c =
 d =
                                Set 2
 e =
 f =
 g =
 h =
arm
                              Expression
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h
Pattern Expression: r

STATUS: M68000--Running in monitor            Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-9. Tracing Last Write to RESULTS (Cont'd)

To begin the trace, select:

`Processor, Go, Pc`

`Analysis, Begin`

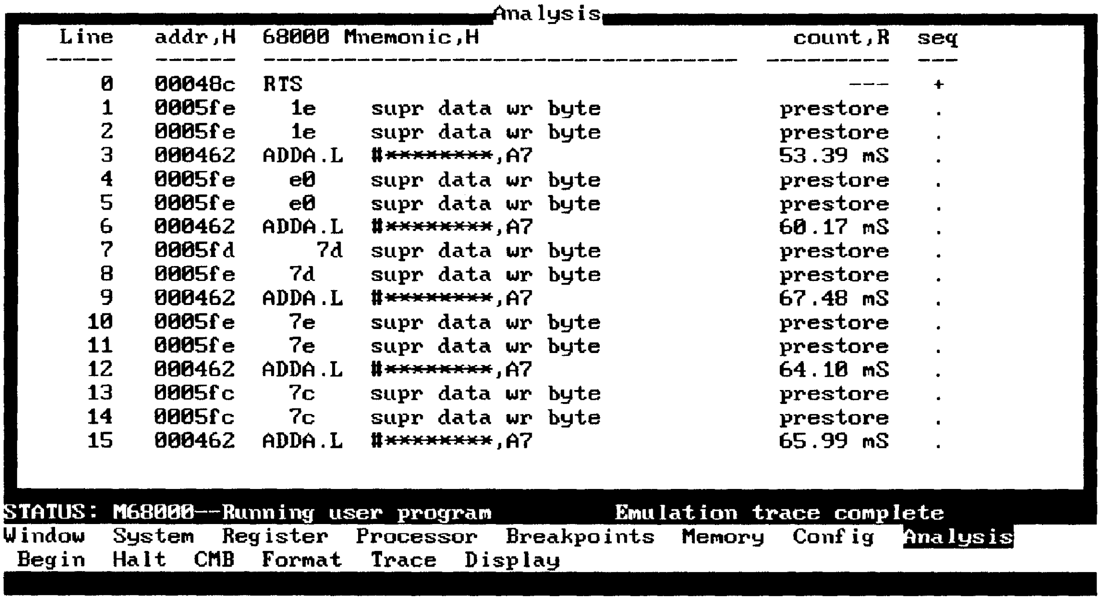
After the "Trace complete" message is shown on the status line, display the trace by selecting the following commands:

`Window, Zoom`

Use the **Tab** key to select the "Analysis" window, and press **Enter**. Now to display the trace, select:

`Analysis, Display`

Enter the first state to display in the starting state field, and the starting state plus 15 into the ending state field, and press **Enter**. The display will be similar to the following display.



```
                                   Analysis
  Line    addr,H   68000 Mnemonic,H                             count,R   seq
  -----   ------   ----------------------------------       ---------   ---
      0   00048c   RTS                                              ---    +
      1   0005fe     1e    supr data wr byte                   prestore    .
      2   0005fe     1e    supr data wr byte                   prestore    .
      3   000462   ADDA.L  #********,A7                        53.39 mS    .
      4   0005fe     e0    supr data wr byte                   prestore    .
      5   0005fe     e0    supr data wr byte                   prestore    .
      6   000462   ADDA.L  #********,A7                        60.17 mS    .
      7   0005fd       7d  supr data wr byte                   prestore    .
      8   0005fe     7d    supr data wr byte                   prestore    .
      9   000462   ADDA.L  #********,A7                        67.48 mS    .
     10   0005fe     7e    supr data wr byte                   prestore    .
     11   0005fe     7e    supr data wr byte                   prestore    .
     12   000462   ADDA.L  #********,A7                        64.10 mS    .
     13   0005fc     7c    supr data wr byte                   prestore    .
     14   0005fc     7c    supr data wr byte                   prestore    .
     15   000462   ADDA.L  #********,A7                        65.99 mS    .

STATUS: M68000--Running user program         Emulation trace complete
Window   System   Register   Processor   Breakpoints   Memory   Config   Analysis
 Begin   Halt  CMB  Format  Trace  Display
```

From the previous trace, you see that the final writes made in the QSORT subroutine are indeed improper values for that part of the RESULTS area. Displaying additional lines of the trace shows you that it is common for bad values to be written to 5FEH. You can set up a trace to trigger on one of the bad writes to 5FEH, and store all the states which lead up to this event. The resulting trace may show you what is wrong with the program.

The sequencer specification which follows will trigger on a write of any value from 80H through 0FFH to 5FEH. The sequencer algorithm to capture the events which lead to a final QSORT write of an inappropriate value to 5FEH is listed below.

1. Search for the beginning of the QSORT routine. (The first occurrence of the INC_LOW address assures that the QSORT routine is actually entered; this eliminates prefetches of the QSORT address from being interpreted as entry into the routine.)

2. If a write of an inappropriate value (80H through 0FFH) to address 5FEH occurs, this may or may not be the trigger event -- another condition must be tested (see 3). Else, if the QSORT routine exits before a write of a bad value to 5FEH occurs, the trigger event has not occurred in this loop of the program; in this case, the sequencer should restart.

3. A write of an inappropriate value to 5FEH has occurred. If the QSORT routine exits without any other value being written to 5FEH, this is the trigger event. Else, if a write of some other value other is made to 5FEH, the previous write is not the event to trigger on, and the sequencer should go back to searching for writes of bad values to 5FEH.

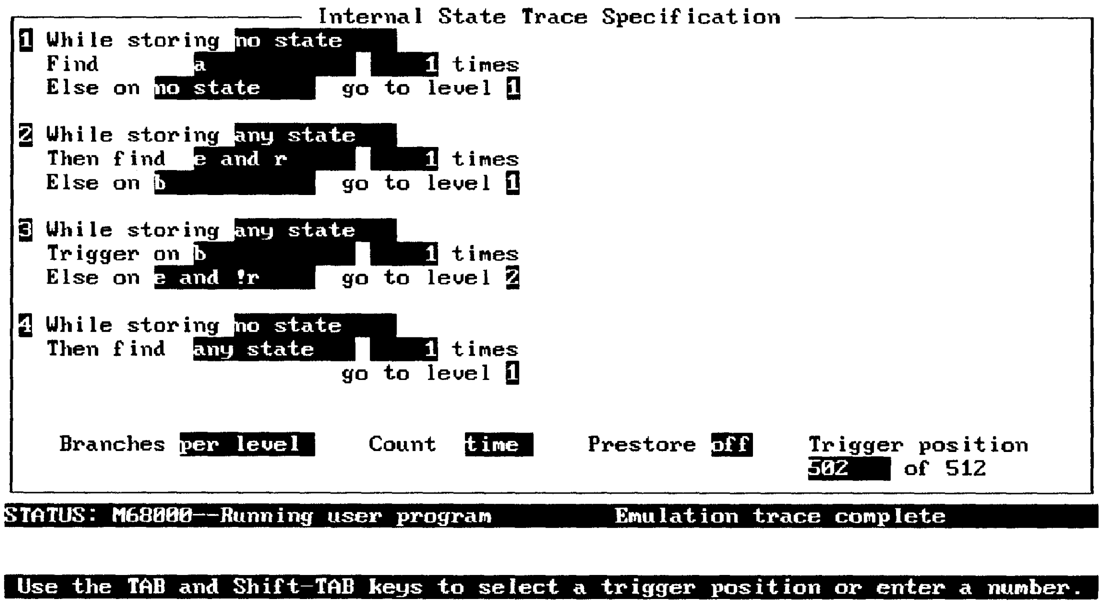
To set up this trace specification, select:

```
   Analysis, Trace, Modify
```

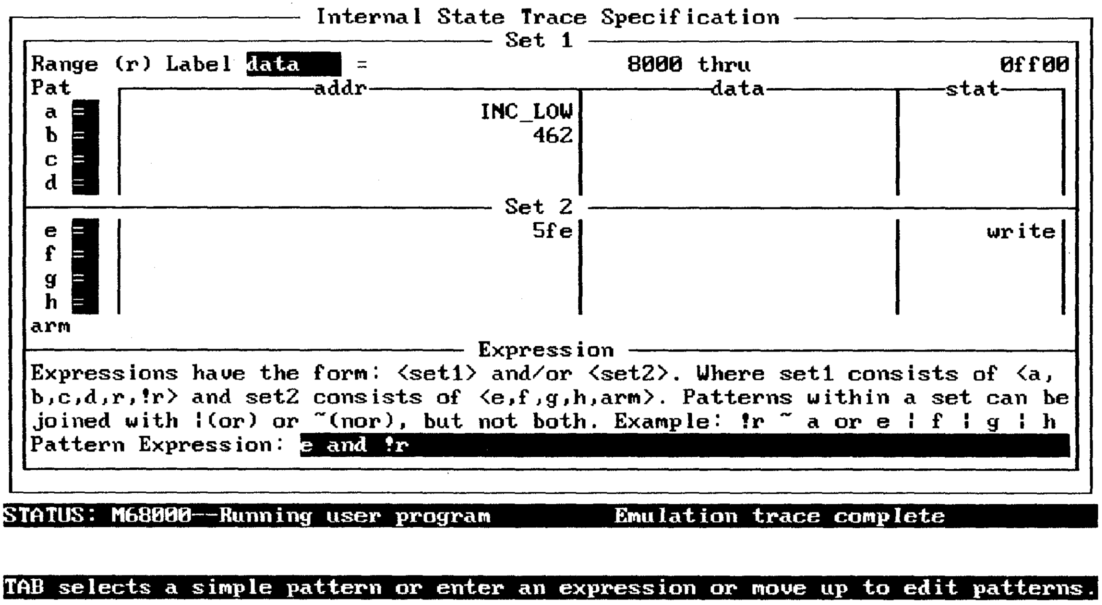
Modify the trace specification as shown in figure 3-10.

Notice that the range of data values are from 8000H through 0FFFFH. Byte values are written to 5FEH, and they are written on the upper 8 bits of the bus. It doesn't matter what value is on the lower 8 bits of the bus, but don't cares are not allowed in values assigned to the range resource.

Notice also that the trigger position is changed to 502 of 512.



```
──────────────── Internal State Trace Specification ────────────────
1 While storing no state
  Find          a              1 times
  Else on no state      go to level 1

2 While storing any state
  Then find  e and r           1 times
  Else on b             go to level 1

3 While storing any state
  Trigger on b                 1 times
  Else on e and !r      go to level 2

4 While storing no state
  Then find  any state         1 times
                        go to level 1


   Branches per level    Count time    Prestore off    Trigger position
                                                       502    of 512

STATUS: M68000--Running user program        Emulation trace complete

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```



```
──────────────── Internal State Trace Specification ────────────────
──────────────────────────────── Set 1 ─────────────────────────────
Range (r) Label data   =                 8000 thru             0ff00
Pat     ┌─────────────addr──────────┬──────────data──────────┬───stat──┐
 a =    │                    INC_LOW│                        │         │
 b =    │                        462│                        │         │
 c =    │                           │                        │         │
 d =    │                           │                        │         │
──────────────────────────────── Set 2 ─────────────────────────────
 e =    │                        5fe│                        │    write│
 f =    │                           │                        │         │
 g =    │                           │                        │         │
 h =    │                           │                        │         │
arm
─────────────────────────────── Expression ─────────────────────────
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with |(or) or ~(nor), but not both. Example: !r ~ a or e | f | g | h
Pattern Expression: e and !r

STATUS: M68000--Running user program        Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-10. Branches "Per Level" Specification

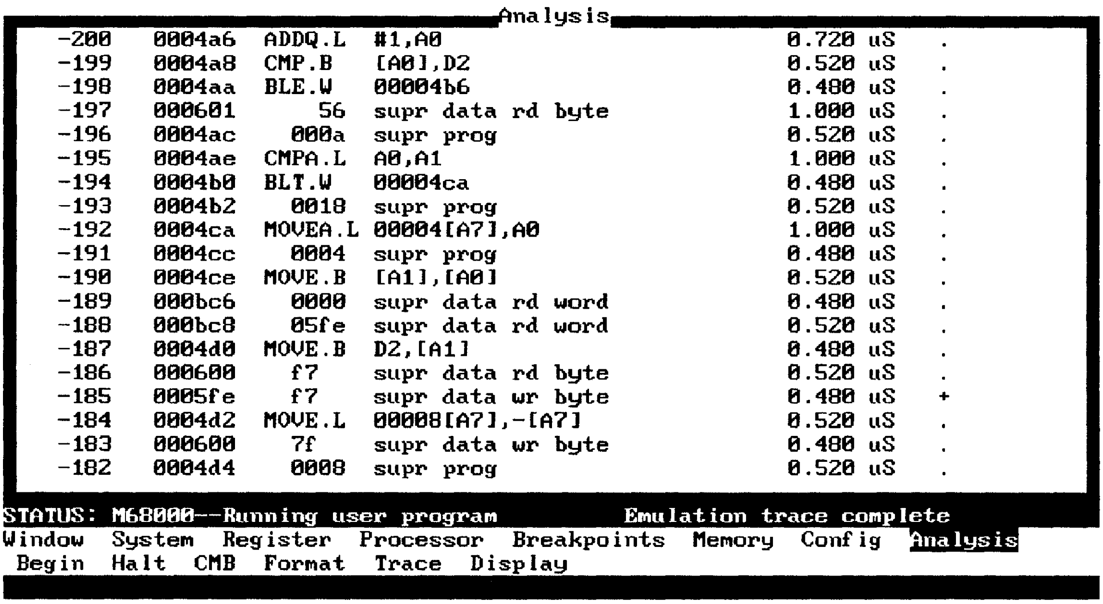
To begin the trace, select:

`Analysis, Begin`

After the "Trace complete" message is shown on the status line, display the trace by selecting:

`Analysis, Display`

Display all the lines in the trace. After the trace is displayed, use the **PgUp** to find the sequence branch prior to the trigger.



```
                                    Analysis
   -200    0004a6   ADDQ.L   #1,A0                               0.720 uS    .
   -199    0004a8   CMP.B    [A0],D2                             0.520 uS    .
   -198    0004aa   BLE.W    00004b6                             0.480 uS    .
   -197    000601       56   supr data rd byte                   1.000 uS    .
   -196    0004ac     000a   supr prog                           0.520 uS    .
   -195    0004ae   CMPA.L   A0,A1                               1.000 uS    .
   -194    0004b0   BLT.W    00004ca                             0.480 uS    .
   -193    0004b2     0018   supr prog                           0.520 uS    .
   -192    0004ca   MOVEA.L  00004[A7],A0                        1.000 uS    .
   -191    0004cc     0004   supr prog                           0.480 uS    .
   -190    0004ce   MOVE.B   [A1],[A0]                           0.520 uS    .
   -189    000bc6     0000   supr data rd word                   0.480 uS    .
   -188    000bc8     05fe   supr data rd word                   0.520 uS    .
   -187    0004d0   MOVE.B   D2,[A1]                             0.480 uS    .
   -186    000600       f7   supr data rd byte                   0.520 uS    .
   -185    0005fe       f7   supr data wr byte                   0.480 uS    +
   -184    0004d2   MOVE.L   00008[A7],-[A7]                     0.520 uS    .
   -183    000600       7f   supr data wr byte                   0.480 uS    .
   -182    0004d4     0008   supr prog                           0.520 uS    .
STATUS: M68000--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

From the trace display above, you can see that the instructions at addresses 4CAH and 4CEH are the ones that cause the problems. These are the instructions associated with the OUT section of the QSORT subroutine. They are used to swap the dividing value and the value at the high index after a segment of the list to be sorted is split. You can see that the high index is address 600H, which it should never be.

Looking back at the program you see that the increment of the high index so that DEC_HIGH works the first time through will cause problems when the BLT OUT instruction gets executed in the INC_LOW loop. Changing the program in the following manner will fix the problem (notice the instructions surrounded by the "#" character).

```
**************************************************
* The QSORT subroutine is passed (on the stack)
* the high and low addresses of an area of bytes
* to be sorted.
**************************************************
QSORT           MOVE.L  8[A7],A1   ; A1 = high index.
                MOVE.L  4[A7],A0   ; A0 = low index.
;-------------------------------------------------
; The following section splits the area to be sorted
; into two areas.  QSORT will be called to sort each
; of these smaller areas.
;
; If high index is less than low index, then sort
; is done.
;-------------------------------------------------
OVER            CMPA.L  A0,A1
                BLE     DONE
;-------------------------------------------------
; D2 = dividing value (from low index).
;-------------------------------------------------
                MOVE.B  [A0],D2
;-------------------------------------------------
; (Increment allows DEC_HIGH loop to work first
; time through.)
;-------------------------------------------------
;#### The following instruction is deleted. ######
;               ADDQ.L  #1,A1
;#################################################
;-------------------------------------------------
; Move low index up until it points to a value
; greater than the dividing value.
;-------------------------------------------------
INC_LOW         ADDQ.L  #1,A0
                CMP.B   [A0],D2
;#### The following instruction is changed. ######
;               BLE     DEC_HIGH
;#################################################
                BLE     NEXT
                CMPA.L  A0,A1
                BLT     OUT
                BRA.B   INC_LOW
;#### The following instruction is new. ##########
NEXT            ADDQ.L  #1,A1
;#################################################
;-------------------------------------------------
; Move high index down until it points to a value
; less than or equal to the dividing value.
;-------------------------------------------------
DEC_HIGH        SUBQ.L  #1,A1
                CMP.B   [A1],D2
                BLT     DEC_HIGH
;-------------------------------------------------
; If high index is less than or equal to low index,
; the area is split; do not swap values.
;-------------------------------------------------
                CMPA.L  A0,A1
                BLT     OUT
```

```
;-------------------------------------------------
; If high index is greater than low index, swap
; values and move indexes again.
;-------------------------------------------------
                MOVE.B  [A0],D3
                MOVE.B  [A1],[A0]
                MOVE.B  D3,[A1]
                BRA.B   INC_LOW
;-------------------------------------------------
; A0 = low address (needed to swap dividing value).
;-------------------------------------------------
OUT             MOVE.L  4[A7],A0)
;-------------------------------------------------
; Swap dividing value and high index value.
;-------------------------------------------------
                MOVE.B  [A1],[A0]
                MOVE.B  D2,[A1]
;-------------------------------------------------
; The area is now split into two smaller areas.
; The last high index value is the middle of the
; two areas.  The high and low addresses for the
; second QSORT call are pushed first.
;-------------------------------------------------
                MOVE.L  8[A7],-[A7]  ; Push high.
                ADDQ.L  #1,A1
                MOVE.L  A1,-[A7]  ; Push middle + 1.
                SUBQ.L  #2,A1
                MOVE.L  A1,-[A7]  ; Push middle - 1.
                MOVE.L  A0,-[A7]  ; Push low.
                BSR.W   QSORT
;-------------------------------------------------
; Increment stack pointer after call.
;-------------------------------------------------
                ADDA.L  #8,A7
                BSR.W   QSORT
                ADDA.L  #8,A7
DONE            RTS
```

Storing "Windows" of Program Execution

One common use for branches "per level" is to trace "windows" of execution. If you're only interested in the execution that occurs between two instructions, you may not want the trace to contain states that occur before and after.

Sequence levels are paired in a typical windowing trace specification. The first sequence level searches for the window enable state, and no states are stored while searching. When the window enable state is found, the analyzer proceeds to the second sequence level which stores the states you're interested in while searching for the window disable state.

If you want to store the window before and after the trigger, use two sets of paired sequence levels: one window enable/disable pair of sequence terms before the trigger, and the another disable/enable pair after the trigger (see figure 3-11). Notice that the order of the second sequence level pair is swapped; if you find the trigger while searching for the window disable state, you want the analyzer to branch to a level that continues to search for the disable state.

For example, to trace only the execution of the sample program's RAND subroutine, you would set up the sequencer specification so that the execution of the first instruction in the RAND subroutine is the window enable state and the address of the RAND subroutine's "return" instruction is the window disable state.

WHILE STORING	FIND	ELSE ON	GO TO
no state	window enable state	no state	
any state	trigger state	window disable state	1
any state	window disable state	no state	
no state	window enable state		3

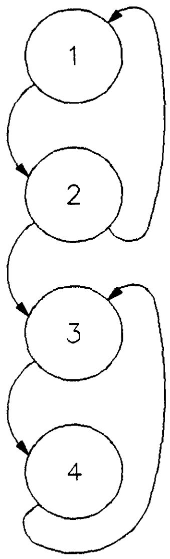


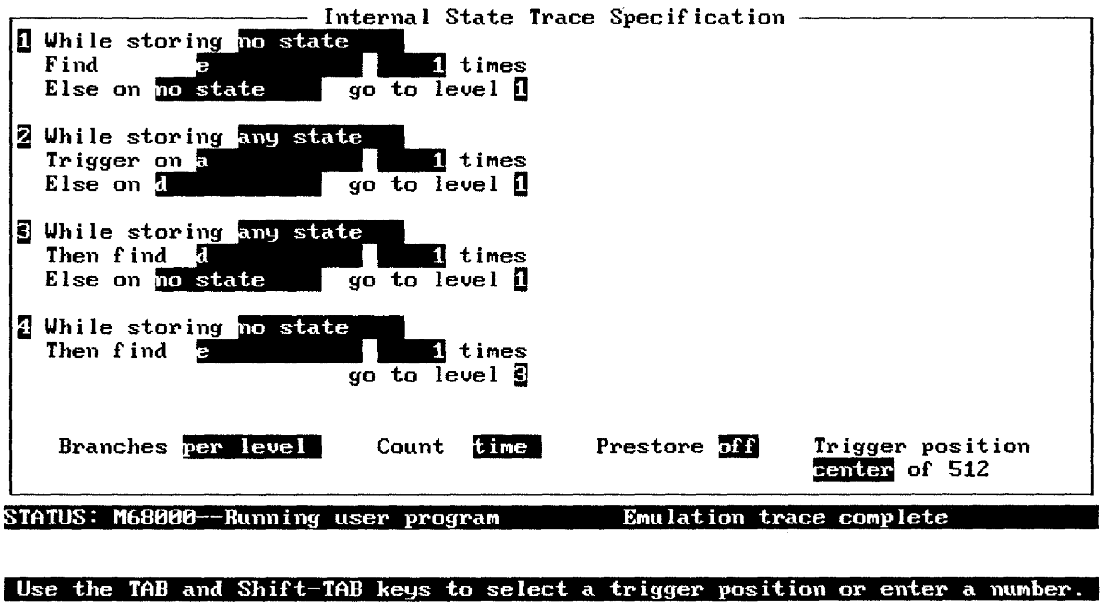
Figure 3-11. Storing a Window of Program Execution

Suppose also that you wish to trigger on the INC_LOW instruction of the QSORT routine.

To set up this trace specification, select:

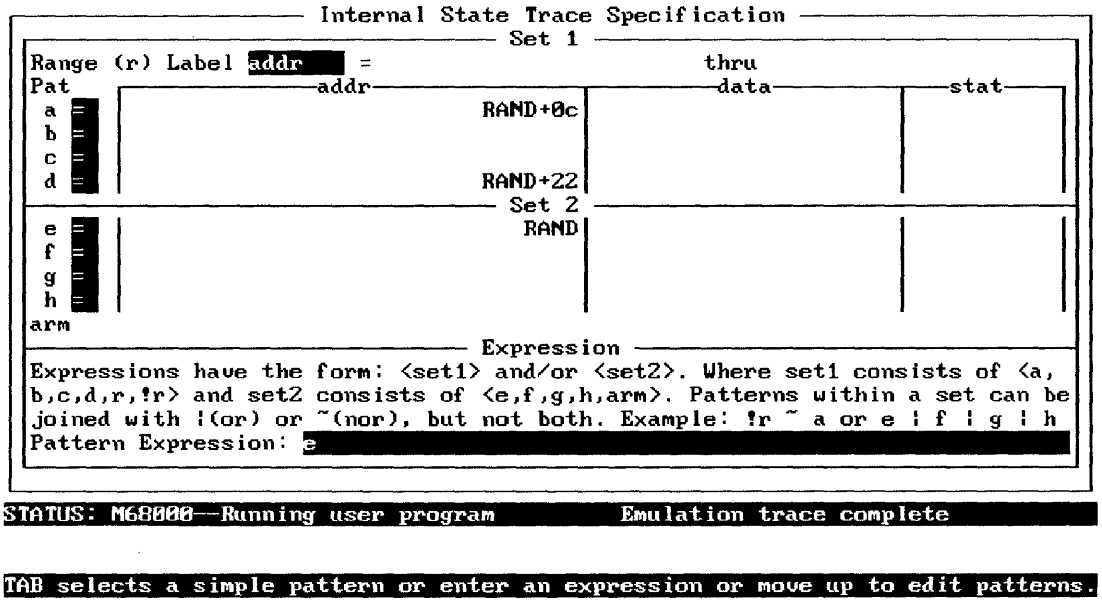
```
Analysis, Trace, Modify
```

Modify the trace specification as shown in figure 3-12.



```
                     Internal State Trace Specification
1 While storing no state
  Find          e            1 times
  Else on no state      go to level 1

2 While storing any state
  Trigger on a               1 times
  Else on d             go to level 1

3 While storing any state
  Then find d                1 times
  Else on no state      go to level 1

4 While storing no state
  Then find e                1 times
                        go to level 3


  Branches per level    Count time    Prestore off    Trigger position
                                                      center of 512
STATUS: M68000--Running user program        Emulation trace complete

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 3-12. Store "Window" Trace Specification



```
 ----------------------- Internal State Trace Specification -------------------------
| ------------------------------------- Set 1 ---------------------------------------|
|Range (r) Label addr    =                          thru                             |
|Pat     ---------------addr-----------------------data---------------stat------     |
| a =   |                            RAND+0c|                  |               |     |
| b =   |                                   |                  |               |     |
| c =   |                                   |                  |               |     |
| d =   |                            RAND+22|                  |               |     |
|-------------------------------------- Set 2 ---------------------------------------|
| e =   |                               RAND|                  |               |     |
| f =   |                                   |                  |               |     |
| g =   |                                   |                  |               |     |
| h =   |                                   |                  |               |     |
|arm                                                                                 |
|------------------------------------ Expression ------------------------------------|
|Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,         |
|b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be        |
|joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h         |
|Pattern Expression: e                                                               |
 ------------------------------------------------------------------------------------
STATUS: M68000--Running user program          Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 3-12. Store "Window" Trace Spec. (Cont'd)

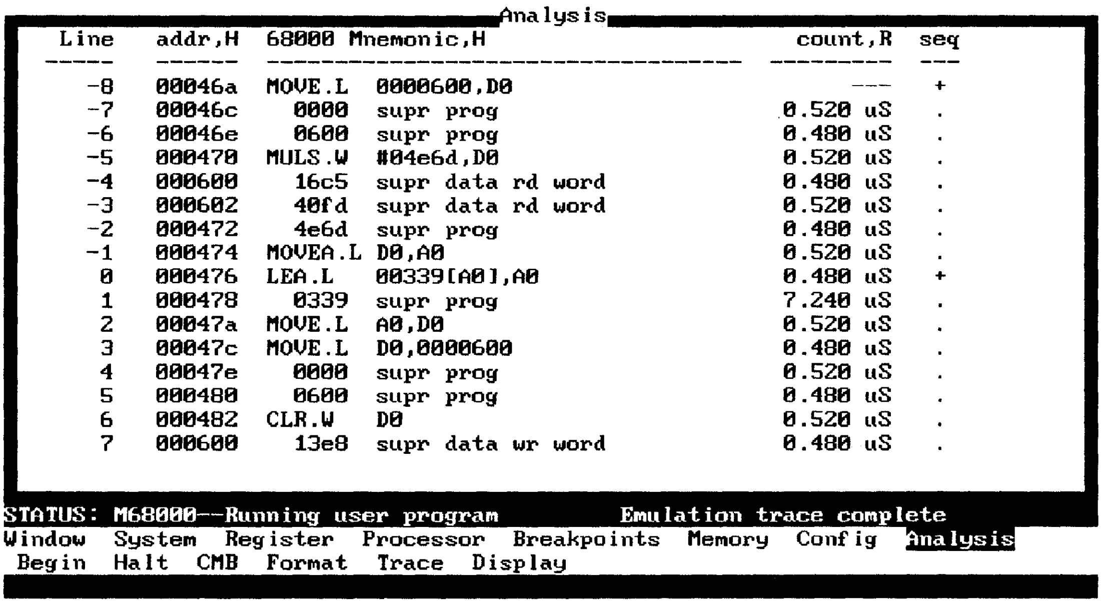
To begin the trace, select:

`Analysis, Begin`

After the "Trace complete" message is shown on the status line, display the trace by selecting:

`Analysis, Display`

Enter -8 in the starting state field and 7 in the ending state field to display the following lines of the trace.

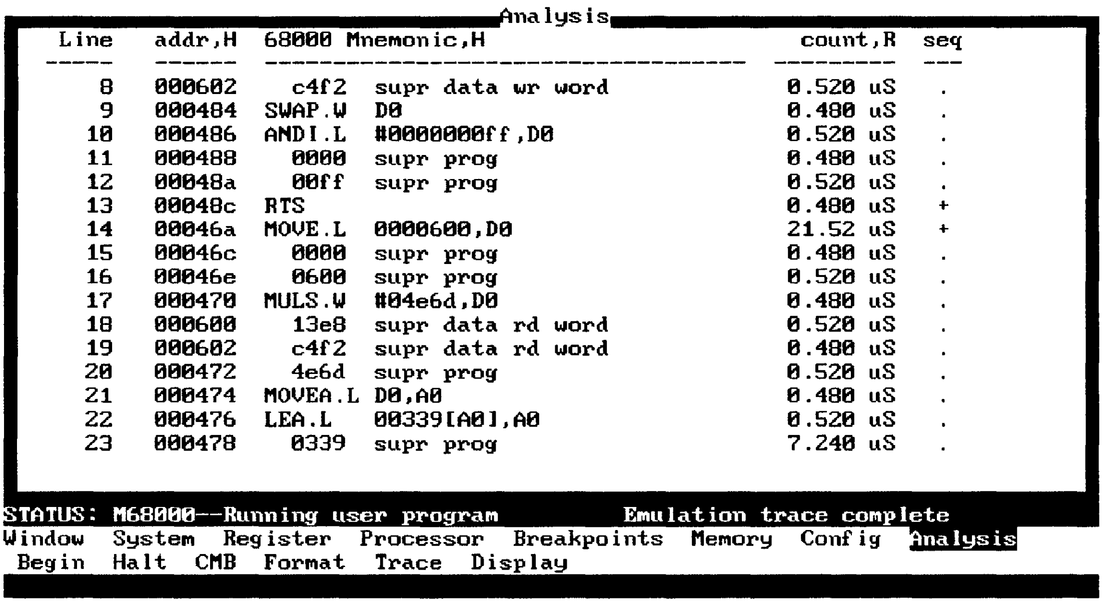


```
                                   Analysis
  Line    addr,H   68000 Mnemonic,H                          count,R   seq
  -----   ------   -------------------------------          ---------  ---
    -8   00046a   MOVE.L  0000600,D0                              ---    +
    -7   00046c     0000  supr prog                         0.520 uS    .
    -6   00046e     0600  supr prog                         0.480 uS    .
    -5   000470   MULS.W  #04e6d,D0                         0.520 uS    .
    -4   000600     16c5  supr data rd word                 0.480 uS    .
    -3   000602     40fd  supr data rd word                 0.520 uS    .
    -2   000472     4e6d  supr prog                         0.480 uS    .
    -1   000474   MOVEA.L D0,A0                             0.520 uS    .
     0   000476   LEA.L   00339[A0],A0                      0.480 uS    +
     1   000478     0339  supr prog                         7.240 uS    .
     2   00047a   MOVE.L  A0,D0                             0.520 uS    .
     3   00047c   MOVE.L  D0,0000600                        0.480 uS    .
     4   00047e     0000  supr prog                         0.520 uS    .
     5   000480     0600  supr prog                         0.480 uS    .
     6   000482   CLR.W   D0                                0.520 uS    .
     7   000600     13e8  supr data wr word                 0.480 uS    .

STATUS: M68000--Running user program        Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

Display more lines of the trace by selecting:

`Analysis, Display`

Press the Enter key twice to select the automatically incremented starting state and ending state numbers to display the following lines of the trace.



```
                                        Analysis
 Line    addr,H   68000 Mnemonic,H                            count,R   seq
-----    ------   ----------------------------------------    ---------  ---
    8    000602     c4f2  supr data wr word                    0.520 uS   .
    9    000484   SWAP.W  D0                                   0.480 uS   .
   10    000486   ANDI.L  #000000ff,D0                         0.520 uS   .
   11    000488     0000  supr prog                            0.480 uS   .
   12    00048a     00ff  supr prog                            0.520 uS   .
   13    00048c   RTS                                          0.480 uS   +
   14    00046a   MOVE.L  0000600,D0                           21.52 uS   +
   15    00046c     0000  supr prog                            0.480 uS   .
   16    00046e     0600  supr prog                            0.520 uS   .
   17    000470   MULS.W  #04e6d,D0                            0.480 uS   .
   18    000600     13e8  supr data rd word                    0.520 uS   .
   19    000602     c4f2  supr data rd word                    0.480 uS   .
   20    000472     4e6d  supr prog                            0.520 uS   .
   21    000474   MOVEA.L D0,A0                                0.480 uS   .
   22    000476   LEA.L   00339[A0],A0                         0.520 uS   .
   23    000478     0339  supr prog                            7.240 uS   .

STATUS: M68000--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

Excluding Windows of Program Execution

You can use the sequencer to exclude windows of program execution by switching the storage qualifiers on the enable and disable sequence levels (see figure 3-13). In other words, store the states you're interested in while searching for the enable state and do not store states while searching for the disable state.

WHILE STORING	FIND	ELSE ON	GO TO
any state	window enable state	no state	
no state	trigger state	window disable state	1
no state	window disable state	no state	
any state	window enable state		3

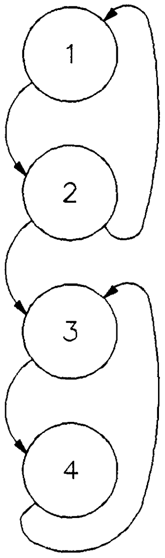


Figure 3-13. Excluding Windows of Program Execution

Triggering on a State Outside the Window

In the previous example, the trigger state occurred inside the window, between two sequence levels that searched for the window enable state. You can set up a trace specification that triggers on a state outside the window by having the trigger occur between two sequence levels that search for the window enable state as shown in figure 3-14.

WHILE STORING	FIND	ELSE ON	GO TO
no state	window enable state	no state	
any state	window disable state	no state	
no state	trigger state	window enable state	2
no state	window enable state	no state	
any state	window disable state		4

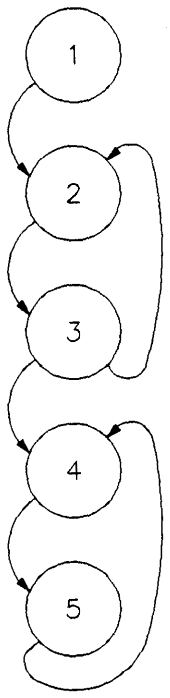


Figure 3-14. Triggering on a State Outside the Window

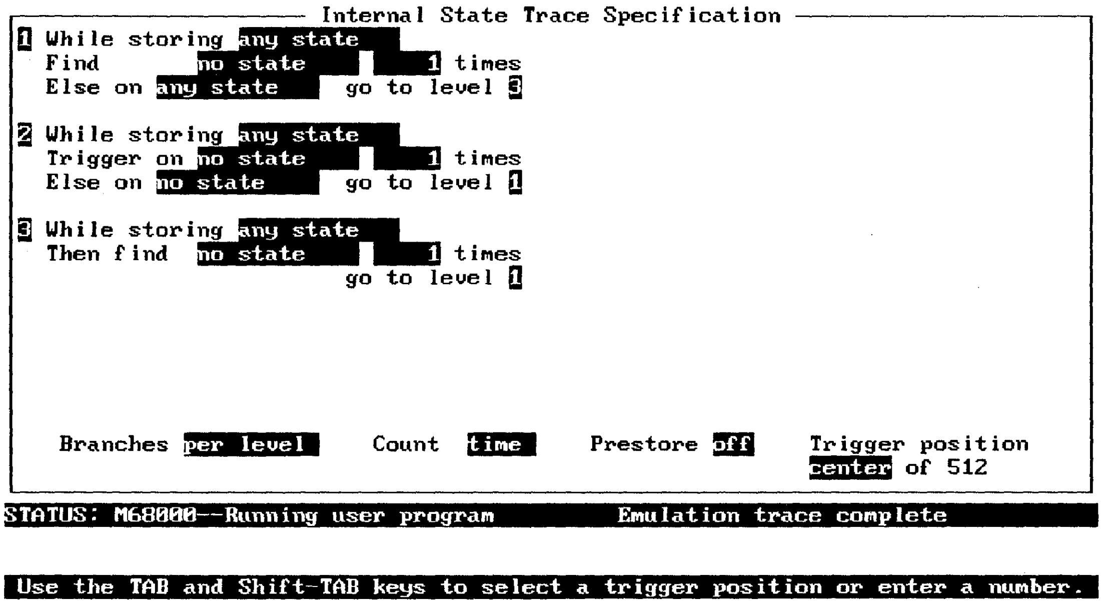
Using Multiple Trigger Levels

It is possible for the analyzer to trigger when finding a state in any sequence level but the last because **the trigger point is not actually when the analyzer finds the state specified in the "Trigger on" field; rather the trigger point is the *first entry* of the sequence level following the level containing the "Trigger on" field.**

To illustrate this point, select:

```
Analysis, Trace, Modify
```

Modify the trace specification as shown in figure 3-15.



```
------------------ Internal State Trace Specification ------------------
1 While storing any state
  Find          no state           1 times
  Else on any state       go to level 3

2 While storing any state
  Trigger on no state              1 times
  Else on no state        go to level 1

3 While storing any state
  Then find     no state           1 times
                          go to level 1




   Branches per level     Count time      Prestore off      Trigger position
                                                             center of 512
STATUS: M68000--Running user program         Emulation trace complete

 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 3-15. Using Multiple Trigger Levels

You might think that because the "Trigger on" field contains "no state" the analyzer will never trigger. But, according to the statement above, the analyzer will trigger because the branch from sequence level one is to the level following the one containing "Trigger on". To verify that this is true, select:

```
Analysis, Begin
```

The "Trace complete" message is shown on the status line, indicating that the analyzer has indeed triggered.

Notes

Using the External Analyzer

Introduction

Most HP 64700 Series emulators may be ordered with an external analyzer. The external analyzer provides 16 external data channels. These data 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 internal emulation analyzer, as an independent external state analyzer, or as an independent external timing analyzer.

The PC Interface is different in the following ways when an external analyzer is present:

1. The "Analysis System" command is available so that you can select the external analyzer mode.

2. The "Internal" and "External" command options are present so that you can specify, when the external analyzer is configured as an independent analyzer, which analyzer the command is for.

3. The "Analysis Format" screen provides fields for selecting the threshold voltages for the external trace signals, shows the activity on the external trace signals, and allows you to define trace labels for the external analyzer trace signals.

4. The predefined "xbits" external analyzer trace label appears in the "Analysis Format" screen and in the trace specification screen.

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.
- Specify the threshold voltages for the external probe signals.
- Label the external trace signals.
- Select the external analyzer mode.

Assembling the Analyzer Probe

The analyzer probe is a two-piece assembly, consisting of 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 they may only be attached 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 4-1.)

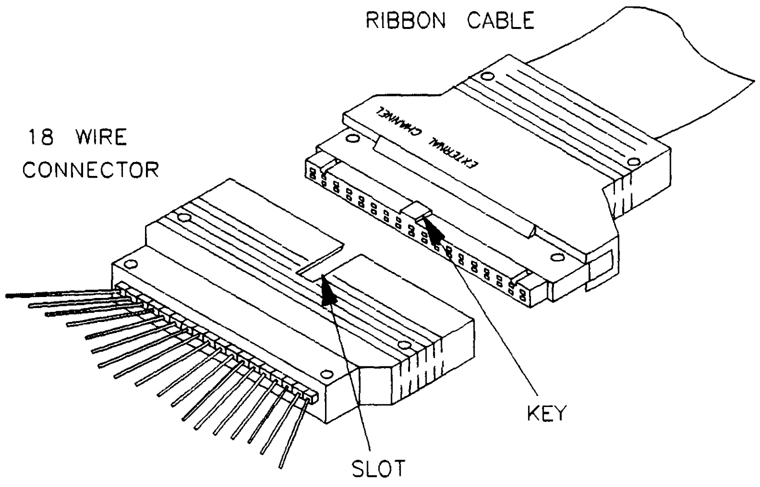


Figure 4-1. Assembling the Analyzer Probe

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 4-2.)

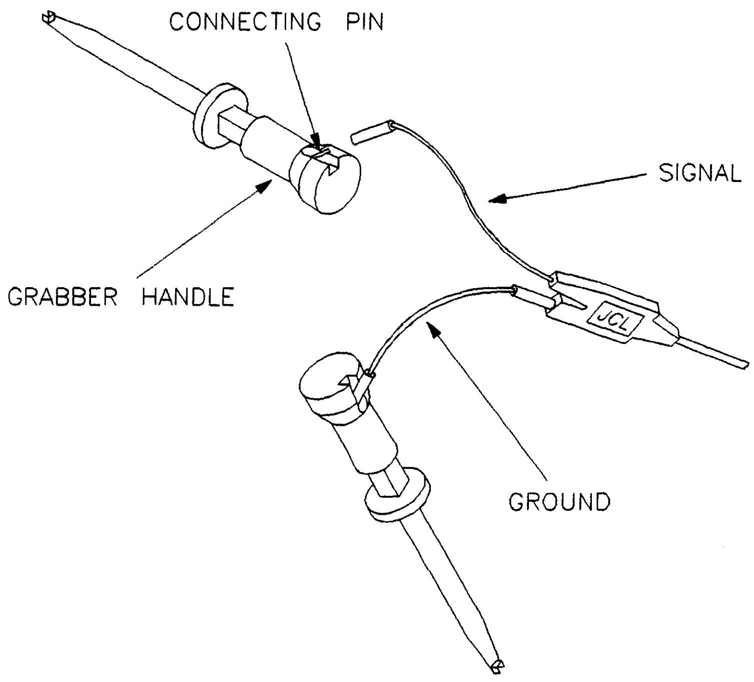


Figure 4-2. Attaching Grabbers to Probe Wires

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 4-3.)

Note

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

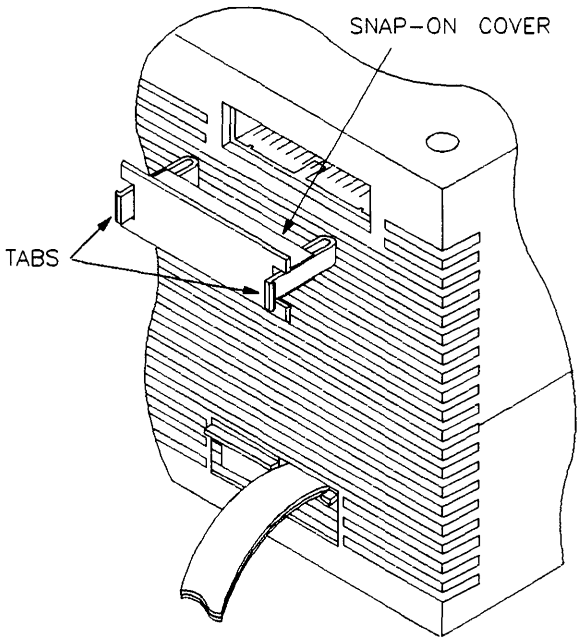


Figure 4-3. Removing Cover to Emulator Connector

Each end of the ribbon cable connector is keyed so that it can be connected 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 4-4.)

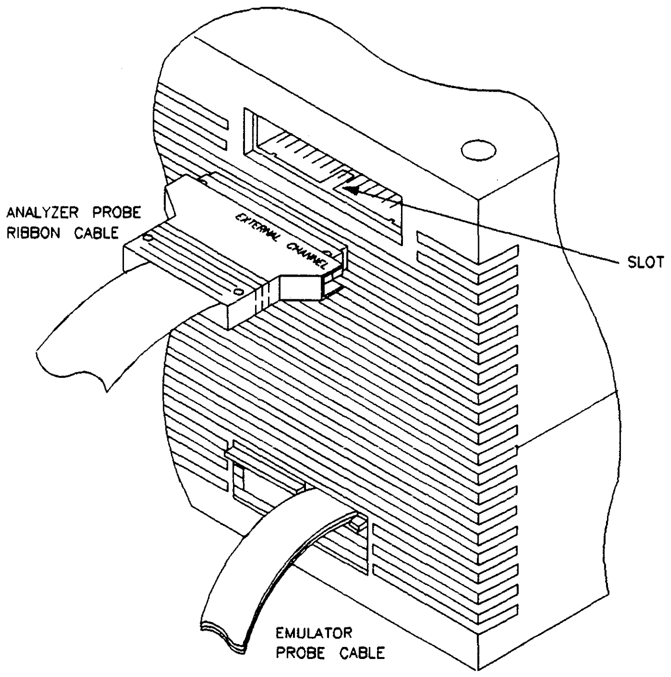


Figure 4-4. Connecting the Probe to the Emulator

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 4-5.)

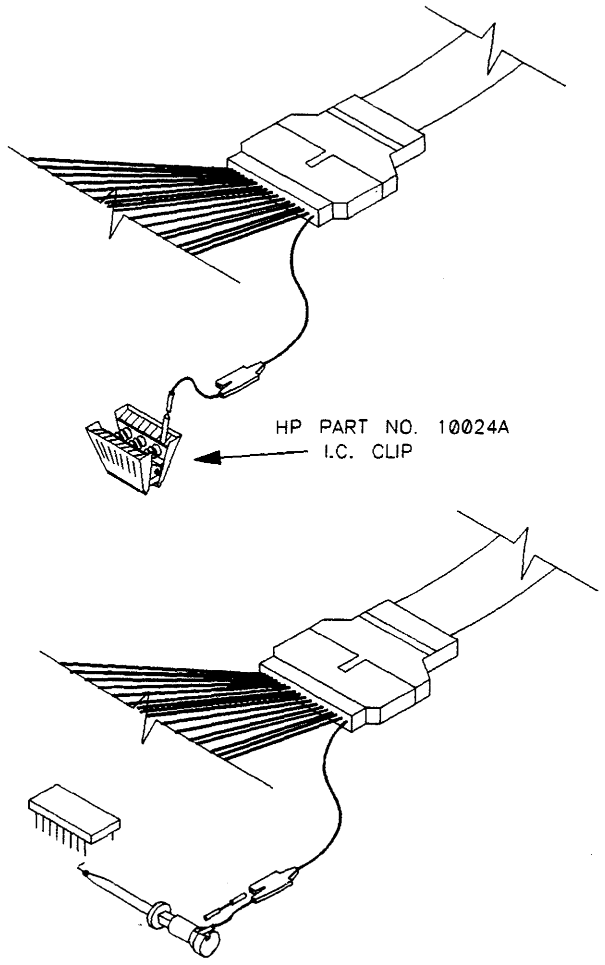


Figure 4-5. Connecting the Probe to the Target System

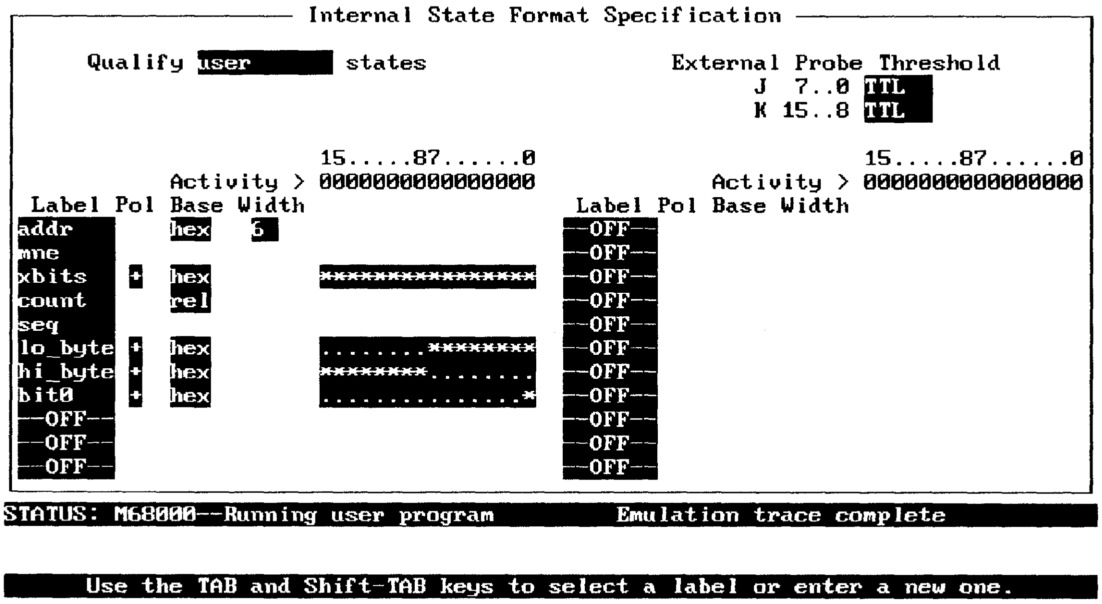
Specifying Threshold Voltages & Defining Labels

To specify threshold voltages for the external probe signals, select:

```
Analysis, Format, Internal
```

An example format specification display is shown in figure 4-6. This screen allows you to specify threshold voltages, informs you of activity on the external analyzer trace signals, and allows you to define external analyzer labels.

When you are done specifying the threshold voltages and defining labels, press **End** and **Enter** to save your specifications. To exit without making changes to this label configuration, press **Esc**.



```
-------------------- Internal State Format Specification --------------------

    Qualify user       states                      External Probe Threshold
                                                        J  7..0 TTL
                                                        K 15..8 TTL

                         15.....87......0                            15.....87......0
              Activity > 0000000000000000                 Activity > 0000000000000000
 Label  Pol Base Width                      Label  Pol Base Width
addr        hex   6                        --OFF--
mne                                        --OFF--
xbits   +   hex       ****************     --OFF--
count       rel                            --OFF--
seq                                        --OFF--
lo_byte +   hex       ........********     --OFF--
hi_byte +   hex       ********........     --OFF--
bit0    +   hex       ...............*     --OFF--
--OFF--                                    --OFF--
--OFF--                                    --OFF--
--OFF--                                    --OFF--

STATUS: M68000--Running user program          Emulation trace complete

     Use the TAB and Shift-TAB keys to select a label or enter a new one.
```

Figure 4-6. Analysis Format Display

Specifying Threshold Voltages

The external probe signals are divided into two groups, the lower byte (channels 0 through 7), and the upper byte (channels 8 through 15). Threshold voltage levels can be specified for each group. When the cursor is in one of the threshold voltage fields, you can use the **Tab** key to select one of the following values, or you can type in the voltage:

TTL	1.4 volts.
CMOS	2.5 volts.
ECL	-1.3 volts.
<number>	You can type in a threshold voltage. Voltages may be from -6.4 volts to 6.35 volts with a 50 mV resolution.

Note

The threshold settings will not take effect until the next time you begin a trace.

External Trace Signal Activity

This line of the display shows the activity on the external trace signals. A trace signal is specified as low (0) when it is below the threshold voltage, high (1) when it is above the threshold voltage, or moving (double headed arrows).

Defining External Analyzer Labels

The format specification screen allows you to define external analyzer labels. When you initially enter the PC Interface, one label, "xbits", is predefined. The "xbits" label includes all 16 external channels as specified by asterisks in the label value field.

You can define labels by moving the cursor to one of the label fields, typing in a label name (embedded spaces are not allowed), and pressing **Enter**. Once you have entered a label name, a label value field appears. The label value field allows you to specify the external channels that make up the label. The label value field allows you to mask off unused or unimportant external channels.

New "label value" fields contain all periods. This means that the label does not currently represent any timing signals (all external channels are masked). To specify channels that the label is to represent, simply move the cursor to the appropriate column, and either use the **Tab** key to select an asterisk (include signal in the label) or type the asterisk key.

In figure 4-6, the external analyzer labels "lo_byte", "hi_byte", and "bit0" are user-defined.

Selecting the External Analyzer Mode

The external analyzer may be configured as an extension to the emulation analyzer, as an independent state analyzer, or as an independent timing analyzer. To specify the external analyzer mode, select:

```
   Anlysis, System
```

Press the **Tab** key to scroll through the various options:

aligned with internal	In this mode, the external analyzer becomes an extension of the internal emulation analyzer. In other words, they operate as one analyzer. The external trace signals may be used to capture target system signals synchronized with the emulation clock.
external state analyzer	In this mode, the external analyzer operates as an independent state analyzer. The independent state analyzer is identical to the emulation analyzer, except that only 16 bits of analysis are available. Your HP 64700 Series emulator now contains two state analyzers; two sets of analyzer resources (trace memory, patterns, qualifiers, etc.) are available, one for the emulation analyzer and one for the independent state analyzer. When one of the independent analyzer modes is selected, you can use one analyzer to arm the

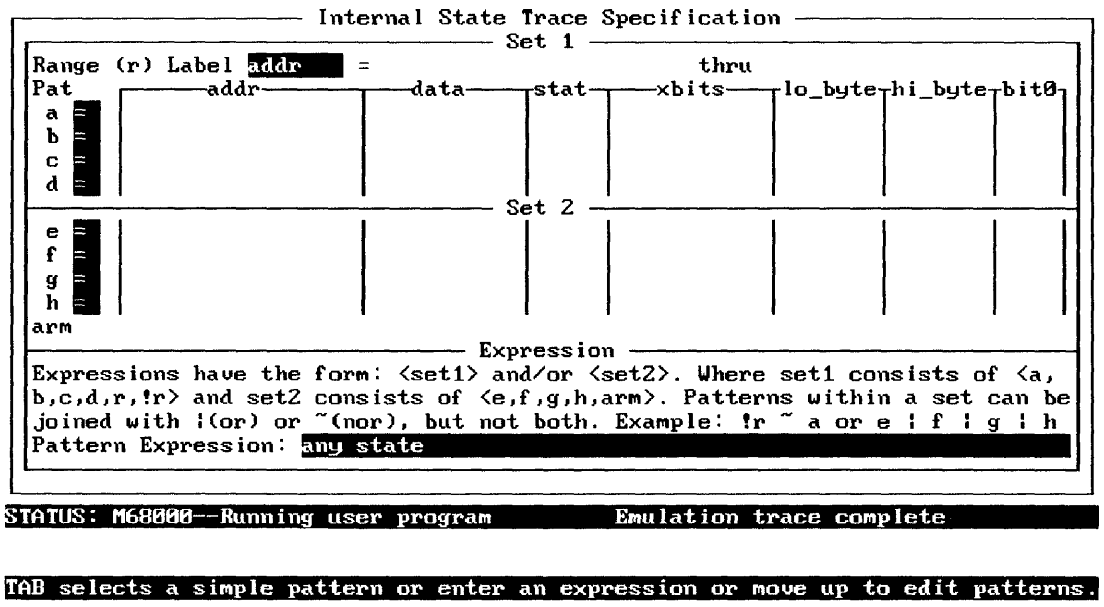
other. You can specify the arm condition as a qualifier, perhaps as the trigger condition (cross-triggering). (Refer to the "Making Coordinated Measurements" chapter for more information on cross-triggering.)

external timing analyzer	In this mode, the external analyzer operates as an independent timing analyzer. Refer to the "Using the External Timing Analyzer" for more information.

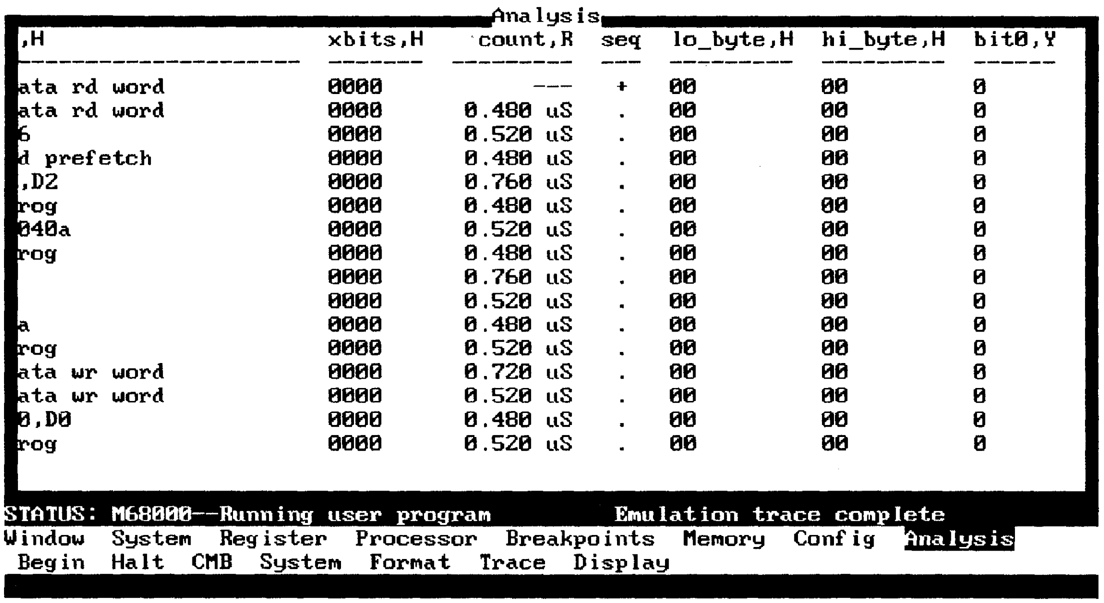
Using the External Analyzer when Aligned with Internal

When the "aligned with internal" mode of the external analyzer is selected, it operates (from the PC Interface) as an extension to the emulation analyzer. In other words, they operate as one analyzer. External data is captured on the emulation clock as are the address, data, and status signals.

When an external analyzer is present, you may enter external data qualifiers in the trace specification, as shown in figure 4-7, by assigning values to "xbits", or to user-defined external analyzer labels such as "lo_byte", "hi_byte", or "bit0". Also, the trace display may contain additional columns showing the data captured on the external channels, as shown in figure 4-7. (You can use the **CTRL** and right-arrow or left-arrow keys to view columns that are beyond the width of the window.)



```
                    Internal State Trace Specification
                                  Set 1
Range (r) Label addr    =                         thru
Pat        addr          data      stat    xbits     lo_byte hi_byte bit0
 a =
 b =
 c =
 d =
                                  Set 2
 e =
 f =
 g =
 h =
arm
                               Expression
Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
joined with |(or) or ~(nor), but not both. Example: !r ~ a or e | f | g | h
Pattern Expression: any state

STATUS: M68000--Running user program          Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```



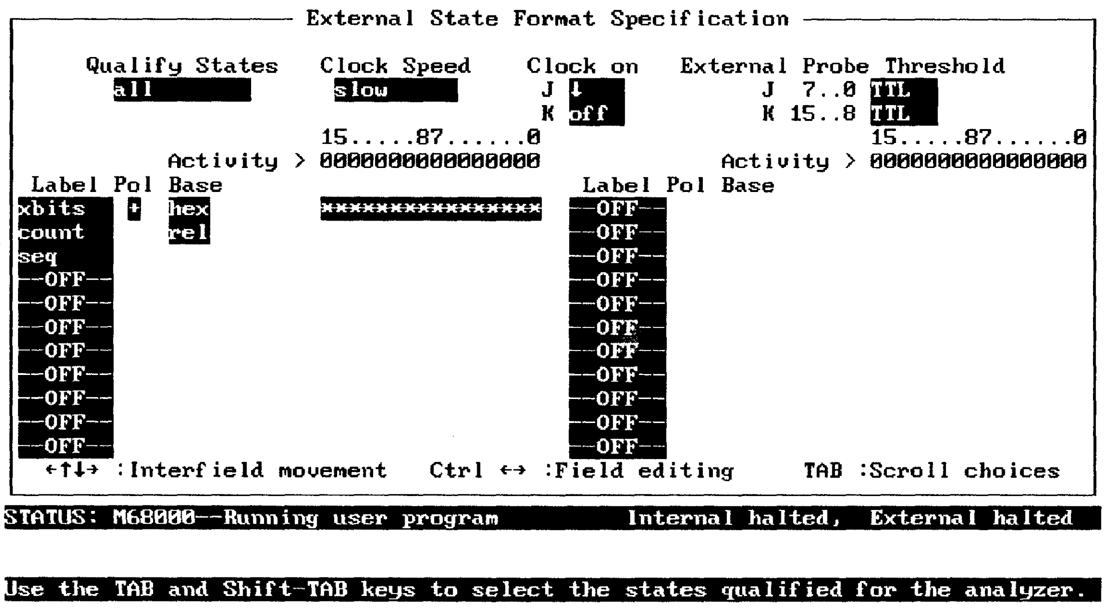
```
                                  Analysis
,H                     xbits,H    count,R   seq  lo_byte,H  hi_byte,H  bit0,Y
--------------------   -------   ---------  ---  ---------  ---------  ------
ata rd word            0000            ---   +   00         00         0
ata rd word            0000      0.480 uS    .   00         00         0
6                      0000      0.520 uS    .   00         00         0
d prefetch             0000      0.480 uS    .   00         00         0
,D2                    0000      0.760 uS    .   00         00         0
rog                    0000      0.480 uS    .   00         00         0
040a                   0000      0.520 uS    .   00         00         0
rog                    0000      0.480 uS    .   00         00         0
                       0000      0.760 uS    .   00         00         0
                       0000      0.520 uS    .   00         00         0
a                      0000      0.480 uS    .   00         00         0
rog                    0000      0.520 uS    .   00         00         0
ata wr word            0000      0.720 uS    .   00         00         0
ata wr word            0000      0.520 uS    .   00         00         0
0,D0                   0000      0.480 uS    .   00         00         0
rog                    0000      0.520 uS    .   00         00         0

STATUS: M68000--Running user program          Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  System  Format  Trace  Display
```

Figure 4-7. External Data in the Trace

Using the External State Analyzer

When you select the "external state analyzer" mode, the external analyzer operates as an independent state analyzer. You use the independent state analyzer in the same way as you use the internal analyzer, except that you must select the external analyzer clock source and specify the maximum qualified clock speed. Both of these specifications are made in the external analysis format screen (see figure 4-8) which you access by selecting:

```
    Analysis, Format, External
```



```
───────────────────── External State Format Specification ─────────────────────

     Qualify States    Clock Speed     Clock on    External Probe Threshold
       all               slow            J ↓             J  7..0 TTL
                                         K off           K 15..8 TTL
                       15.....87......0                         15.....87......0
              Activity > 0000000000000000                Activity > 0000000000000000
  Label Pol Base                                  Label Pol Base
xbits   +   hex        ****************           --OFF--
count       rel                                   --OFF--
seq                                               --OFF--
--OFF--                                           --OFF--
--OFF--                                           --OFF--
--OFF--                                           --OFF--
--OFF--                                           --OFF--
--OFF--                                           --OFF--
--OFF--                                           --OFF--
--OFF--                                           --OFF--
--OFF--                                           --OFF--
   ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing     TAB :Scroll choices

STATUS: M68000--Running user program          Internal halted,  External halted

Use the TAB and Shift-TAB keys to select the states qualified for the analyzer.
```

Figure 4-8. Analysis Format Display

Selecting the Clock Source

The independent state analyzer may be clocked with target system clock signals connected to the JCL and KCL external clock inputs.

Notice that there are two "Clock on" fields (see figure 4-8), one for the J signal and one for the K signal. Position the cursor in one of these fields and press the **Tab** key repeatedly to view the following selections.

<rising edge>	Selects the rising edge of this signal as the clock.
<falling edge>	Selects the falling edge of this signal as the clock.
<both edges>	Specifies that the external analyzer be clocked on both the rising and falling edges of this signal.
high	Specifies this signal as a qualifying signal that only allows the other signal to clock the analyzer when the qualifying signal is higher than the threshold voltage (see figure 4-9).
low	Specifies this signal as a qualifying signal that only allows the other signal to clock the analyzer

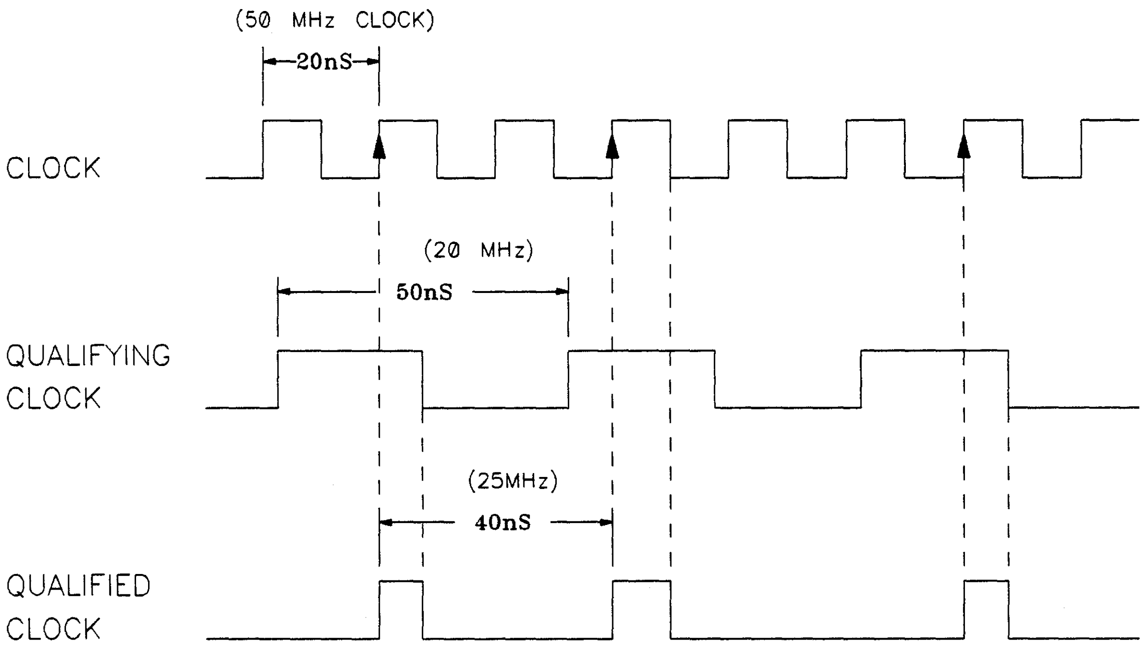


Figure 4-9. Qualified Clocks

	when the qualifying signal is lower than the threshold voltage.
off	Specifies that this signal is not used.

When edges are specified for both the J and K signals, the external analyzer clocks on any of the specified edges.

Setup time for qualifying signals is approximately 20 nanoseconds. Qualifying signal hold time is approximately 5 nanoseconds.

Specifying the Maximum Qualified Clock Speed

The maximum qualified clock rate is the repetition rate of all specified clock signals (see figure 4-9). When selecting the maximum qualified clock speed of the analyzer, there are tradeoffs involving the trace count qualifier to be considered. You select the maximum qualified clock speed in the "Clock Source" field of the "Analysis Format External" screen (see figure 4-8). There are three maximum speeds that can be specified:

slow	Slow specifies a maximum qualified clock rate of 16 MHz. When "slow" is selected, there are no restrictions on the trace count qualifier.
fast	Fast specifies a maximum qualified clock rate of 20 MHz. When "fast" is selected, the trace count qualifier may be used to count states but not time.
very fast	Very fast specifies a maximum qualified clock rate of 25 MHz. When "very fast" is selected, the trace count qualifier may not be used at all (in other words, the count qualifier field in the trace specification screen must be "off").

External Analyzer Specifications

- Threshold Accuracy = +/- 50 mV.
- Threshold Voltage Range = 6 V to -6 V.
- 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%.

External State Analyzer Specifications

- 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.

5

Using the External Timing Analyzer

Introduction

The external analyzer can be aligned with the internal emulation analyzer, configured as an external state analyzer, or configured as an external timing analyzer. This chapter shows you how to use the external timing analyzer. The main sections in this chapter:

- Show you how to configure the external analyzer as a timing analyzer.
- Show you how to specify threshold voltages and define labels for the external probe signals.
- Describe the timing specifications.
- Describe the timing waveform display.
- Show you examples of using the external timing analyzer.

Note

Two versions of the PC Interface are shipped with your emulator; one contains interface software for the timing analyzer and the other does not. The version that contains the timing interface software is larger and requires most of the PC's 640K bytes of RAM. The version that does not contain the timing interface software requires approximately 480K bytes of RAM.

Prerequisites

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

- You must be using the version of the PC Interface that gives you access to the external timing analyzer.
- The external analyzer probe must be assembled and connected to signals of interest as shown in the "Using the External Analyzer" chapter.

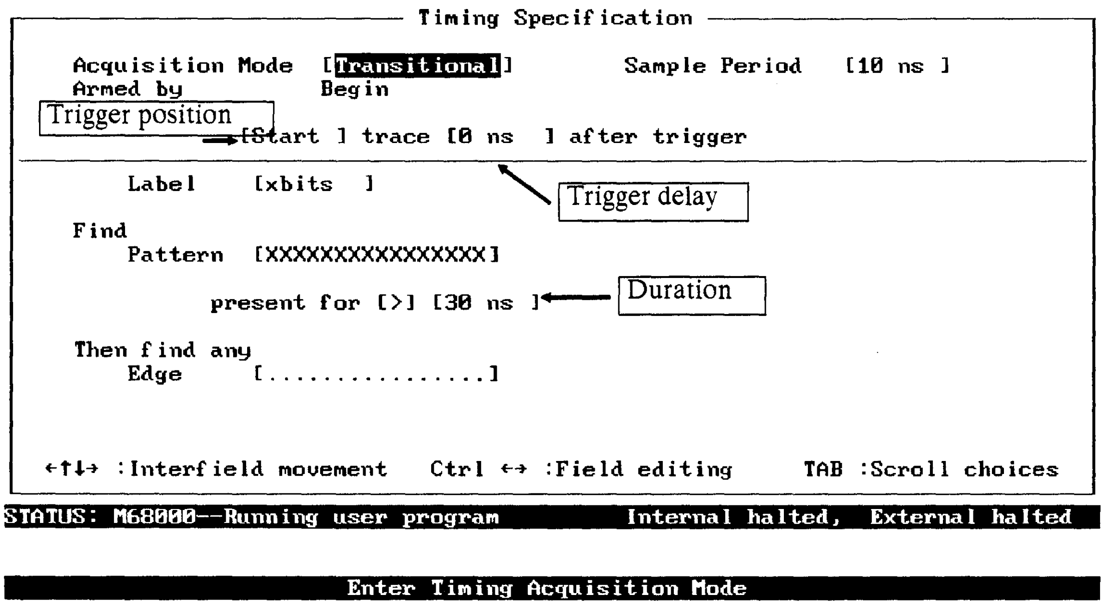
Configuring for Timing Analysis

To configure your emulator's external analyzer as a timing analyzer, select:

```
Analysis, System
```

Now, use the **Tab** key to select "external timing analyzer" and press **Enter**.

When the external analyzer is configured as a timing analyzer, the "Analysis Trace Modify External" command gives you a special timing analyzer trace specification screen (see figure 5-1) and the "Analysis Display External" command gives you a timing waveform display.



```
──────────────────── Timing Specification ────────────────────

  Acquisition Mode  [Transitional]        Sample Period   [10 ns ]
  Armed by          Begin

                  [Start ] trace [0 ns   ] after trigger
──────────────────────────────────────────────────────────────
      Label     [xbits  ]

  Find
      Pattern   [XXXXXXXXXXXXXXXX]

             present for [>] [30 ns ]

  Then find any
      Edge      [................]


 ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: M68000--Running user program         Internal halted,  External halted

                       Enter Timing Acquisition Mode
```

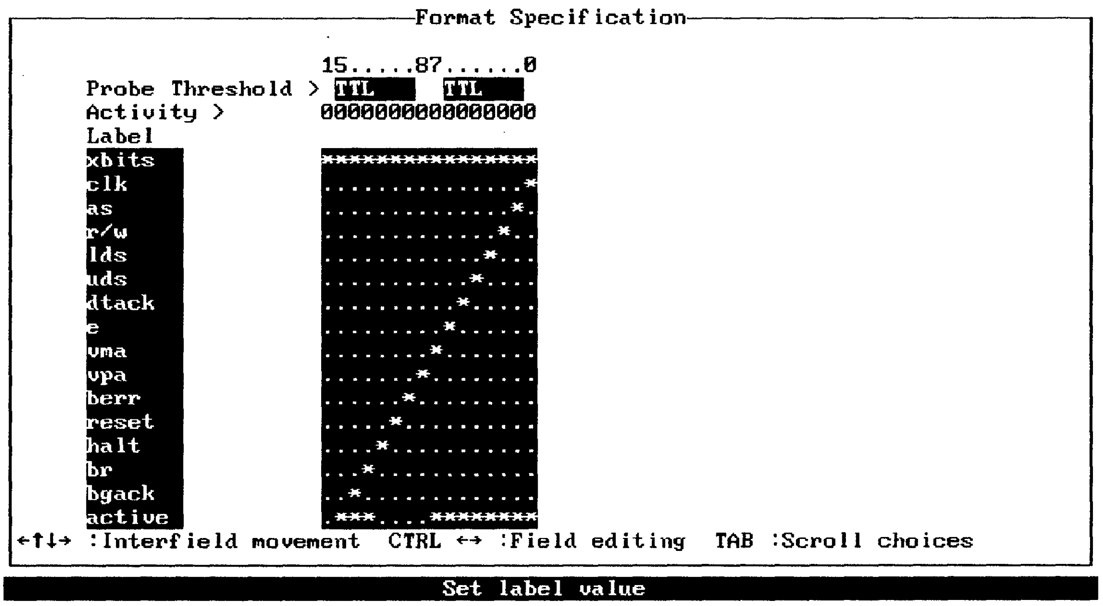
Figure 5-1. Timing Interface Main Display

Specifying Threshold Voltages & Defining Labels

To specify threshold voltages for the external probe signals, select:

```
Analysis, Format, External
```

An example format specification display is shown in figure 5-2. This screen allows you to specify threshold voltages, informs you of activity on the external analyzer trace signals, and allows you to define timing labels.



```
                                   Format Specification
                              15.....87......0
     Probe Threshold >  TTL      TTL
     Activity >            0000000000000000
     Label
     xbits                 ****************
     clk                   ...............*
     as                    ..............*.
     r/w                   .............*..
     lds                   ............*...
     uds                   ...........*....
     dtack                 ..........*.....
     e                     .........*......
     vma                   ........*.......
     vpa                   .......*........
     berr                  ......*.........
     reset                 .....*..........
     halt                  ....*...........
     br                    ...*............
     bgack                 ..*.............
     active                .***....********
←↑↓→ :Interfield movement  CTRL ←→ :Field editing  TAB :Scroll choices
                                   Set label value
```

Figure 5-2. Timing Label Specification

When you are done specifying the threshold voltages and defining labels, press **End** and **Enter** to save your specifications. To exit without making changes to this label configuration, press **Esc**.

Threshold Voltages

The external probe signals are divided into two groups, the lower byte (channels 0 through 7), and the upper byte (channels 8 through 15). Threshold voltage levels can be specified for each group. When the cursor is in one of the threshold voltage fields, you can use the **Tab** key to select one of the following values, or you can type in the voltage:

TTL	1.4 volts.
CMOS	2.5 volts.
ECL	-1.3 volts.

<number>	You can type in a threshold voltage. Voltages may be from -6.4 volts to 6.35 volts with a 50 mV resolution.

Activity

This line of the display shows the activity on the external trace signals. A trace signal is specified as low (0) when it is below the threshold voltage, high (1) when it is above the threshold voltage, or moving (double headed arrows).

Labels

The format specification screen allows you to define timing labels. When you initially enter the PC Interface, one label, "xbits", is predefined. The "xbits" label includes all 16 external channels as specified by asterisks in the label value field.

You can define labels by moving the cursor to one of the label fields, typing in a label name (embedded spaces are not allowed), and pressing **Enter**. Once you have entered a label name, a label value field appears. The label value field allows you to specify the external channels that make up the label. The label value field allows you to mask off unused or unimportant external channels.

New "label value" fields contains all periods. This means that the label does not currently represent any timing signals (all external channels are masked). To specify channels that the label is to represent, simply move the cursor to the appropriate column, and either use the **Tab** key to select an asterisk (include signal in the label) or type the asterisk key.

Labels can be used for signal identification in the waveform display. When a label consists of more than one signal, the signals within that label are numbered from the least significant active channel. For example, in figure 5-2, external channel 13 may be represented in the waveform display with the following names: xbits 13, bgack 00, or active 9.

Timing Specification

This section describes the options available when modifying the timing specification. To modify the timing specification, select:

```
   Analysis, Trace, Modify, External
```

Brackets, [ ], in the timing specification display indicate fields in which the **Tab** key may be used to select choices.

When you are done modifying the timing specification, press **End** and **Enter** to save your specifications. To exit without making changes to the timing specification, press **Esc**.

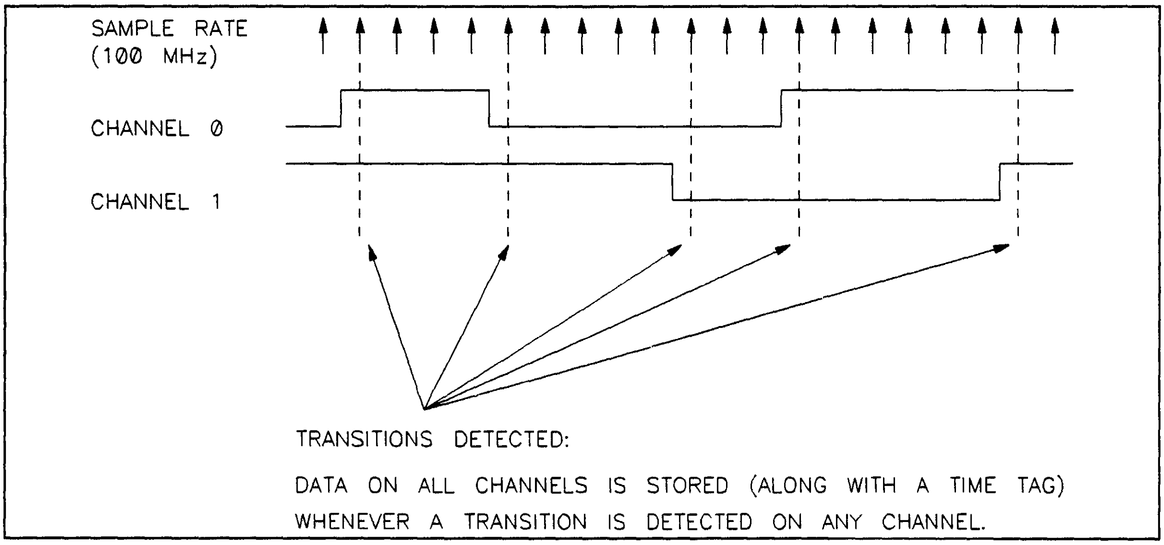

SAMPLE RATE (100 MHz)
CHANNEL 0
CHANNEL 1
TRANSITIONS DETECTED:
DATA ON ALL CHANNELS IS STORED (ALONG WITH A TIME TAG) WHENEVER A TRANSITION IS DETECTED ON ANY CHANNEL.

Figure 5-3. Transitional Acquisition Mode

Acquisition Modes

You can use the external timing analyzer in one of three modes: transitional, standard (data acquisition), or glitch (data and glitch acquisition).

Transitional

In the transitional mode, data is sampled at 100 MHz, but stored only when an input transition (on any channel) is detected. A time tag is also be stored so that the data can be accurately placed in the display. This mode allows the effective time window stored to be increased while maintaining resolution.

The number of data transitions stored depends on the incoming data transmission rate. The minimum number of data transitions that will be stored is 128. Because time tags are also stored, the maximum number of transitions that can be stored is 512. See figure 5-3.

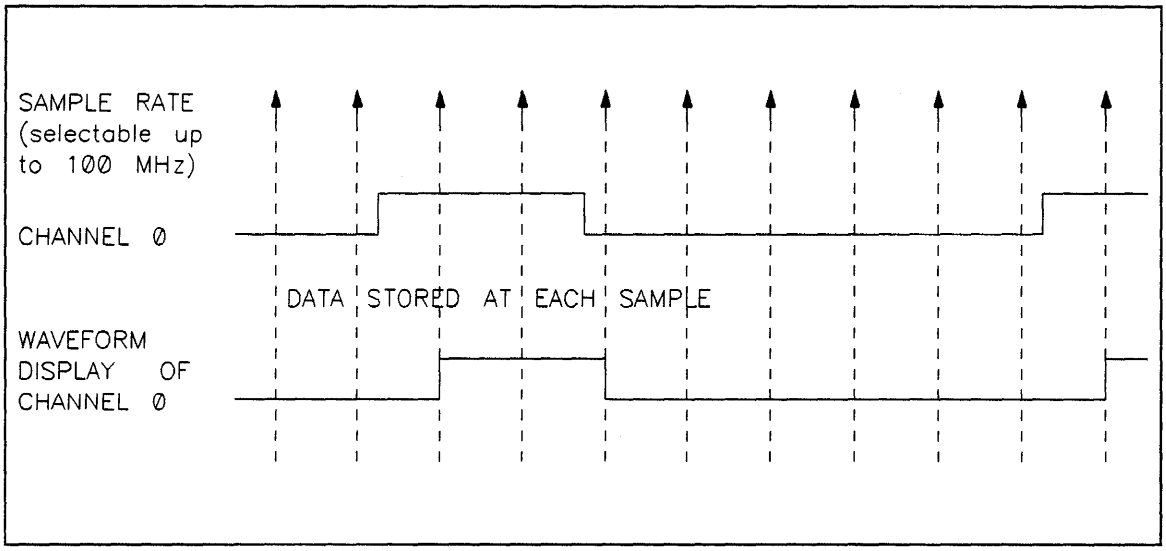

SAMPLE RATE (selectable up to 100 MHz)
CHANNEL 0
DATA STORED AT EACH SAMPLE
WAVEFORM DISPLAY OF CHANNEL 0

Figure 5-4. Standard Acquisition Mode

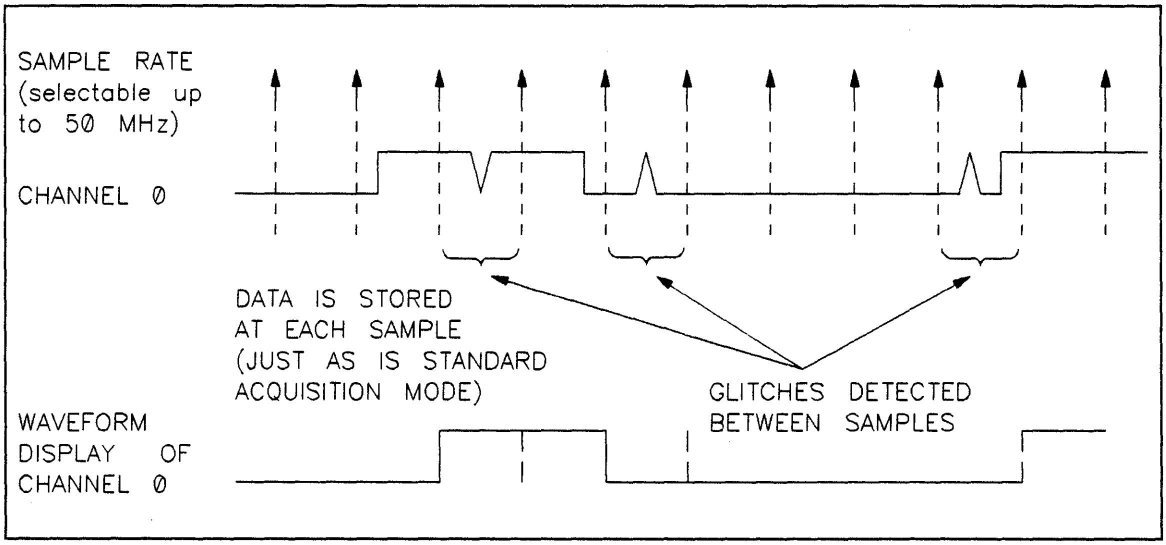

SAMPLE RATE (selectable up to 50 MHz)
CHANNEL 0
DATA IS STORED AT EACH SAMPLE (JUST AS IS STANDARD ACQUISITION MODE)
GLITCHES DETECTED BETWEEN SAMPLES
WAVEFORM DISPLAY OF CHANNEL 0

Figure 5-5. Glitch Acquisition Mode

Standard

In the standard mode, the timing analyzer samples data on the external analyzer probe at the selected sample rate. Up to 1024 samples can be stored, and the maximum sample rate is 100 MHz (10 ns intervals). See figure 5-4.

Glitch

This is the same as the standard acquisition mode except that glitch information is also stored at 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 512, and the maximum sample rate is 50 MHz (20 ns intervals). See figure 5-5.

Sample Period

The sample period in the transitional acquisition mode is always 10 ns.

Valid periods in the standard acquisition mode are between 10 ns and 50 ms in a 1/2/5 sequence.

Valid periods in the glitch acquisition mode are between 20 ns and 50 ms in a 1/2/5 sequence.

The accuracy of the sample rate is that of the crystal oscillator, approximately +/- 0.01%.

Armed By

The "Armed by" field reflects the arm condition of the external analyzer.

Begin

When "Armed by Begin" is shown, the external analyzer is always armed. This means that the analyzer can perform measurements at any time.

TRIG1 or TRIG2

When "Armed by TRIG1" or "Armed by TRIG2" is shown, the external analyzer cannot perform a measurement until the arm is received from an external trigger signal.

External trigger signals can drive the external analyzer over the TRIG1 or TRIG2 signals, as specified in the Trigger Configuration (see the "Allowing CMB and BNC Triggers to Arm the Analyzer" section in the "Coordinated Measurements" chapter).

Trigger Position

The trigger position option allows you to place the trigger at the start, center, or end of the trace. For example, if you wanted to look at events before the trigger, you would place the trigger at the end of the trace.

Trigger Holdoff (Prestore)

Trigger holdoff is the number of samples that are stored before the trigger will be recognized. Trigger holdoff is as follows for the three timing modes:

Transitional Mode	None.
Standard Mode	Start trace = 60 samples. Center trace = 512 samples. End trace = 960 samples.
Glitch Mode	Start trace = 30 samples. Center trace = 256 samples. End trace = 480 samples.

Trigger Delay

Timing 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.

Label Qualifier

The "Label" field in the timing specification display allows you to select one of the defined labels. You can use the **Tab** key to scroll through the defined labels. Once a label is selected, you can then specify patterns, edges, or glitches on the data channels associated with that label. The trigger condition is the combined specifications of all labels (that is, the pattern, edge, or glitch specifications for each label are ORed together to form the complete specification for the 16 external data channels).

All defined labels can appear in the waveform display regardless of the label qualifier selected.

Specifying the Trigger Condition

The timing specification options described above tell the timing analyzer how to capture data and how to display it, but they do not tell the analyzer when to capture data; this is done by specifying the trigger condition. The "Find" and "Then Find" options allow you to specify the trigger condition.

There are three types of trigger conditions:

1. **Pattern Trigger**. This type of trigger condition occurs when no edges (or glitches if the glitch acquisition mode is enabled) are specified in the "Then Find" options. In order for the trigger condition to become true, the pattern specified must be present for greater than or less than a specified amount of time (duration).

2. **Edge Trigger**. This type of trigger condition can only occur when edges are specified in the "Then Find" options and a greater than duration is specified. In order for this trigger condition to become true, the specified pattern trigger must be found and be present for greater than the specified period of time, and a specified edge must be found while the pattern is still present.

3. **Glitch Trigger**. This type of trigger condition can only occur in the glitch acquisition mode when glitches are specified in the "Then Find" options and a greater than duration is specified. In order for this trigger condition to become true, the specified pattern must be found and be present for greater than the specified period of time, and a specified glitch must be found while the pattern is still present.

These three types of trigger conditions are described below.

Find Pattern

Find pattern allows you to specify a data pattern consisting of 1's, 0's, or X's (don't cares) across the 16 channels. The most significant bit is channel 15 and least significant bit is channel 0.

Use the field editing keys to position the cursor to the different channels, and use the **Tab** key to select the appropriate value.

Duration

The "present for" option allows you to specify a pattern duration greater than or less than a specified amount of time before the pattern trigger condition is 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.

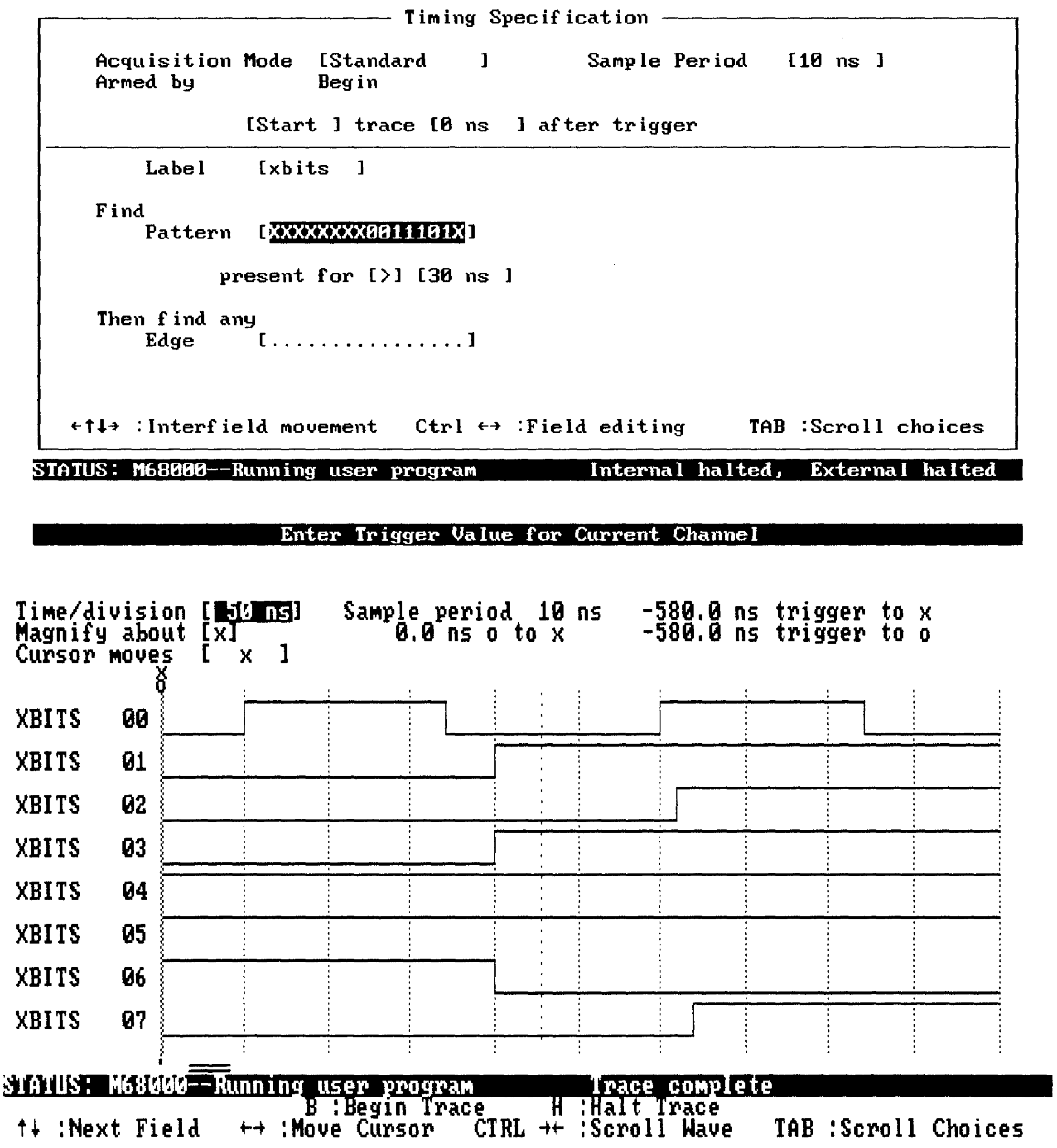
Greater Than Duration

The trigger occurs after the pattern is present on the probe for the specified time duration. Selectable from 30 ns to 10 ms in 10 ns increments.

Less Than Duration

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

An example pattern trigger, with a duration greater than 30 ns, is shown in figure 5-6.



```
--------------------------- Timing Specification ---------------------------

    Acquisition Mode  [Standard     ]        Sample Period   [10 ns ]
    Armed by          Begin

                [Start ] trace [0 ns    ] after trigger
----------------------------------------------------------------------------
        Label     [xbits   ]

    Find
        Pattern   [XXXXXXXX00111101X]

                present for [>] [30 ns ]

    Then find any
        Edge      [................]


  ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing     TAB :Scroll choices
STATUS: M68000--Running user program        Internal halted,  External halted
```

```
                    Enter Trigger Value for Current Channel

Time/division [50 ns]   Sample period  10 ns    -580.0 ns trigger to x
Magnify about [x]           0.0 ns o to x       -580.0 ns trigger to o
Cursor moves  [  x  ]
```

```
STATUS: M68000--Running user program        Trace complete
                        B :Begin Trace      H :Halt Trace
↑↓ :Next Field   ←→ :Move Cursor   CTRL →← :Scroll Wave   TAB :Scroll Choices
```

Figure 5-6. Pattern Trigger

Then Find Edge or Glitch

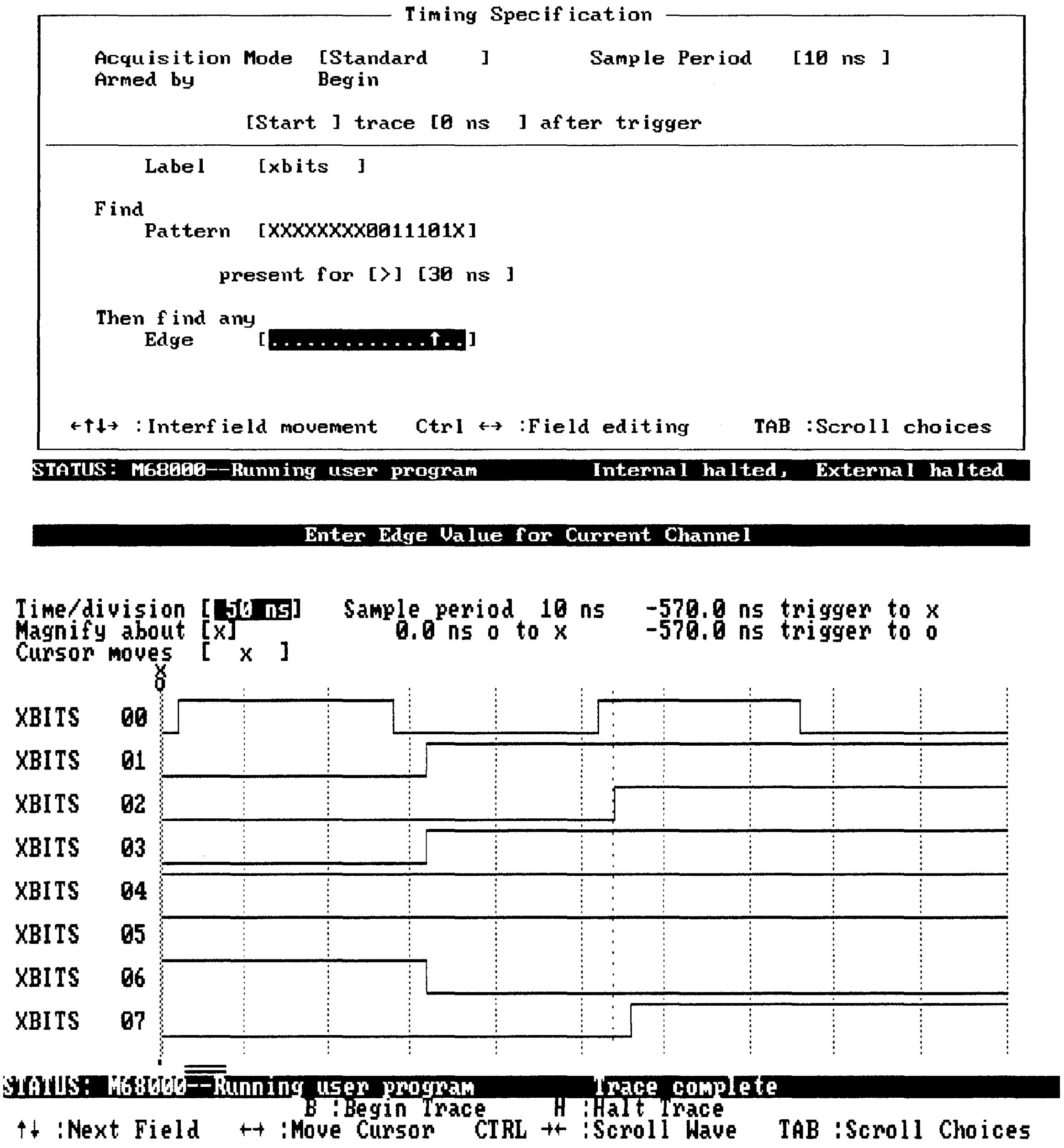
Edge or glitch triggers occur after the pattern has been present on the probe for the specified duration and, while the pattern is still present, an edge or glitch occurs on any of the selected channels. Edges on each channel can be specified as rising, falling, or either rising or falling.

Edge triggers may only be specified with "greater than" durations. See figure 5-7 for an edge trigger example.

Glitch triggers may only be specified while in the glitch acquisition mode and with "greater than" durations. See figure 5-8 for a glitch trigger example.

Reset

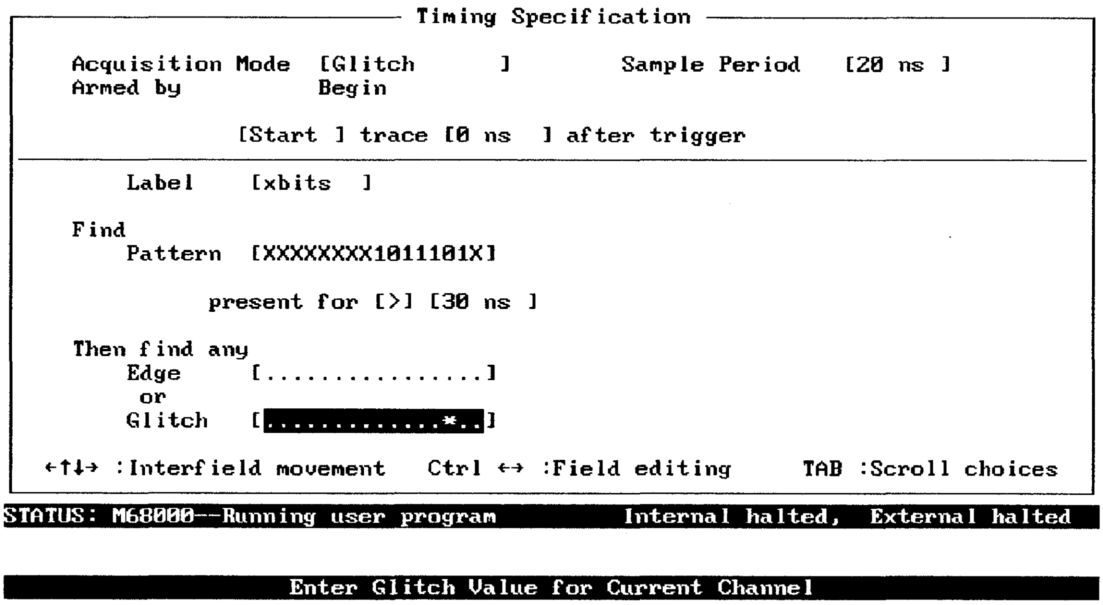
The timing specification reset command allows you to return the timing specification to its default values (see figure 5-1).



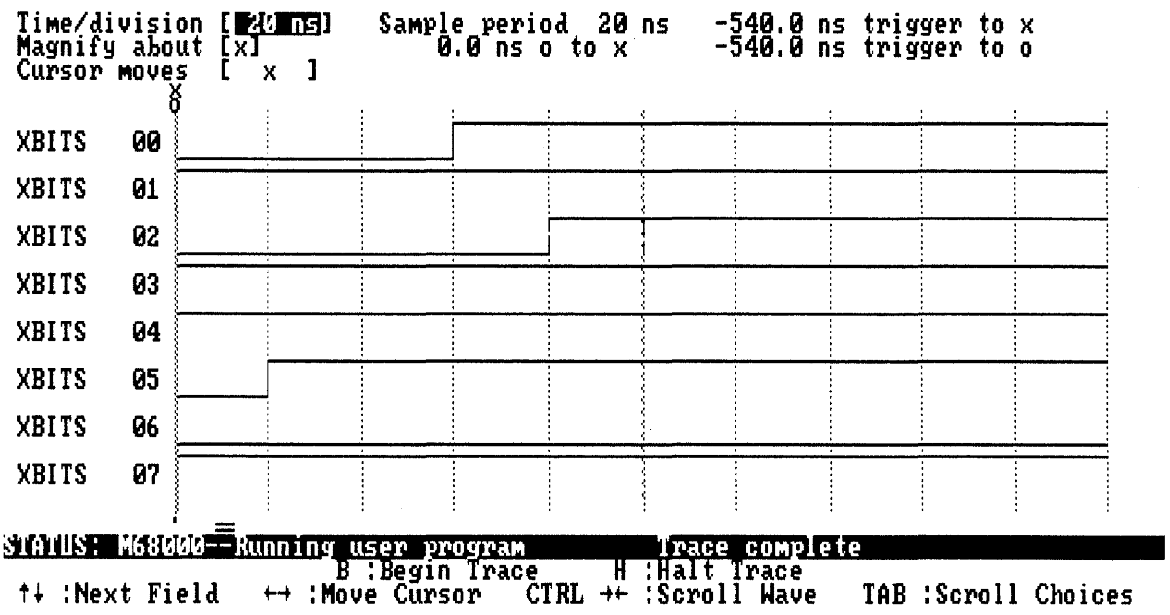
```
─────────────────────── Timing Specification ───────────────────────

   Acquisition Mode  [Standard     ]        Sample Period   [10 ns ]
   Armed by          Begin

              [Start ] trace [0 ns   ] after trigger
─────────────────────────────────────────────────────────────────────
        Label     [xbits   ]

   Find
        Pattern   [XXXXXXXX0011101X]

              present for [>] [30 ns ]

   Then find any
        Edge      [.............↑..]


 ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing     TAB :Scroll choices
STATUS: M68000--Running user program          Internal halted,  External halted

                     Enter Edge Value for Current Channel
```

```
Time/division [50 ns]   Sample period  10 ns     -570.0 ns trigger to x
Magnify about [x]             0.0 ns o to x      -570.0 ns trigger to o
Cursor moves  [  x  ]

XBITS   00
XBITS   01
XBITS   02
XBITS   03
XBITS   04
XBITS   05
XBITS   06
XBITS   07

STATUS: M68000--Running user program          Trace complete
                        B :Begin Trace       H :Halt Trace
↑↓ :Next Field    ←→ :Move Cursor    CTRL →← :Scroll Wave    TAB :Scroll Choices
```

Figure 5-7. Edge Trigger



```
──────────────────────── Timing Specification ────────────────────────

   Acquisition Mode  [Glitch      ]        Sample Period   [20 ns ]
   Armed by          Begin

              [Start ] trace [0 ns    ] after trigger
──────────────────────────────────────────────────────────────────────
        Label     [xbits   ]

   Find
        Pattern   [XXXXXXXX1011101X]

              present for [>] [30 ns ]

   Then find any
        Edge      [................]
         or
        Glitch    [.............*..]

 ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: M68000--Running user program         Internal halted,  External halted

                  Enter Glitch Value for Current Channel
```



```
Time/division [20 ns]   Sample period  20 ns    -540.0 ns trigger to x
Magnify about [x]           0.0 ns o to x       -540.0 ns trigger to o
Cursor moves  [  x  ]

XBITS   00
XBITS   01
XBITS   02
XBITS   03
XBITS   04
XBITS   05
XBITS   06
XBITS   07

STATUS: M68000--Running user program          Trace complete
                   B :Begin Trace       H :Halt Trace
↑↓ :Next Field   ←→ :Move Cursor   CTRL →← :Scroll Wave    TAB :Scroll Choices
```

Figure 5-8. Glitch Trigger

Timing Waveform Display

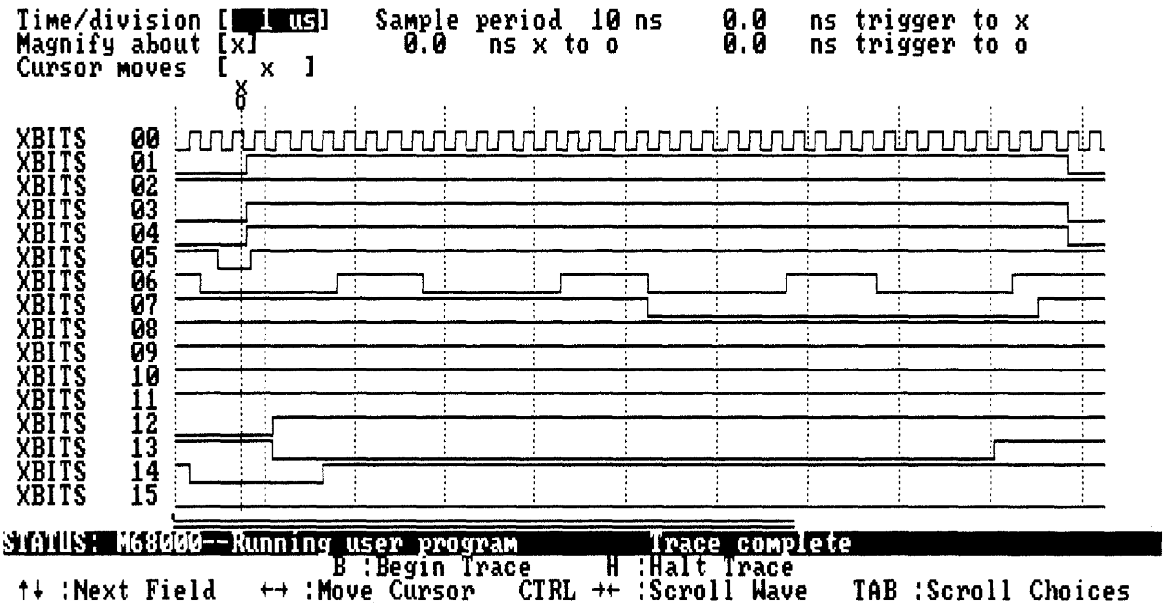
To view a timing waveform display, select:

```
Analysis, Display, External
```

You can exit the waveform display by pressing **ESC**, and you can re-enter the waveform display by selecting the "Analysis Display External" command. When you enter the waveform display, there are several fields which allow you to change how the trace information is displayed. This section describes the waveform display and the features that allow you to change the display. An example timing waveform display is shown in figure 5-9.

Setting the Time per Division

By moving the cursor to the time/division field and using the **Tab** key, you can change the time per division on the display. There are 10 divisions on the display, represented by vertical dotted lines.



```
Time/division [  1 us]   Sample period  10 ns     0.0   ns trigger to x
Magnify about [x]          0.0   ns x to o        0.0   ns trigger to o
Cursor moves  [  x  ]

XBITS   00
XBITS   01
XBITS   02
XBITS   03
XBITS   04
XBITS   05
XBITS   06
XBITS   07
XBITS   08
XBITS   09
XBITS   10
XBITS   11
XBITS   12
XBITS   13
XBITS   14
XBITS   15
STATUS: M68000--Running user program          Trace complete
                         B :Begin Trace      H :Halt Trace
↑↓ :Next Field   ←→ :Move Cursor   CTRL →← :Scroll Wave   TAB :Scroll Choices
```

Figure 5-9. Timing Waveform Display

Scrolling through the Waveform Display

When the time per division is decreased, not all of the trace can be displayed on the screen. However, you can scroll the trace across the screen by pressing the **CTRL** left arrow and **CTRL** right arrow keys. The portion of the trace that is currently displayed is represented by a bar at the bottom of the screen.

Reference Points

Three reference points appear in the waveform display. The first is the trigger point. The two remaining reference points are movable, and you can place them at areas of interest in the trace.

Trigger

The trigger point is represented by the vertical dashed line. The trigger point is the actual trigger condition (pattern, edge, or glitch) plus the trigger delay.

User Defined (X, O)

The user defined reference points can be moved anywhere in the display. These points are also represented by vertical dashed lines, and they are labeled "x" and "o". The top of the waveform display shows time between these two points and the time between them and the trigger.

Cursor moves (X, O, X & O) The third field from the top allows you to select which reference points the cursor will move, "x", "o", or both "x & o".

Magnify About (X or O) Toggling the "display about" value from "x" to "o" causes the waveforms to be displayed about the different reference points.

Inserting/Deleting Channels

The **Insert** key allows you to increase the number of channels included in the waveform display. By inserting channels, you can display a maximum of 16 channels on the display.

Each channel can be used to display any of the defined labels.

The **Delete** key allows you to decrease the number of waveforms included on the display.

Examples

Starting the Timing Measurement

To start a timing measurement, select:

```
Analysis, Begin, External
```

Also, while in the waveform display, you can enter the "B" key to start a timing measurement.

Halting the Trace

To halt a currently running timing measurement, select:

```
Analysis, Halt, External
```

Also, while in the waveform display, you can enter the "H" key to halt a timing measurement.

External Timing Analyzer Specifications

Sample Rate Accuracy = 0.01%

Asynchronous Pattern. Trigger on pattern less than or greater than specified duration. Pattern is logical AND of specified low, high, or "don't care" for each channel. If 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 is 30 ns to 20 ms. Resolution is +/- 10 ns or 0.01%, whichever is greater. Accuracy is (10 ns + 0.01%) to (-30 ns - 0.01%).

Less Than Duration. Range is 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 (30 ns + 0.01%) to (-10 ns - 0.01%).

Delay Accuracy = 0.01% +/- 10 ns

Minimum Detectable Glitch = 5 ns at threshold

Typical Skew < 4 ns

Making Coordinated Measurements

Introduction

Coordinated measurements are measurements synchronously made in multiple emulators or analyzers. Coordinated measurements can be made between HP 64700 Series emulators which communicate over the Coordinated Measurement Bus (CMB). Coordinated measurements can also be made between an emulator and some other instrument connected to the BNC connector.

This chapter will describe coordinated measurements which involve the emulation or external analyzers. These types of coordinated measurements are:

- Starting a trace on reception of the CMB EXECUTE signal.
- Using the analyzer trigger to break emulator execution into the monitor.
- Using the analyzer trigger to drive the CMB or BNC TRIGGER signal.
- Allowing CMB or BNC TRIGGER signals to arm the analyzer.
- Using the external analyzer to arm the emulation analyzer, and vice-versa.

Note

You must use the background emulation monitor to perform coordinated measurements. Refer to your *PC Interface: Emulator User's Guide* for more information on the emulation monitor.

Three signal lines on the CMB are active and serve the following functions when enabled:

TRIGGER	Active low. The analyzer trigger line on the CMB and on the BNC serve the same logical purpose. They provide a means for the analyzer to drive its trigger signal out of the system or for external trigger signals to arm the analyzer or break the emulator into its monitor.
READY	Active high. This line is for synchronized, multi-emulator start and stop. When CMB run control interaction is enabled, all emulators are required to break to background upon reception of a false READY signal and will not return to foreground until this line is known to be in a true state.
EXECUTE	Active low. This line serves as a global interrupt signal. Upon reception of an enabled EXECUTE signal, each emulator is to interrupt whatever it is doing and execute a previously defined process, typically, run the emulator or start a trace measurement.

Tracing at EXECUTE

To specify that an analyzer measurement begin upon reception of the CMB EXECUTE signal, select the following PC Interface command:

```
Analysis, CMB, Begin
```

The trace measurement defined by the current trace specification will be started when the EXECUTE signal becomes active. When the trace measurement begins, you will see the message "ALERT: CMB execute; emulation trace started".

Using the Analyzer Trigger to Break into the Monitor

To cause emulator execution to break into the monitor when the analyzer trigger condition is found, you must modify the trigger configuration. To access the trigger configuration, select:

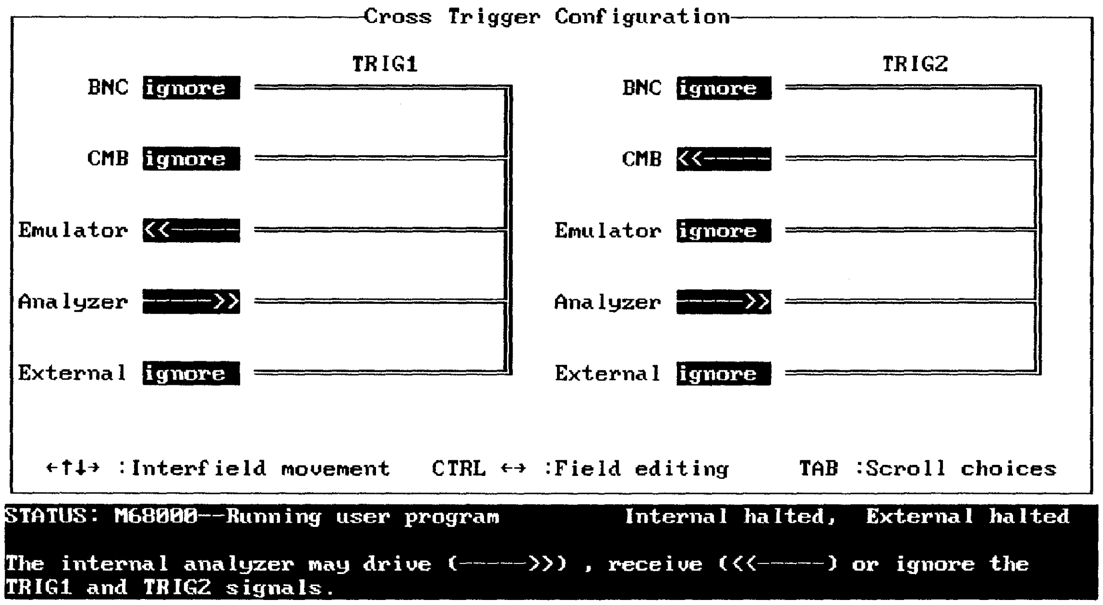
```
Config, Trigger
```

The trigger configuration display contains two diagrams, one for each of the internal TRIG1 and TRIG2 signals.

To use the internal TRIG1 signal to connect the analyzer trigger to the emulator break line, highlight the "Analyzer" field in the TRIG1 portion of the display, and use the **Tab** key to select the single-headed arrow pointing towards TRIG1; this shows that the analyzer is driving TRIG1.

Next, highlight the "Emulator" field and use the **Tab** key to select the arrow pointing towards the emulator; this specifies that emulator execution will break into the monitor when the TRIG1 signal is driven.

The resulting trigger configuration display is shown in figure 6-1.



```
                         Cross Trigger Configuration
                       TRIG1                                  TRIG2
    BNC ignore                                 BNC ignore
    CMB ignore                                 CMB <<-----
Emulator <<-----                          Emulator ignore
Analyzer ----->>                          Analyzer ----->>
External ignore                           External ignore

  ←↑↓→ :Interfield movement    CTRL ←→ :Field editing    TAB :Scroll choices
STATUS: M68000--Running user program        Internal halted,  External halted

The internal analyzer may drive (----->>) , receive (<<-----) or ignore the
TRIG1 and TRIG2 signals.
```

Figure 6-1. Cross Trigger Configuration

Both the emulation analyzer and the external analyzer may drive the same internal signal; in this case, the internal signal will be driven by the analyzer that finds its trigger condition (point) first.

Using the Analyzer Trigger to Drive External Signals

To specify that external TRIGGER signals (either CMB or BNC) become active when the analyzer trigger condition is found, access the trigger configuration by selecting:

```
Config, Trigger
```

To use the internal TRIG2 signal to connect the analyzer trigger to the CMB TRIGGER line, highlight the "Analyzer" field in the TRIG2 portion of the trigger configuration display, and use the

Tab key to select the single-headed arrow pointing towards TRIG2; this shows that the analyzer trigger is driving TRIG2.

Next, highlight the "CMB" field and use the **Tab** key to select the single-headed arrow pointing towards CMB; this shows that the CMB TRIGGER signal becomes active when the TRIG2 signal is driven.

This trigger configuration is also shown in figure 6-1.

Allowing CMB or BNC TRIGGERs to Arm the Analyzer

The two previous examples show how the analyzer can be used to drive signals. You can also specify that the analyzer be armed (turned on, or enabled) on the reception of CMB or BNC TRIGGER signals. (The analyzer is always armed unless one of these signals is specifically used to arm the analyzer.) Until the analyzer is armed, trace measurements cannot be performed.

Arming the Internal Emulation Analyzer

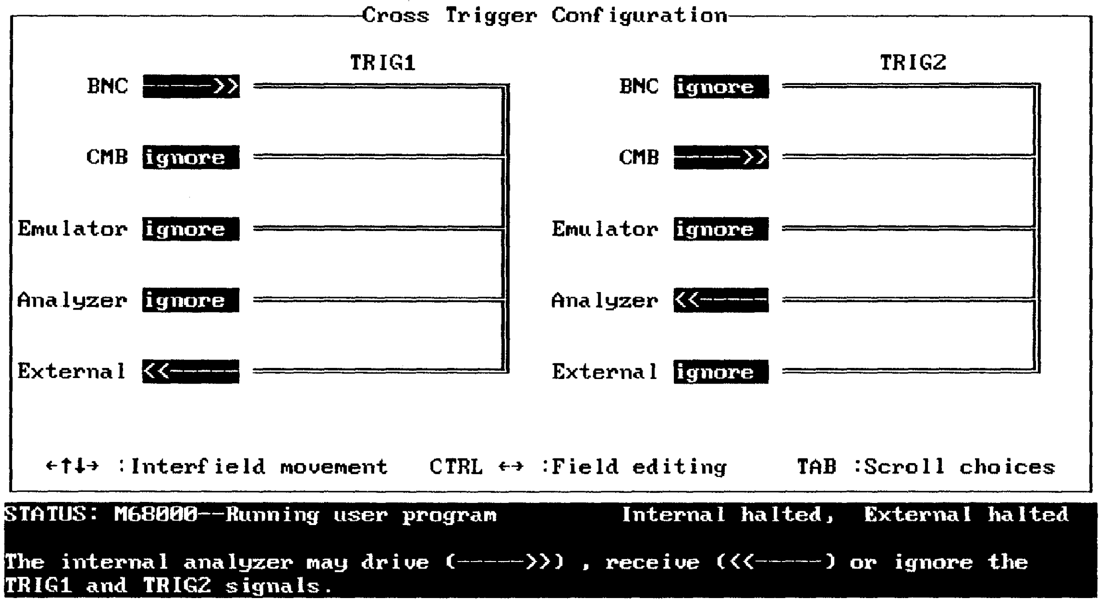
To arm the internal emulation analyzer on reception of an external CMB or BNC TRIGGER signal, select the following command from within the PC Interface:

```
Config, Trigger
```

For example, to specify that the state analyzer be armed by the CMB TRIGGER over TRIG2, highlight the "CMB" field and use the **Tab** key to select the single-headed arrow pointing towards TRIG2; this shows that the TRIG2 signal is driven when the CMB TRIGGER signal becomes active.

Next, highlight the "Analyzer" field in the TRIG2 portion of the display, and use the **Tab** key to select the arrow which points towards "Analyzer"; this shows that the analyzer trigger receives TRIG2.

This trigger configuration is shown in figure 6-2.



```
                         Cross Trigger Configuration
                          TRIG1                                    TRIG2
      BNC  ----->>                              BNC  ignore
      CMB  ignore                               CMB  ----->>
 Emulator  ignore                          Emulator  ignore
 Analyzer  ignore                          Analyzer  <<-----
 External  <<-----                         External  ignore

  ←↑↓→ :Interfield movement    CTRL ←→ :Field editing     TAB :Scroll choices
STATUS: M68000--Running user program         Internal halted,   External halted

The internal analyzer may drive (----->>) , receive (<<-----) or ignore the
TRIG1 and TRIG2 signals.
```

Figure 6-2. Receiving External Signals

In the internal emulation analyzer trace specification display, you can use the "arm" resource in any of the state qualifier fields. For example, if "arm" is shown in the "trigger on" field, the analyzer triggers when it is armed.

Arming the External Analyzer

To arm the external analyzer on reception of an external CMB or BNC TRIGGER signal, select the following command from within the Timing Interface:

```
Config, Trigger
```

For example, to specify that the timing analyzer be armed by the BNC TRIGGER over TRIG1, highlight the "BNC" field and use the **Tab** key to select the single-headed arrow pointing towards TRIG1; this shows that the TRIG1 signal is driven when the BNC TRIGGER signal becomes active.

Next, highlight the "External" field in the TRIG1 portion of the display, and use the **Tab** key to select the arrow which points towards "External"; this shows that the analyzer trigger receives TRIG1.

This trigger configuration is shown in figure 6-2.

In the external state analyzer trace specification display, you can use the "arm" resource in any of the state qualifier fields. For example, if "arm" is shown in the "trigger on" field, the analyzer triggers when it is armed.

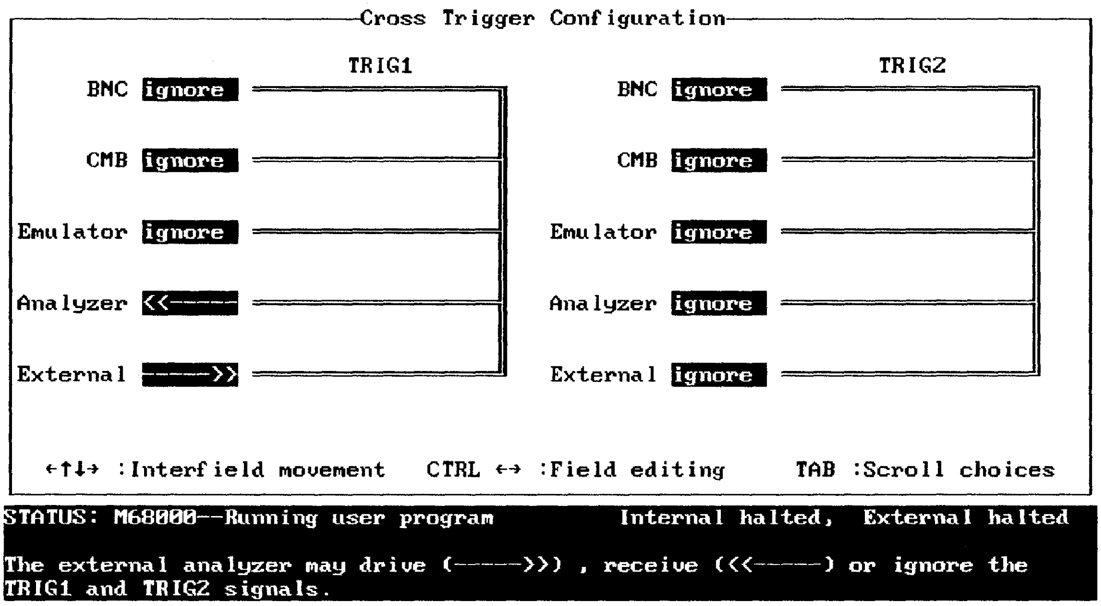
When the external analyzer is configured as a timing analyzer and is to be armed by an external CMB or BNC TRIGGER signal, the timing trace specification display is updated to show "Armed by External". The timing measurement begins when the arm signal is received.

Using One Analyzer to Arm the Other

You can use the trigger condition of the external analyzer to arm the emulation analyzer, and vice-versa. To make a specification like this, you also select:

```
Config, Trigger
```

For example, in figure 6-3, the TRIG1 signal is used to arm the internal analyzer when the external analyzer's trigger condition is found.



```
                              Cross Trigger Configuration
                          TRIG1                                     TRIG2
   BNC  ignore                                  BNC  ignore

   CMB  ignore                                  CMB  ignore

Emulator ignore                              Emulator ignore

Analyzer <<-----                             Analyzer ignore

External ----->>                             External ignore

  ←↑↓→ :Interfield movement   CTRL ←→ :Field editing     TAB :Scroll choices
STATUS: M68000--Running user program        Internal halted,  External halted

The external analyzer may drive (----->>) , receive (<<-----) or ignore the
TRIG1 and TRIG2 signals.
```

Figure 6-3. Using One Analyzer to Arm the Other

Other Trigger Combinations

You may have noticed that the CMB or BNC TRIGGER signals may be driven by the internal TRIG1 and TRIG2 signals, be received by external sources, or both. When a CMB or BNC signal is received from an external source, it may be used to break emulator execution into the monitor. When a CMB or BNC signal is both driven and received (double-headed arrow shown in the associated field), you can set up the emulation analyzer to drive the internal signal; if the internal signal drives the emulator break line, the emulator will break into the monitor on either the analyzer trigger or on the reception of an external TRIGGER signal.

When an emulator break occurs due to the analyzer trigger, the analyzer will stop driving the internal signal that caused the break. Therefore, if TRIG2 is used both to break and to drive the CMB TRIGGER (for example), TRIGGER will go true when the trigger is found and then will go false after the emulator breaks. However, if TRIG1 is used to cause the break and TRIG2 is used to drive the CMB TRIGGER, TRIGGER will stay true until the trace is halted or until the next trace starts.

Notes

Index

Notes

HEWLETT
PACKARD

Hewlett-Packard
Printed in the USA