Help Volume

© 1992-2001 Agilent Technologies. All rights reserved.

System: Measurement Examples

Measurement Examples

These quick reminders on how to perform common measurements are grouped by the development phase in which the measurement typically occurs:

- "Hardware Turn-On" on page 10 — Hardware designers take a loaded first cut printed-circuit board and verify its basic operation before delivering it to driver writers and software developers.
- "Firmware Development" on page 143 — Given a printed-circuit board that has been turned-on, firmware developers create drivers and operating system calls that control and communicate with the hardware. They deliver stable hardware with a low-level software interface to application software developers.
- "Software Development" on page 198 — Given stable hardware and low-level driver software, software developers verify real-time application software execution.
- "System Integration" on page 257 — When system problems are discovered, system integrators determine whether the problem is being caused by hardware, software, or both. Also, they analyze system performance.

See Also

"Contents" on page 6

"Measurement Tips & Tricks" on page 300

Main System Help (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Glossary (see page 311)

Contents

1

Measurement Examples

Contents

Hardware Turn-On

Hardware Turn-On

Looking at Signal Parameters

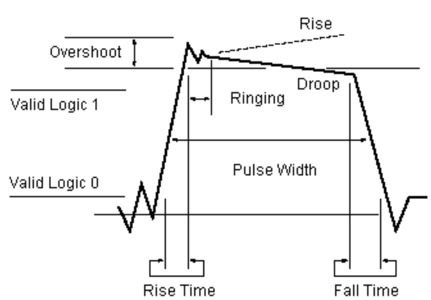
Looking at Signal Edges, Patterns, and Glitches

Measuring Conformance to Specifications

Looking at State Events

Looking at Signal Parameters

To make basic oscilloscope measurements



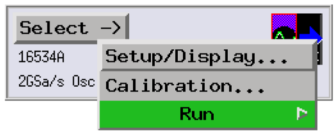
Possible uses:

- To measure the analog parameters of signals.
- To trace to the root cause of noise, crosstalk, or ground bounce problems when combined with a logic analyzer.

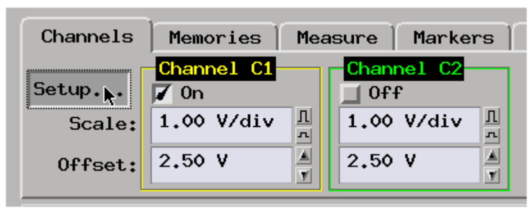
Probing the Target System

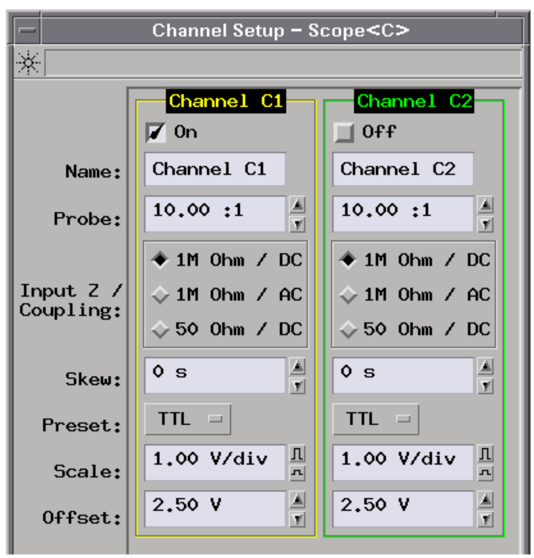
1. Connect the oscilloscope channel probes to signals of interest in the target system.

2. Display the oscilloscope window.



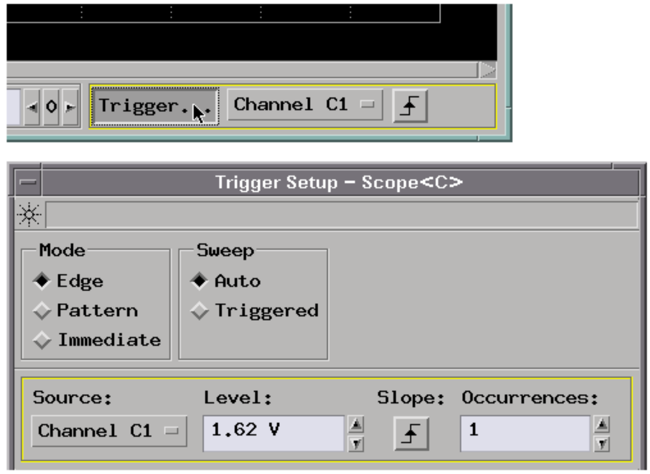
3. Select the Channels tab, and set up the channels.





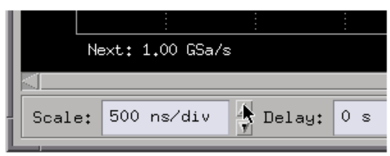
Capturing the Data

1. Set up the trigger.



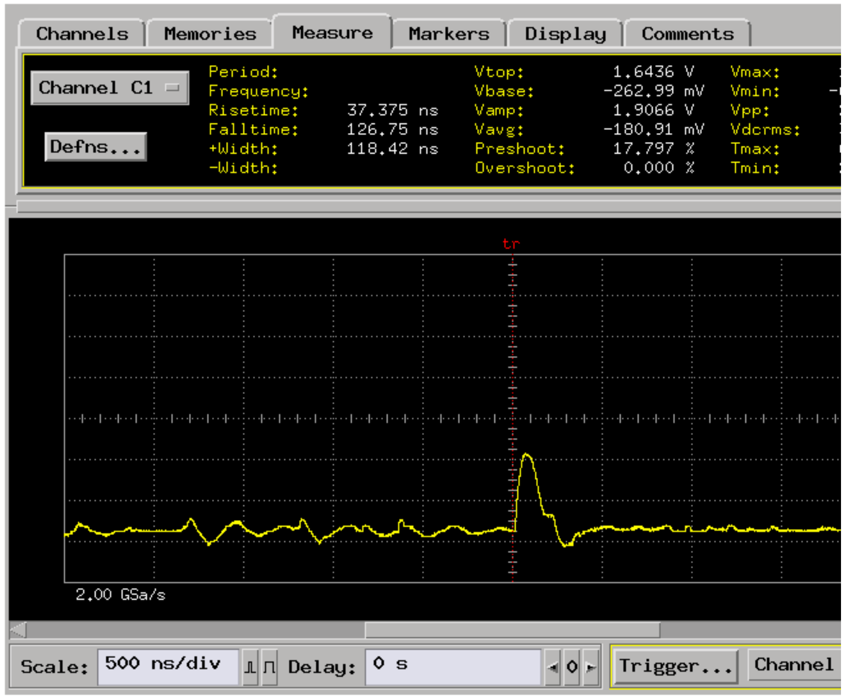
2. Select the Run button to capture an oscilloscope trace.

   You may want to change the time/div scale and select the Run button again to capture data with a different sample rate.

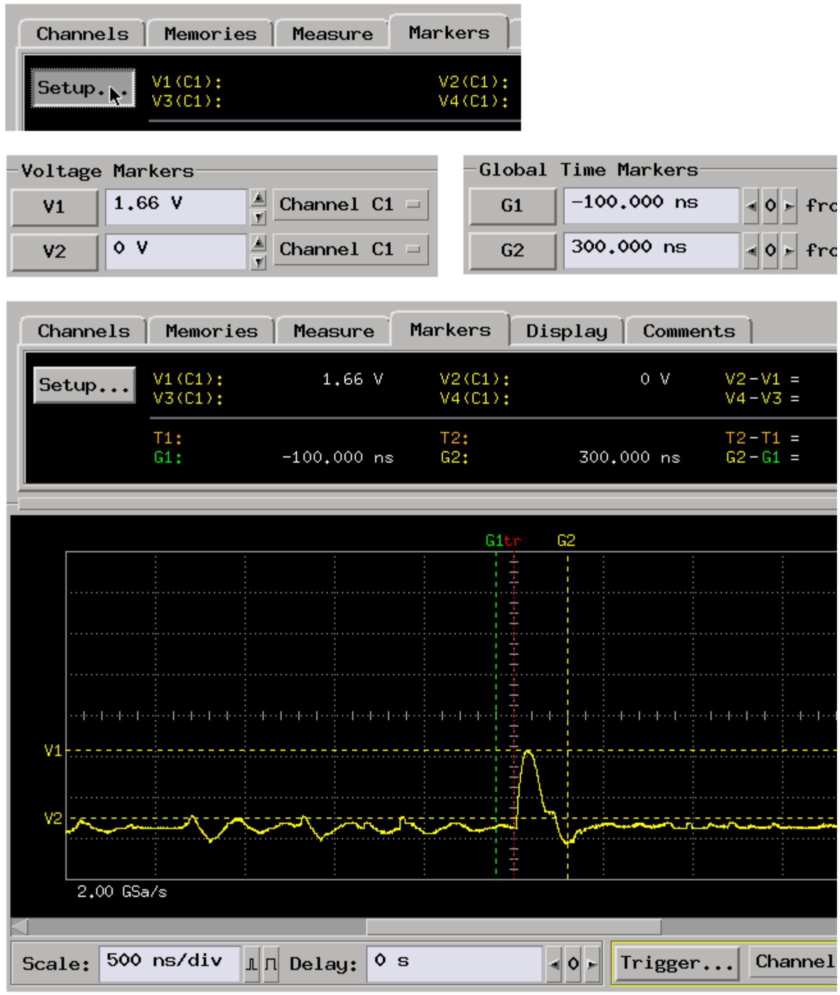


Displaying the Data

1. Select the Measure tab to view the data on the analog parameters of the captured signal.



2. Select the Markers tab to set up voltage and time markers on the display.



See Also

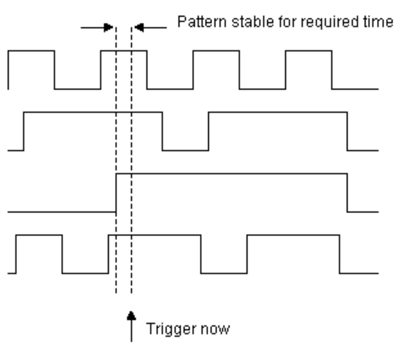
"To capture software execution when a scope triggers" on page 258

"To arm an oscilloscope when the analyzer triggers" on page 277

"To trigger an oscilloscope when a source line executes" on page 294

Looking at Signal Edges, Patterns, and Glitches

To trigger on a stable pattern

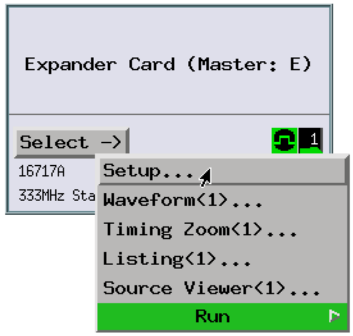


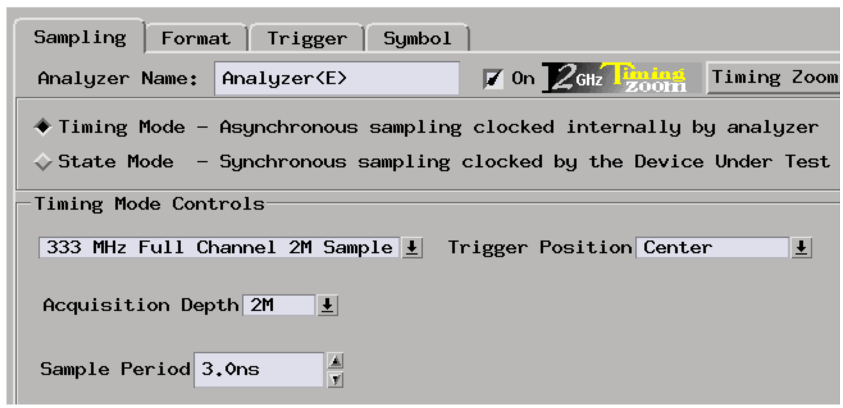
Possible uses:

- To wait for all status lines to finish transitioning before triggering.
- To filter out spurious triggers because of transitions that occur when the target system's state machine is indeterminate.

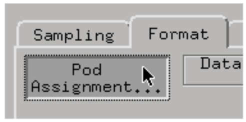
Probing the Target System

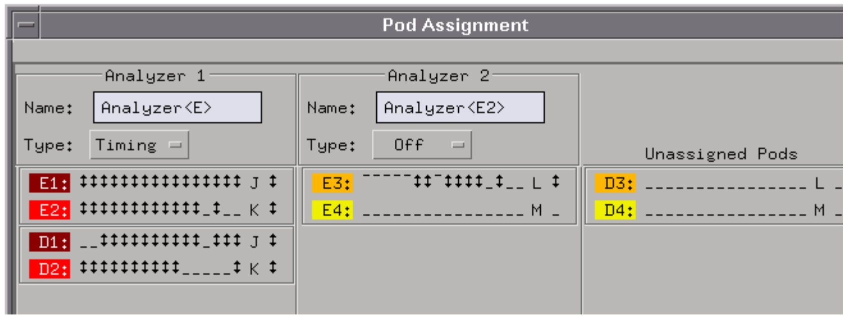
1. Connect the logic analyzer probes to the signals on which you will look for the pattern.
2. Configure a timing analysis machine.



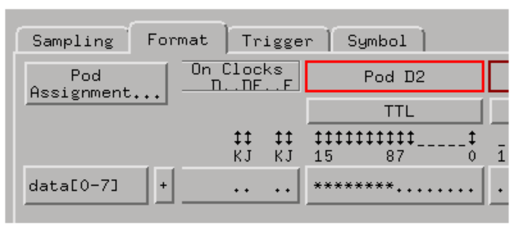


3. Assign pods if necessary.



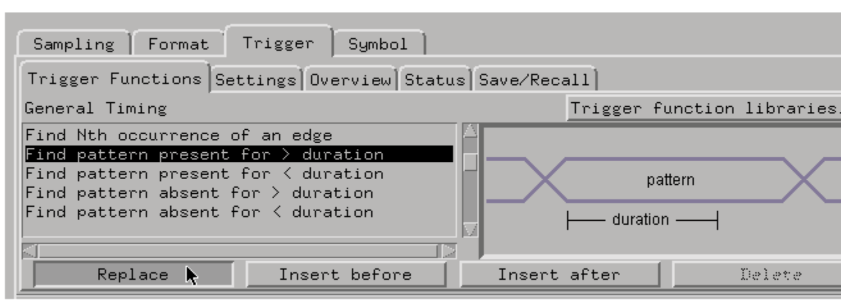


4. Format a label for the signals on which you will look for a stable pattern.

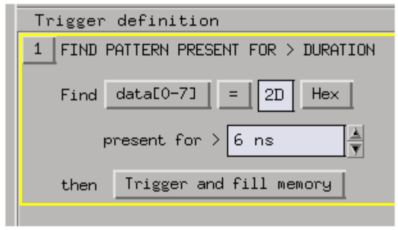


Capturing the Data

1. Use the "Find pattern present for > duration" trigger function.



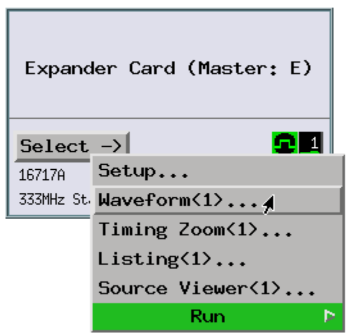
2. In the trigger definition, specify the pattern, and enter the time that pattern must be stable for.

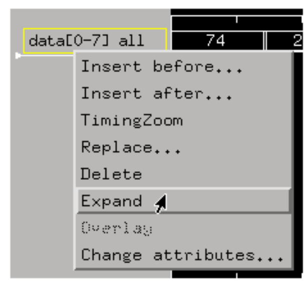


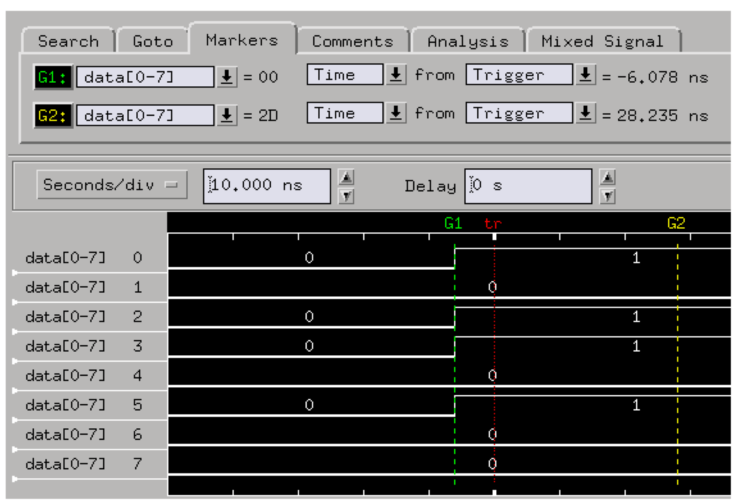
3. Select the Run button to start the measurement.

Displaying the Data

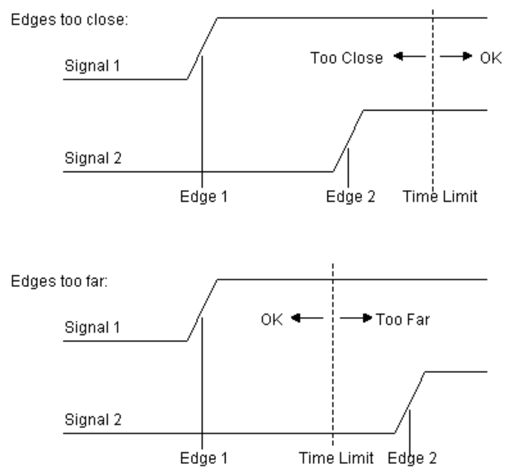
1. Use the Waveform display to verify that the pattern was stable for the specified time before the trigger.







To find edges that are too close or too far

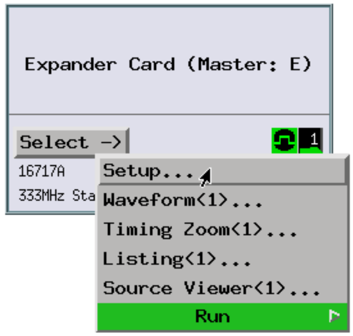


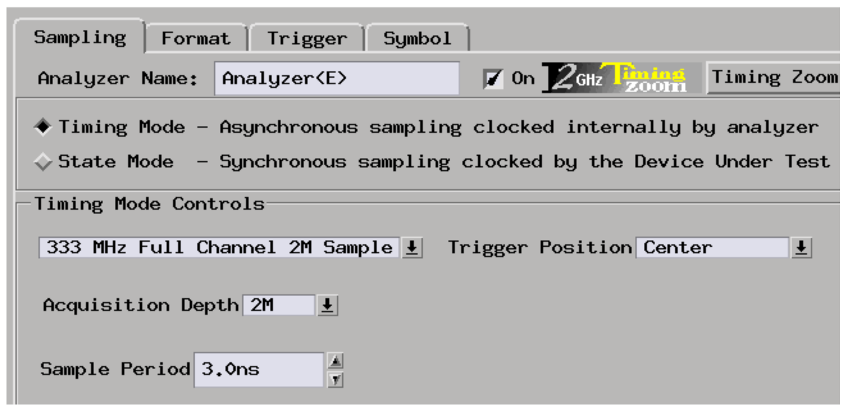
Possible uses:

- To check DRAM row/column address strobe timing.

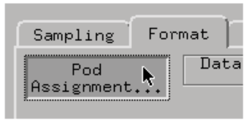
Probing the Target System

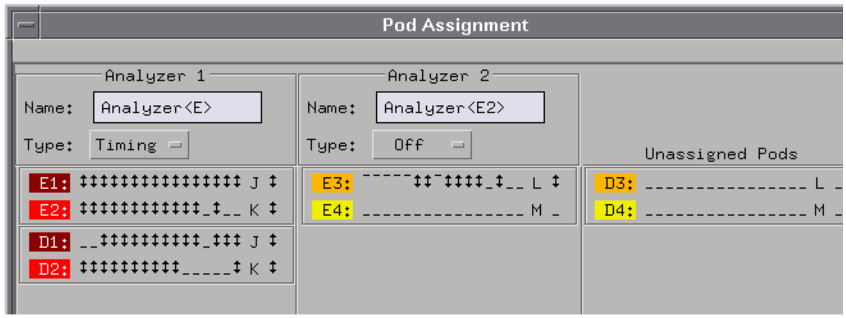
1. Connect logic analyzer probes to the signals whose edges you wish to look at.
2. Configure a timing analysis machine.



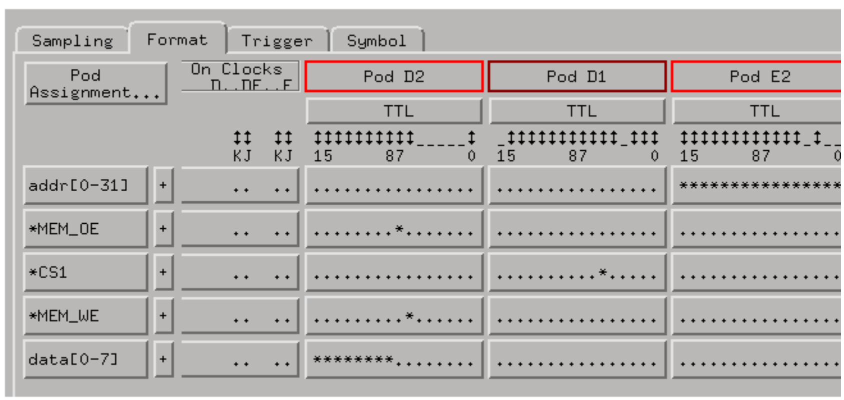


3. Assign pods if necessary.



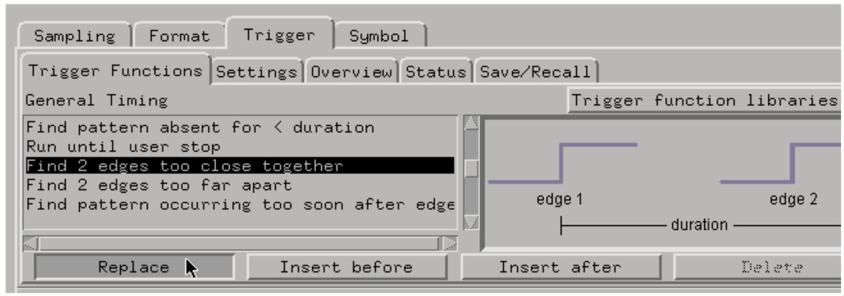


4. Format labels for the signals of interest.

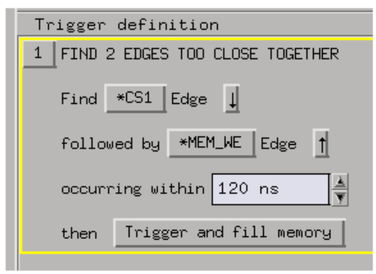


Capturing the Data

1. Use the "Find 2 edges too close together" or the "Find 2 edges too far apart" trigger function.



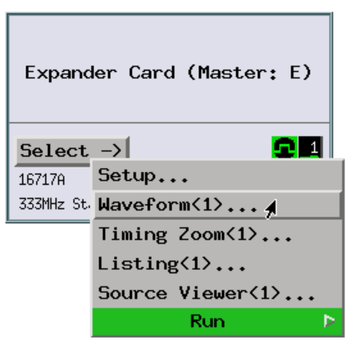
2. In the trigger definition, select edges and time.

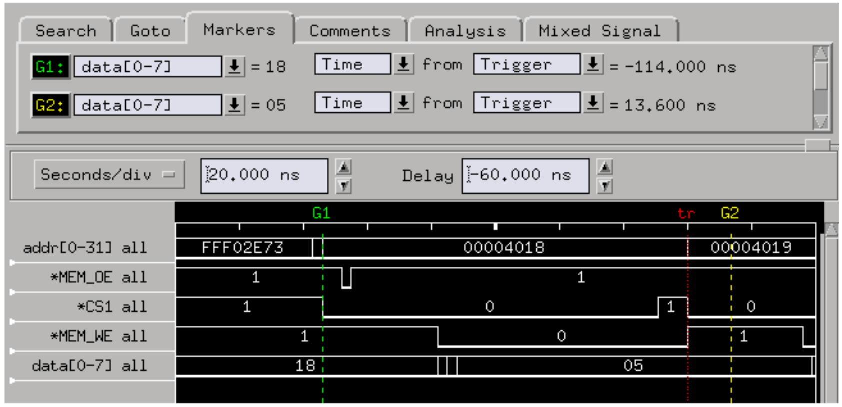


3. Select the Run button to start the measurement.

Displaying the Data

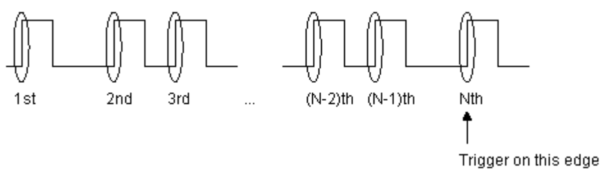
1. Open the Waveform display and use the global markers to show the time between the edges.





See Also "Use trigger functions for easy measurement set up" on page 305

To find the Nth transition of a signal

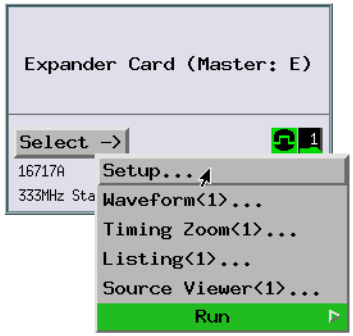


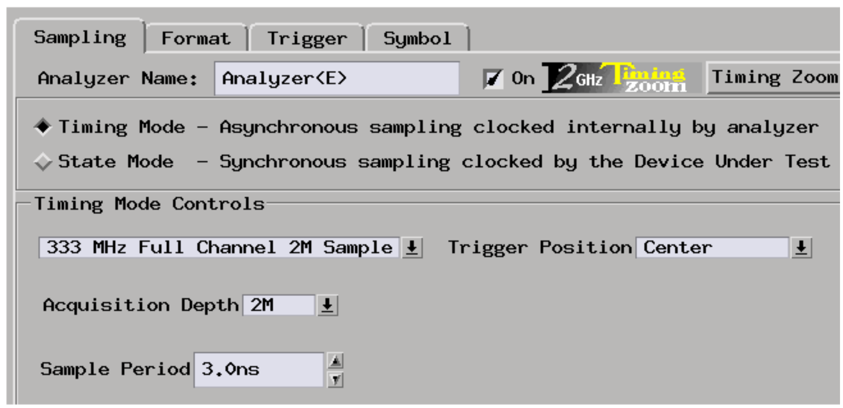
Possible uses:

- To find the 3rd occurrence of the start of a data transfer.
- To find the 1000th occurrence of a chip select line being asserted.

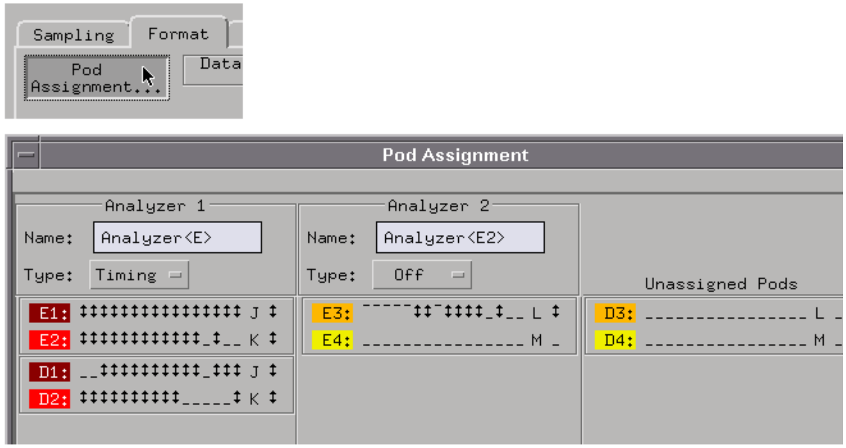
Probing the Target System

1. Connect a logic analyzer probe to the signal of interest.
2. Configure a timing analysis machine.

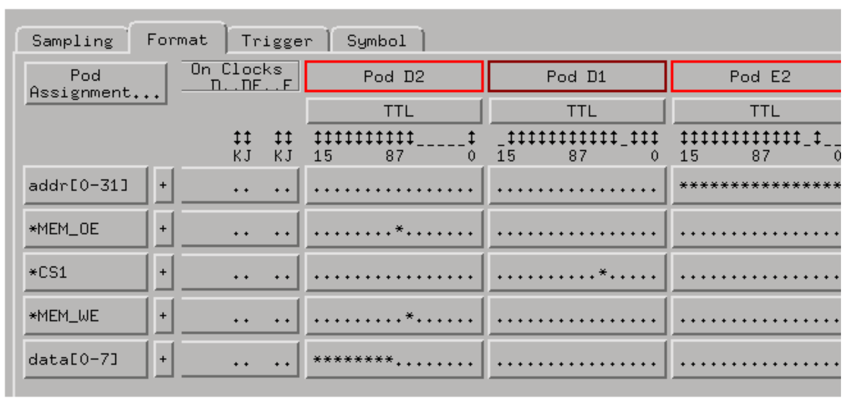




3. Assign pods if necessary.

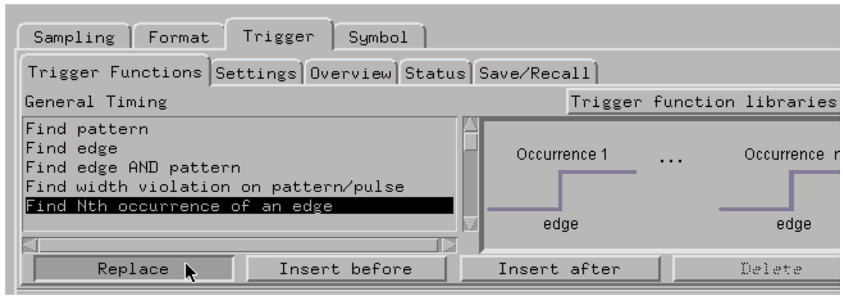


4. Format labels for the signals of interest.

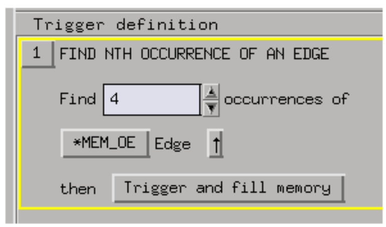


Capturing the Data

1. Use the "Find Nth occurrence of an edge" trigger function.



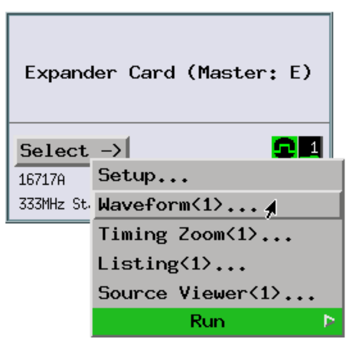
2. In the trigger definition, select the edge label and number of occurrences.

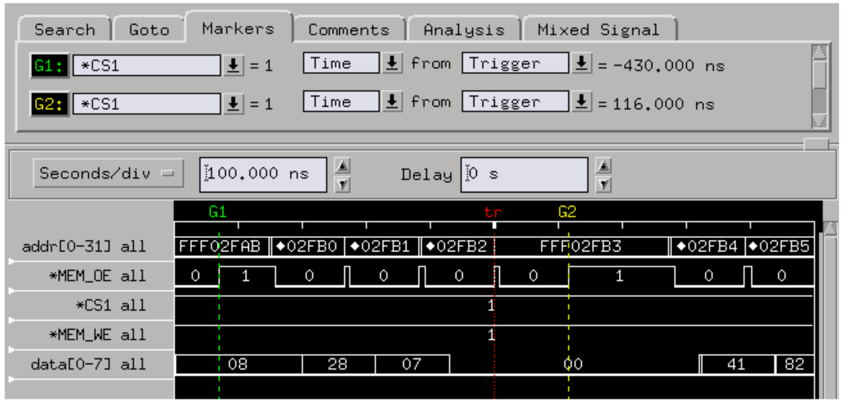


3. Select the Run button to start the measurement.

Displaying the Data

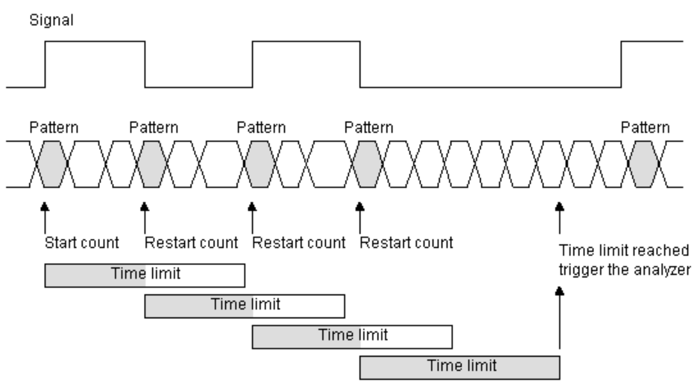
1. In the Waveform window (depending on the time between signal transitions) you may be able to see that you've triggered on the Nth transition of the signal.





See Also "Use trigger functions for easy measurement set up" on page 305

To find when a signal or pattern stops



You can count time by counting occurrences of sampled data or by using a timer.

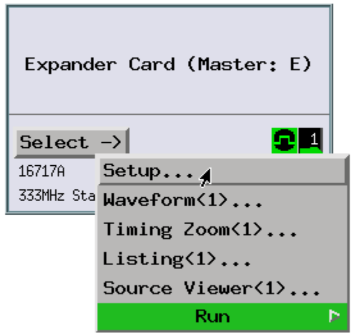
Possible uses:

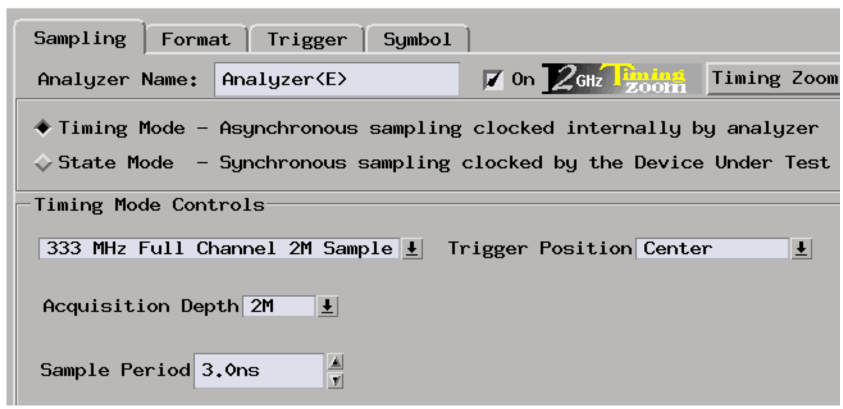
- To find when signals are inactive for too long a time.
- To check when execution leaves an address range.
- To check when expected variable values stop being written.

- To capture what leads up to an unexpected condition.

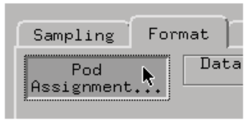
Probing the Target System

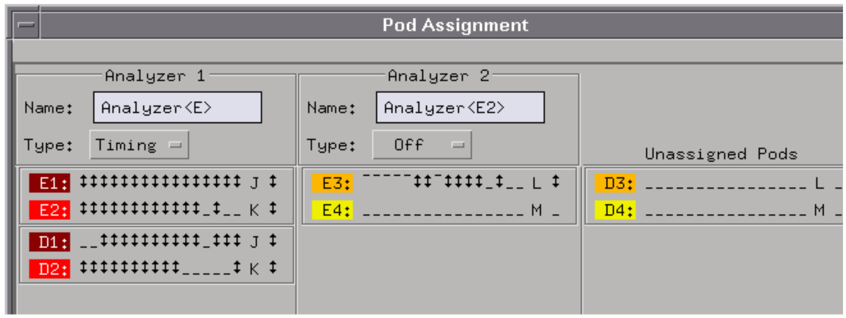
1. Configure a timing machine to look at signal edges or patterns.



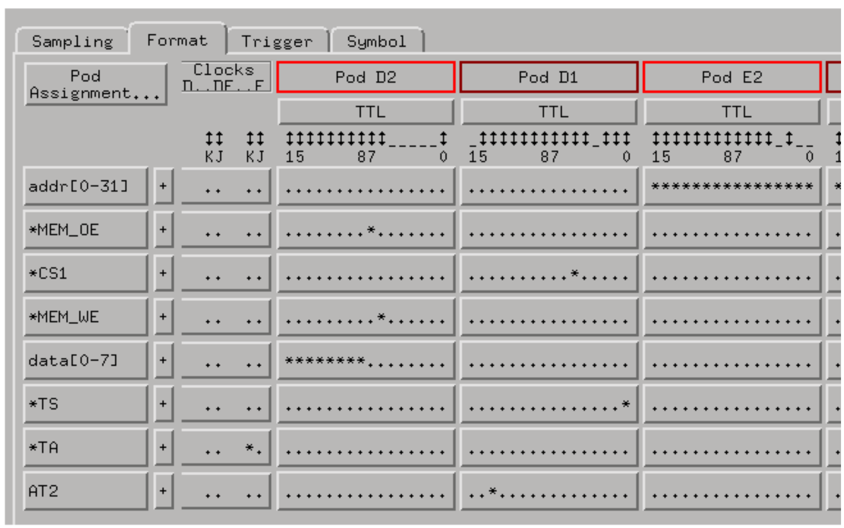


2. Assign pods if necessary.



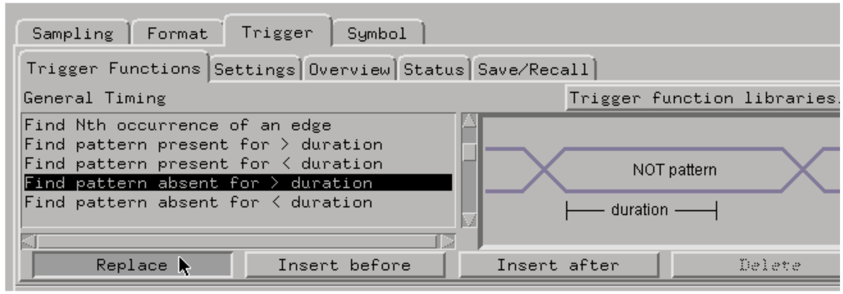


3. Format labels for the signals or patterns of interest.

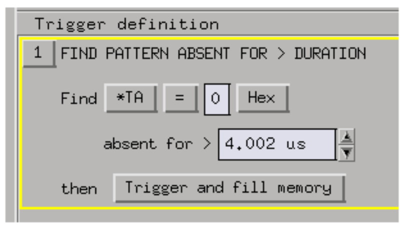


Capturing the Data

1. Use the "Find pattern absent for > duration" trigger function.



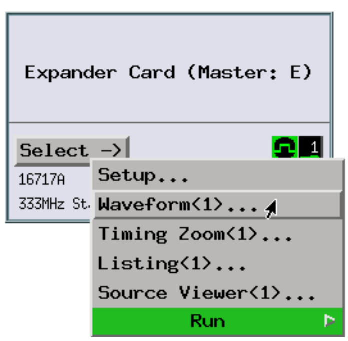
2. In the trigger definition, specify the pattern and the amount of time absent.

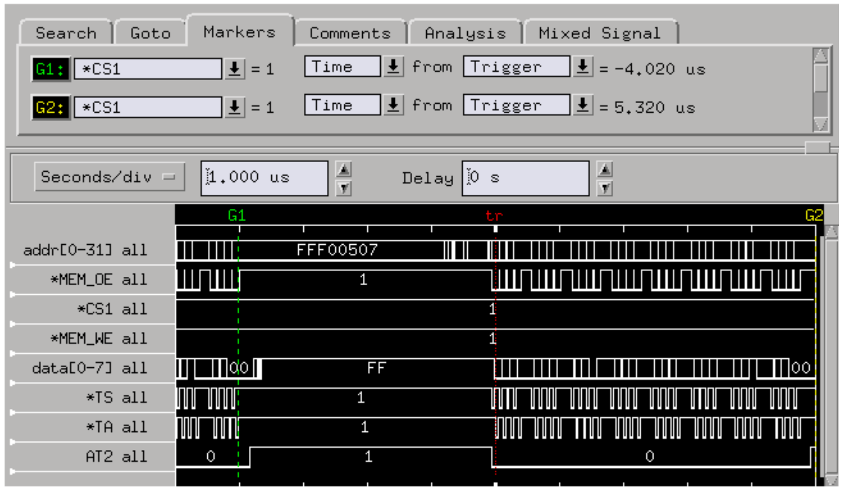


3. Select the Run button to start the measurement.

Displaying the Data

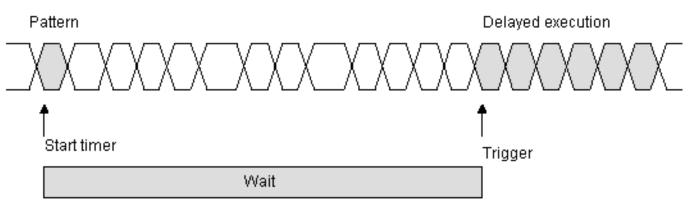
1. In the Waveform display (depending on the time between the edge/pattern and the trigger) you may be able to see the last time the edge/pattern occurred.





See Also "Use trigger functions for easy measurement set up" on page 305

To delay capture after a pattern

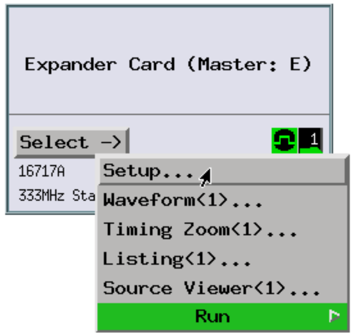


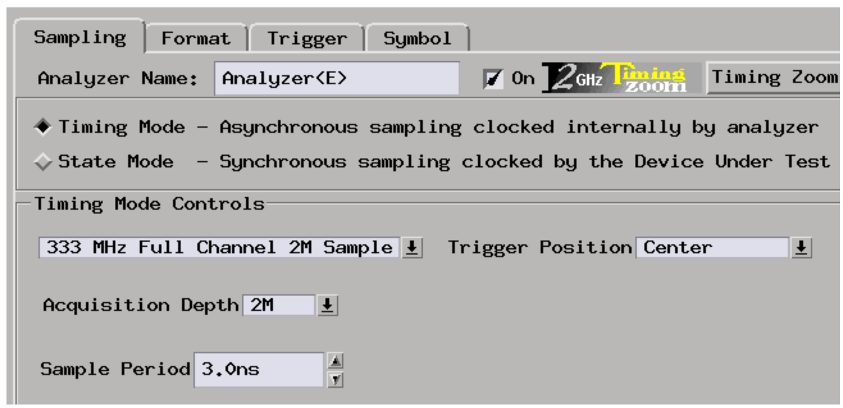
Possible uses:

- To hold off the trigger and look at control signals later than when the address bus pattern becomes invalid.
- To look for a receiver's response which is supposed to occur 3 milliseconds after a transmission.

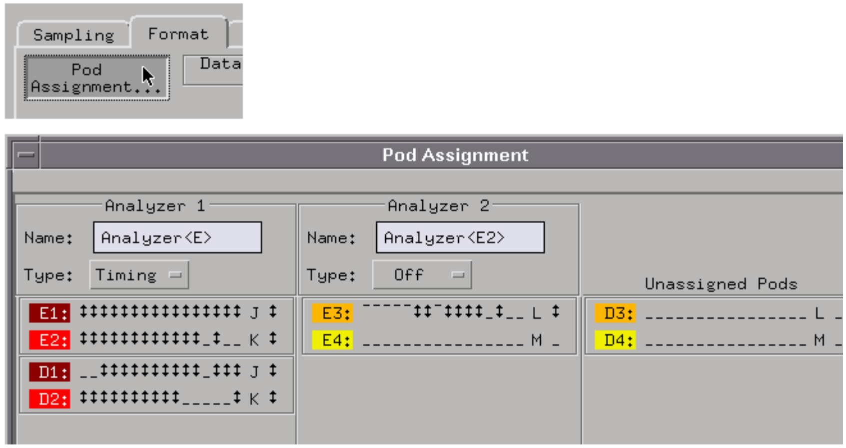
Probing the Target System

1. Configure a timing analysis machine.

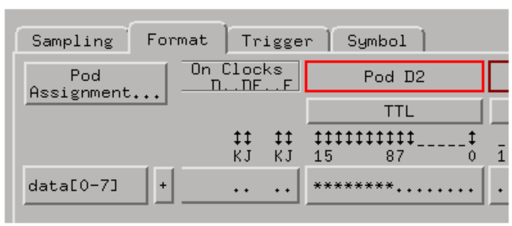




2. Assign pods if necessary.

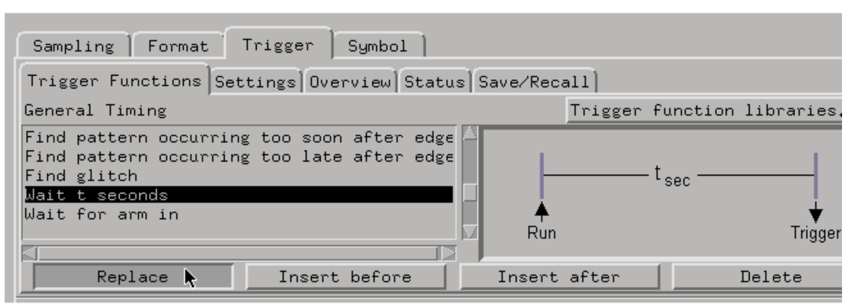


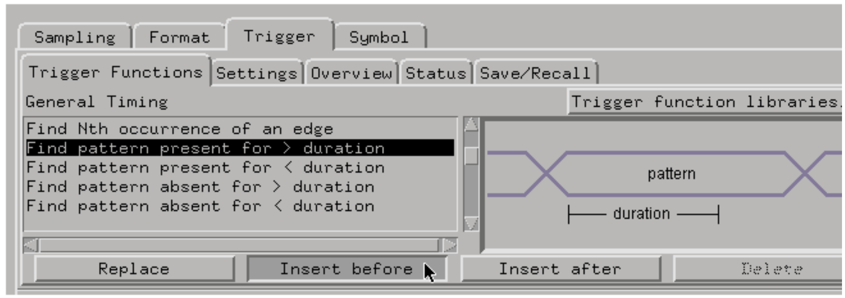
3. Format a label for the signals on which you will look for a stable pattern.



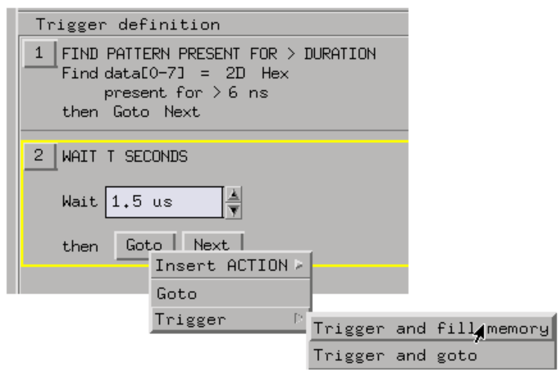
Capturing the Data

1. Build a two level trigger setup using the "Find pattern present for > duration" and "Wait t seconds" trigger functions.





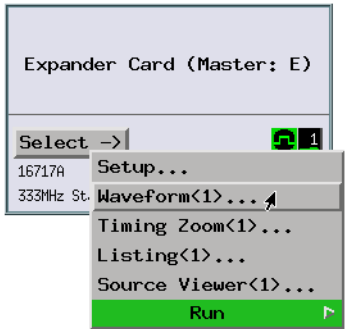
2. In the trigger definition, specify the pattern and the amount of time present. Then, enter the amount of time to wait.

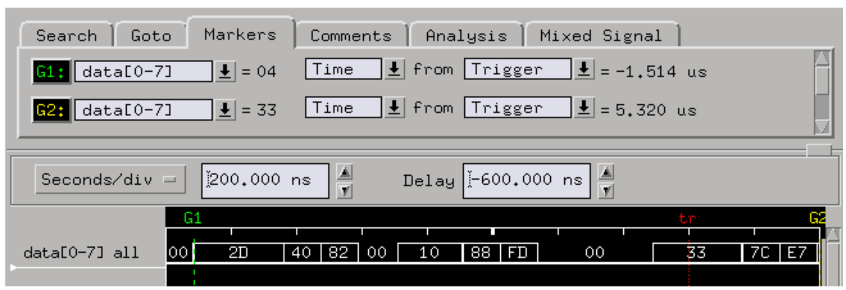


3. Select the Run button to start the measurement.

Displaying the Data

1. In the Waveform display (depending on the time between the pattern and the trigger) you may be able to see the last time the pattern occurred.

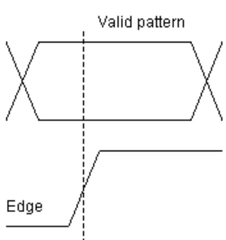




See Also

"Use trigger functions for easy measurement set up" on page 305

To find an edge during a valid pattern

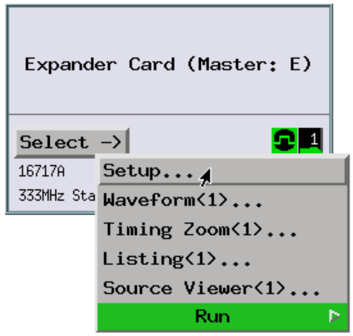


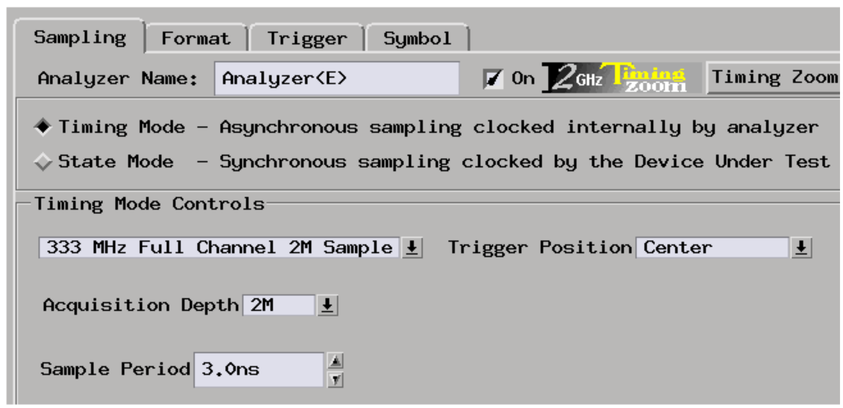
Possible uses:

- To capture a memory chip's select line at a given address.
- To view the timing of a write signal to a peripheral.

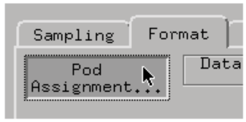
Probing the Target System

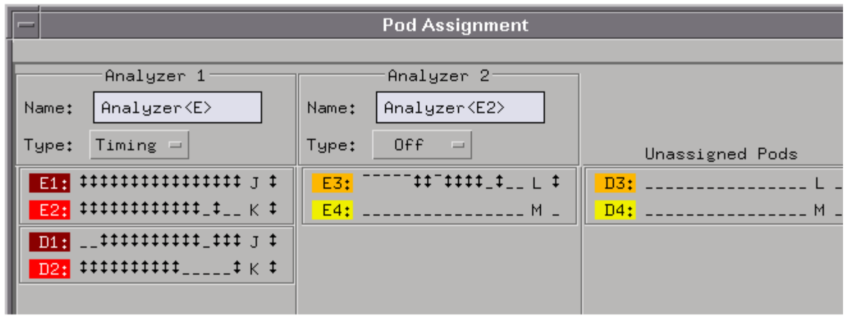
1. Configure a timing analysis machine.



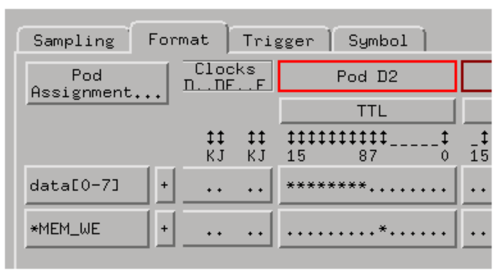


2. Assign pods if necessary.



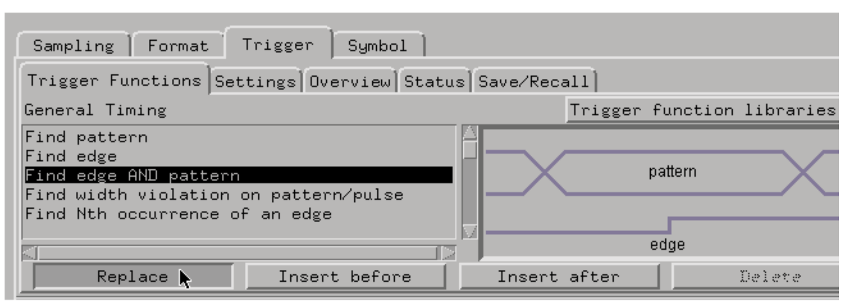


3. Format one label for the signals on which you will look for a pattern and another label for the signal on which you will look for the edge.

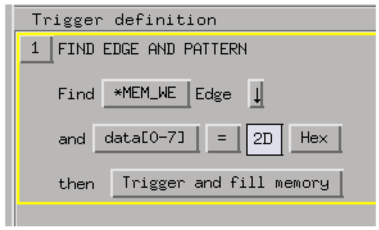


Capturing the Data

1. Use the "Find edge AND pattern" trigger function.



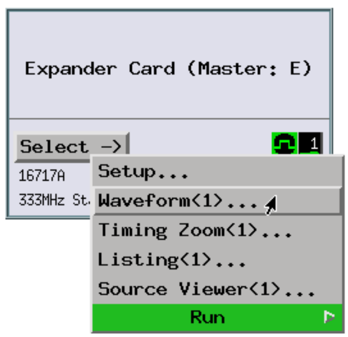
2. In the trigger definition, specify the edge and the pattern.

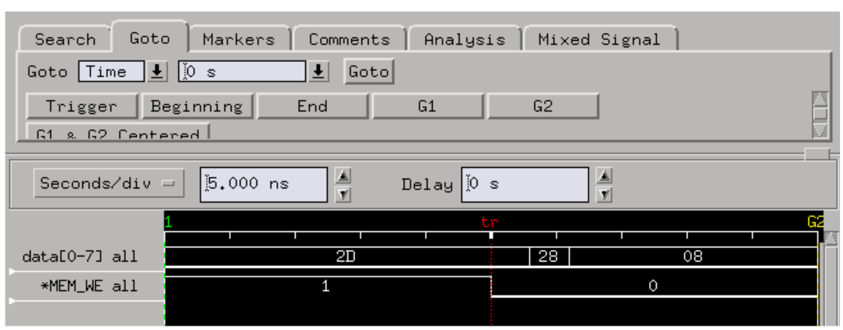


3. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Waveform display to show the edge in relation to the pattern.





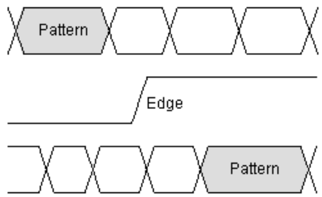
If the analyzer never triggers, it could mean the pattern was never found or the edge never occurs when the pattern is valid.

See Also

"Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

To find a pattern, an edge, and another pattern

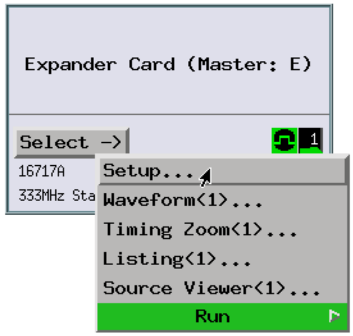


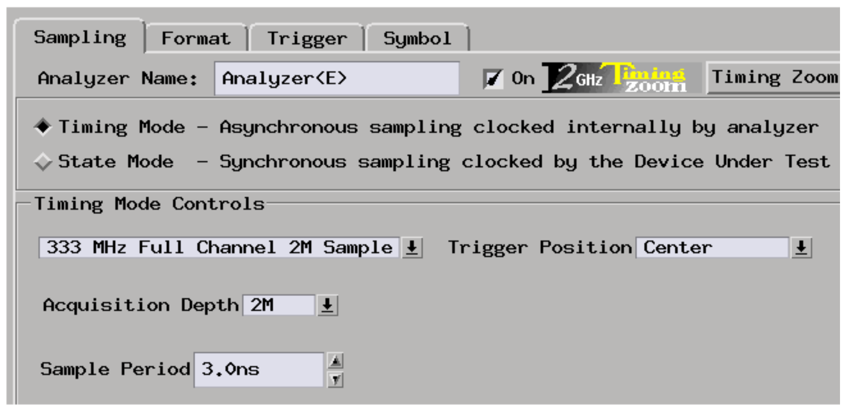
Possible uses:

- To view a correct address bus, control signal, data bus sequence.
- To check whether a data packet was sent, a handshake signal followed, and an acknowledgement was returned.

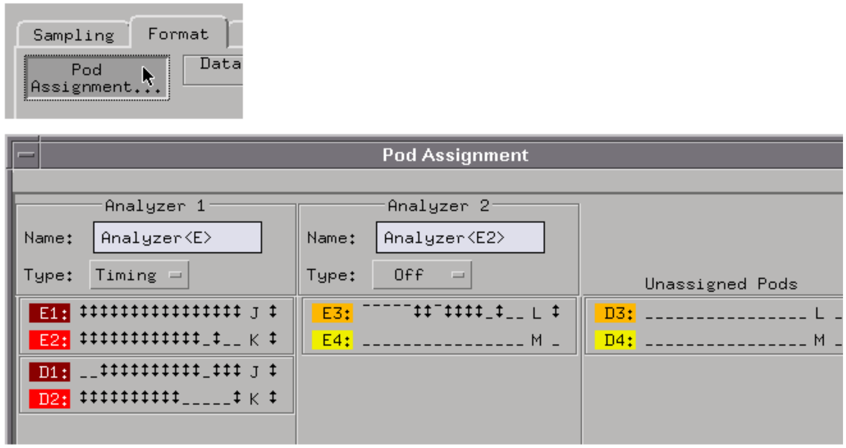
Probing the Target System

1. Configure a timing analysis machine.

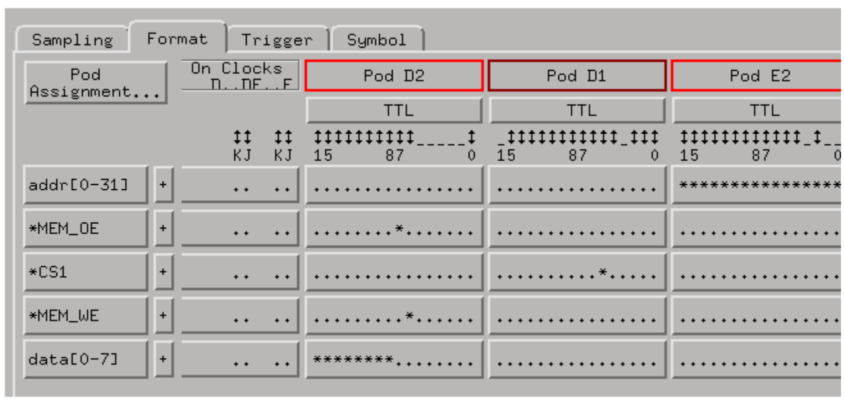




2. Assign pods if necessary.

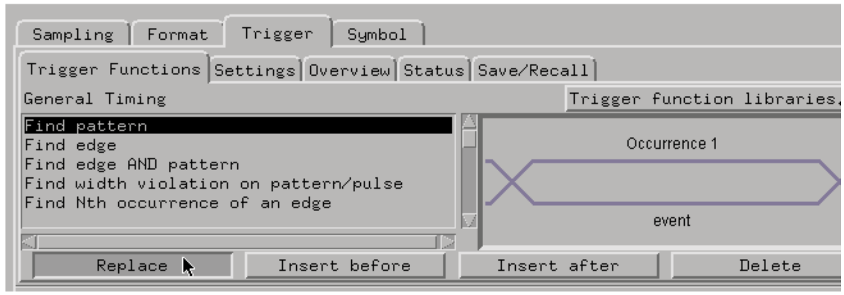


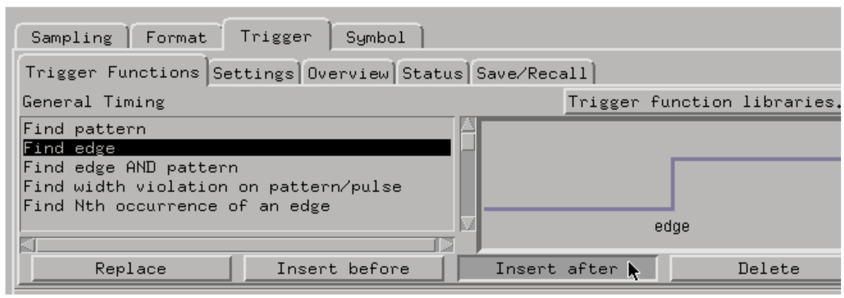
3. Format labels for the signals on which you will look for the edge and patterns.

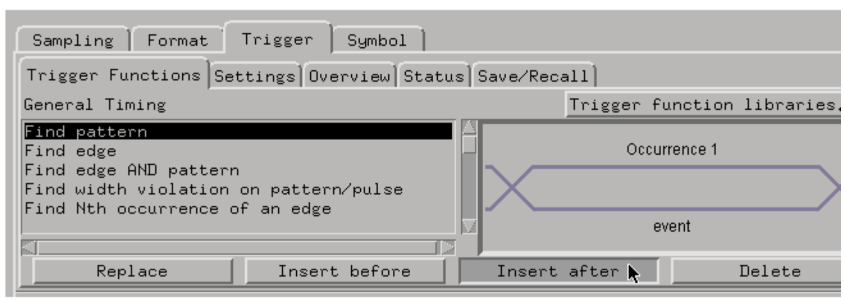


Capturing the Data

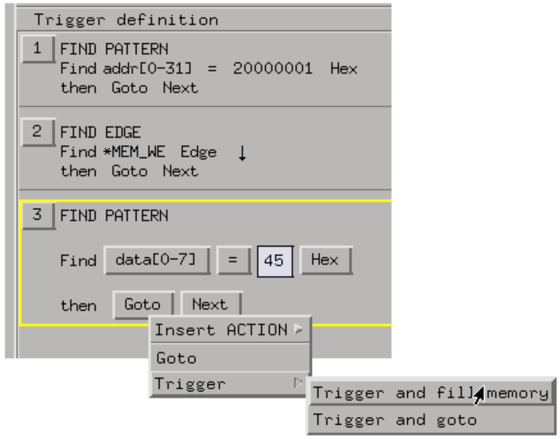
1. Build a three level trigger setup using the "Find pattern", "Find edge", and "Find pattern" trigger functions.







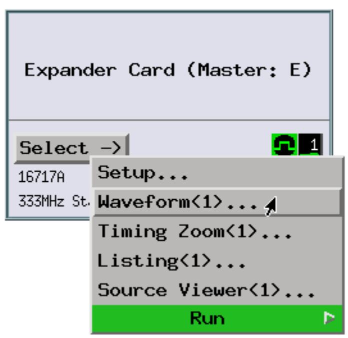
2. In the trigger definition, specify the first pattern and the amount of time present. Then, specify the edge. Finally, specify the second pattern and the amount of time present.

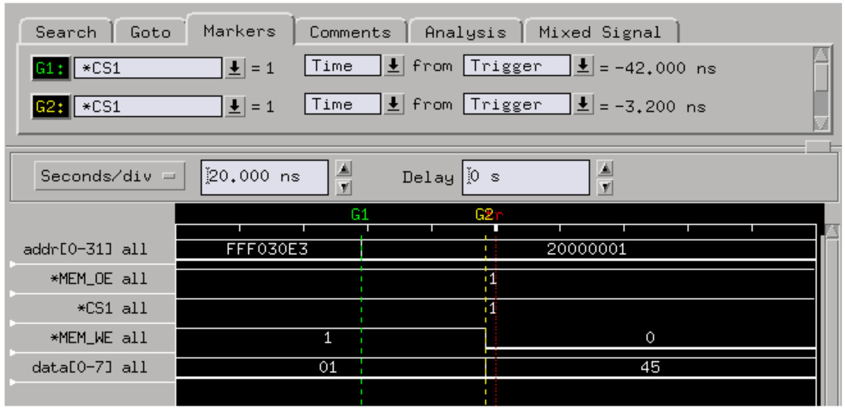


3. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Waveform display to show the proper sequence was captured.





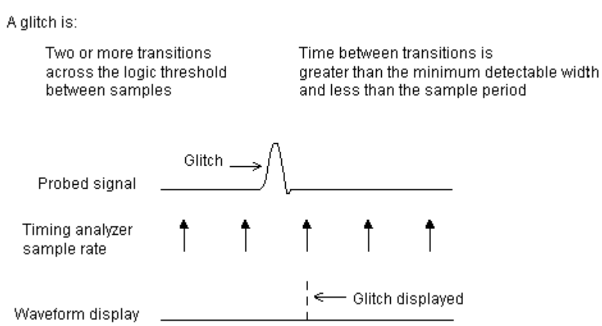
If the analyzer never triggers, the proper sequence does not occur. Depending on the level that was reached in the sequence above, you will need to set up a different trigger to see what actually occurs.

See Also

"Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

To find signal glitches



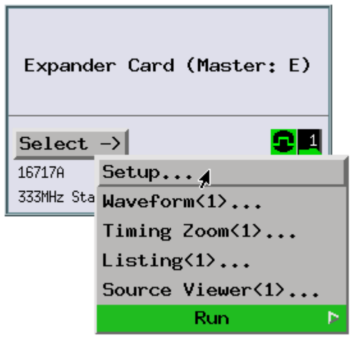
Possible uses:

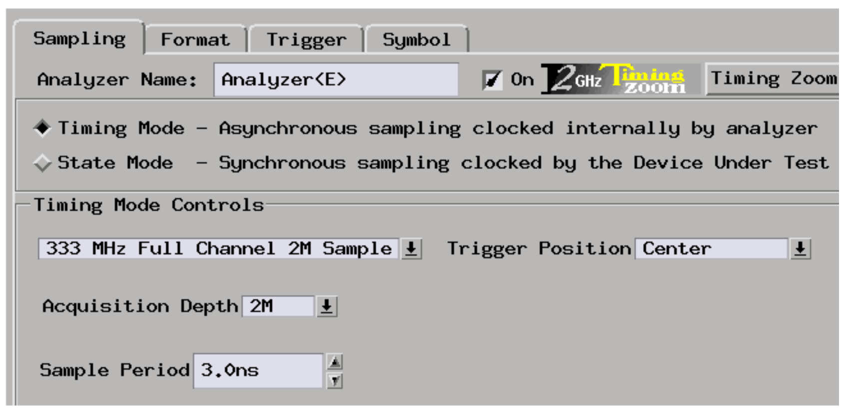
- To look for pulses more narrow than the minimum pulse width.

Probing the Target System

1. Connect the logic analyzer probes to the signals on which you will look for glitches.

2. Configure a timing analysis machine.



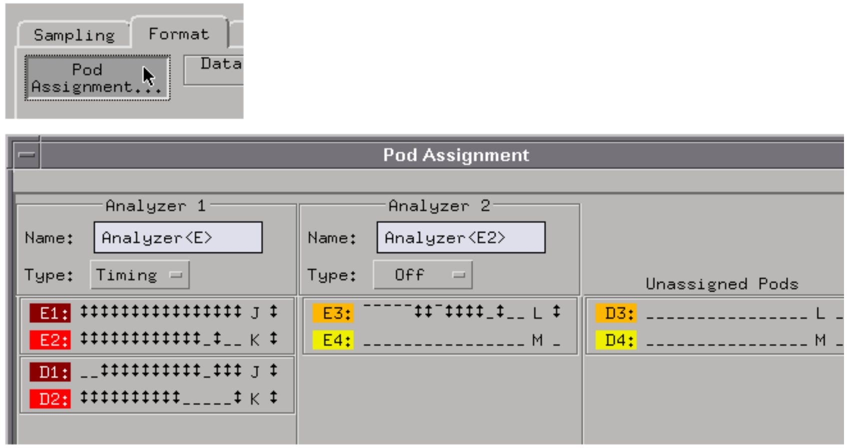


The sample period will specify what is interpreted as a glitch.

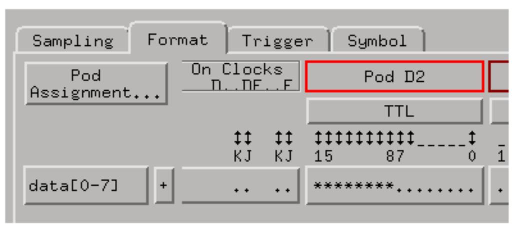
If the logic analyzer has a special acquisition mode for capturing glitches, select that mode.

NOTE: You must select the glitch capture mode in order to see the glitch symbol in the Waveform display.

3. Assign pods if necessary.



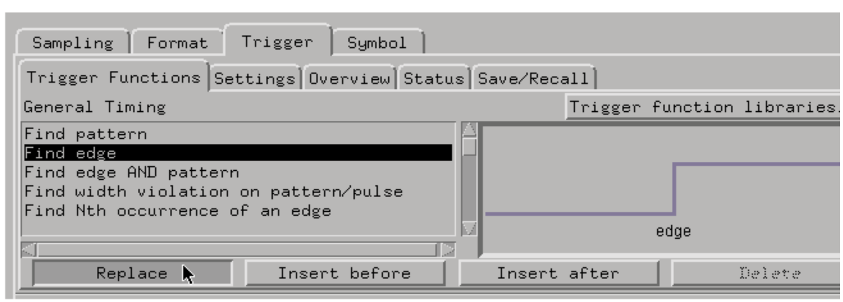
4. Format labels for the signals of interest.



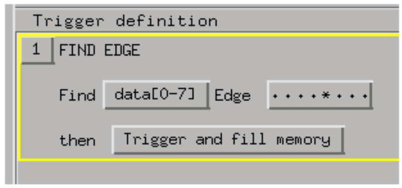
Capturing the Data

You can trigger on anything, on any glitch, or on a particular glitch.

1. Use the "Find edge" trigger function.



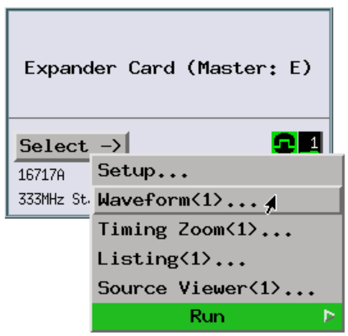
2. In the trigger definition, specify the signal on which you will look for glitches.

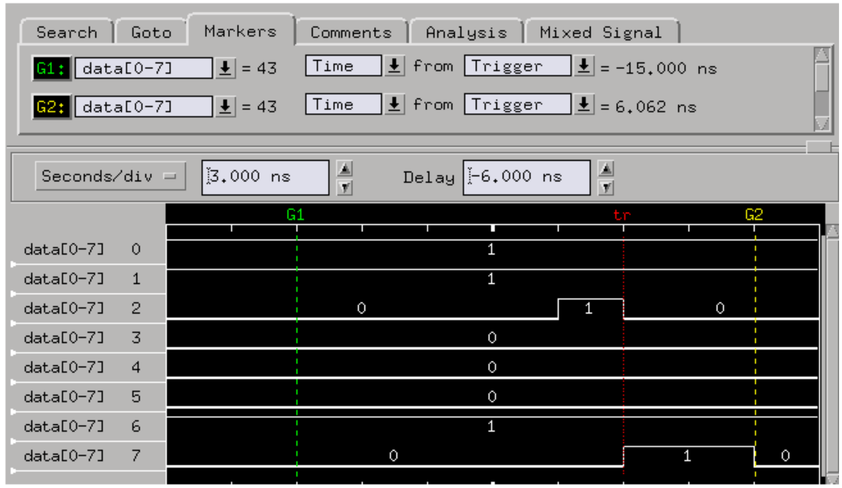


3. Select the Run button to start the measurement.

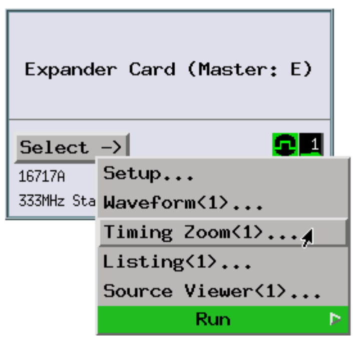
Displaying the Data

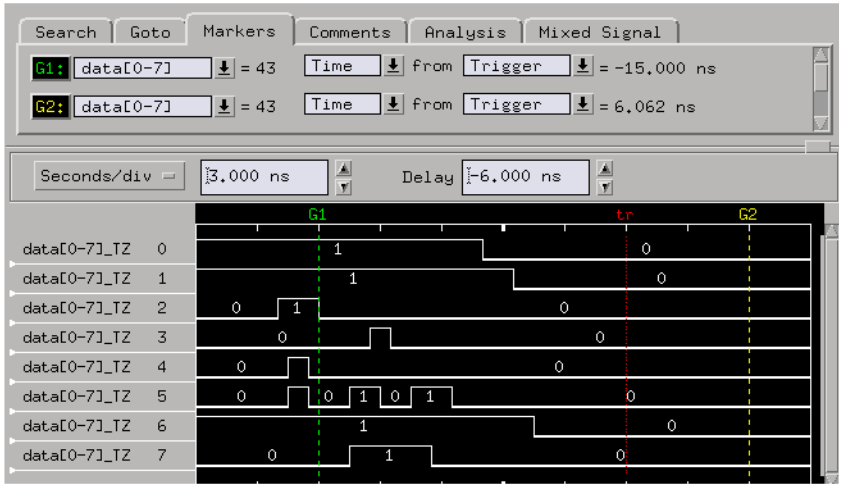
1. Use the Waveform display to show the captured glitches.





2. Use the TimingZoom display to get a better picture of the glitch.





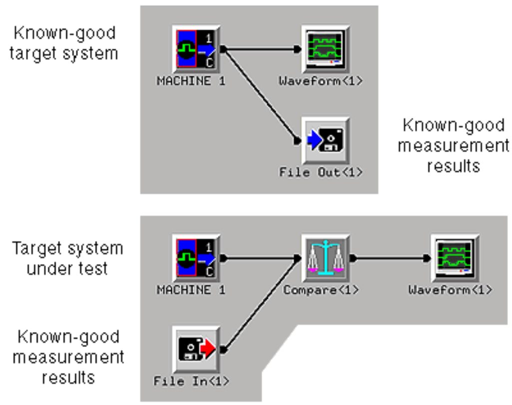
See Also

"To arm an oscilloscope when the analyzer triggers" on page 277

Measuring Conformance to Specifications

- "To measure conformance to specs (with the Compare tool)" on page 51
- "To find setup and hold violations" on page 56
- "To trigger if a pattern doesn't follow an edge" on page 59
- "To verify pulse widths" on page 63
- "To trigger on a violation of an edge sequence" on page 67
- "To trigger when two edges are asserted simultaneously" on page 71
- "To generate pattern stimulus on devices" on page 75
- "To analyze jitter or time dispersion (with SPA)" on page 81
- "To analyze bus stability (with SPA)" on page 88

To measure conformance to specs (with the Compare tool)

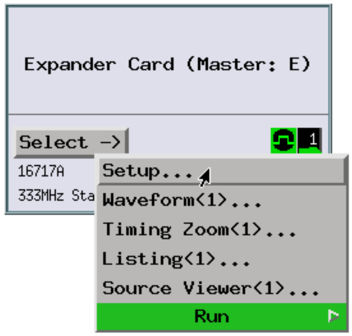


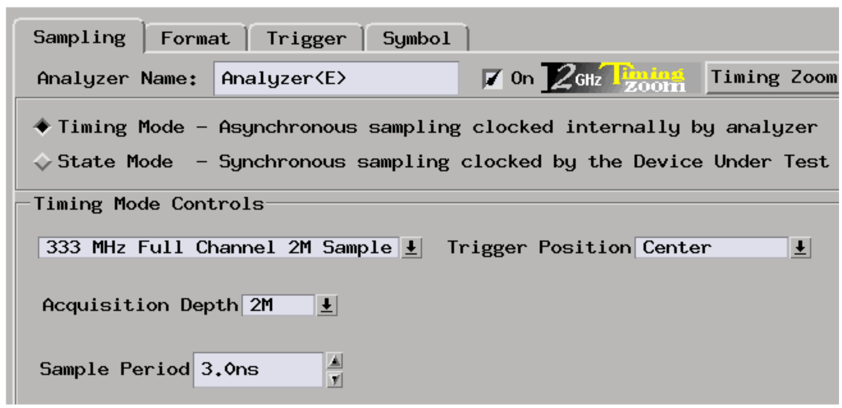
Possible uses:

- To measure specifications conformance against known-good circuitry.
- To measure specifications conformance under component stress conditions.

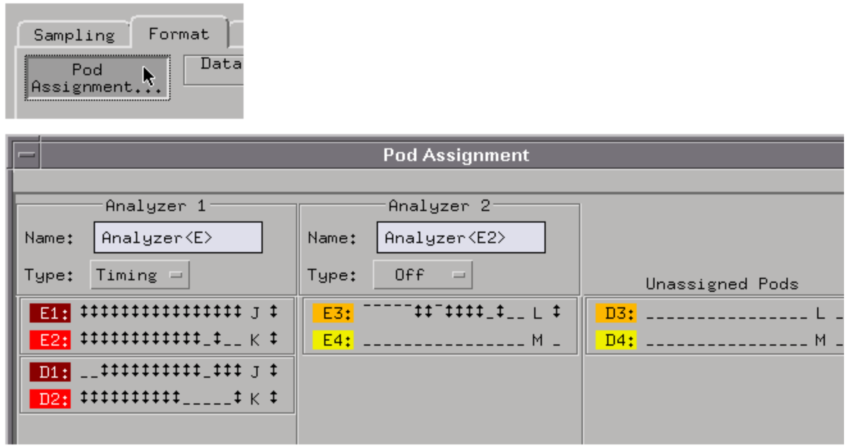
Probing the Target System

1. Configure a timing or state machine.

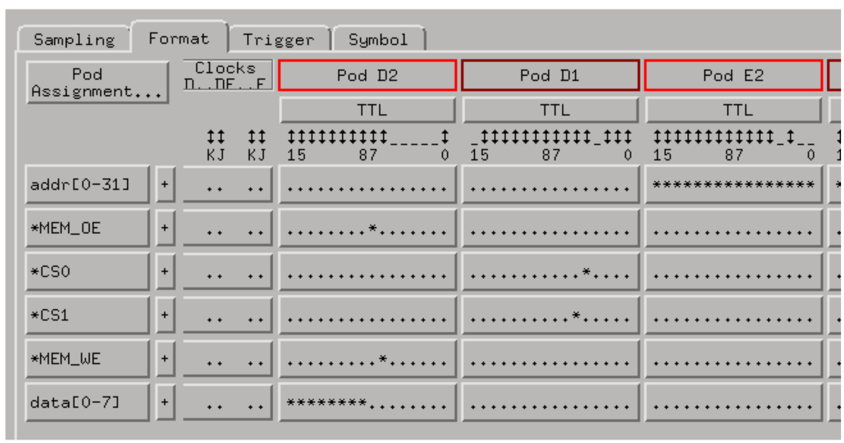




2. Assign pods if necessary.

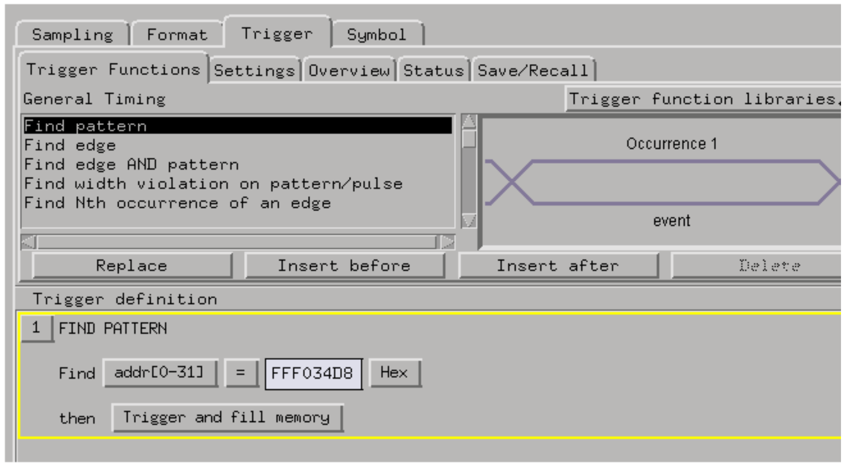


3. Format labels for the signals of interest.



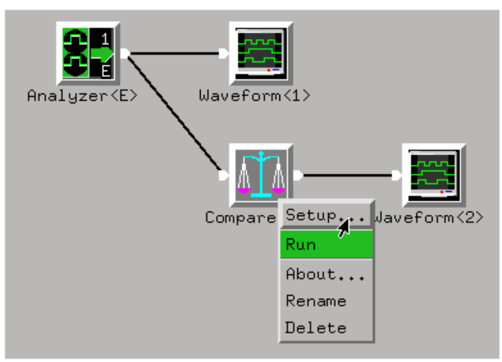
Capturing the Data

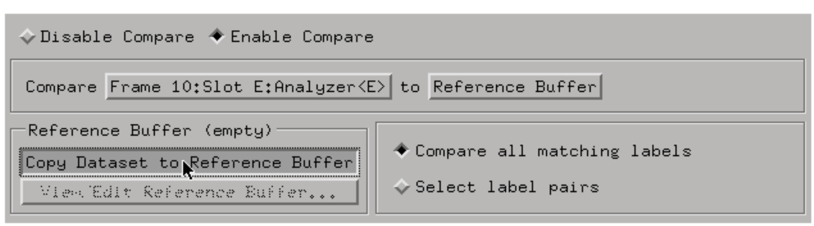
1. Set up a trigger specification.



2. Select the Run button to start the measurement.

3. In the Workspace window, add the Compare tool to your analyzer configuration and copy the known-good dataset to the Compare tool's *reference buffer*.





4. If you want to turn OFF the analyzer and probe a different target system, save the Compare tool configuration. This will save the contents of the Reference Buffer to the logic analyzer's disk.

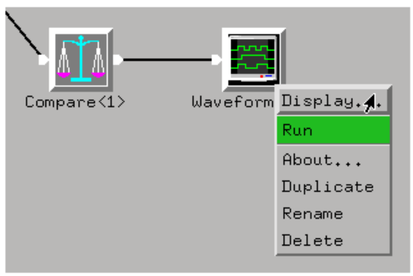
5. Probe a different target system or add the component stress conditions.

6. If you saved a Compare tool configuration, load it.

7. Select the Run button to repeat the measurement.

Displaying the Data

1. In the display window, differences in the measurement results will be highlighted with gray.

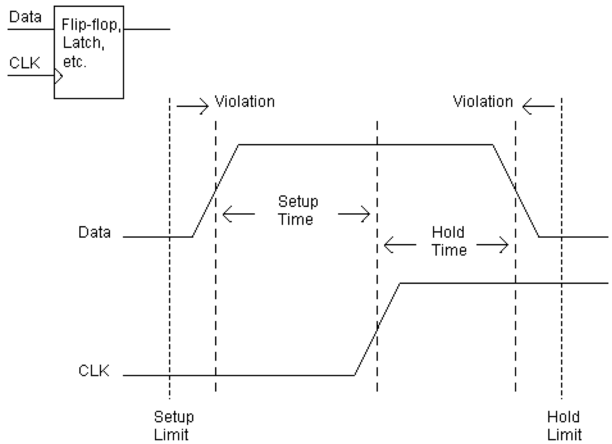


Note that a *difference flag* label is generated so you can search for differences.

See Also

The Compare tool online help (see the *Compare Tool* help volume) for more information.

To find setup and hold violations



Possible uses:

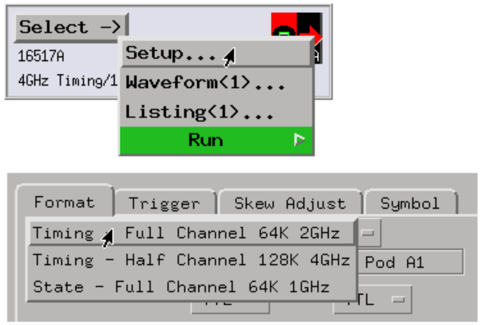
- Verifying that design timing meets setup and hold specifications of flip-flops, latches, and other memory element circuitry.

Requirements:

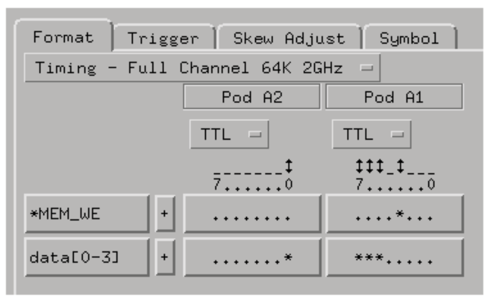
- The Agilent Technologies 16517A 4GHz Timing/1GHz State Logic Analyzer can look for setup and hold violations on multiple channels (for example, a data bus).

Probing the Target System

1. Configure a timing analysis machine.

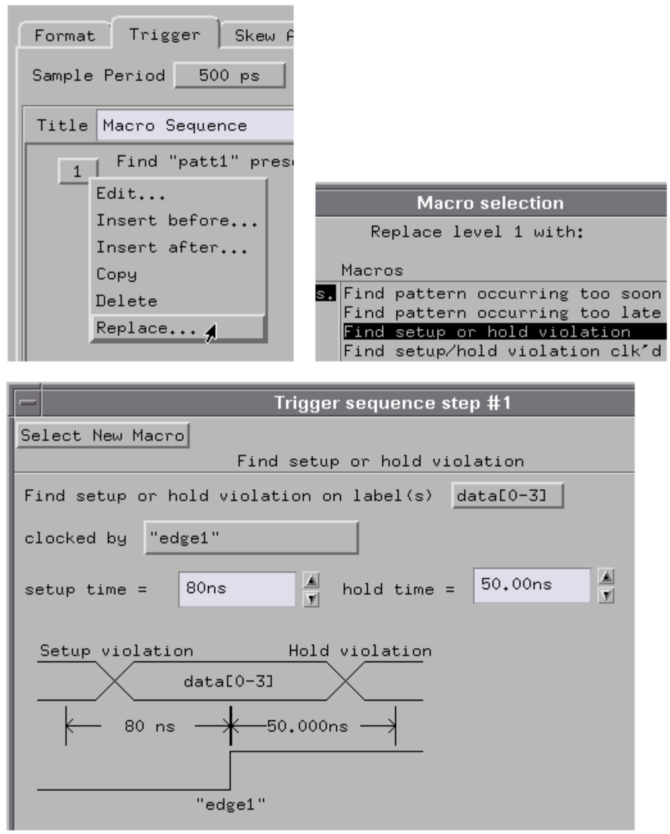


2. Format labels for the signals of interest.

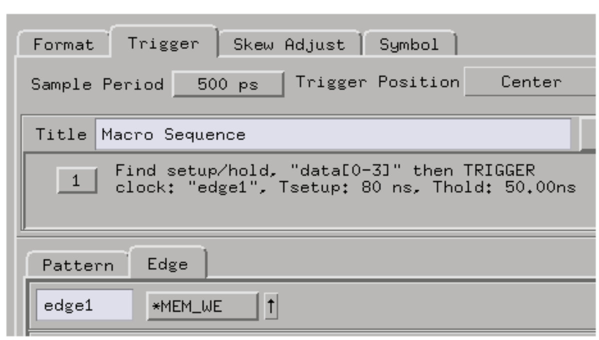


Capturing the Data

1. In the Trigger window, replace level 1 of the trigger specification with the "Find setup or hold violation" macro. In the macro dialog, select the label, edge, setup time limit, and hold time limit.



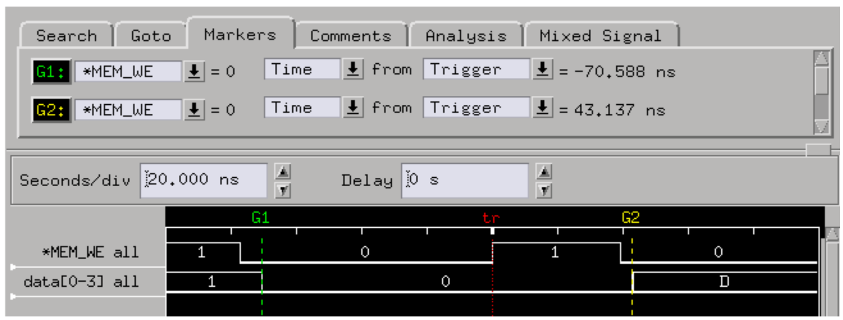
2. Set up the edge resource.



3. Select the Run button to start the measurement.

Displaying the Data

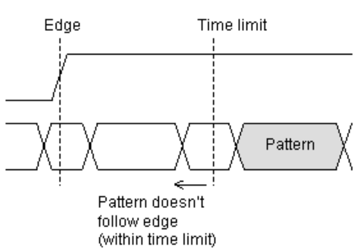
1. If the analyzer triggers, a setup or hold violation occurs. Open the Waveform display and use the global markers to see the actual setup or hold time.



See Also

To see how setup and hold violations affect software execution, (see "To capture SW execution on a setup or hold violation" on page 289).

To trigger if a pattern doesn't follow an edge



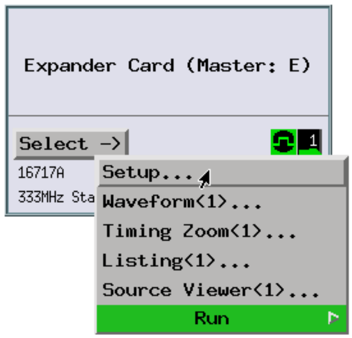
Possible uses:

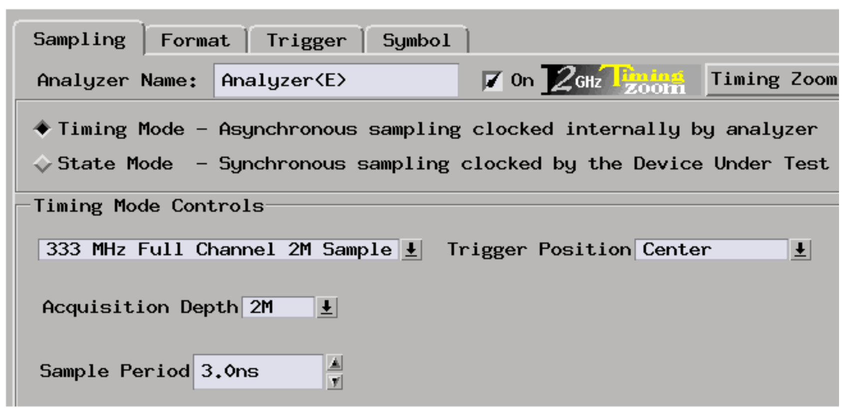
- To measure interrupt response time.

- To trigger when expected data does not appear on the data bus from a remote device when requested.

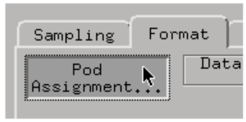
Probing the Target System

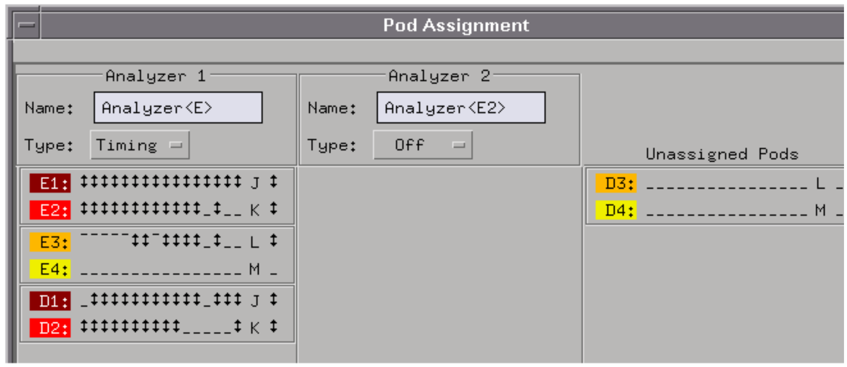
1. Configure a timing analysis machine.



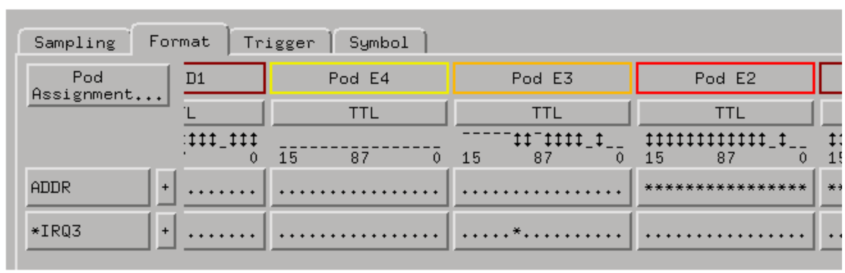


2. Assign pods if necessary.



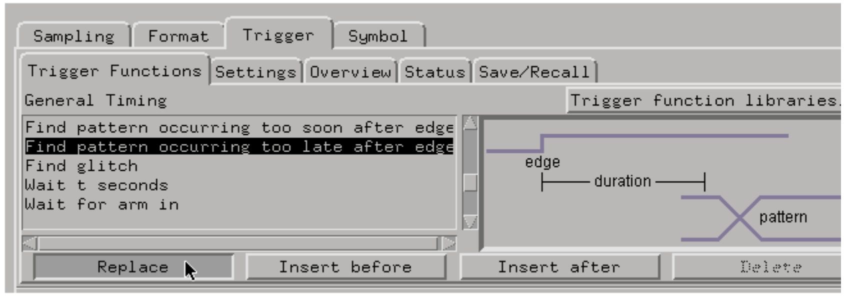


3. Format one label for the signals on which you will look for a pattern and another label for the signal on which you will look for the edge.

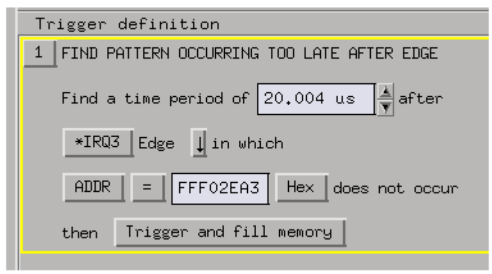


Capturing the Data

1. Use the "Find pattern occurring too late after edge" trigger function.



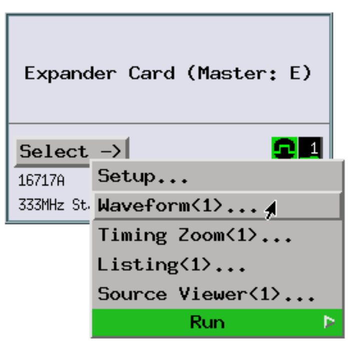
2. In the trigger definition, enter the time period and specify the edge and the pattern.

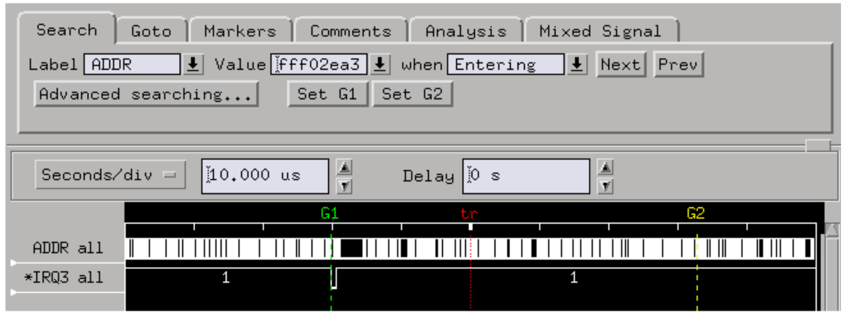


3. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Waveform display to show that the pattern didn't occur after the edge. Use the Search dialog to find the edge and the pattern.





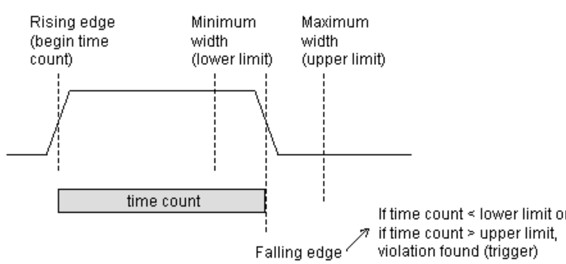
If the analyzer never triggers, the pattern always occurs after the edge within the time specified.

See Also

"Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

To verify pulse widths

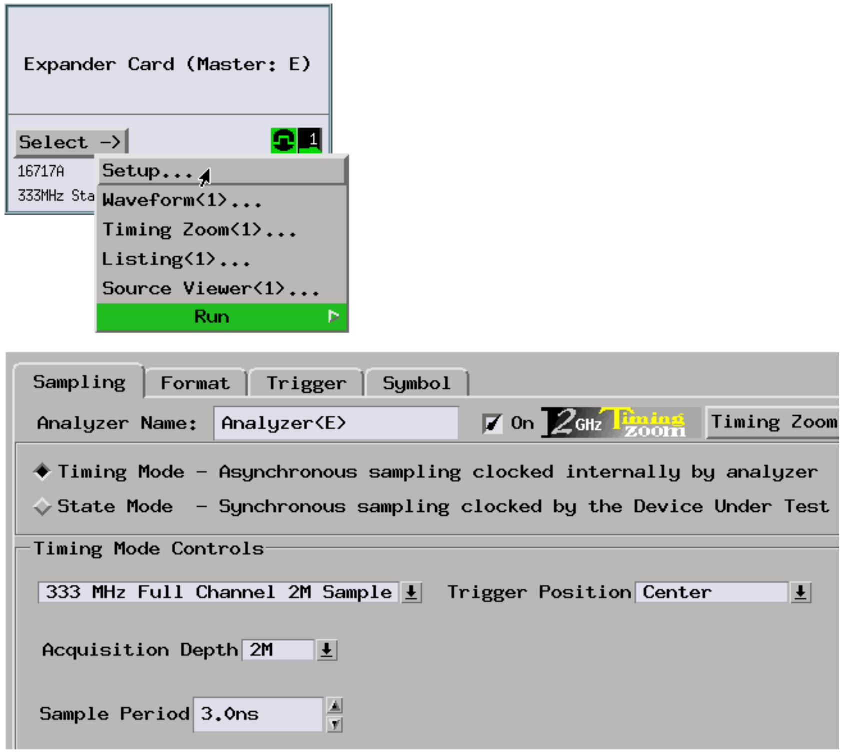


Possible uses:

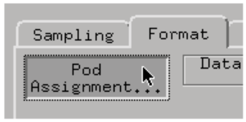
- To test minimum and maximum pulse limits.
- To verify that all pulses controlling a mechanical device fall within specifications.

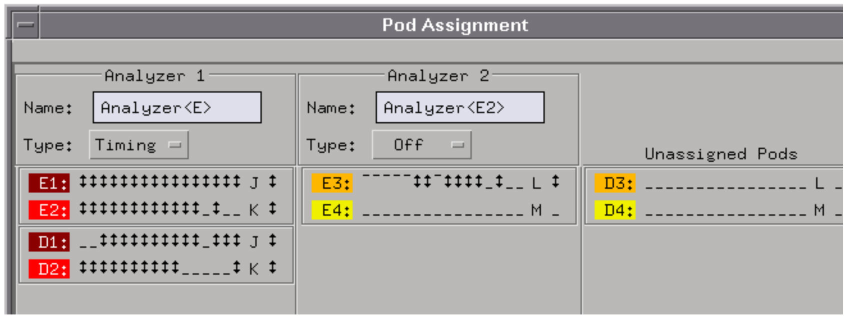
Probing the Target System

1. Configure a timing analysis machine.

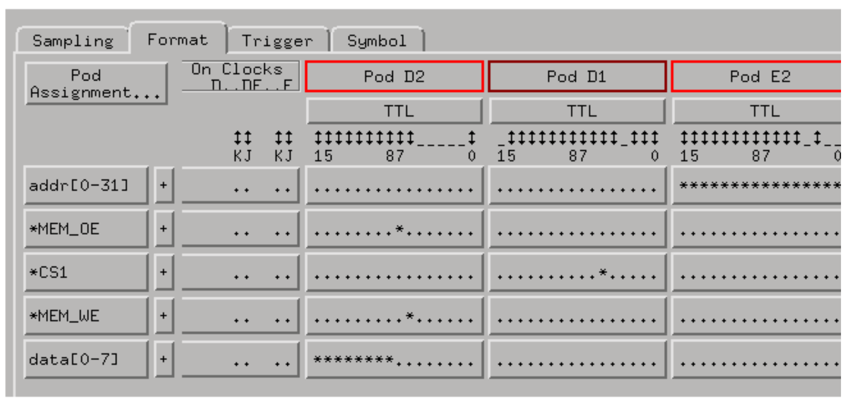


2. Assign pods if necessary.



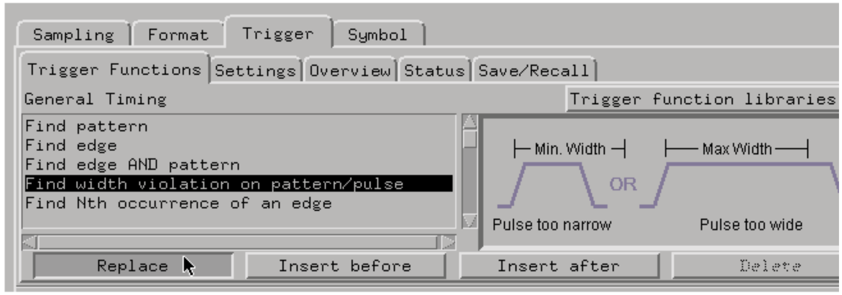


3. Format a label for the signal whose pulses you will be looking at.

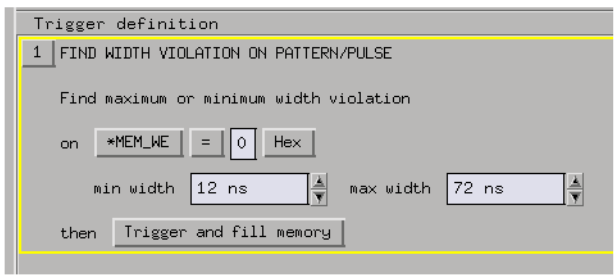


Capturing the Data

1. Use the "Find width violation on pattern/pulse" trigger function.



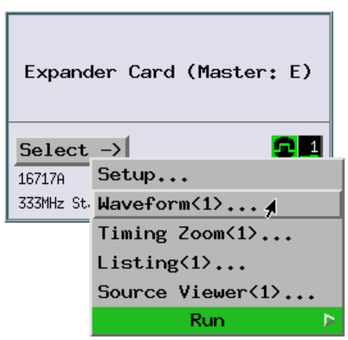
2. In the trigger definition, specify the pattern and enter the minimum and maximum widths.

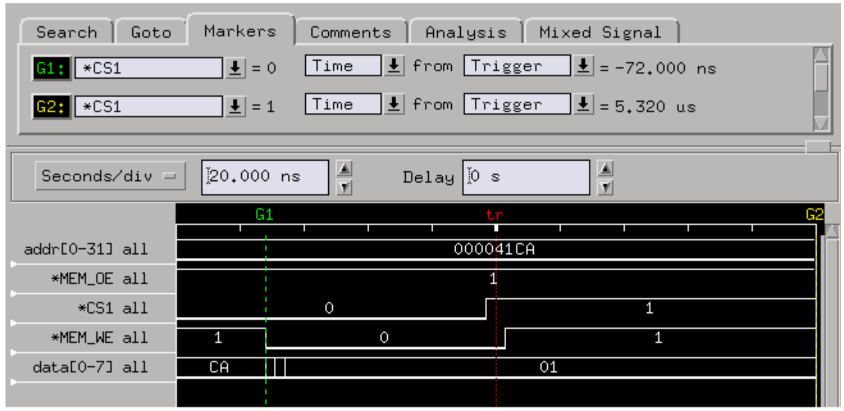


3. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Waveform display to show that the trailing edge of the pulse didn't occur within the defined interval.





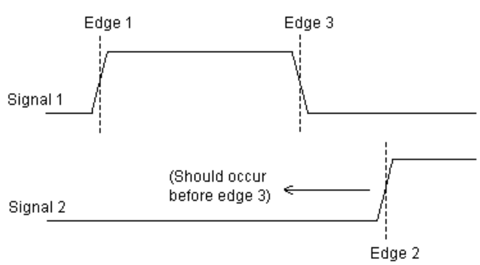
If the analyzer never triggers, the falling edge occurs within the defined interval.

See Also

"Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

To trigger on a violation of an edge sequence

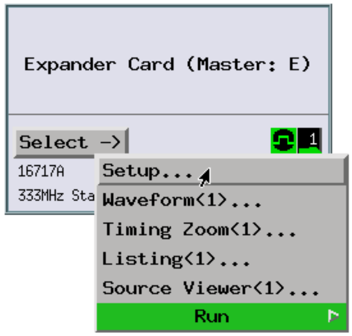


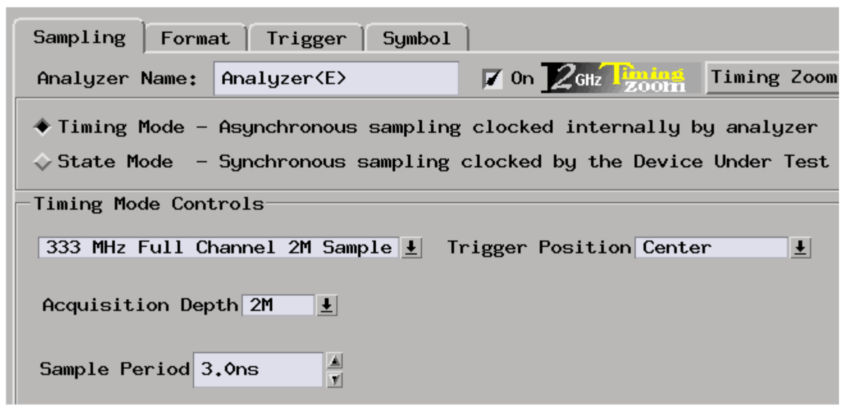
Possible uses:

- To detect a handshake violation.
- To trigger on incorrect control signal generation from a Programmable Logic Device (PLD).

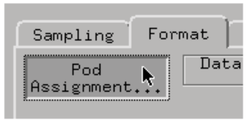
Probing the Target System

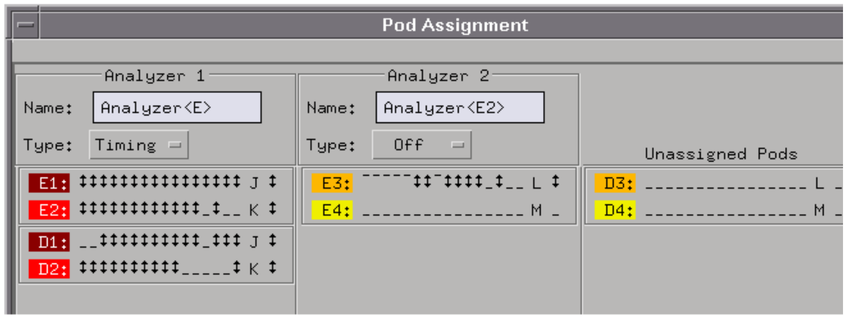
1. Configure a timing analysis machine.



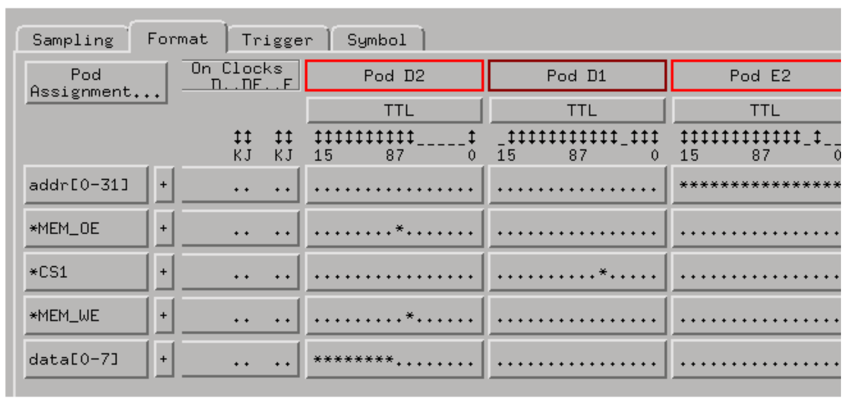


2. Assign pods if necessary.



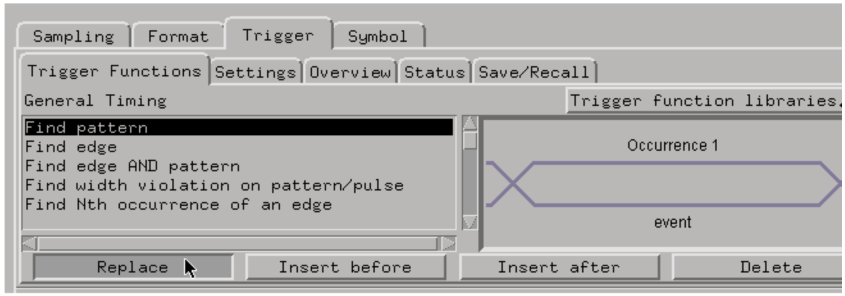


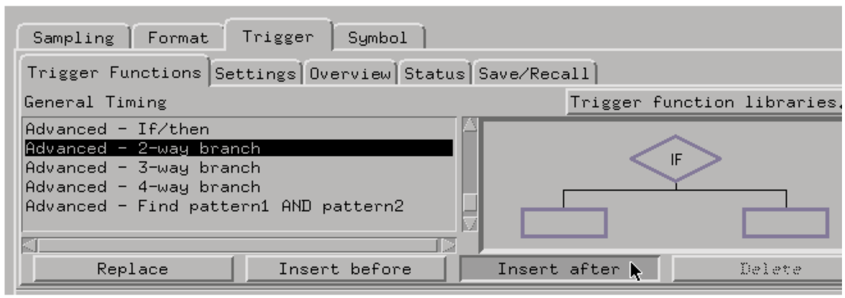
3. Format labels for the signals whose edges you will be looking at.



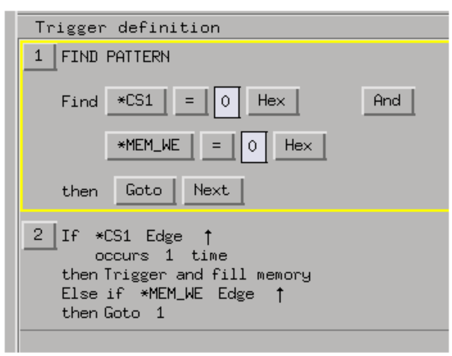
Capturing the Data

1. Build a two level trigger setup using the "Find pattern" and "Advanced 2-way branch" trigger functions. The pattern will identify when the edge sequence is about to occur and the 2-way branch will check for the sequence of edges.





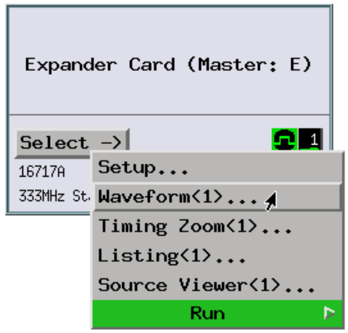
2. In the trigger definition, specify the pattern and the edges. If edges occur in the wrong sequence, trigger the logic analyzer; otherwise, go back and look for the next occurrence of the edges.

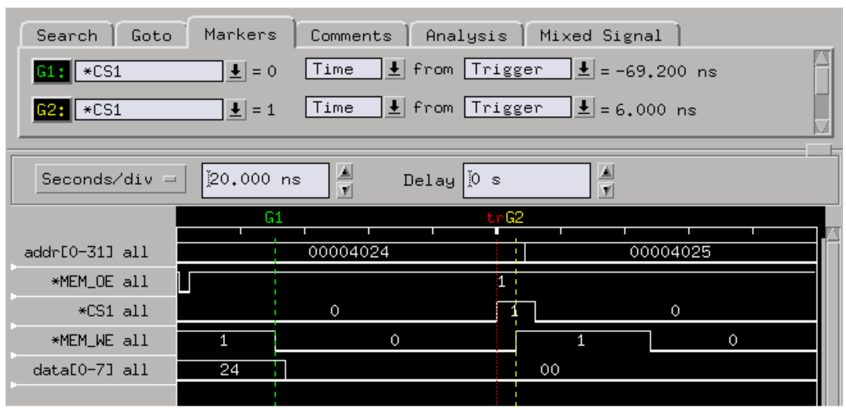


3. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Waveform display to show the edge sequence violation.



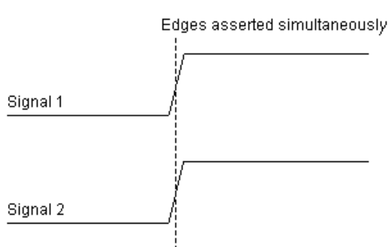


If the analyzer never triggers, the edges occur in the proper sequence.

See Also

"If the trigger doesn't occur as expected" on page 309

To trigger when two edges are asserted simultaneously

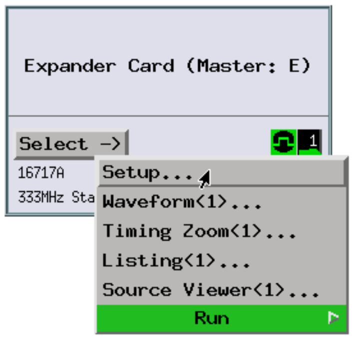


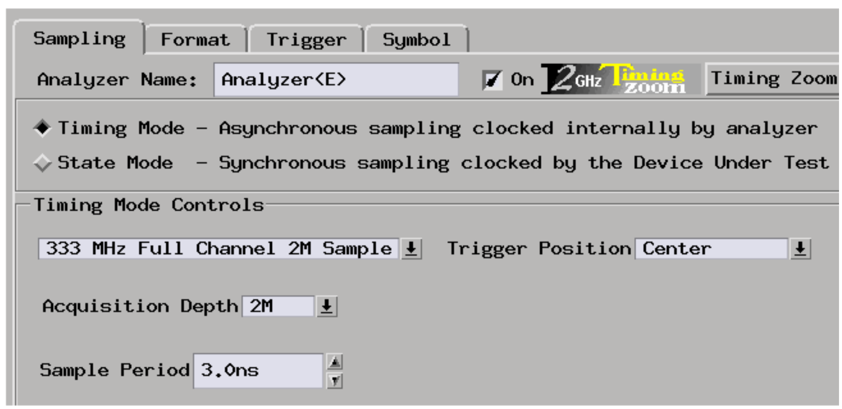
Possible uses:

- To detect bus contention.
- To view system activity when two entities are trying to seize a digital communications channel at once.

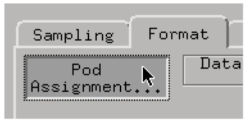
Probing the Target System

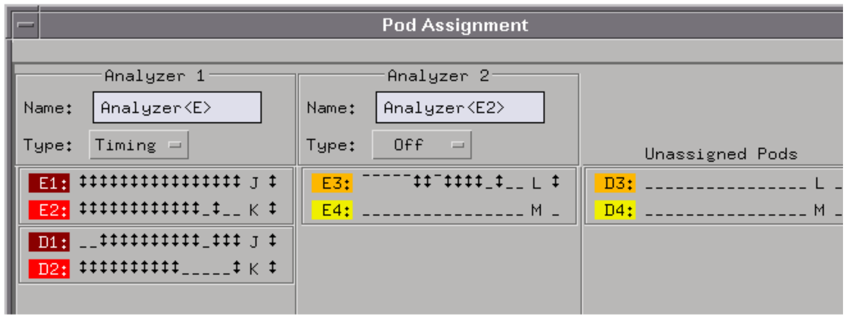
1. Configure a timing analysis machine.



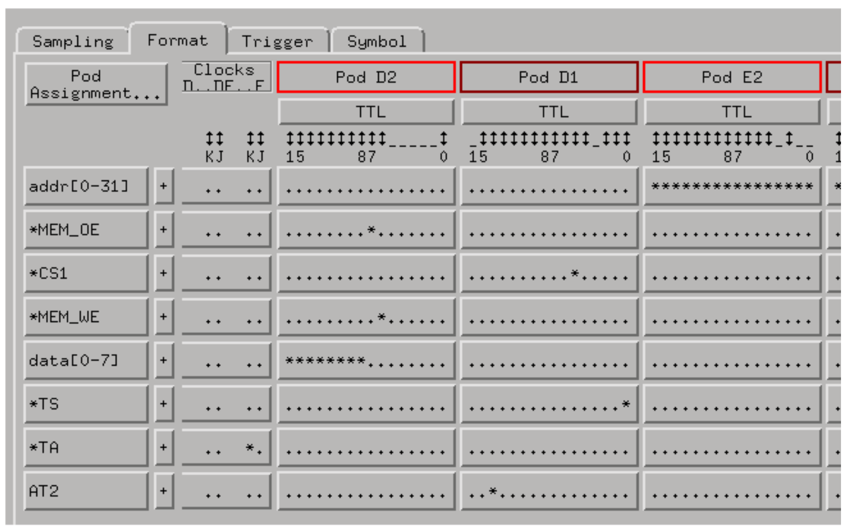


2. Assign pods if necessary.



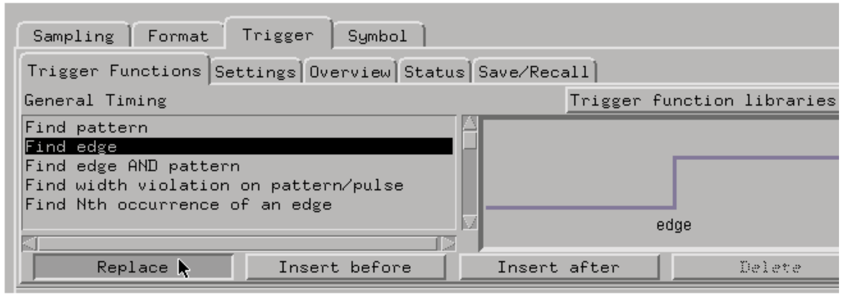


3. Format labels for the signals whose edges you will be looking at.

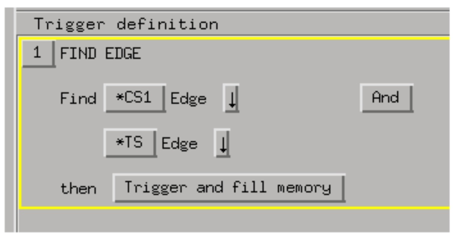


Capturing the Data

1. Use the "Find edge" trigger function.



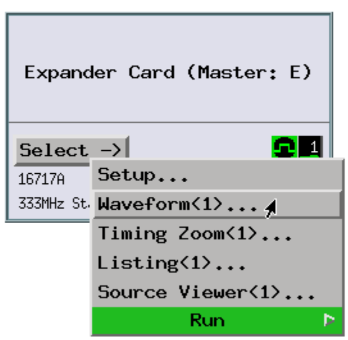
2. In the trigger definition, select the button after Find, and choose the "Insert LABEL after" item. Specify the two edges.

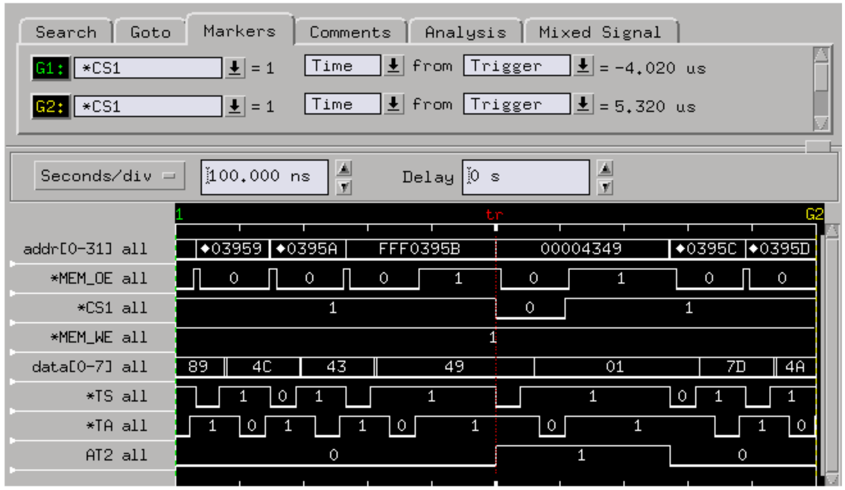


3. Select the Run button to start the measurement.

Displaying the Data

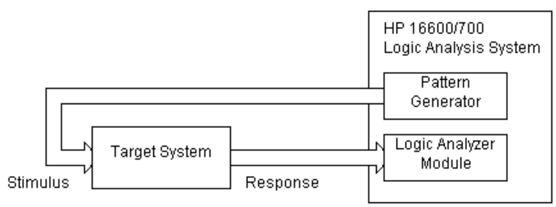
1. When the analyzer triggers, use the Waveform display to show the two edges that occur at the same time.





If the analyzer never triggers, the two edges don't occur within the same sample period.

To generate pattern stimulus on devices



Requirements:

- This measurement requires a pattern generator module (Agilent Technologies 16522A).

Possible uses:

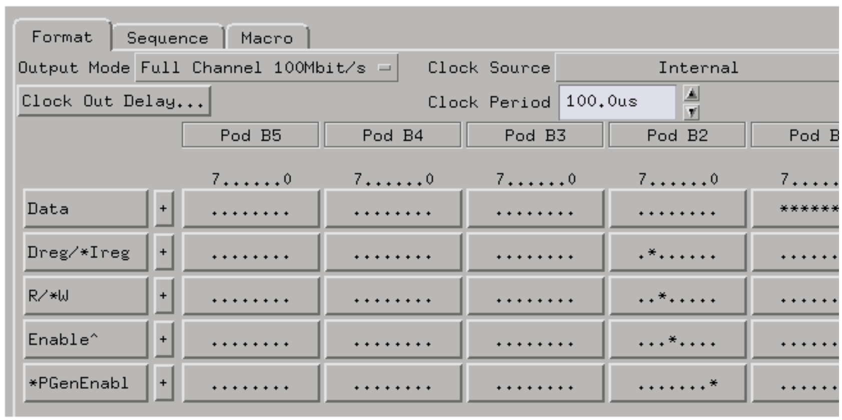
- To simulate digital circuitry that isn't available.
- To generate signals for functionally testing prototype hardware.

Connecting Pattern Generator Outputs to the Target System

1. Connect the pattern generator lines to your target system using the appropriate TTL, CMOS, or ECL data or clock pods (see the pattern generator documentation (see the *Agilent Technologies 16522A 200M Vectors/s Pattern Generator* help volume) for more information).

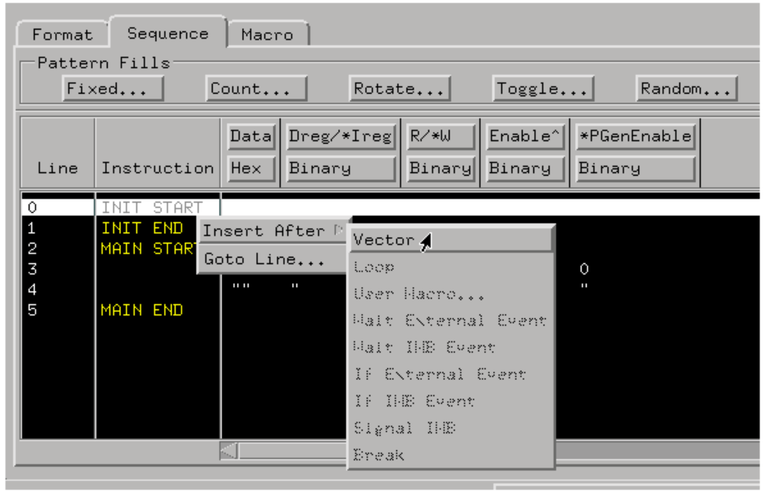
Configuring the Pattern Generator & Labeling Outputs

1. In the Format tab, select the output mode and the clock source, and label the output signals.



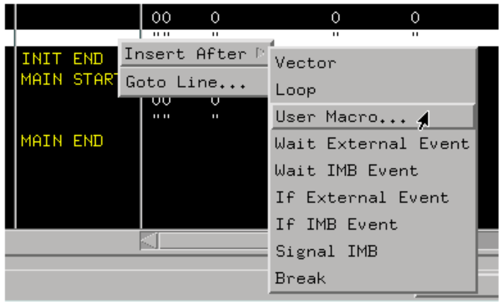
Building the Test Vectors

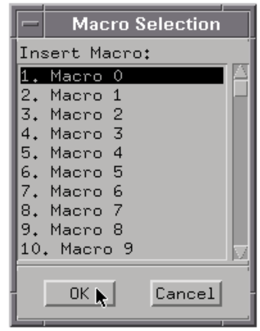
1. In the Sequence tab, insert vectors. Select data values, and enter the new values. A double-quote character means the same value as above.

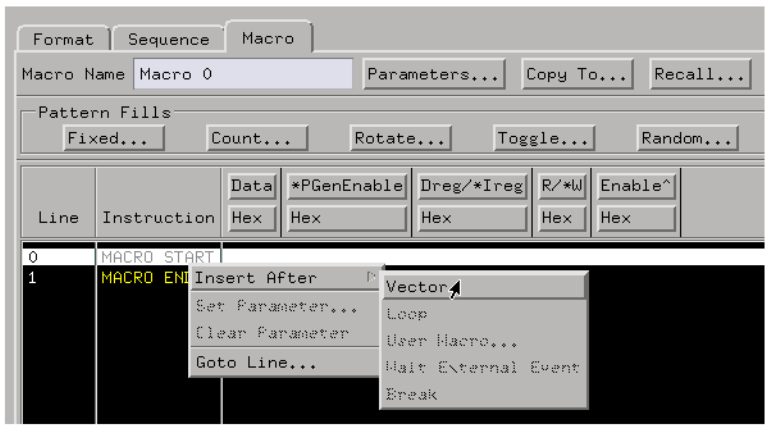


During a repetitive run, the vectors in the INIT section are only executed once.

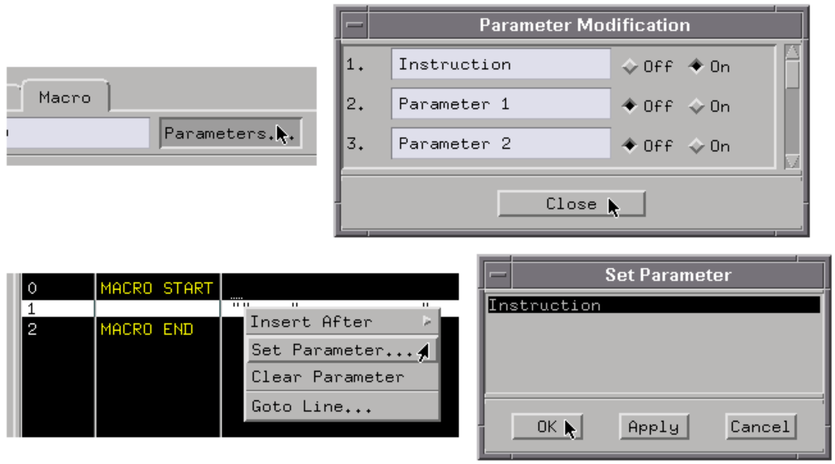
You can use macros to insert vectors that need to be repeated.





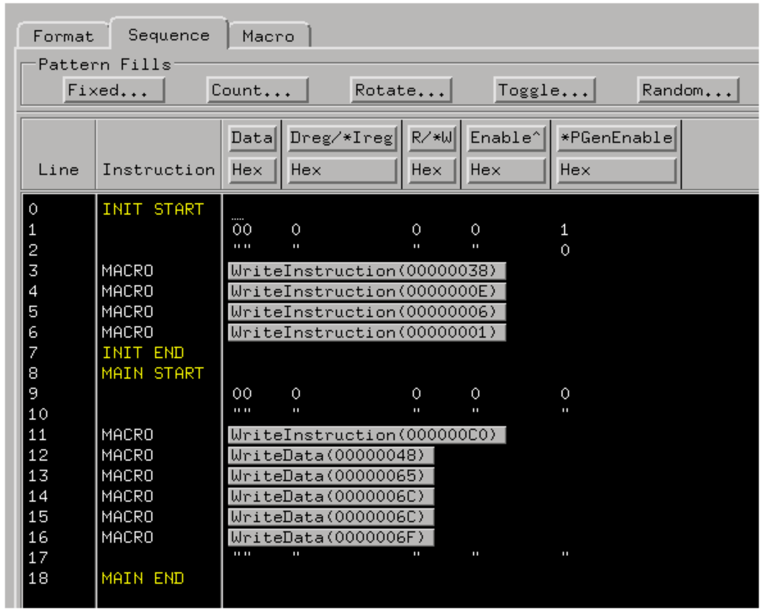


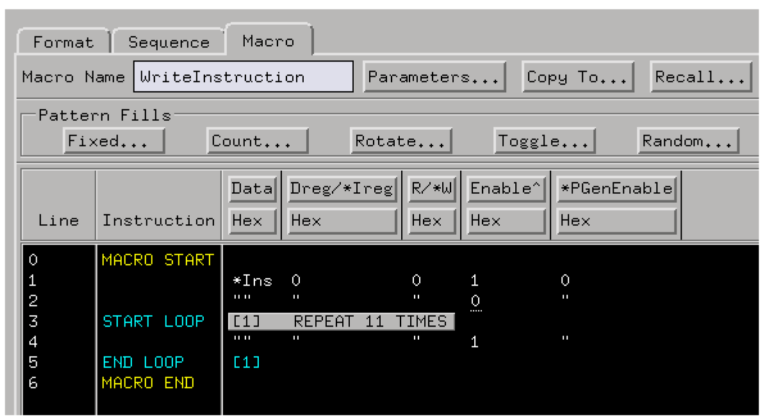
You can pass parameters to macros.

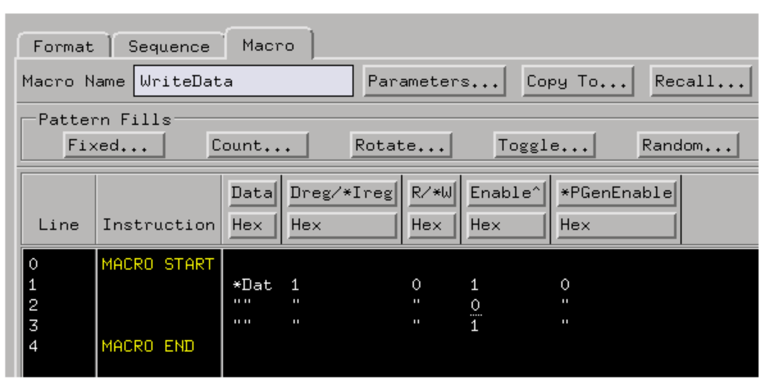


Running the Pattern Generator

1. When you've finished building the test vectors, select the Run button to cause the pattern generator to output the vectors.


Format
Sequence
Macro
Pattern Fills
Fixed...
Count...
Rotate...
Toggle...
Random...
Line
Instruction
Data
Dreg/*Ireg
R/*W
Enable^
*PGenEnable
Hex
0 INIT START
1 00 0 0 0 1
2 0
3 MACRO WriteInstruction(00000038)
4 MACRO WriteInstruction(0000000E)
5 MACRO WriteInstruction(00000006)
6 MACRO WriteInstruction(00000001)
7 INIT END
8 MAIN START
9 00 0 0 0 0
10
11 MACRO WriteInstruction(000000C0)
12 MACRO WriteData(00000048)
13 MACRO WriteData(00000065)
14 MACRO WriteData(0000006C)
15 MACRO WriteData(0000006C)
16 MACRO WriteData(0000006F)
17
18 MAIN END


Format
Sequence
Macro
Macro Name
WriteInstruction
Parameters...
Copy To...
Recall...
Pattern Fills
Fixed...
Count...
Rotate...
Toggle...
Random...
Line
Instruction
Data
Dreg/*Ireg
R/*W
Enable^
*PGenEnable
Hex
0 MACRO START
1 *Ins 0 0 1 0
2 0
3 START LOOP [1] REPEAT 11 TIMES
4 1
5 END LOOP [1]
6 MACRO END

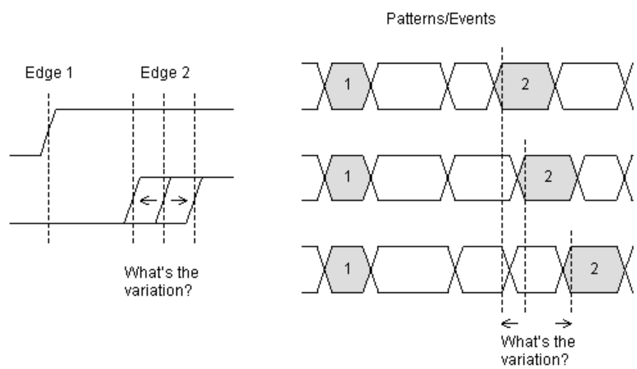

Format
Sequence
Macro
Macro Name
WriteData
Parameters...
Copy To...
Recall...
Pattern Fills
Fixed...
Count...
Rotate...
Toggle...
Random...
Line
Instruction
Data
Dreg/*Ireg
R/*W
Enable^
*PGenEnable
Hex
0 MACRO START
1 *Dat 1 0 1 0
2 0
3 1
4 MACRO END

See Also

"To simulate particular interrupt sequences" on page 191

"To generate patterns when a source line executes" on page 262

To analyze jitter or time dispersion (with SPA)



Requirements:

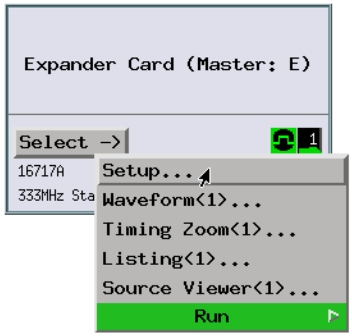
- This measurement requires the system performance analyzer (SPA) tool set.

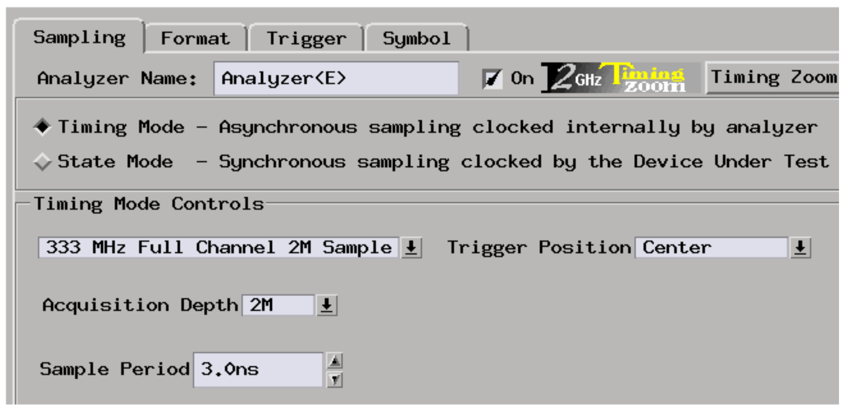
Possible uses:

- To measure the jitter between two edges.
- To measure the variation between two bus states.
- To measure setup and hold times.

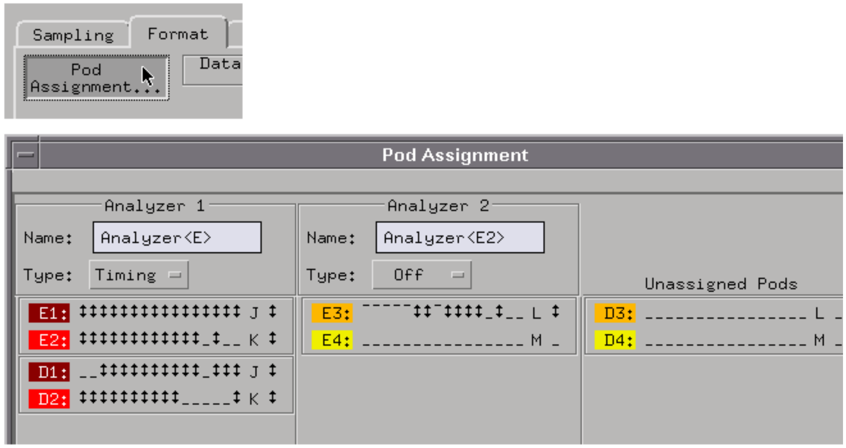
Probing the Target System

1. Probe the signals of interest.
2. Configure a timing or state analyzer, depending on whether you want to look at signal edges, patterns, or events.

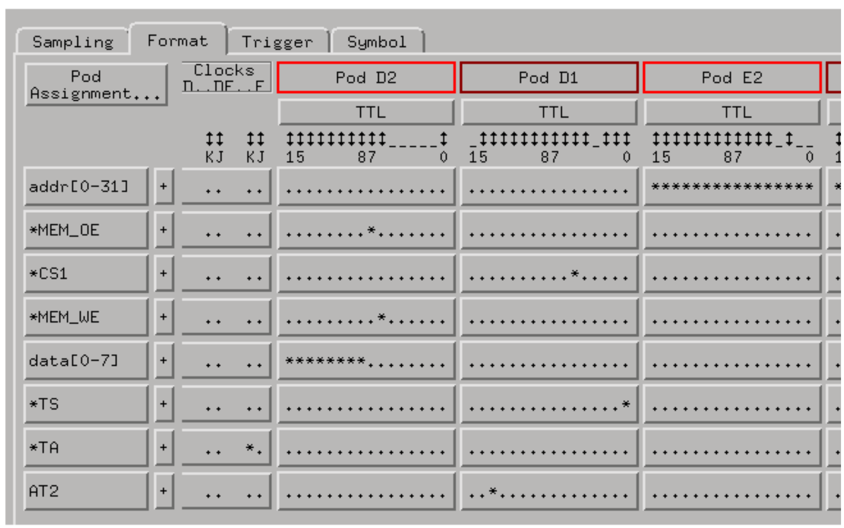




3. Assign pods if necessary.

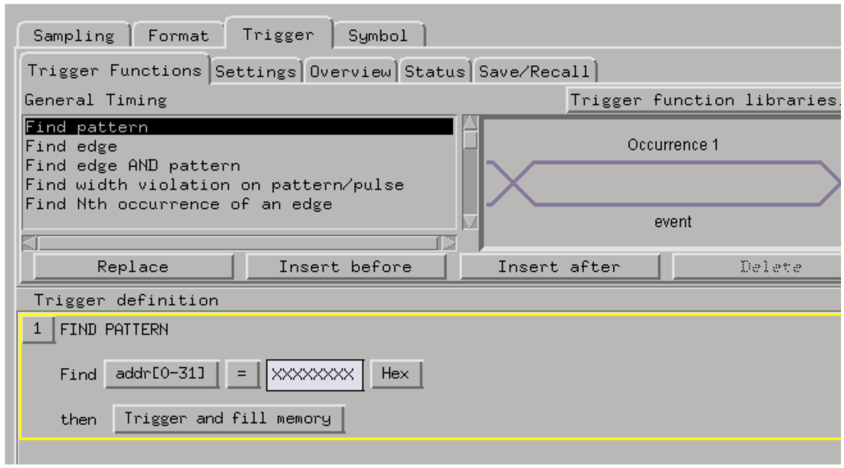


4. Label the logic analyzer channels. (If you're using an analysis probe, you can configure the analyzer and set up labels by loading the included configuration files.)

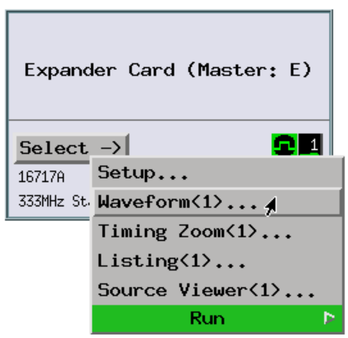


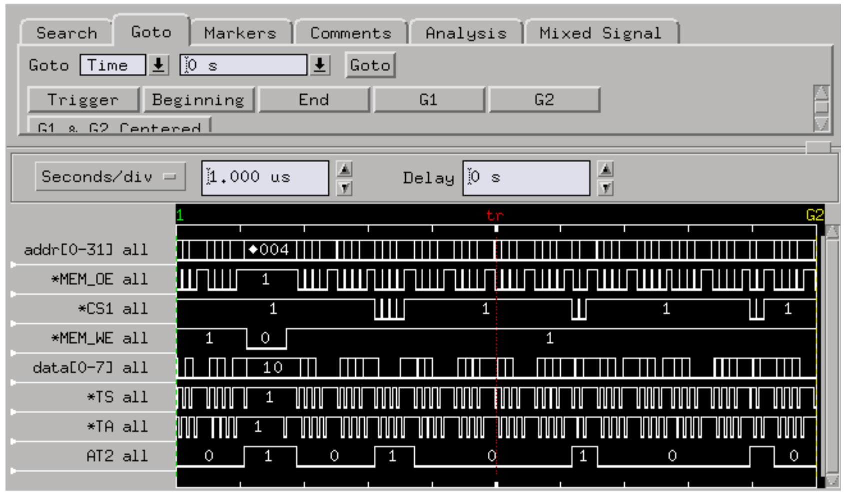
Capturing the Data

1. Set up a trigger specification to capture the signal edges, patterns, or events you're interested in.



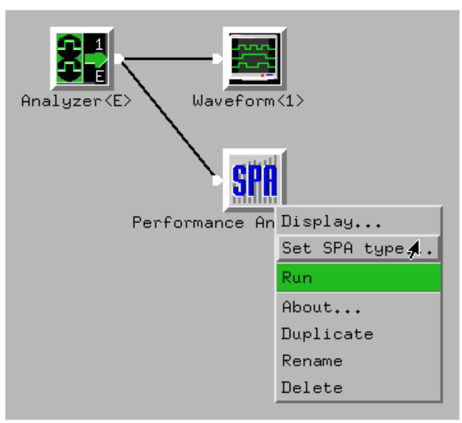
2. Select the Run button to start the measurement.
3. Display the captured waveforms.



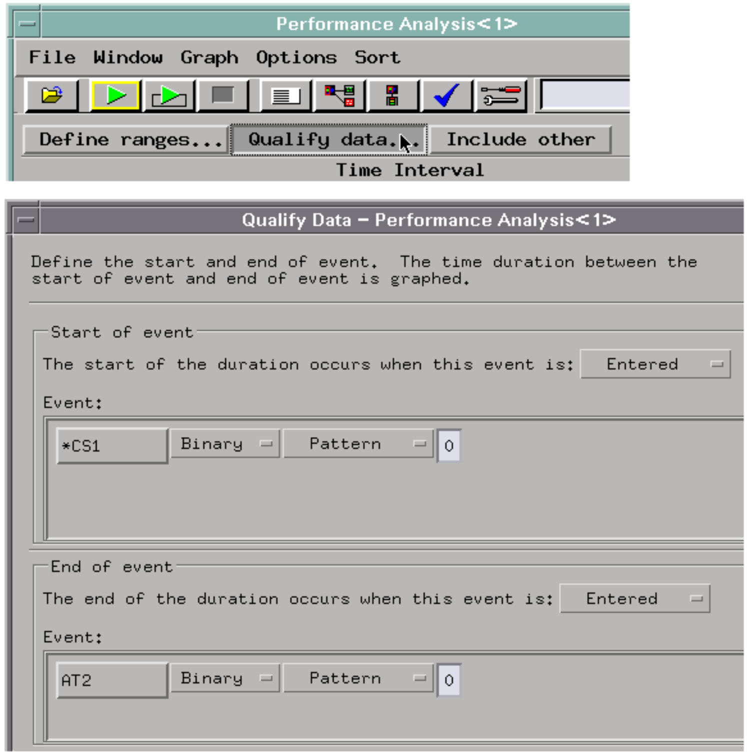


Displaying the Data

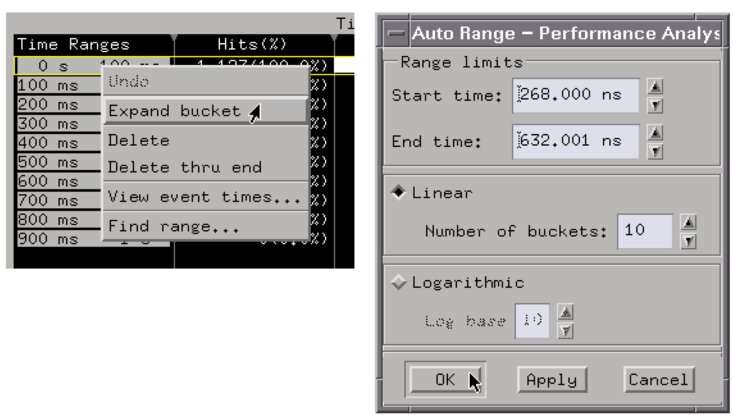
1. Use the system performance analyzer's Time Interval display to view the captured data.



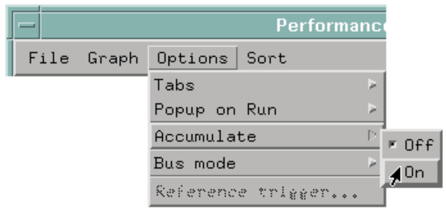
2. Define the start and end of the event whose time variations you wish to measure.

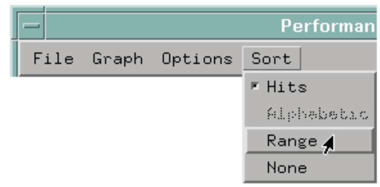


3. Define *buckets* for expected time ranges.



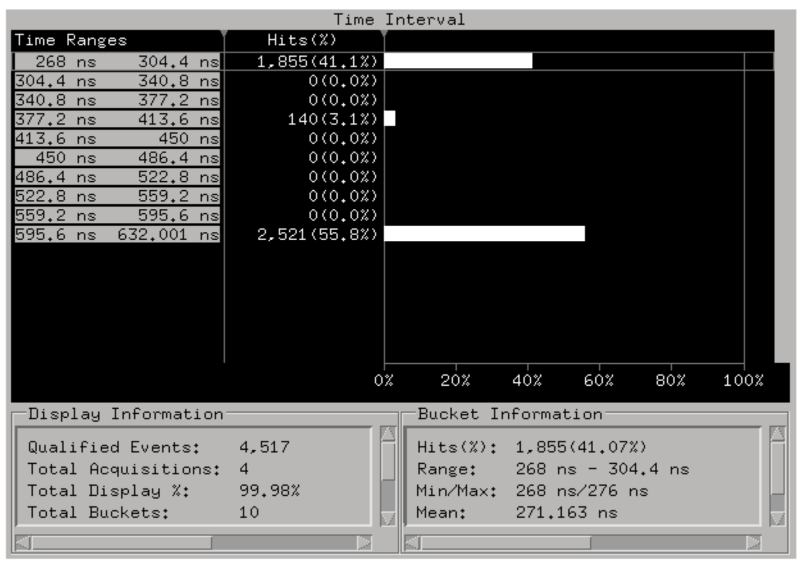
4. Set the appropriate data gathering and display options.





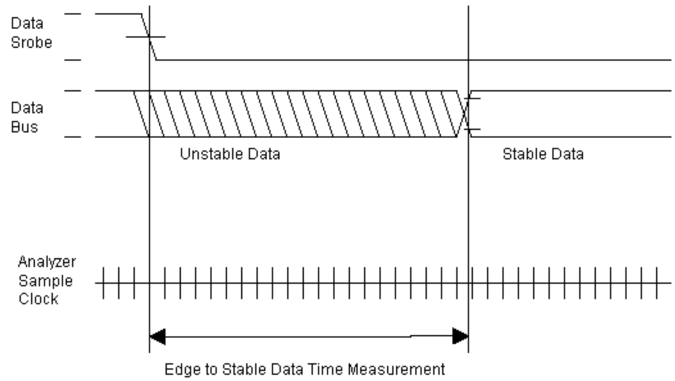
Use Accumulate Mode to analyze the behavior of your system over a long period of time (and, perhaps, run the measurement repetitively).

5. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.



Statistics such as the maximum time, minimum time, standard deviation, and mean help you document system behavior.

To analyze bus stability (with SPA)



The stability of a bus is defined by two or more consecutive acquisitions of the same data value on the bus.

For example, if you analyze a microprocessor's access to a RAM, you want to be sure that the data is stable when it is strobed.

In this context, the system performance analyzer helps you characterize areas of stability or instability for this bus.

Requirements:

- This measurement requires the system performance analyzer (SPA) tool set.

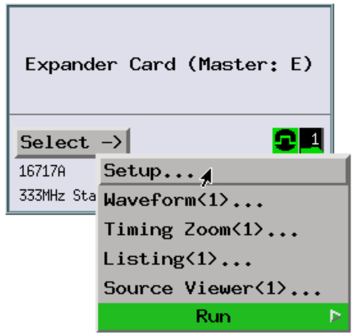
Possible uses:

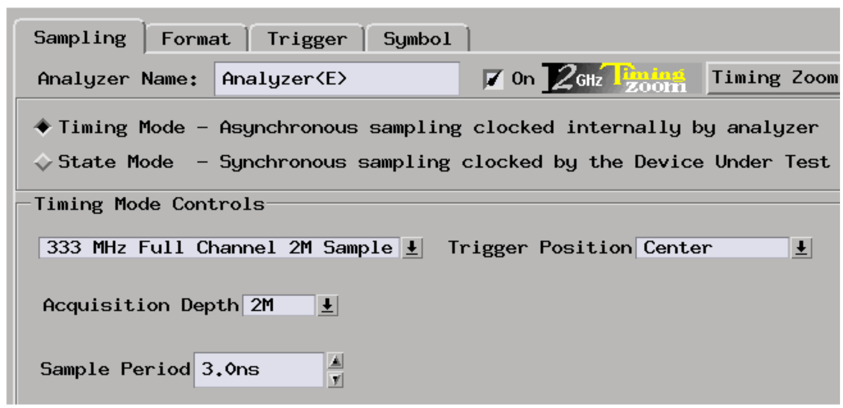
- To measure the correlation between a signal (such as a strobe or an edge) and the presence of valid, stable information on a bus (or a label with one or more channels).
- To search on the entry or exit of a stable or unstable bus condition.
- To focus on bus transactions.
- To search for stability within a defined time range or outside a defined time range.

Probing the Target System

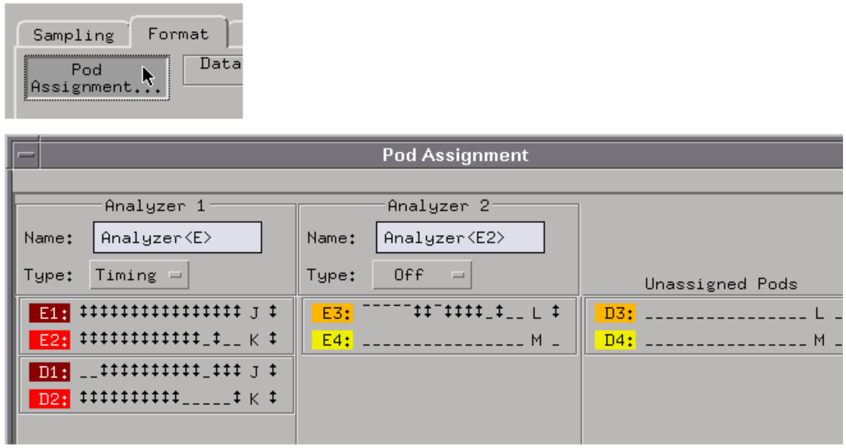
1. Probe the bus and strobe signals of interest.

2. Configure a timing analyzer.

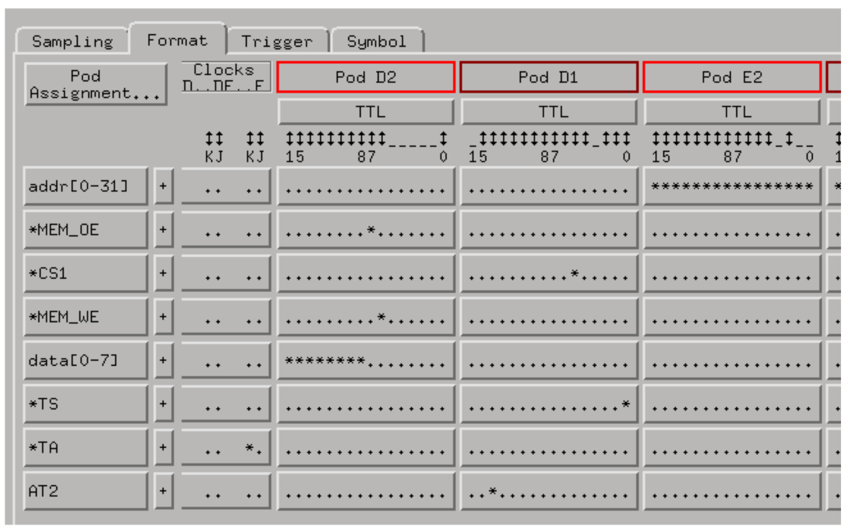




3. Assign pods if necessary.

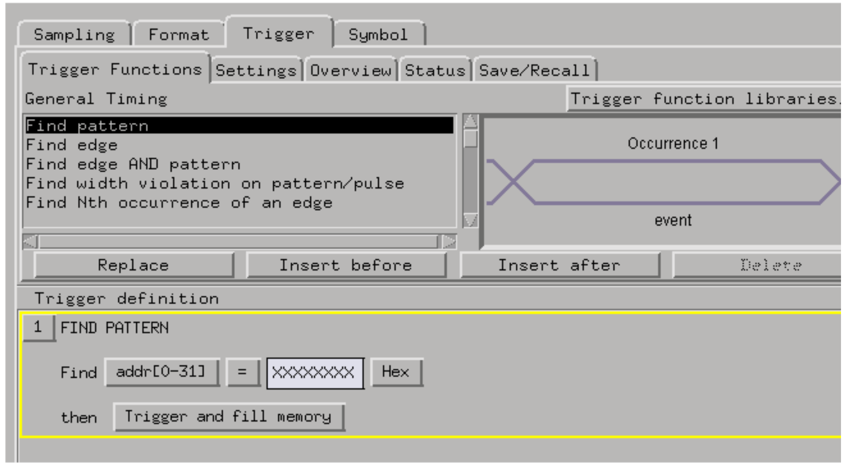


4. Label the logic analyzer channels.



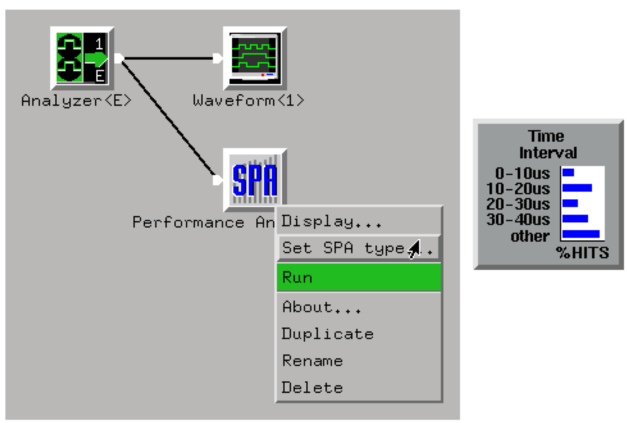
Capturing the Data

1. Set up a trigger specification to capture the strobe signal edge and the bus signals.

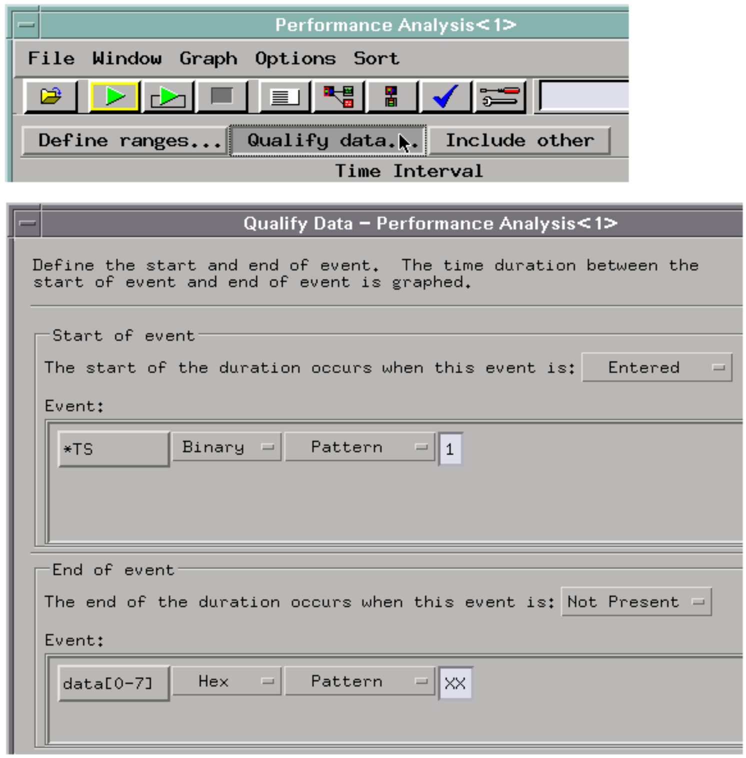


Displaying the Data

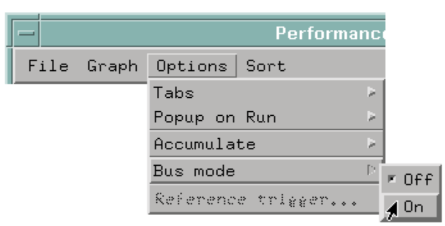
1. Use the system performance analyzer's Time Interval display to view the captured data.



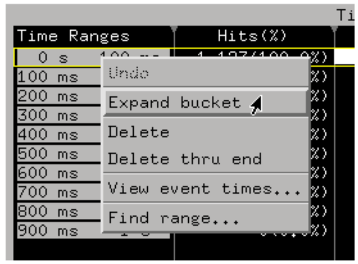
2. Define the start of the event as the data strobe signal going active, and define the end of event as the bus being stable (that is, a "don't care" pattern *Not Present*).

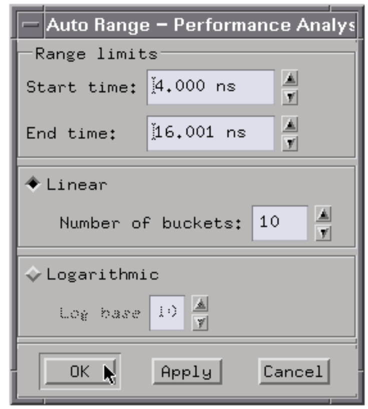


3. Turn ON the system performance analyzer’s bus mode.

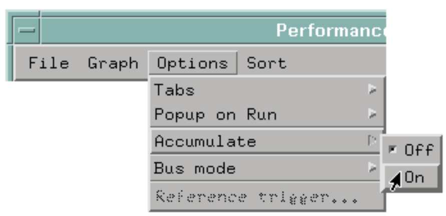


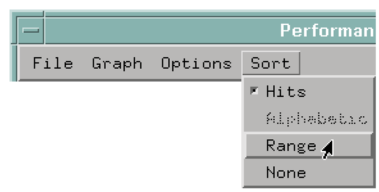
4. Define *buckets* for expected time ranges.





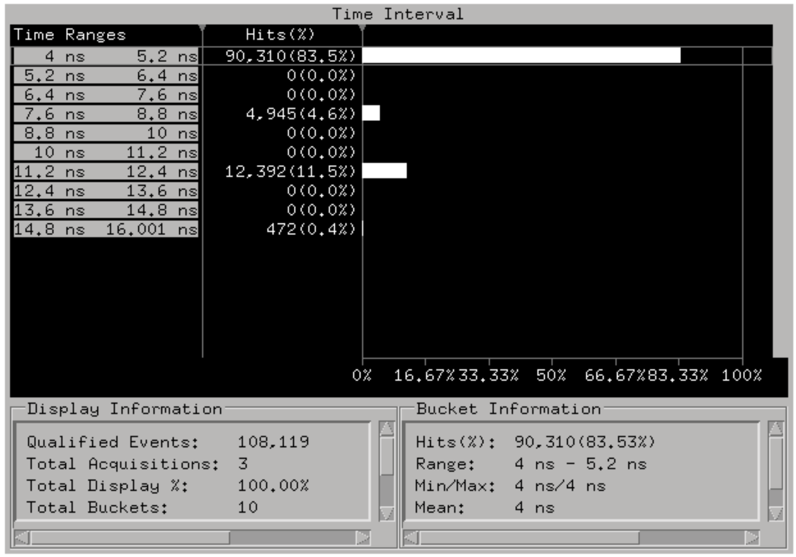
5. Set the appropriate data gathering and display options.



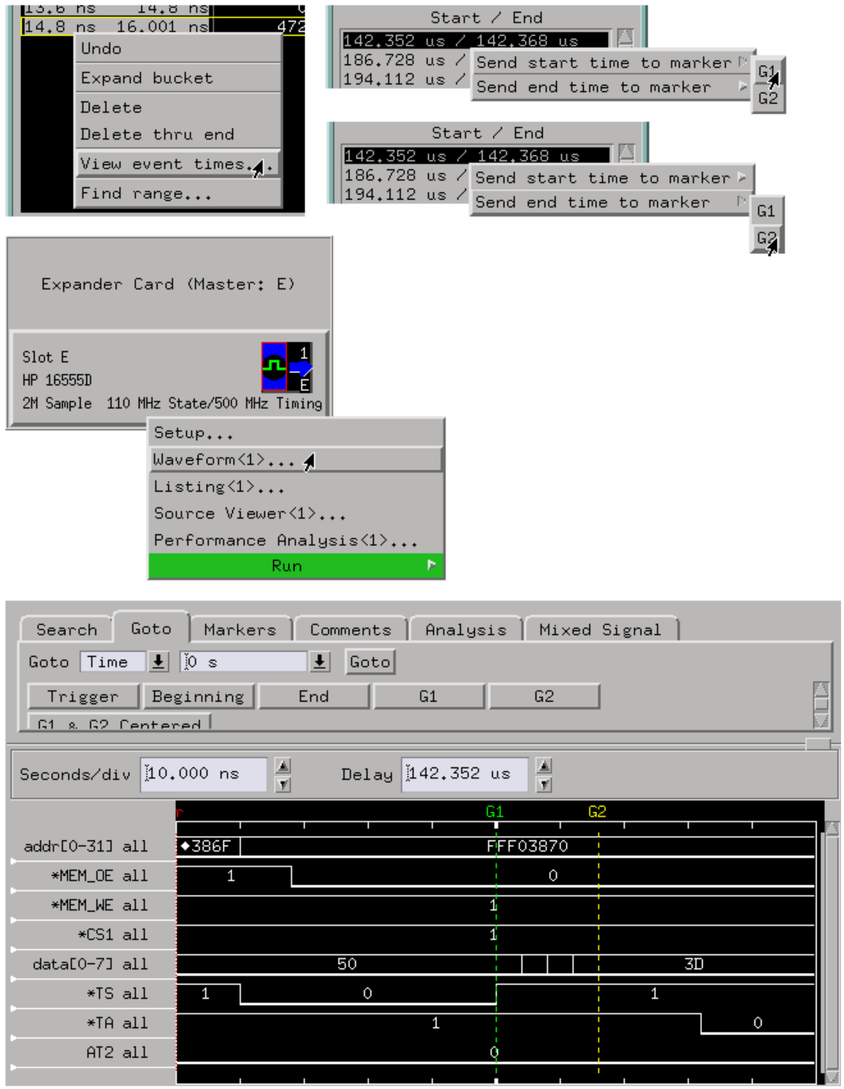


Use Accumulate Mode to analyze the behavior of your system over a long period of time (and, perhaps, run the measurement repetitively).

6. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.



You can use recorded event times to view an event in the Waveform display.

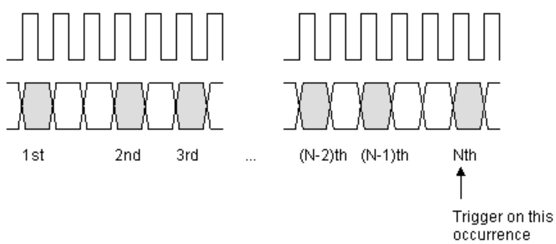


See Also

Bus Mode Search Criteria (see the *System Performance Analyzer* help volume) in the system performance analyzer help volume.

Looking at State Events

To trigger on the Nth occurrence of an event

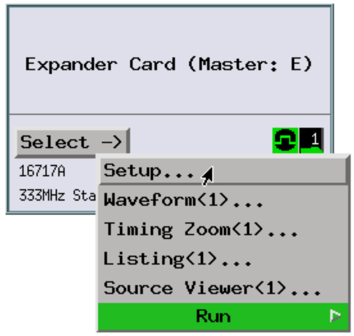


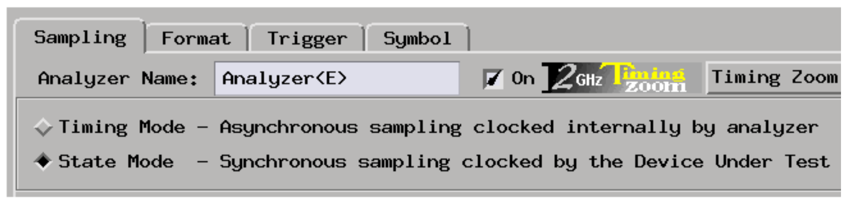
Possible uses:

- To find the 50th occurrence of a digital signal processing (DSP) subroutine.
- To trigger on the 3rd write to a specific memory address.

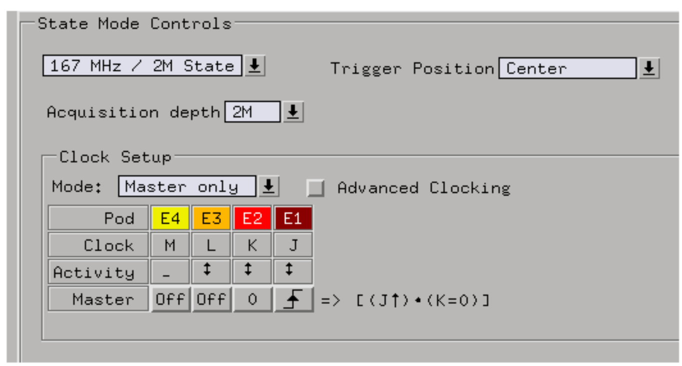
Probing the Target System

1. Configure a state analysis machine.

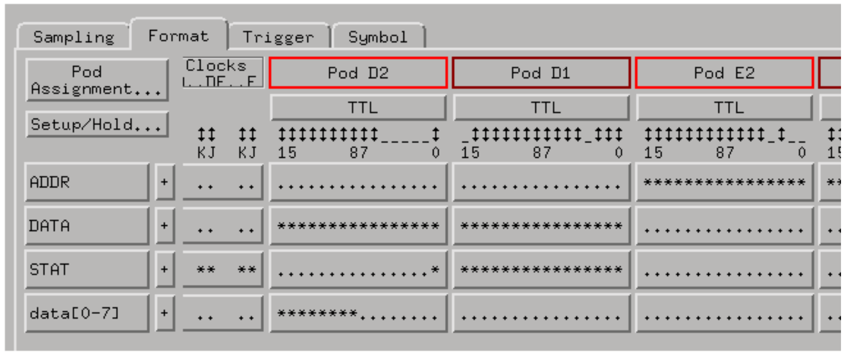




2. Select the state analyzer's clock input.

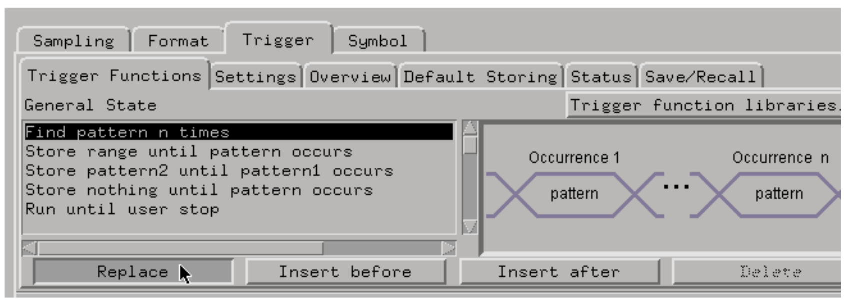


3. Format labels for the signals on which you will look for the event.

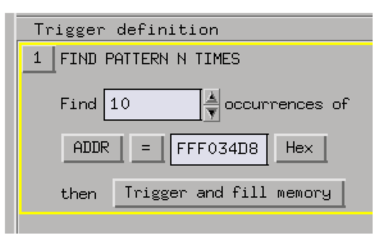


Capturing the Data

1. In the Trigger tab, use the "Find pattern n times" trigger function.



2. In the trigger definition, enter the number of occurrences and specify the pattern.

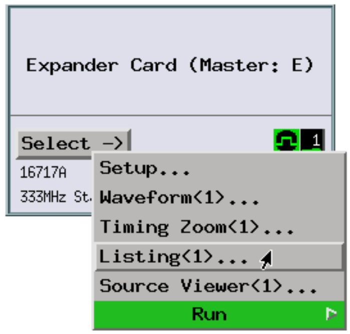


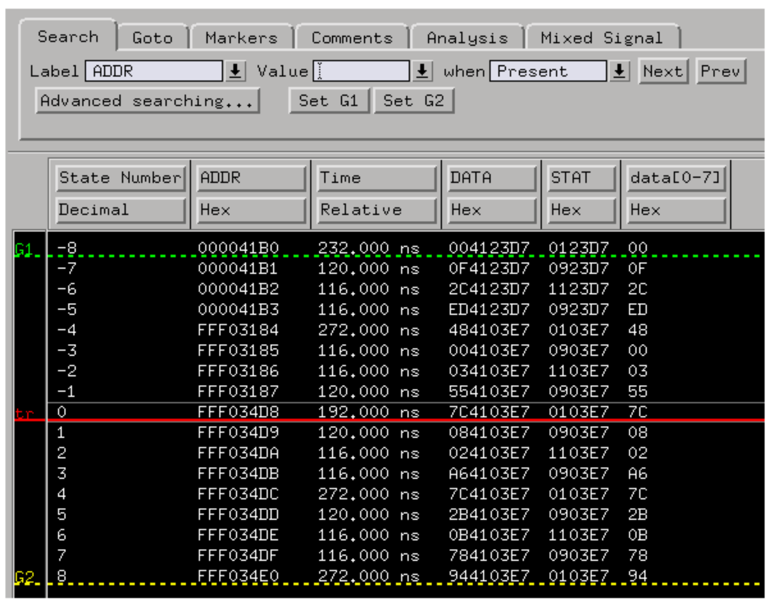
Use the default storage qualifier that is initially on and stores all states.

3. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Listing display to show that the event occurred the number of times you specified.





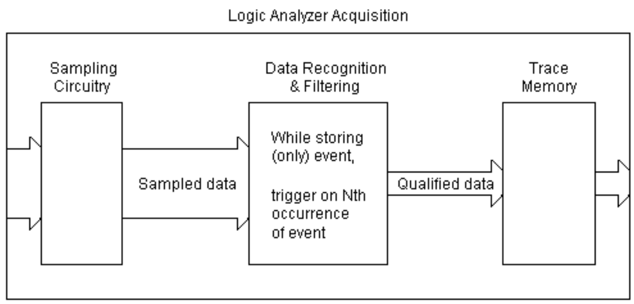
If the analyzer never triggers, the event does not occur the number of times specified. You can stop the measurement and look at the Listing display to see how many times the event did occur.

See Also

"Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

To store N samples of an event



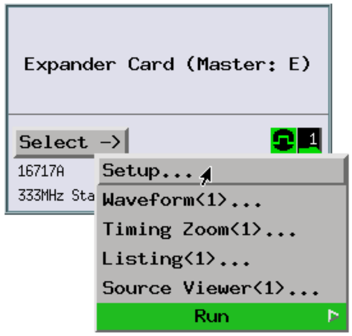
You can limit the data that is stored in trace memory at each trace sequence level or by specifying whether trace level branches are stored.

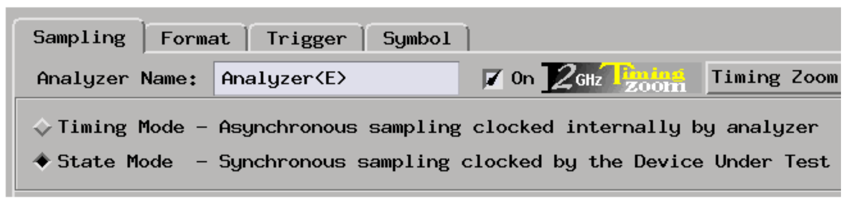
Possible uses:

- To view the first 200 reads and writes to a FIFO.
- To look at 75 pushes onto the stack.

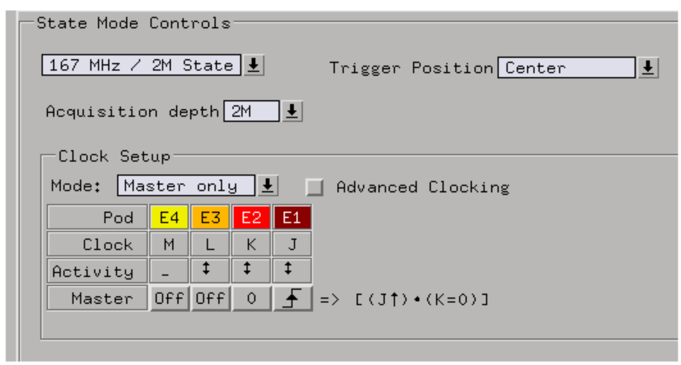
Probing the Target System

1. Configure a state analysis machine.

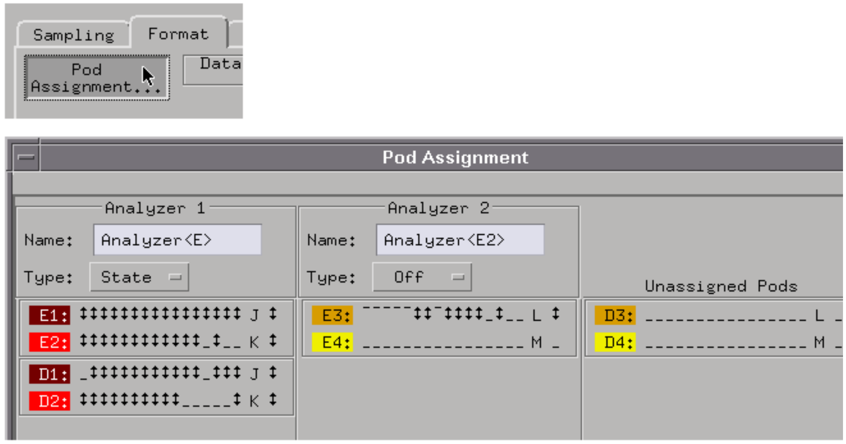




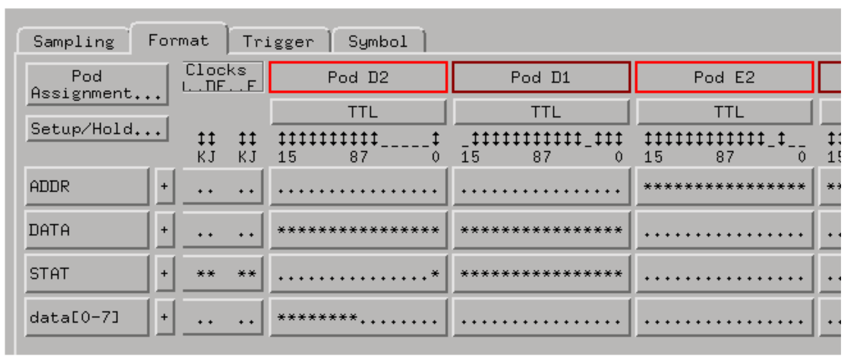
2. Select the state analyzer's clock input.



3. Assign pods if necessary.

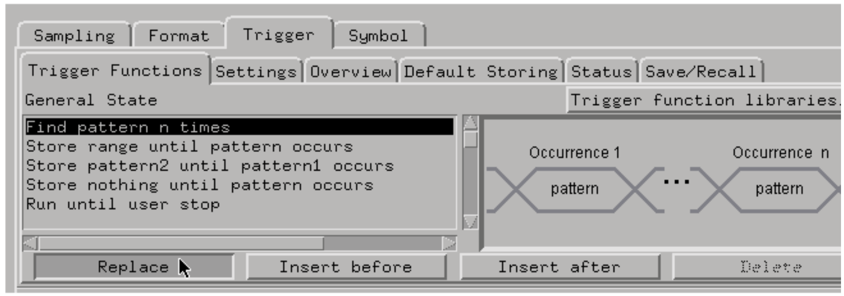


4. Format labels for the signals on which you will look for the event.

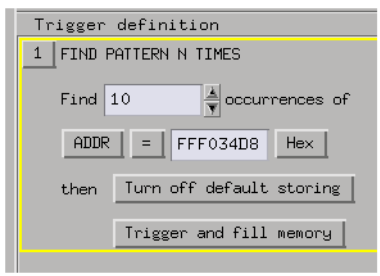


Capturing the Data

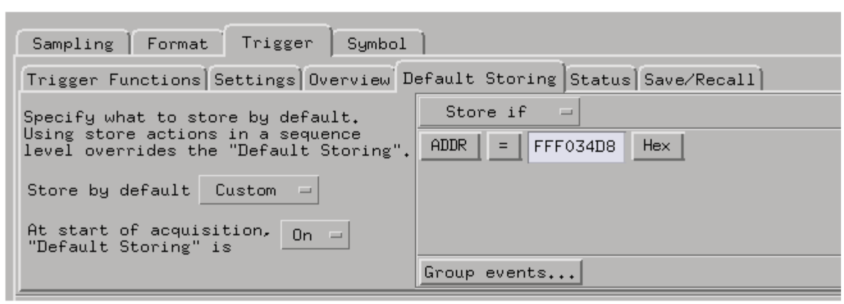
1. In the Trigger tab, use the "Find pattern n times" trigger function.



2. In the trigger definition, specify the event and the number of samples you want to store. Also, insert an action to turn off default storing after the number of events occur.



3. Set up the default storing to store only the event you're interested in.



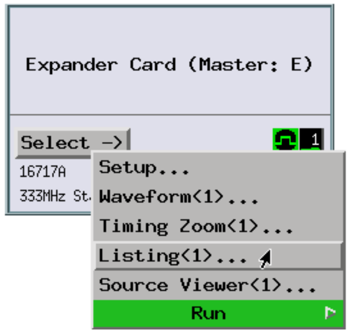
4. Select the Run button to start the measurement.

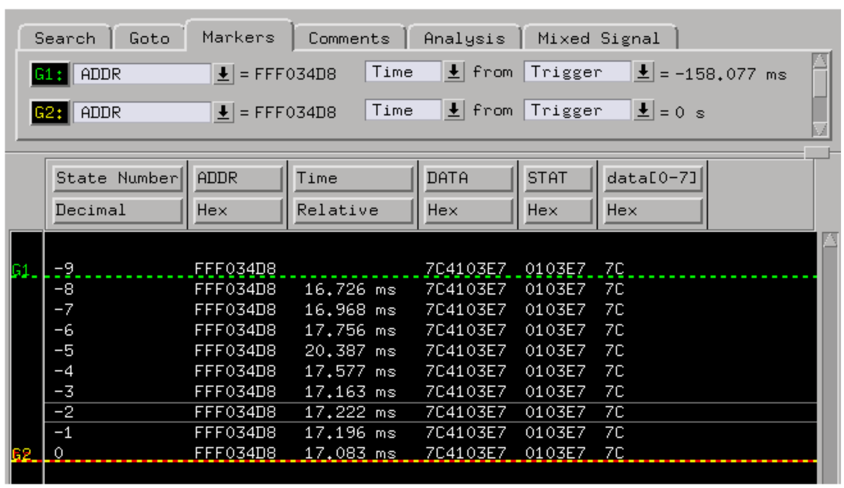
5. Select the Stop  button after the trigger occurs. (Because default

storing is turned off after the trigger, trace memory will not fill up.)

Displaying the Data

1. When the analyzer triggers, use the Listing display to show that N event occurrences are stored.



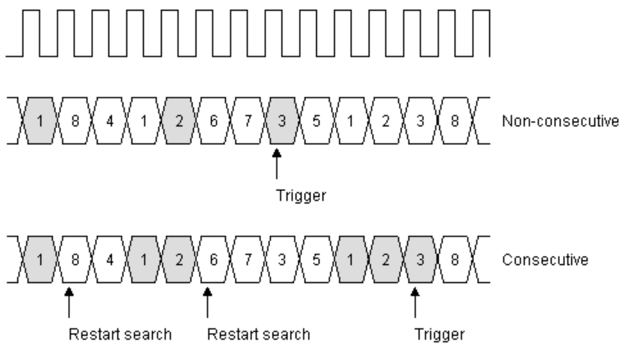


If the analyzer never triggers, the event does not occur N times. You can stop the measurement and look at the Listing display to see how many times the event did occur.

See Also

"If the trigger doesn't occur as expected" on page 309

To trigger on a sequence of events

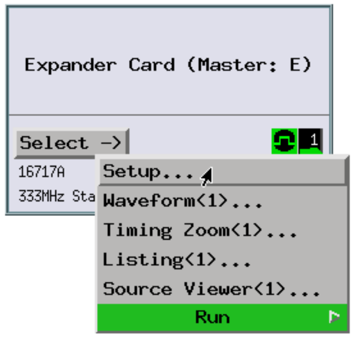


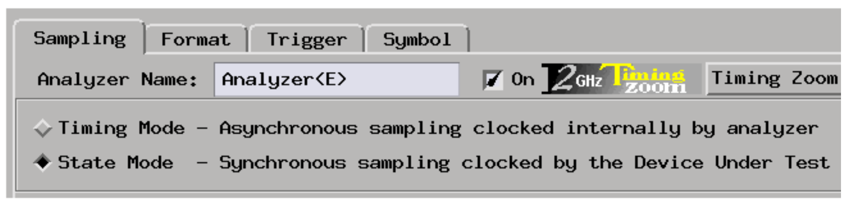
Possible uses:

- To trigger on the occurrence of a calculation subroutine *after* two initialization subroutines have executed (non-consecutive sequence of events).
- To trigger on the access to an I/O port *after* its two I/O registers have been set (non-consecutive sequence of events).
- To trigger on the occurrence of a subroutine only when it has been called from a specific branch of the main program (consecutive sequence of events).
- To look for data writes to 4 consecutive memory locations with no reads in-between (consecutive sequence of events).

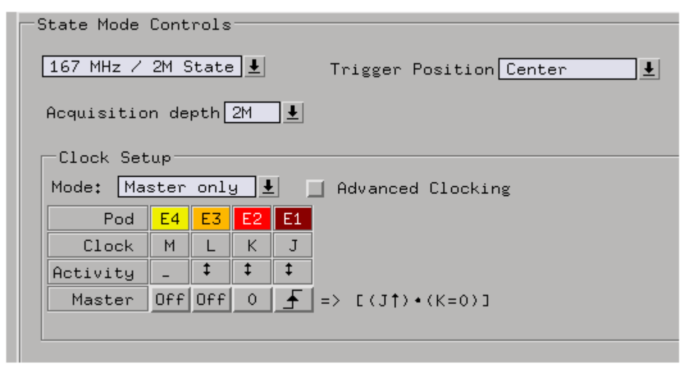
Probing the Target System

1. Configure a state analysis machine.

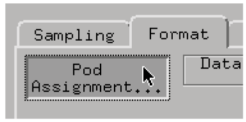


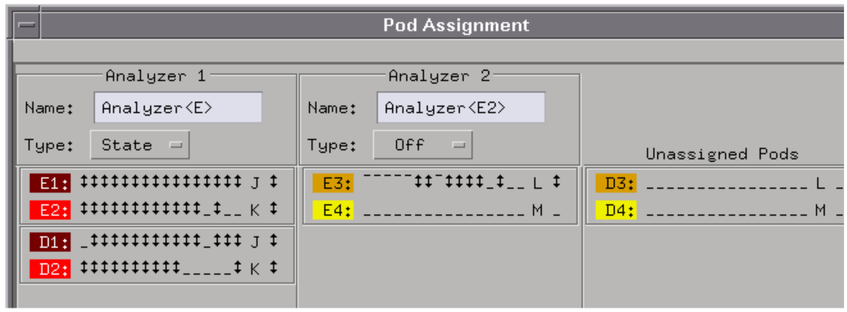


2. Select the state analyzer's clock input.

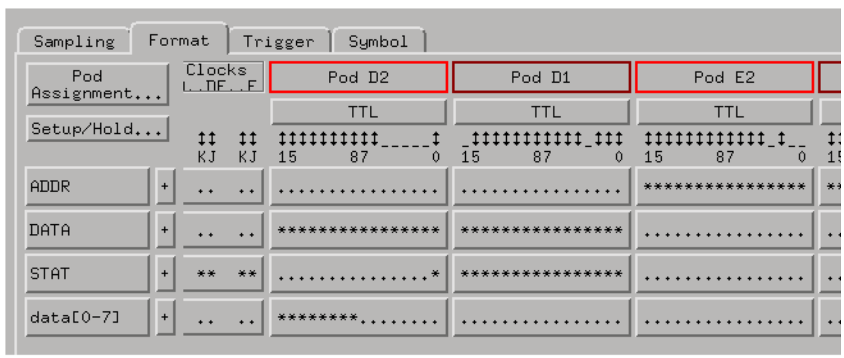


3. Assign pods if necessary.



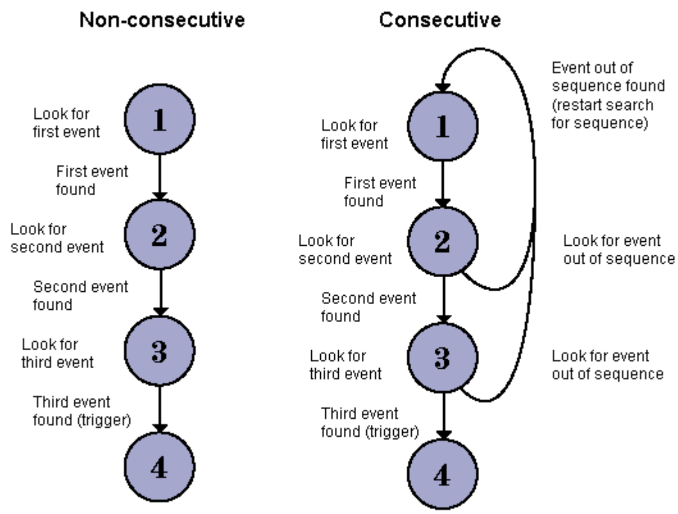


4. Format labels for the signals on which you will look for the event.



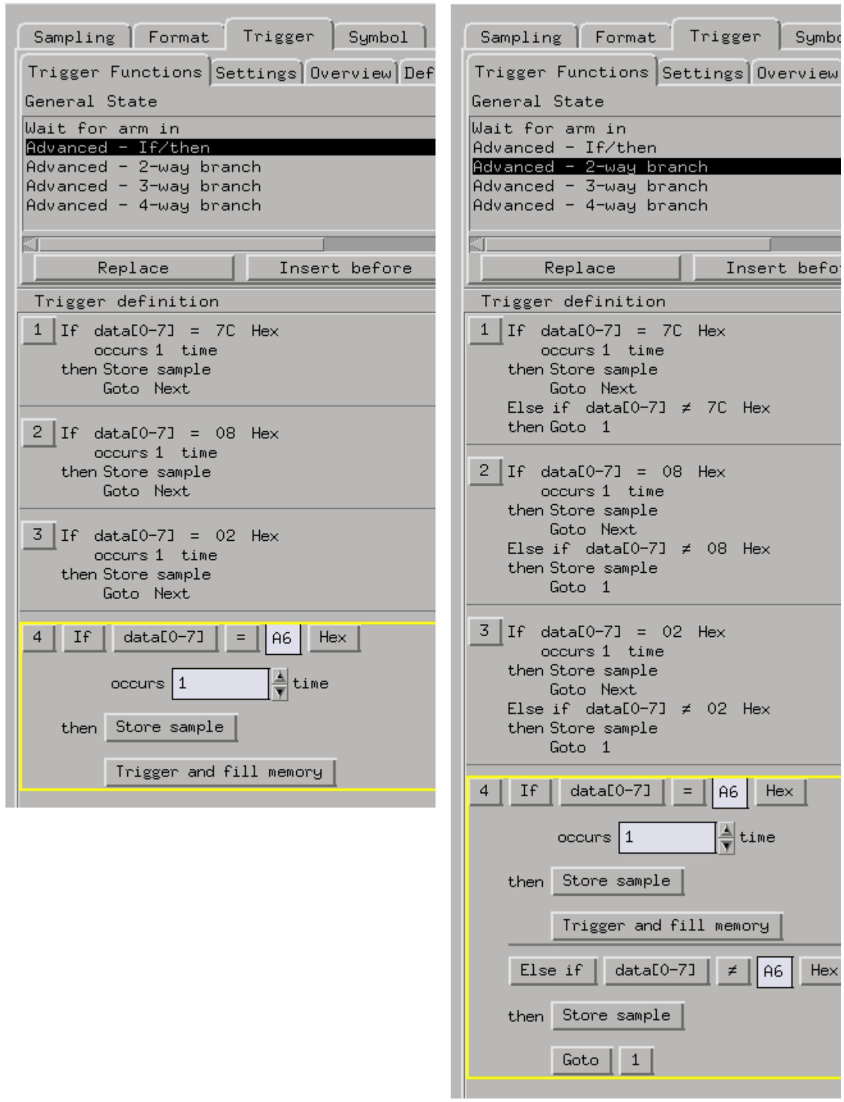
Capturing the Data

1. To look for a sequence of non-consecutive events, set up a trigger sequence where level 1 looks for the first event; when it's found, level 2 looks for the second event, and so on. The second to last sequence level looks for the last event and triggers the analyzer when it's found.



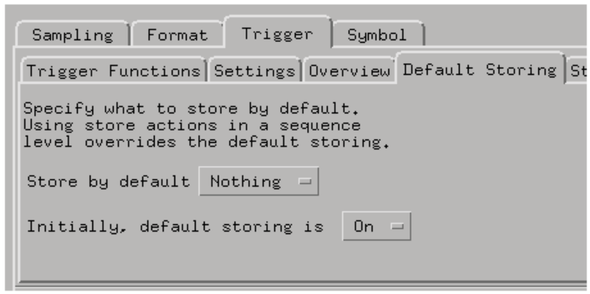
To look for a sequence of consecutive events, set up a trigger sequence where level 1 looks for the first event; when it's found, level 2 looks for the second event (which if found causes a branch to the next level) or a state that is not the second event or indicates an event out of sequence (which if found causes a branch back to the first level to restart the search).

Subsequent levels are the same as level 2. When the last event is found, the analyzer triggers, and the last sequence level is used to specify what is stored after the trigger.



Note that, when looking for a consecutive sequence of events, any event that disqualifies the sequence you're looking for can be used in the "else on" branches. For example, instead of looking for a particular sequence of states, you can look for a particular sequence of function calls.

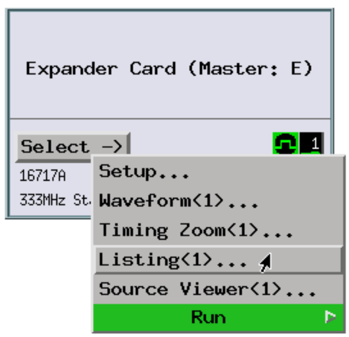
To show the differences between these trigger definitions, nothing is stored by default.

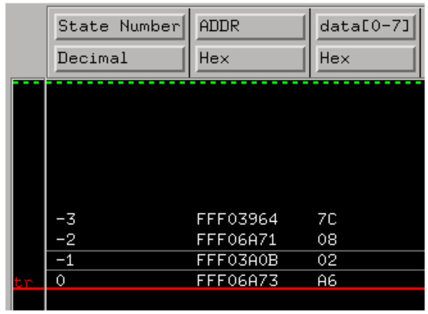


2. Select the Run button to start the measurement.

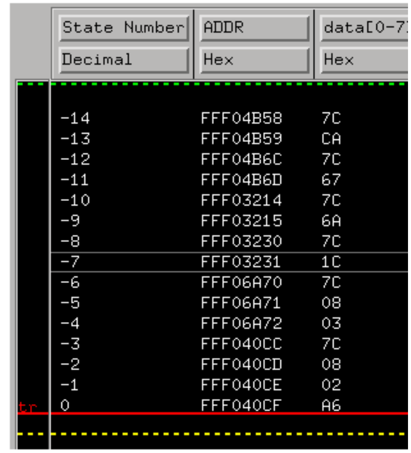
Displaying the Data

1. When the analyzer triggers, use the Listing display to show that the sequence of events occurred.





State Number Decimal	ADDR Hex	data[0-7] Hex
-3	FFF03964	7C
-2	FFF06A71	08
-1	FFF03A0B	02
0	FFF06A73	A6



State Number Decimal	ADDR Hex	data[0-7] Hex
-14	FFF04B58	7C
-13	FFF04B59	CA
-12	FFF04B6C	7C
-11	FFF04B6D	67
-10	FFF03214	7C
-9	FFF03215	6A
-8	FFF03230	7C
-7	FFF03231	1C
-6	FFF06A70	7C
-5	FFF06A71	08
-4	FFF06A72	03
-3	FFF040CC	7C
-2	FFF040CD	08
-1	FFF040CE	02
0	FFF040CF	A6

If the analyzer never triggers, at least one of the events in the sequence never occurs. You can stop the measurement and look at the Listing display to see which events in the sequence were captured.

See Also "If the trigger doesn't occur as expected" on page 309

To trigger when a program loop exits

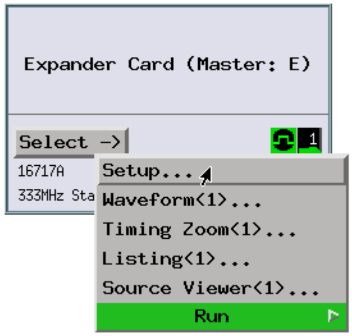
```
...
Before_Loop
Loop_Start
    In_Loop
    ...
Loop_End
After_Loop
...
```

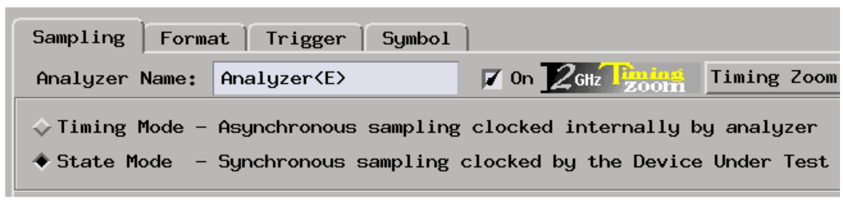
Possible uses:

- To capture execution after a background monitor loop that runs until a control key is pressed.
- To verify that all stacks and registers are restored correctly before exiting a subroutine (that is, to look at code execution before exit).

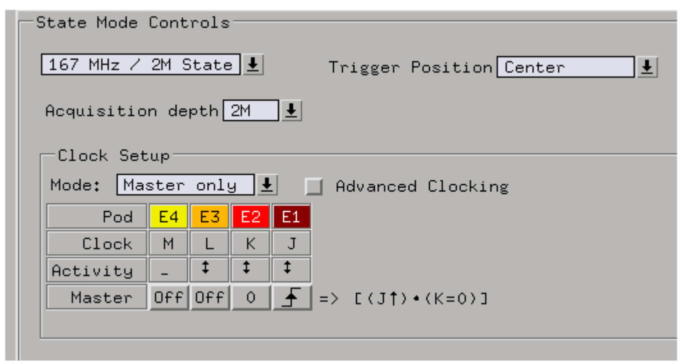
Probing the Target System

1. Configure a state analysis machine.

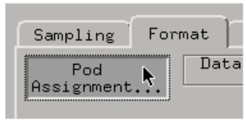


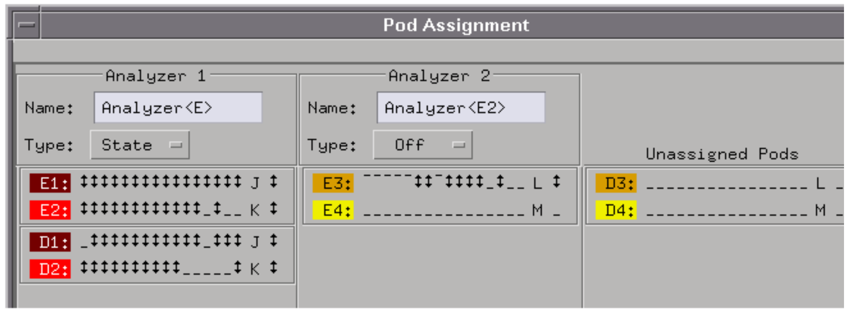


2. Select the state analyzer’s clock input.

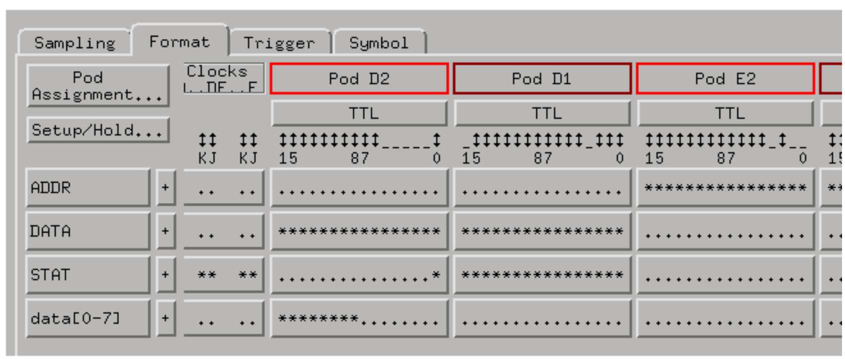


3. Assign pods if necessary.



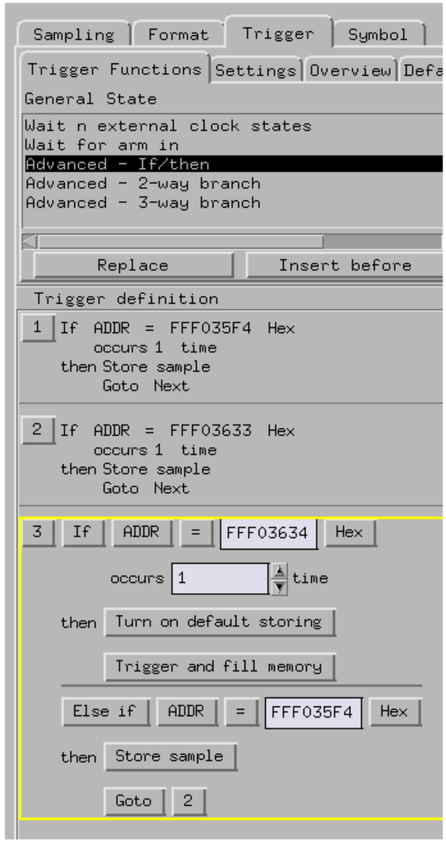


4. Format a label for the address bus signals on which you will look for loop start and loop end events. (If you are using an analysis probe, the included configuration files will format an ADDR label.)

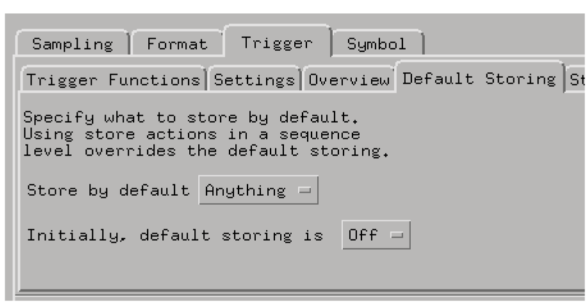


Capturing the Data

1. Set up a trigger sequence where level 1 looks for the loop start event; when it's found, level 2 looks for the loop end event; when it's found level 3 looks for an event that isn't the loop start event (which if found triggers the analyzer) or the loop start event (which if found causes a branch back to level 2 where the loop end event is searched for).



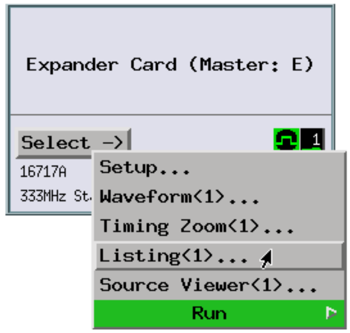
To show only loop start and loop end events until the logic analyzer triggers, turn default storing off initially.

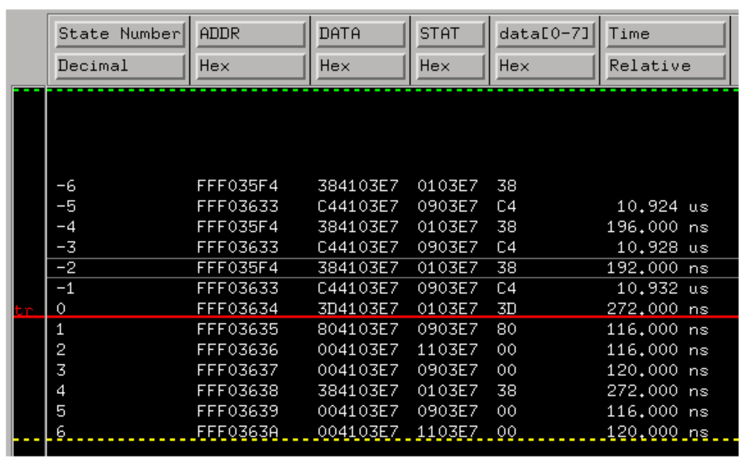


2. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Listing display to show that the program loop exited.



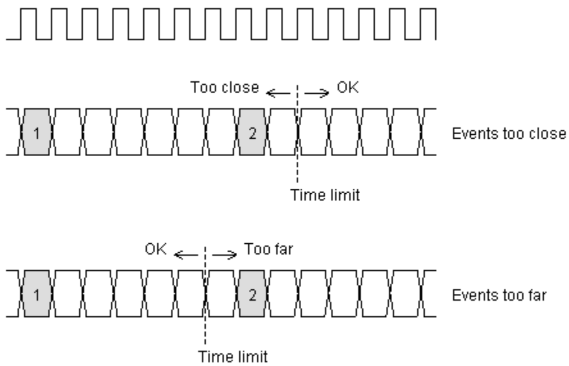


If the analyzer never triggers, you can look at the run status message line to see which sequence levels are visited, and you can learn more about why the trigger never occurred.

See Also

"If the trigger doesn't occur as expected" on page 309

To find events that are too close or too far



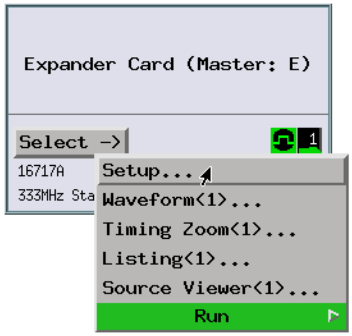
You can measure time by using a timer or by counting states or the occurrences of an event.

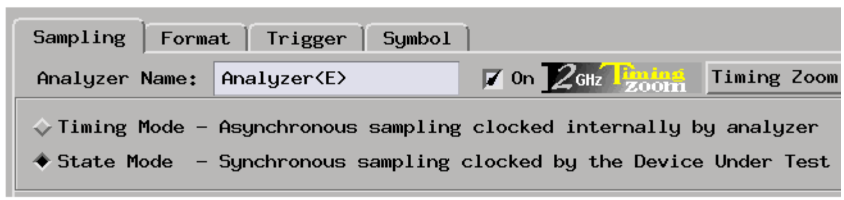
Possible uses:

- To detect when a subroutine is exited prematurely from any number of exit points (events too close, not enough cycles between subroutine entry and exit).
- To find a protocol violation in sending control messages to a peripheral (where events that are too close violate the protocol).
- To trigger when secondary cache must be accessed between 2 consecutive memory reads, producing extra cycles (events too far, that is, there are too many cycles between consecutive memory reads).
- To detect when an interrupt routine is executing for an excessive number of cycles (events too far, that is, there are too many cycles between interrupt entry and exit).
- To examine code execution when circuitry issues a data request interrupt more than N times during the execution of a time-critical subroutine (events too far, that is, there are too many interrupts between subroutine entry and exit).
- To trigger if a loop is executed more than 10 times between 2 non-consecutive routines (events too far, that is, there are too many loop executions between two routines).

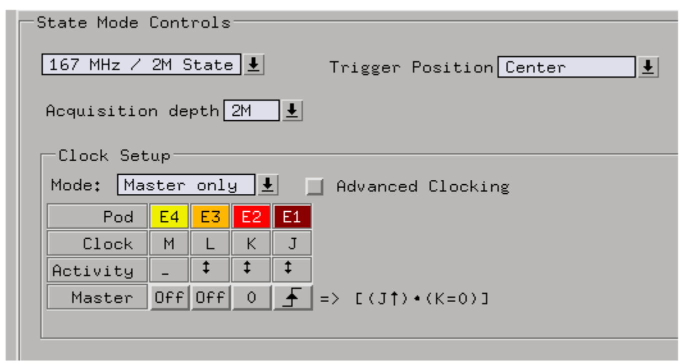
Probing the Target System

1. Configure a state analysis machine.

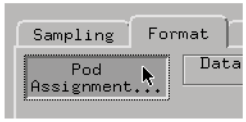


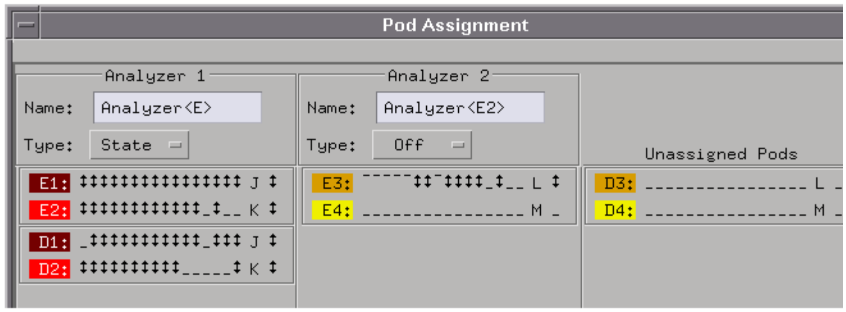


2. Select the state analyzer's clock input.

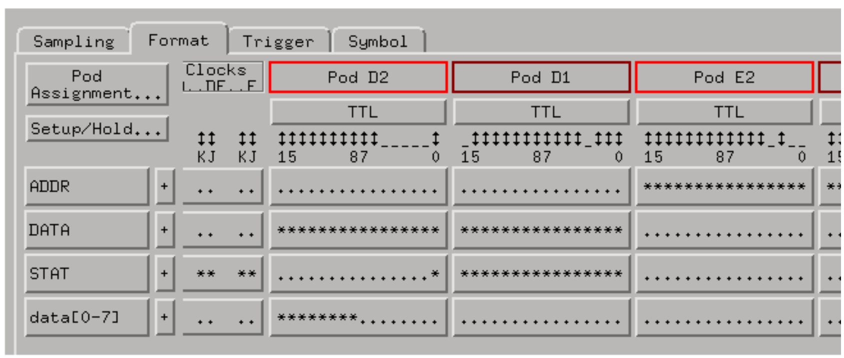


3. Assign pods if necessary.





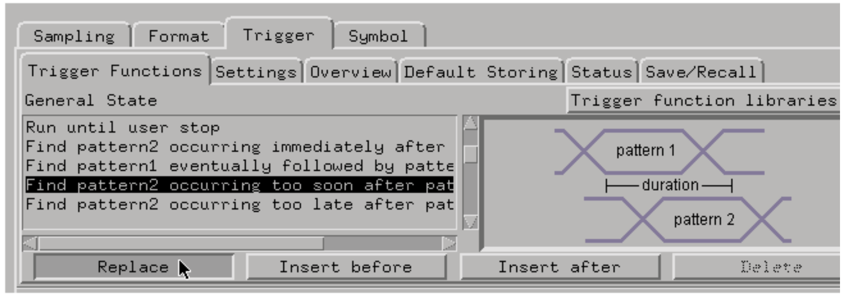
4. Format labels for the signals on which you will look for the events.



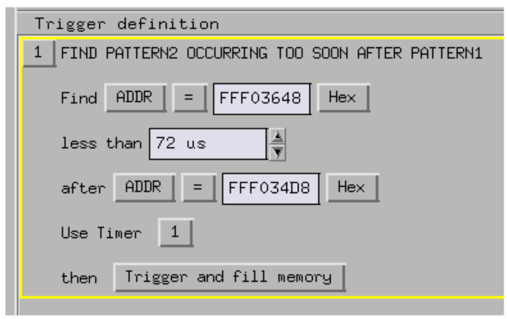
Capturing the Data

1. In the Trigger window, replace level 1 of the trigger specification with one of the following trigger functions:

   - Find pattern2 occurring too soon after pattern1
   - Find too few states between pattern1 and pattern2
   - Find pattern2 occurring too late after pattern1
   - Find too many states between pattern1 and pattern2



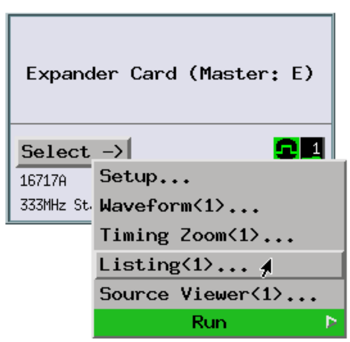
2. In the trigger definition, specify the patterns and enter the time limit.

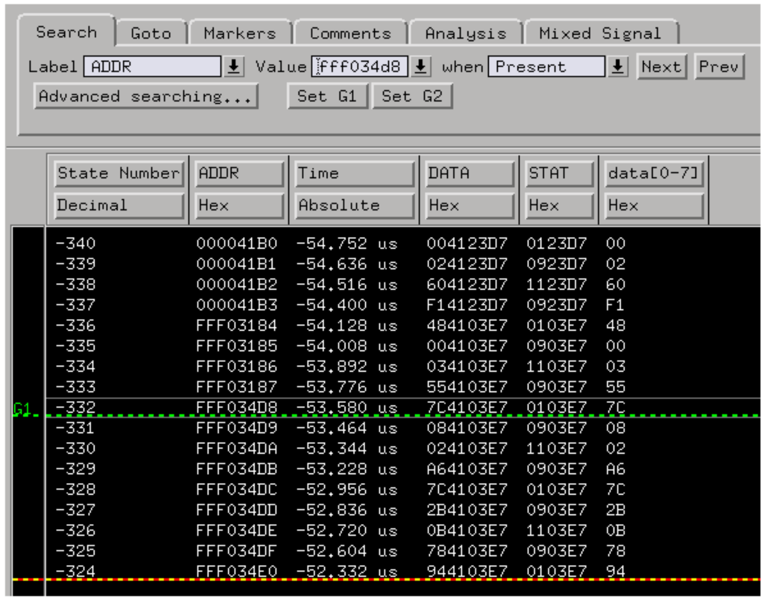


3. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Listing display to show that fewer or more than N cycles, events, or some amount of time occurred between the two events.



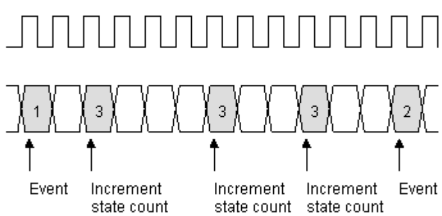


See Also

"Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

To count occurrences of an event between two events

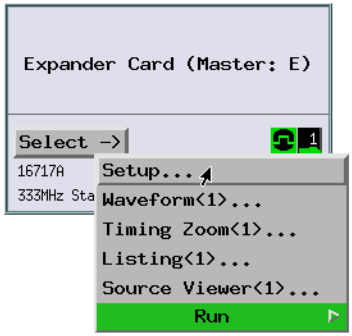


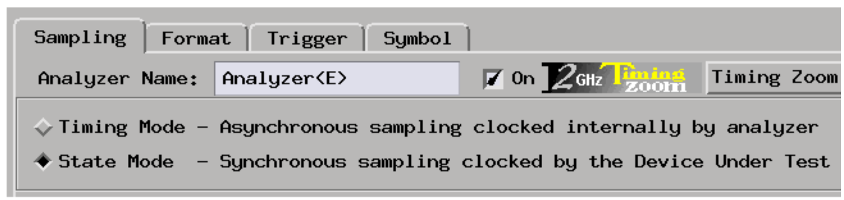
Possible uses:

- To verify that a memory refresh routine is executing the number of times expected.
- To count the number of memory write cycles within a segment of code.

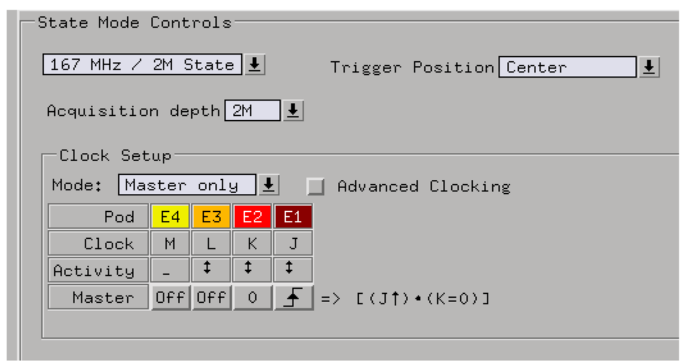
Probing the Target System

1. Configure a state analysis machine.

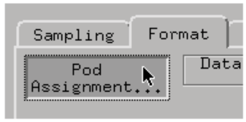


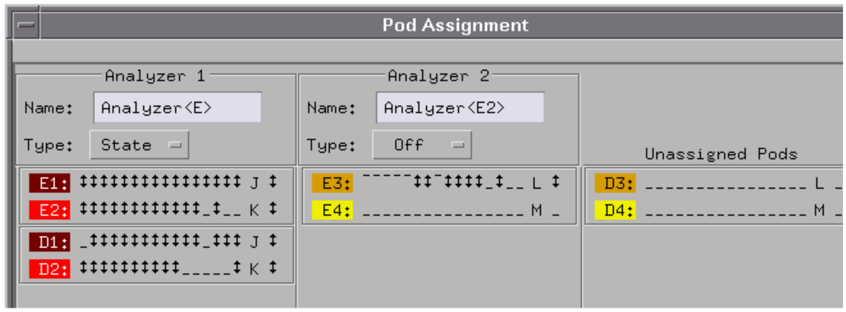


2. Select the state analyzer's clock input.

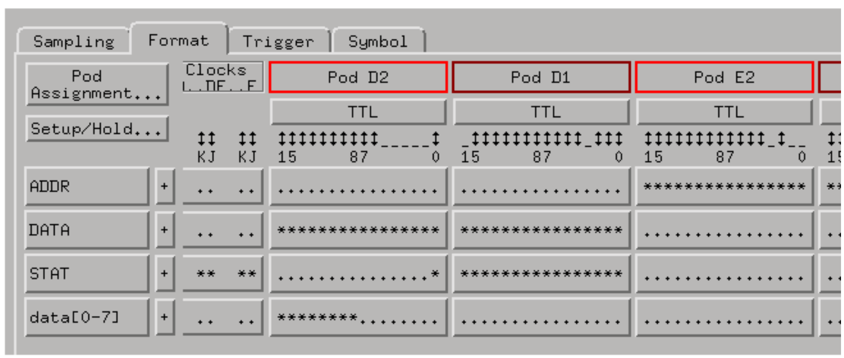


3. Assign pods if necessary.



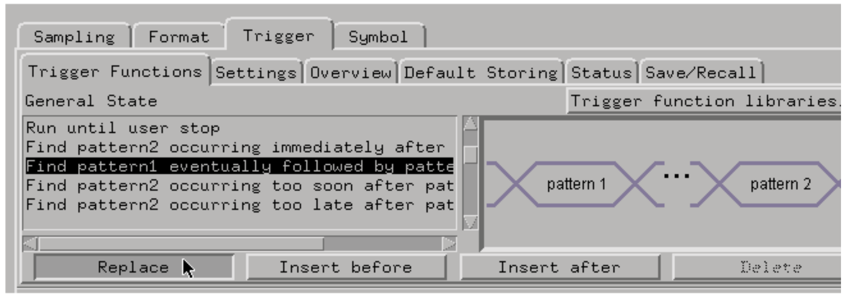


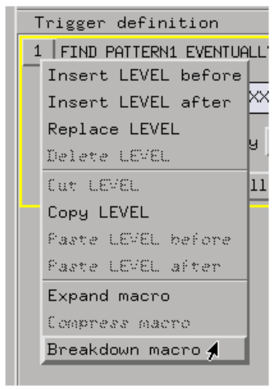
4. Format labels for the signals on which you will look for the events.

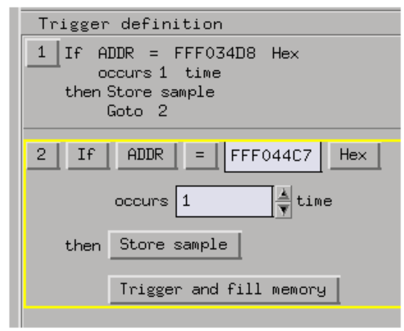


Capturing the Data

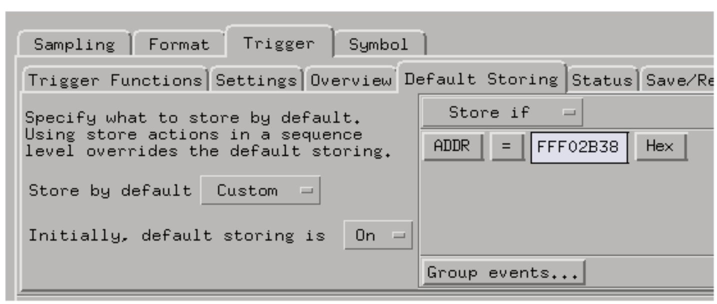
1. Set up a trigger sequence where level 1 looks for the start event; when it’s found, level 2 looks for the second event; when it’s found trigger the analyzer.



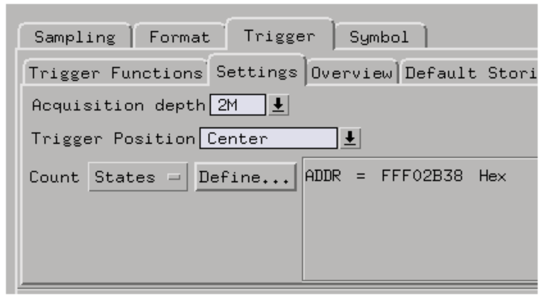




2. Store the start and end events and the event you want to count.



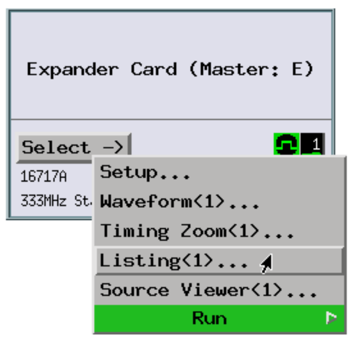
3. Change the count qualifier to count the event you are interested in.

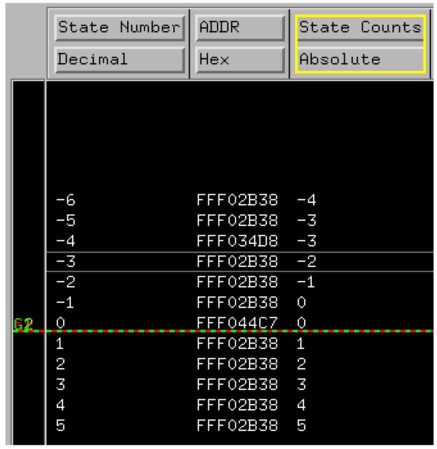


4. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Listing display to show the start and end events. The difference in count values shows the number of times the count event occurred between the start and end events.



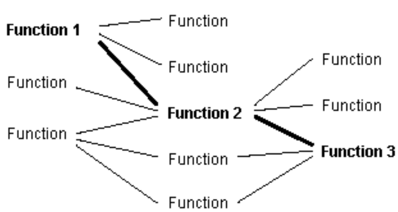


If the analyzer never triggers, the start or end events were never found. Look at the run status message to see which sequence levels are visited; this will tell you which event was not found.

See Also

"If the trigger doesn't occur as expected" on page 309

To trigger on a function call sequence

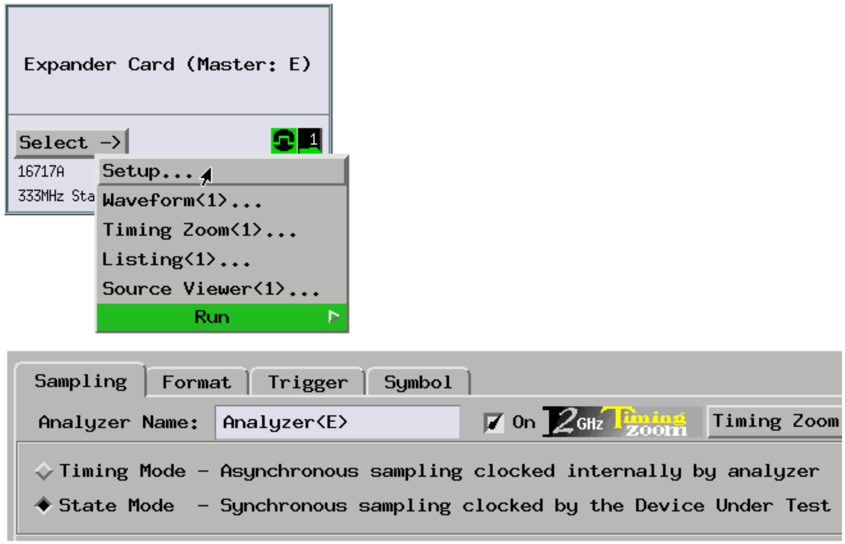


Possible uses:

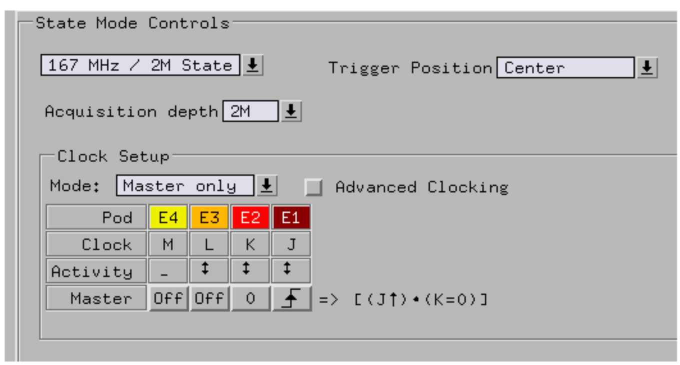
- To trigger when procedure 3 displays an error message, but only when it's called by procedure 2 and procedure 1 before that.
- To trigger on the 3rd nested occurrence of a recursive subroutine.

Probing the Target System

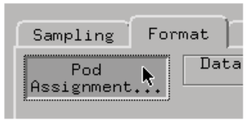
1. Configure a state analysis machine.

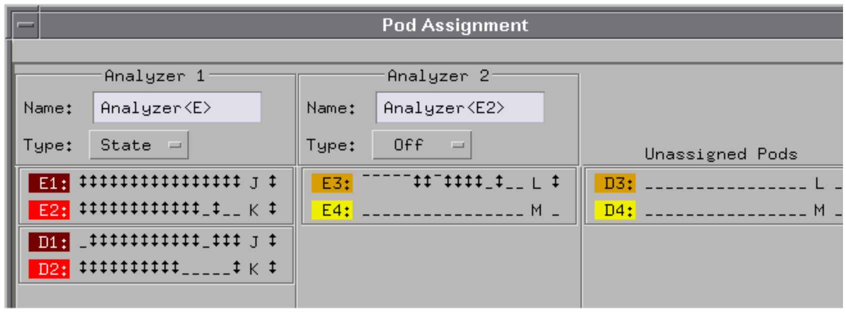


2. Select the state analyzer's clock input.

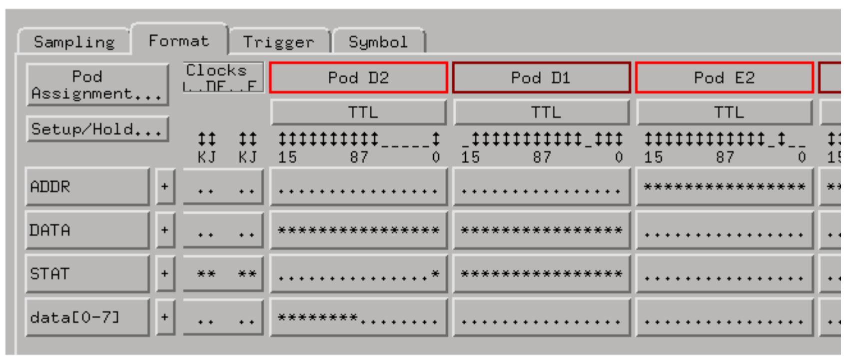


3. Assign pods if necessary.

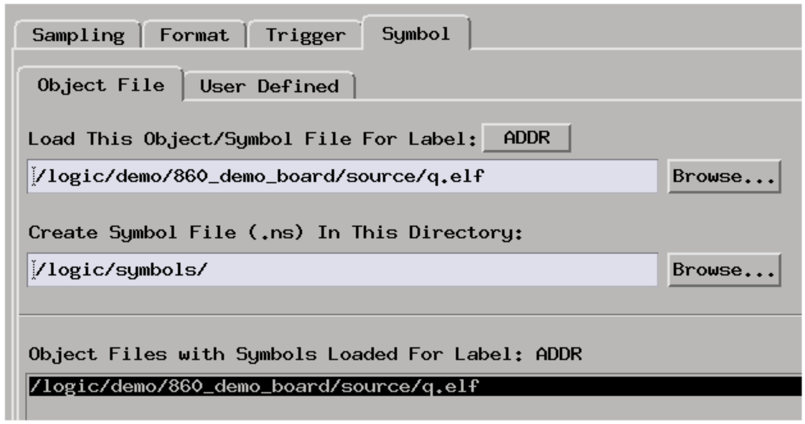




4. Format labels for the signals on which you will look for the program flow events.

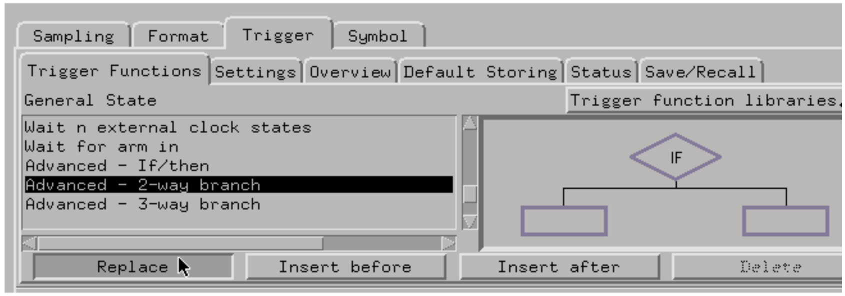


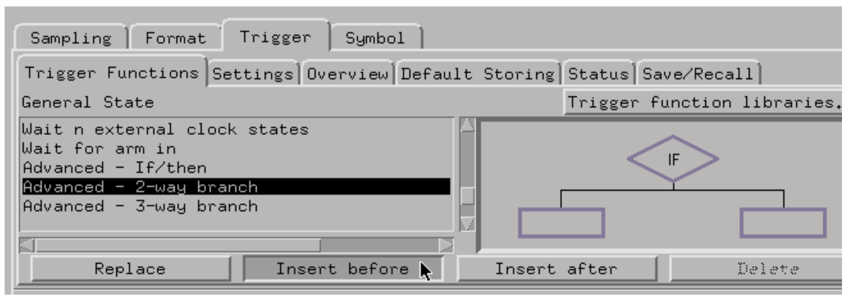
5. Load symbols from your program's object module file.

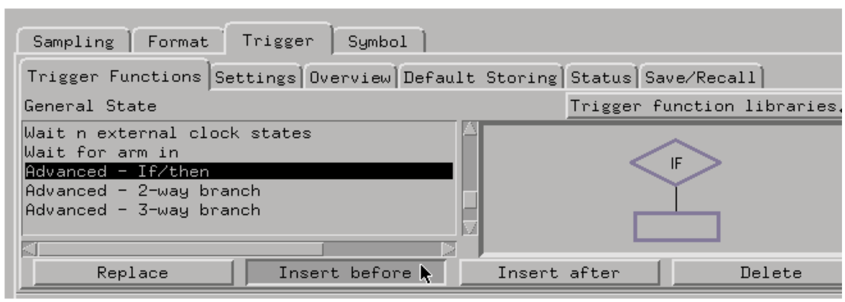


Capturing the Data

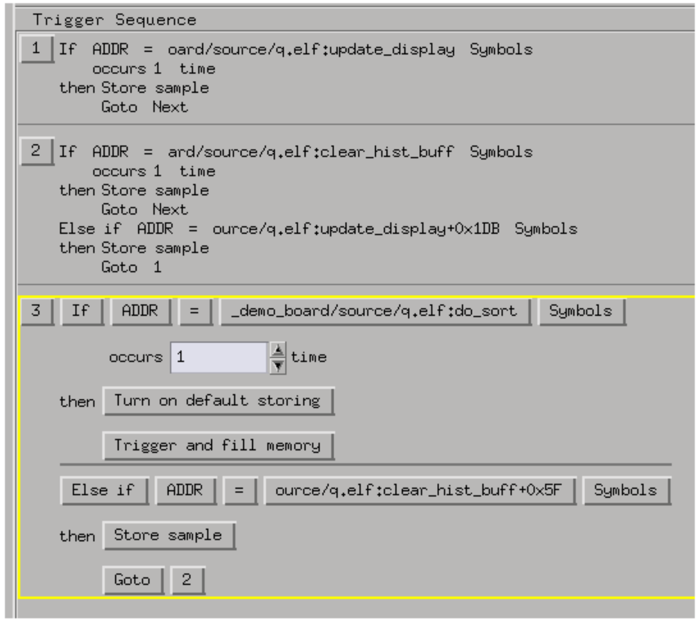
1. Set up a trigger sequence where level 1 looks for the procedure 1; when it's found, level 2 looks for procedure 2 or the end of procedure 1 (which if found will restart the search). Level 3 looks for procedure 3 (which if found will trigger the analyzer) or the end of procedure 2 (which if found will branch back to level 2).



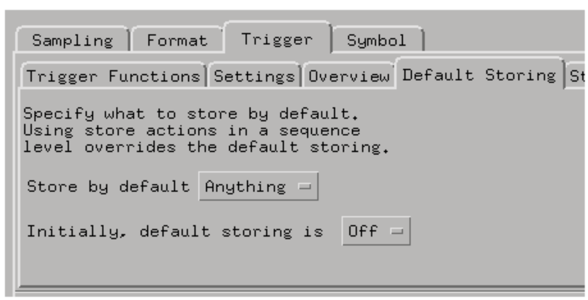




2. In the trigger definition, specify the program flow events.



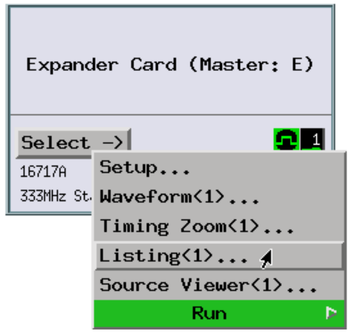
3. Set up default storing to be initially off.

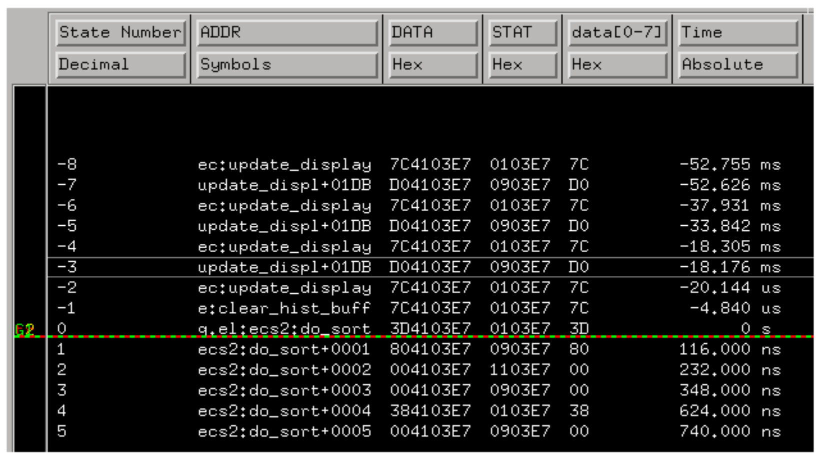


4. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Listing display to show the particular sequence was captured.





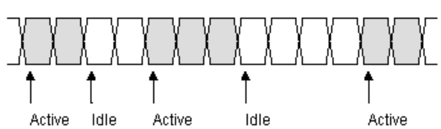
State Number Decimal	ADDR Symbols	DATA Hex	STAT Hex	data[0-7] Hex	Time Absolute
-8	ec:update_display	7C4103E7	0103E7	7C	-52.755 ms
-7	update_displ+01DB	D04103E7	0903E7	D0	-52.626 ms
-6	ec:update_display	7C4103E7	0103E7	7C	-37.931 ms
-5	update_displ+01DB	D04103E7	0903E7	D0	-33.842 ms
-4	ec:update_display	7C4103E7	0103E7	7C	-18.305 ms
-3	update_displ+01DB	D04103E7	0903E7	D0	-18.176 ms
-2	ec:update_display	7C4103E7	0103E7	7C	-20.144 us
-1	e:clear_hist_buff	7C4103E7	0103E7	7C	-4.840 us
0	g.el:ecs2:do_sort	3D4103E7	0103E7	3D	0 s
1	ecs2:do_sort+0001	804103E7	0903E7	80	116.000 ns
2	ecs2:do_sort+0002	004103E7	1103E7	00	232.000 ns
3	ecs2:do_sort+0003	004103E7	0903E7	00	348.000 ns
4	ecs2:do_sort+0004	384103E7	0103E7	38	624.000 ns
5	ecs2:do_sort+0005	004103E7	0903E7	00	740.000 ns

If the analyzer never triggers, look at the run status message to see which sequence levels are visited; this will tell you which event was not found.

See Also

"If the trigger doesn't occur as expected" on page 309

To analyze bus occupation & bandwidth (with SPA)



Bus occupation and bandwidth measurements generally show the amount of idle bus states among all bus states.

Requirements:

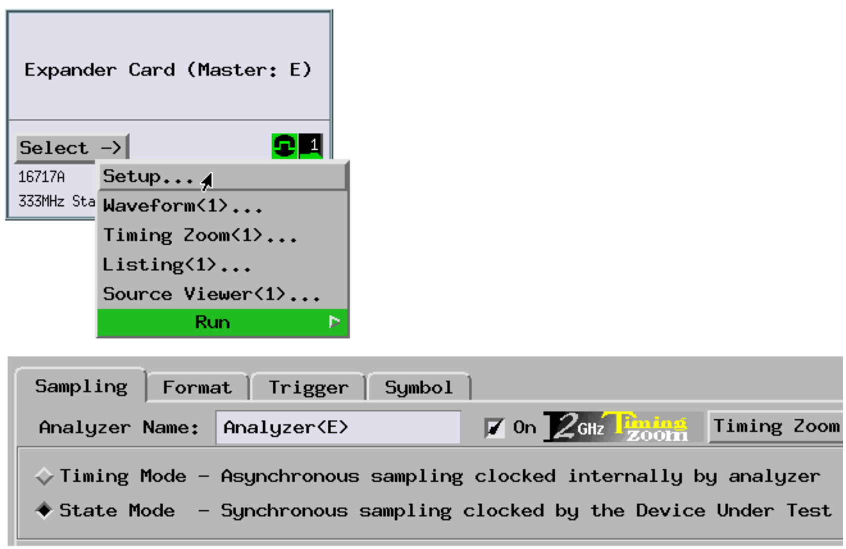
- This measurement requires the system performance analyzer (SPA) tool set.

Possible uses:

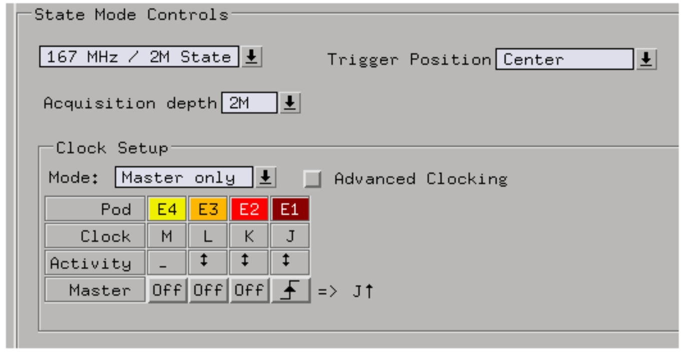
- To show the share of the workload that each processor in a multiple-processor system carries, or to determine if the system is balanced.
- To analyze headroom by examining the percentage of idle bus states.
- To analyze cache hits and misses.

Probing the Target System

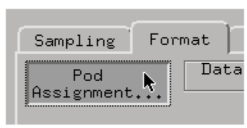
1. Probe the bus you wish to analyze.
2. Configure a state analysis machine.

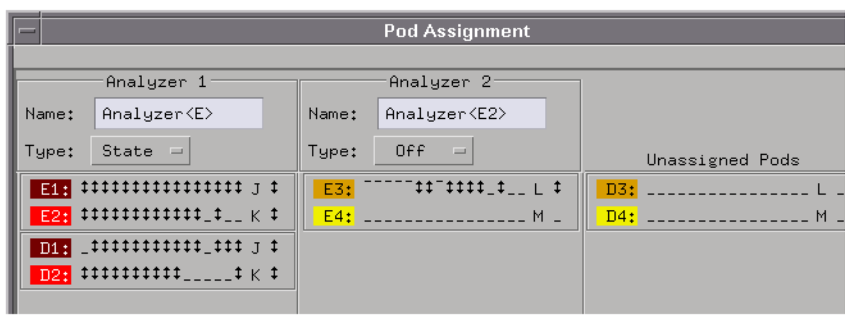


3. Select the state analyzer's clock input.

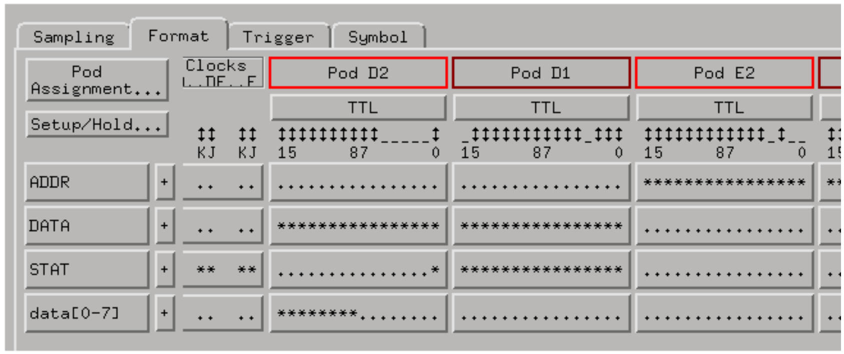


4. Assign pods if necessary.



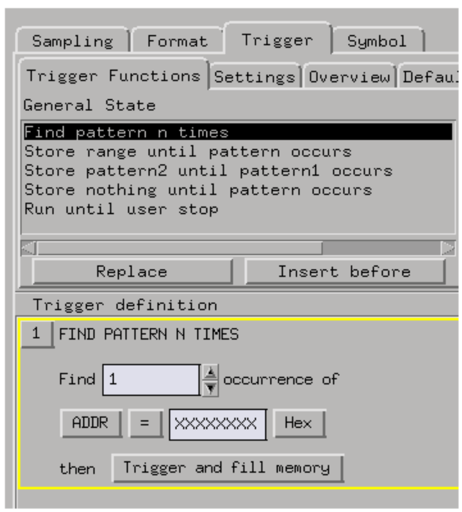


5. Label the logic analyzer signals. If you are using an analysis probe, you can configure and label signals with provided configuration files.



Capturing the Data

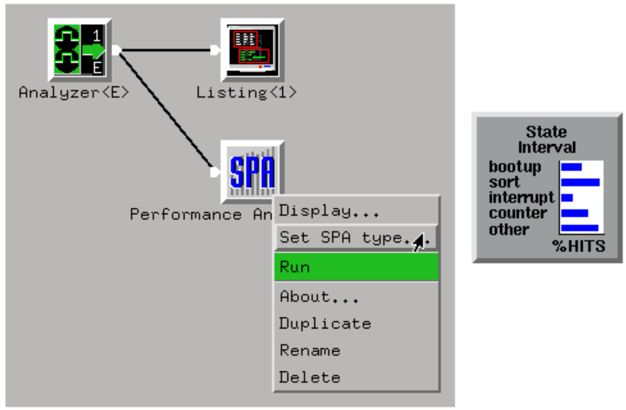
1. Set up a trigger specification to capture all activity on the bus.



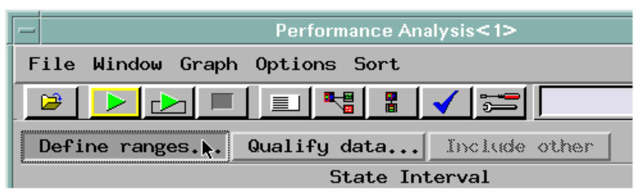
2. Select the Run button to start the measurement.

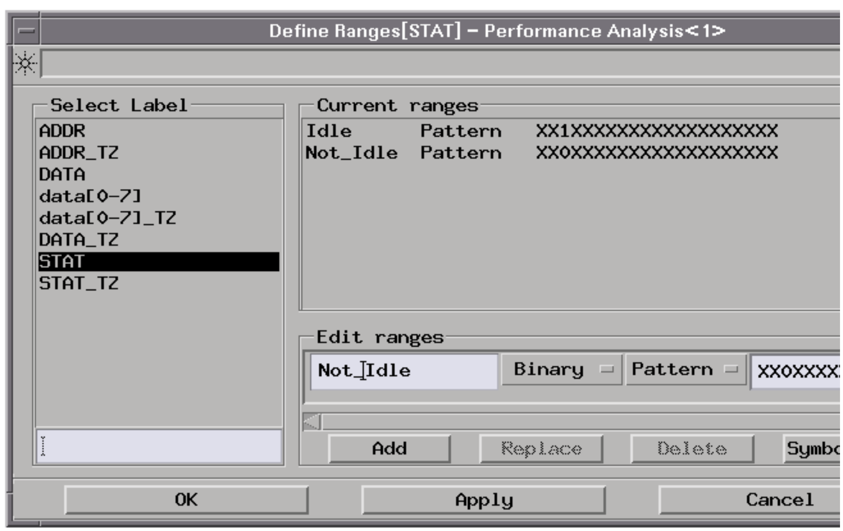
Displaying the Data

1. Use the System Performance Analyzer's State Interval display to view the measurement result.



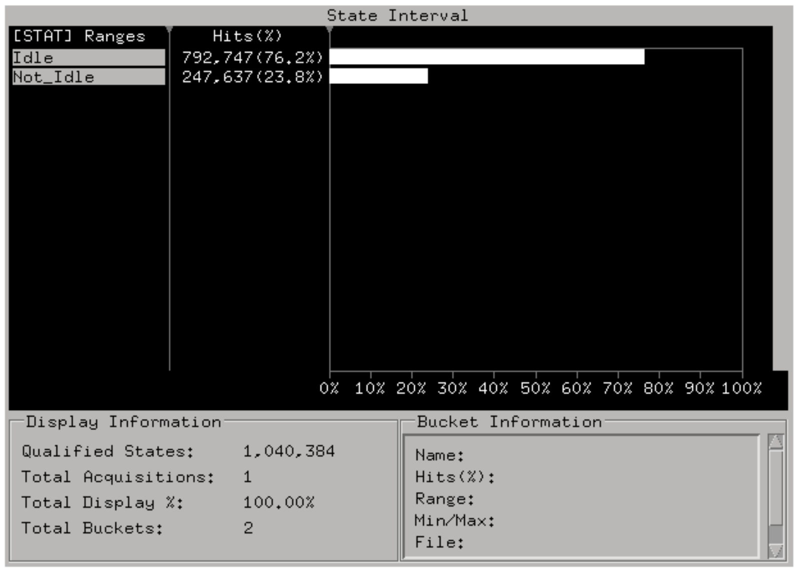
2. Define a state range that corresponds to idle bus states.





There is no limit to the number of ranges that can be simultaneously defined and displayed. The ranges can be sorted alphabetically or by number of hits.

3. Run the measurement and view the results.



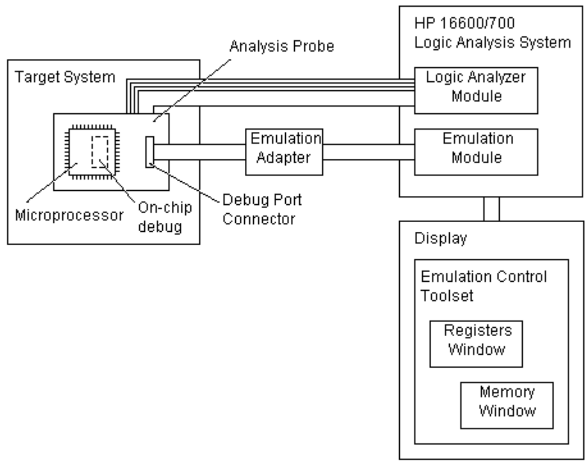
See Also

"To view bus activity" on page 174

Exercising the Microprocessor (with the Emulation Probe)

- "To initialize registers, access memory" on page 137
- "To use the emulation probe as a test tool" on page 140

To initialize registers, access memory



Requirements:

- Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

  The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

- You need either the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system or you need third-party debugger software to control the microprocessor debug interaction.

Possible uses:

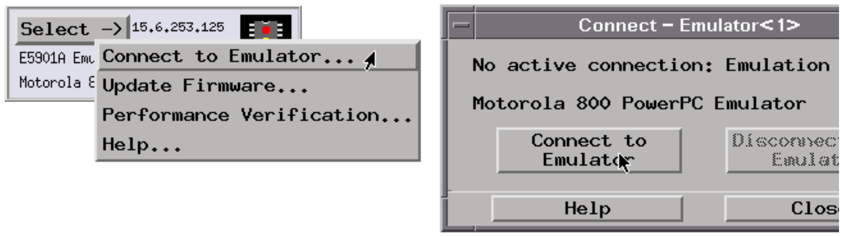
- To test microprocessor access to target system memory or I/O.
- To modify the contents of microprocessor data or configuration registers.
- To prepare the target system for downloading code to RAM.

Probing the Target System

1. Make sure the emulation probe (or emulation module and emulation adapter) has been connected to the target system.
2. Set up the emulation control tool set or third-party debugger connection to the emulation probe.

Starting the Emulation Control Software

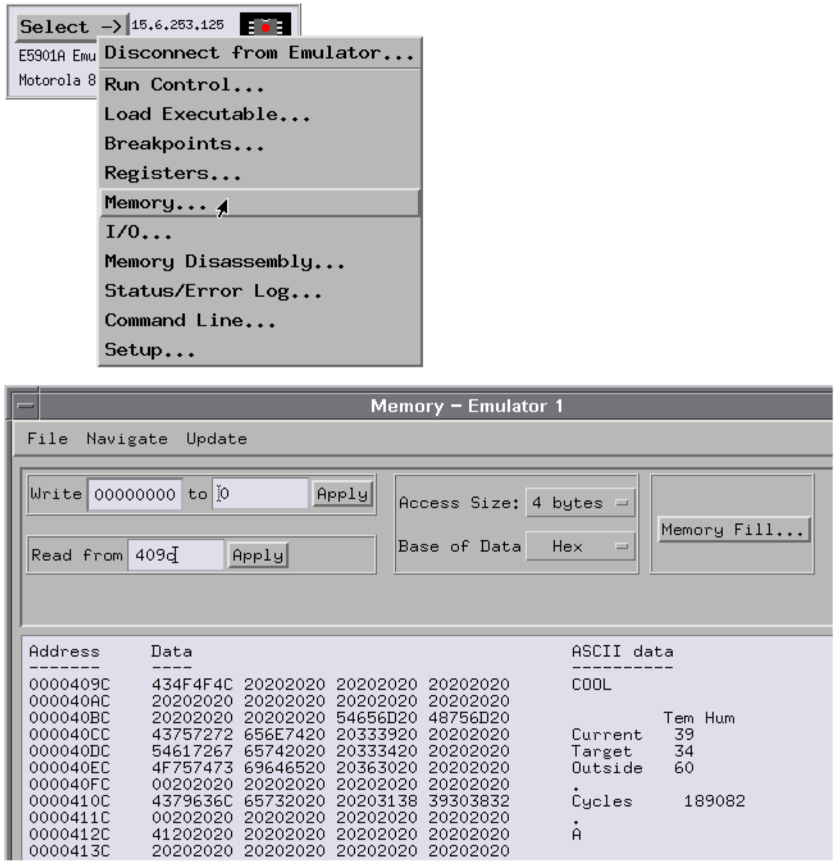
1. Start an emulation module session.



If you have third-party debugger software (on a computer in the network), start that software, and connect it to the emulation probe.

Accessing/Modifying Registers, Memory, or I/O

1. In the emulation control tool set, open the Register, Memory, or I/O window, display the locations you're interested in, and modify particular locations.



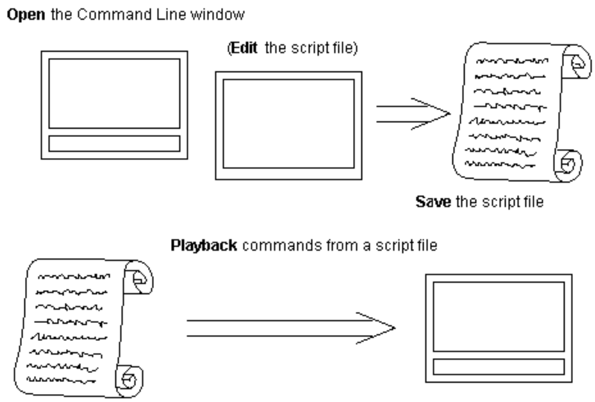
If you are using a third-party debugger, perform these tasks using its interface.

See Also

"To download boot code" on page 144

"To use the emulation probe as a test tool" on page 140

To use the emulation probe as a test tool



Requirements:

- Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

  The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

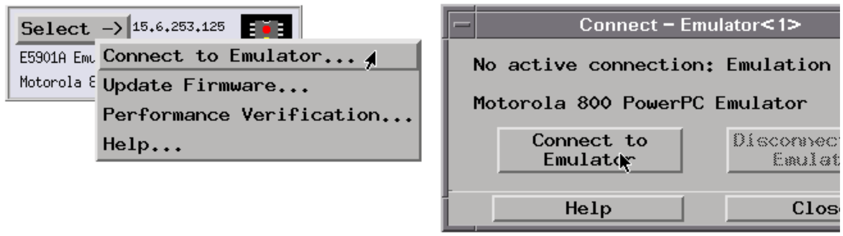
- You need the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system.

Possible uses:

- To automate a sequence of register, memory, or I/O access commands.
- To control the execution of the microprocessor as part of a system test.

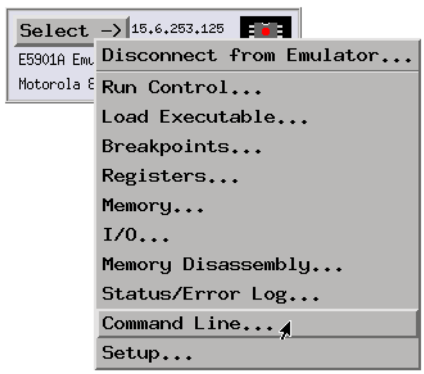
Probing the Target System

1. Make sure the emulation probe (or emulation module and emulation adapter) has been connected to the target system.
2. Set up the emulation control tool set connection to the emulation probe.



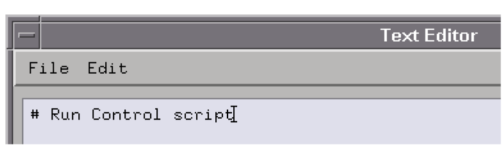
Creating/Editing the Command Script

1. Access the Command Line window.

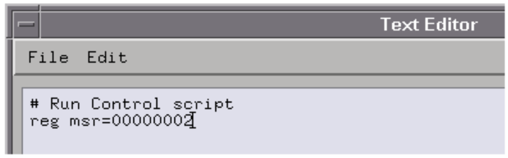


2. Select the Edit Script button.

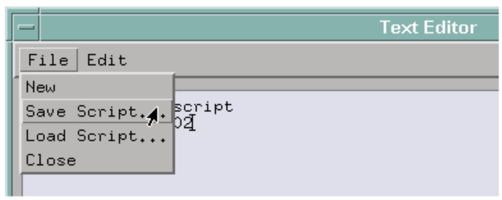
3. Enter the comment line that identifies the file as a run control script.



4. Enter commands.

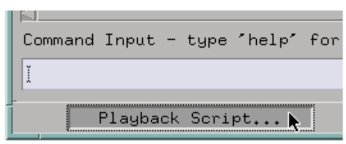


5. Save the script to a file.



Playing Back the Command Script

1. Select the Playback Script button and choose the run control script file.



See Also

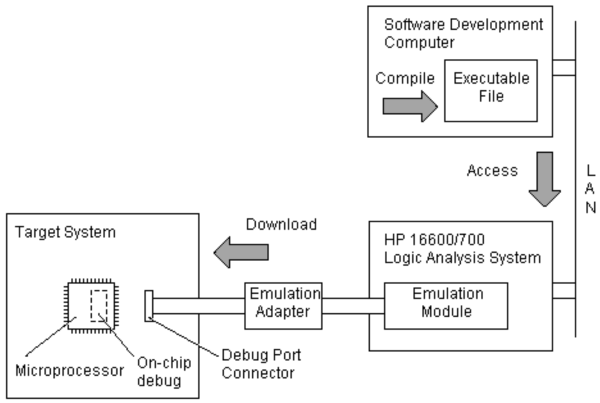
"To initialize registers, access memory" on page 137

Firmware Development

Testing Boot Code (with the Emulation Probe)

- "To capture startup execution" on page 152

To download boot code



Requirements:

- Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

  The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

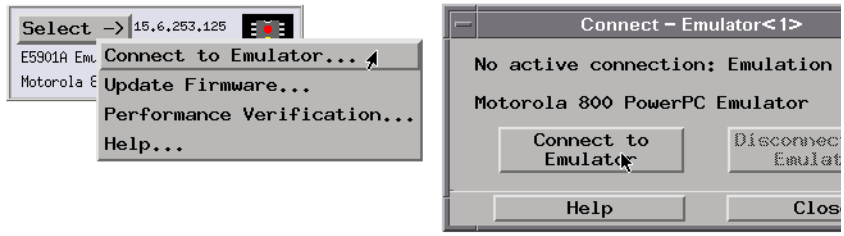
- You need the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system.

Possible uses:

- To move boot code into target system RAM for execution.

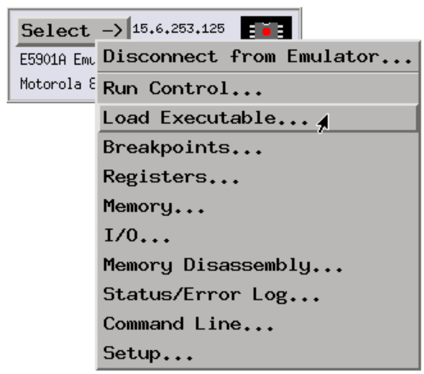
Probing the Target System

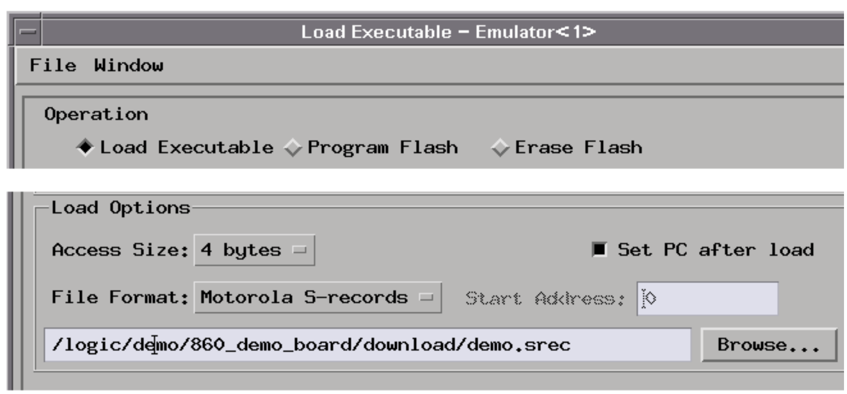
1. Make sure the emulation probe (or emulation module and emulation adapter) has been connected to the target system.
2. Set up the emulation control tool set connection to the emulation probe.



Downloading Boot Code

1. Access the Load Executable window.



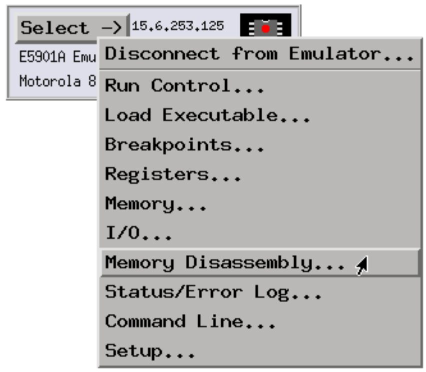


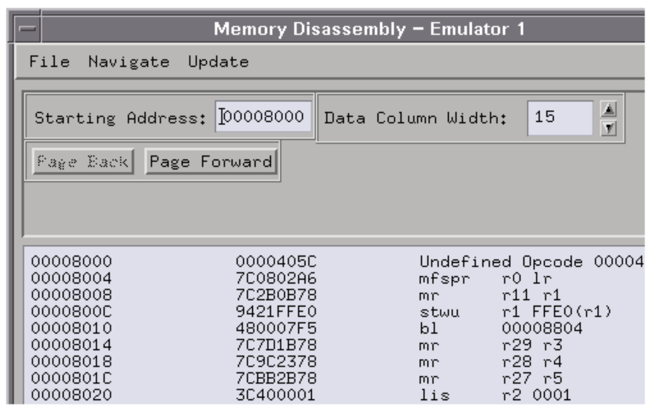
2. Select the appropriate file format, options, and executable file name.

3. Select the Apply button to download the executable file.

Verifying the Download

1. You can verify the download by looking at the memory locations in disassembled format.

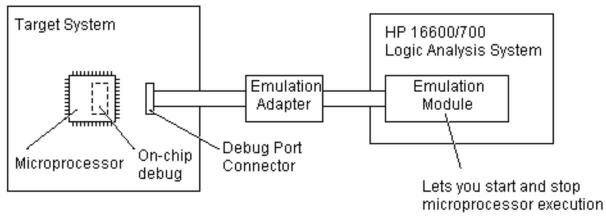




See Also

"To initialize registers, access memory" on page 137 (if you need to initialize registers before code download)

To start or stop processor execution



Requirements:

- Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

  The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

- You need either the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system or you need third-party debugger software to control the microprocessor debug interaction.

Possible uses:

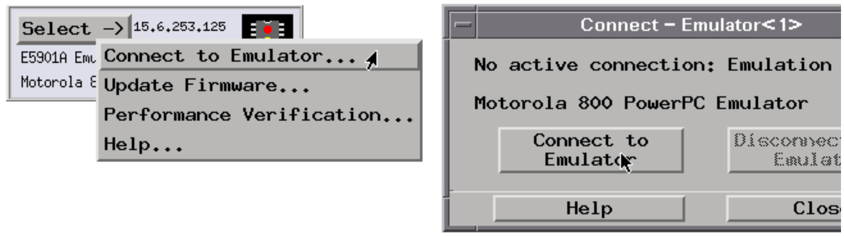
- To control the target system boot up sequence.
- To view the state of the microprocessor at particular points during program execution.

Probing the Target System

1. Make sure the emulation probe (or emulation module and emulation adapter) has been connected to the target system.
2. Set up the emulation control tool set or third-party debugger connection to the emulation probe.

Starting the Emulation Control Software

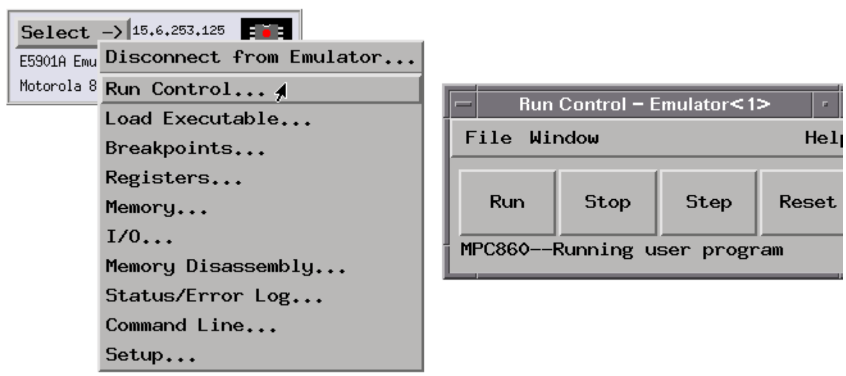
1. If you have the emulation control tool set in your Agilent Technologies 16700A/B-series logic analysis system, drag the icon to the workspace, and connect to the emulation probe.



If you have third-party debugger software, start that software, and connect to the emulation probe.

Controlling Processor Execution

1. In the emulation control tool set, open the Run Control window, and select the Run or Stop buttons.

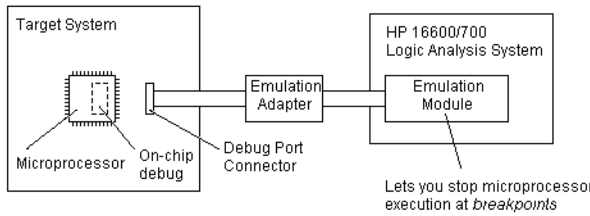


If you have a third-party debugger, perform these tasks using its interface.

See Also

"To stop processor execution using breakpoints" on page 149

To stop processor execution using breakpoints



Requirements:

- Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

  The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

- You need either the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system or you need third-party debugger software to control the microprocessor debug interaction.

Possible uses:

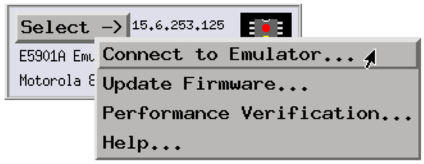
- To stop microprocessor execution on a particular line of source code.
- To view the state of the microprocessor at particular points during program execution.

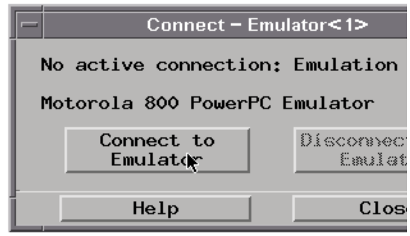
Probing the Target System

1. Make sure the emulation probe (or emulation module and emulation adapter) has been connected to the target system.
2. Set up the emulation control tool set or third-party debugger connection to the emulation probe.

Starting the Emulation Control Software

1. If you have the emulation control tool set in your Agilent Technologies 16700A/B-series logic analysis system, drag the icon to the workspace, and connect to the emulation probe.

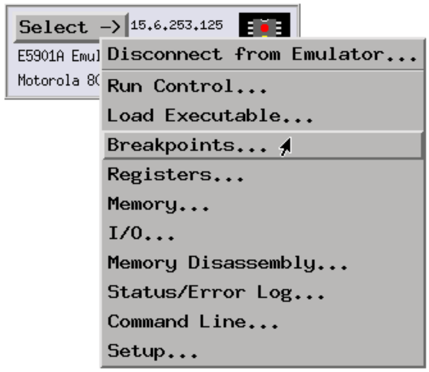


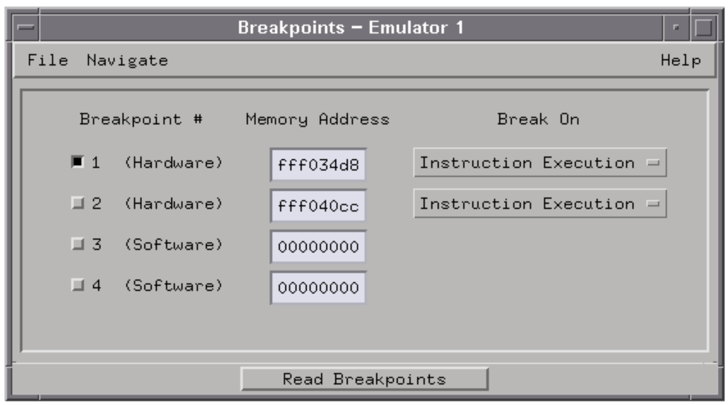


If you have third-party debugger software, start that software, and connect to the emulation probe.

Setting Breakpoints

1. In the emulation control tool set, open the Breakpoints window, select a breakpoint to use, and enter the address at which microprocessor execution should stop.





Hardware breakpoints can be used for addresses in target system ROM.

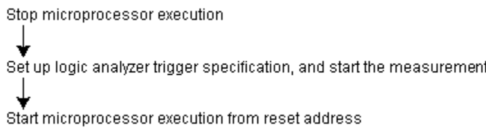
Software breakpoints replace existing code with a *breakpoint instruction*, so they only work for addresses in target system RAM.

If you have a third-party debugger, perform these tasks using its interface.

See Also

"To start or stop processor execution" on page 147

To capture startup execution

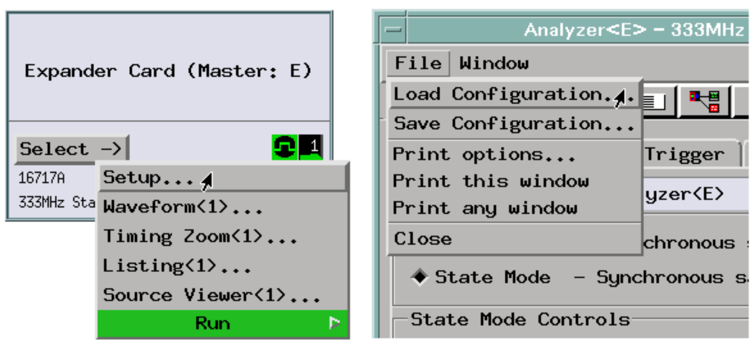


Possible uses:

- To verify boot code operation.

Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

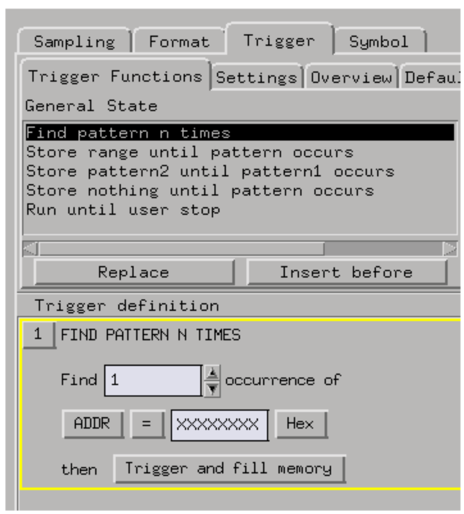


You may also want to connect an emulation probe to the target system microprocessor's debug port (either via a connector designed into the target system or the connector provided on the analysis probe). The emulation control tool set's reset/run control lets you make this measurement without cycling power in your target system.

Capturing the Data

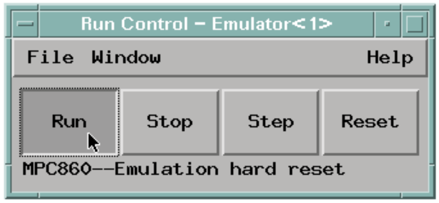
1. Stop microprocessor execution (either by turning OFF power to the target system or using the emulation control tool set's Reset command).

2. Set up a trigger specification to trigger on anything and capture everything.



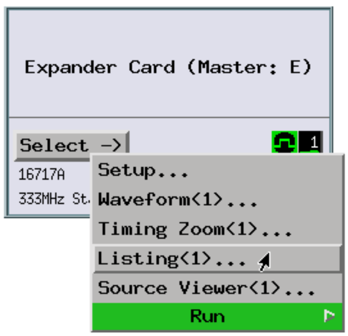
3. Select the Run button to start the measurement.

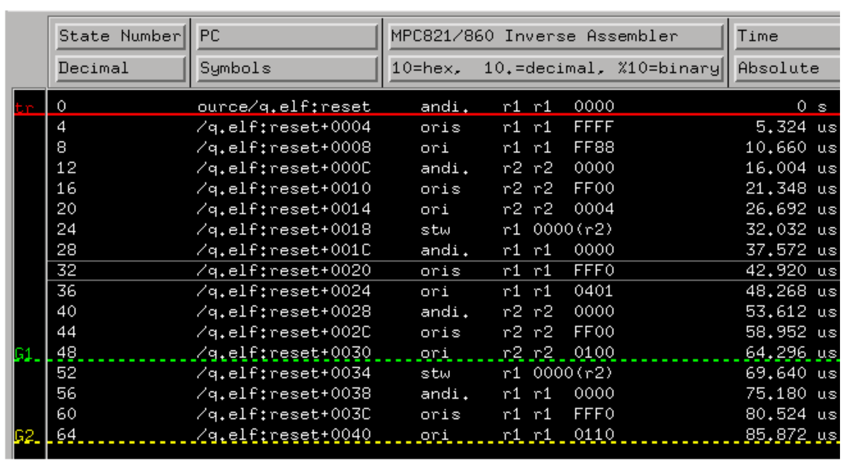
4. Start microprocessor execution from its reset address (either by turning power ON to the target system or by using the emulation control tool set's Reset and Run commands).



Displaying the Data

1. Use the Listing display to show the data that was captured when the target system started up.





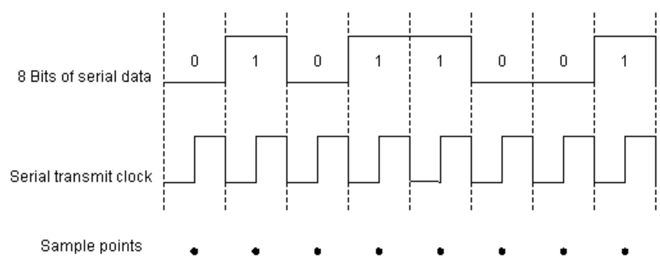
See Also

"To start or stop processor execution" on page 147

Making Driver Development Measurements

- "To trigger on an 8-bit serial pattern" on page 155
- "To view serial data in parallel" on page 160
- "To capture driver execution (& view HW and SW)" on page 165
- "To capture execution up to a failure or halt" on page 171
- "To view bus activity" on page 174
- "To capture simple program messages" on page 175
- "To trigger on packet data (with DataComm Analysis)" on page 177

To trigger on an 8-bit serial pattern



Possible uses:

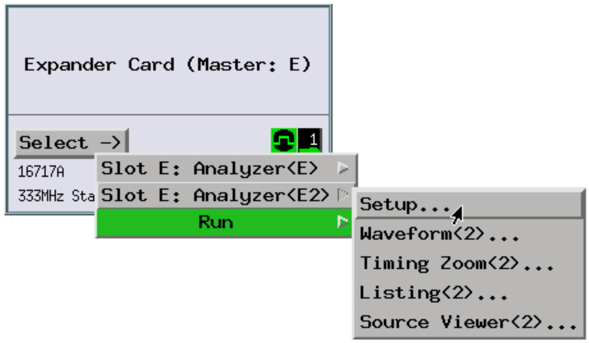
- To view system activity after pattern transmission.
- To look at system status when an error pattern is detected.

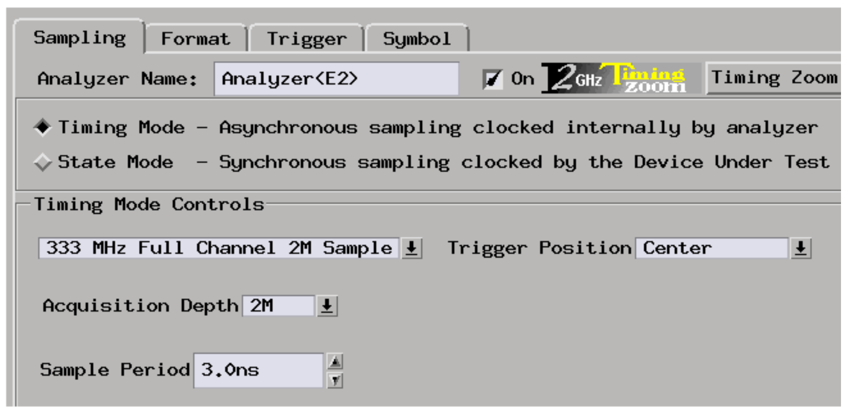
Probing the Target System

1. Connect a logic analyzer probe to the target system's serial signal, and connect the analyzer's clock input probe to the serial transmit/receive clock signal.
2. Configure a state analysis machine and specify the serial transmit/receive clock as the analyzer's clock input.

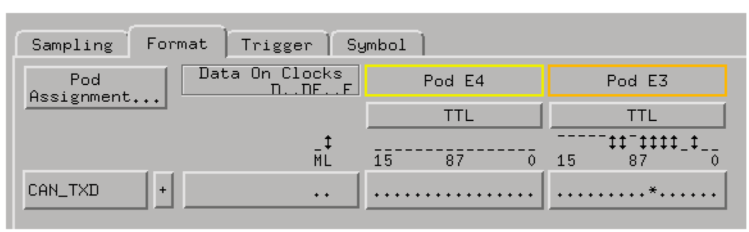
   Or:

   Configure a timing analysis machine and make the sample rate the same as the serial transmit/receive clock.





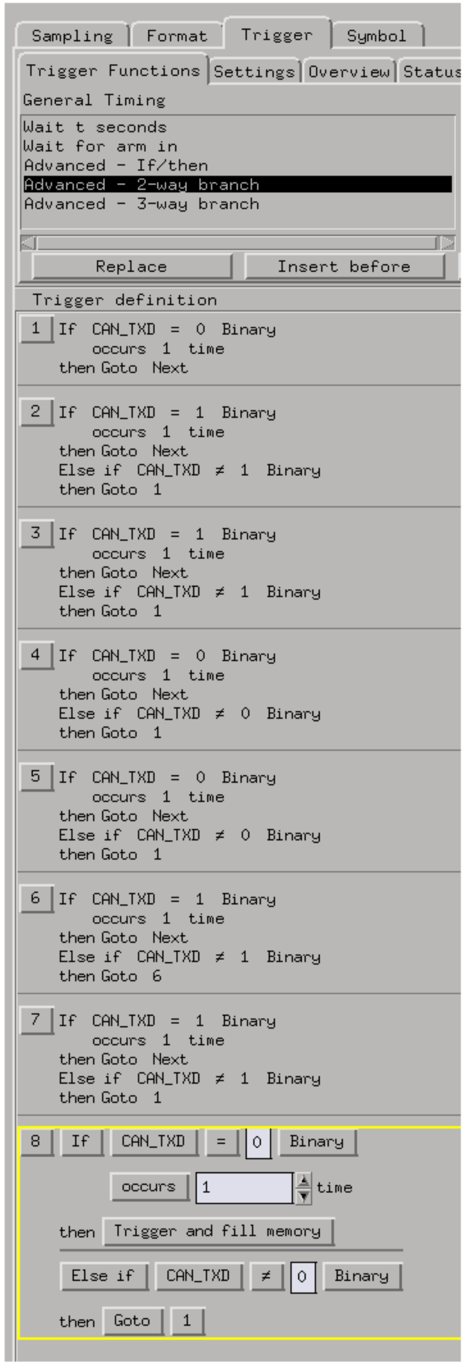
3. Format a label for the signal on which you will look for the serial pattern.



Capturing the Data

1. Set up a trigger sequence where level 1 looks for the first logic level of the pattern; when the pattern is found, level 2 looks for the second logic level of the pattern, etc. If, in each of levels 2 through 8, the appropriate logic

level is not found, branch back to level 1 to start looking for the serial pattern again.

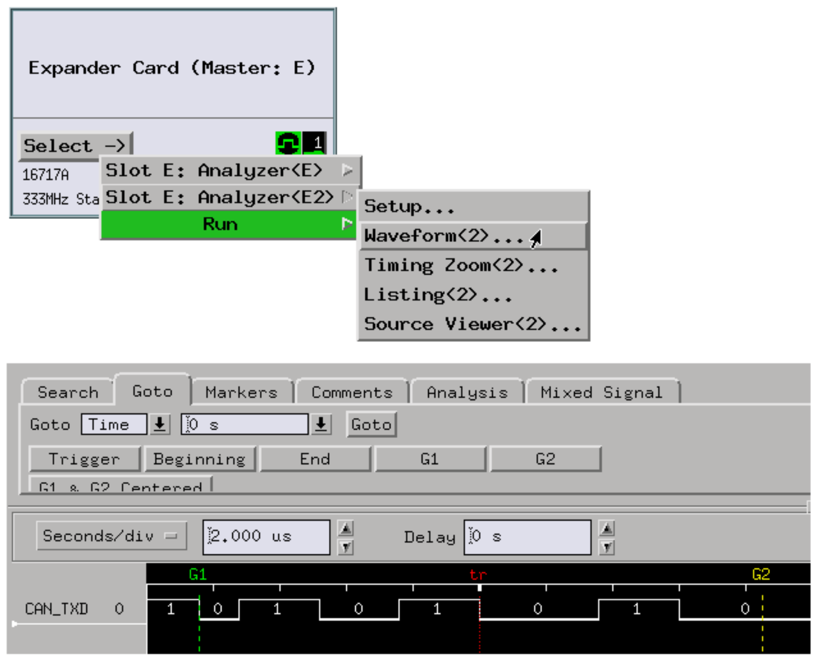

Sampling
Format
Trigger
Symbol
Trigger Functions
Settings
Overview
Status
General Timing
Wait t seconds
Wait for arm in
Advanced - If/then
Advanced - 2-way branch
Advanced - 3-way branch
Replace
Insert before
Trigger definition
1 If CAN_TXD = 0 Binary occurs 1 time then Goto Next
2 If CAN_TXD = 1 Binary occurs 1 time then Goto Next Else if CAN_TXD ≠ 1 Binary then Goto 1
3 If CAN_TXD = 1 Binary occurs 1 time then Goto Next Else if CAN_TXD ≠ 1 Binary then Goto 1
4 If CAN_TXD = 0 Binary occurs 1 time then Goto Next Else if CAN_TXD ≠ 0 Binary then Goto 1
5 If CAN_TXD = 0 Binary occurs 1 time then Goto Next Else if CAN_TXD ≠ 0 Binary then Goto 1
6 If CAN_TXD = 1 Binary occurs 1 time then Goto Next Else if CAN_TXD ≠ 1 Binary then Goto 6
7 If CAN_TXD = 1 Binary occurs 1 time then Goto Next Else if CAN_TXD ≠ 1 Binary then Goto 1
8 If CAN_TXD = 0 Binary occurs 1 time then Trigger and fill memory Else if CAN_TXD ≠ 0 Binary then Goto 1

To look look for longer (or shorter) patterns, use more (or fewer) sequence levels.

2. Select the Run button to start the measurement.

Displaying the Data

1. When the analyzer triggers, use the Waveform display to show the 8-bit serial pattern was captured.



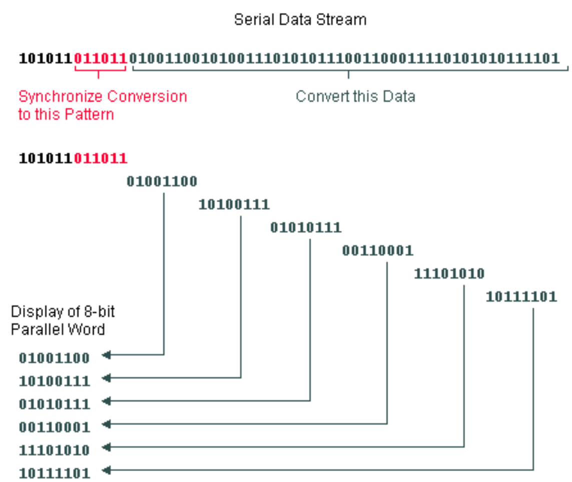
If the analyzer never triggers, the serial pattern does not occur. Depending on the level that was reached in the sequence above, you can see how much (or how little) of the pattern was found.

See Also

"To view serial data in parallel" on page 160

"If the trigger doesn't occur as expected" on page 309

To view serial data in parallel



Requirements:

- This measurement requires the serial to parallel tool set.

Possible uses:

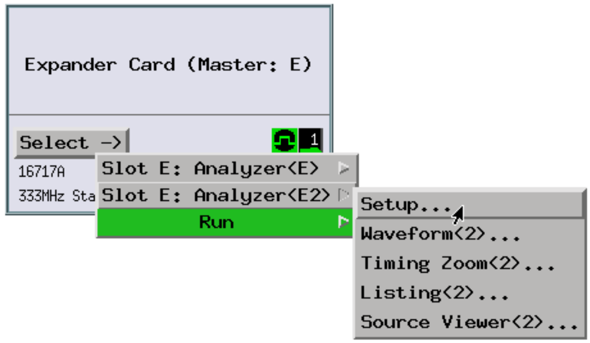
- To view serial data in a format consistent with a particular protocol format.

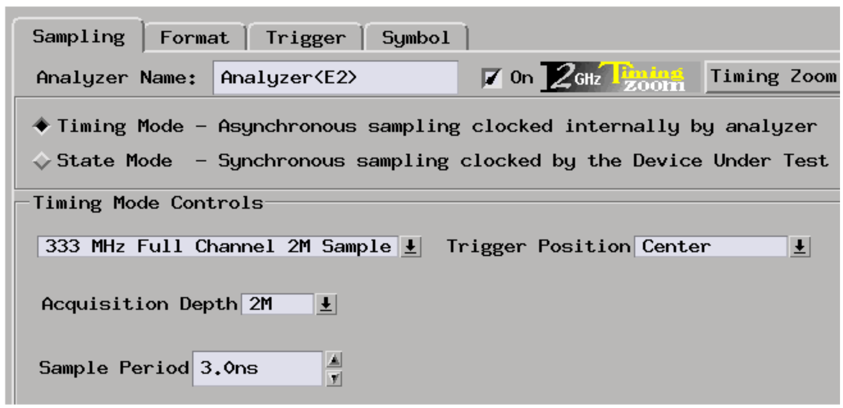
Probing the Target System

1. Connect a logic analyzer probe to the serial data signal, and connect the serial data transmit (or receive, etc.) clock signal to one of the analyzer's clock input channels.
2. Configure the logic analyzer as a state analyzer and use the serial transmit/ receive clock as the analyzer's clock input.

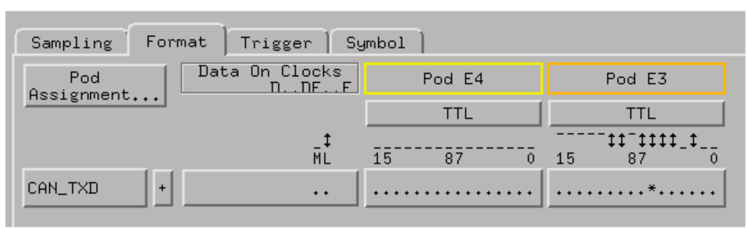
   Or:

   Configure the analyzer as a timing analyzer and use the Serial To Parallel tool's *Clock Recovery* option.



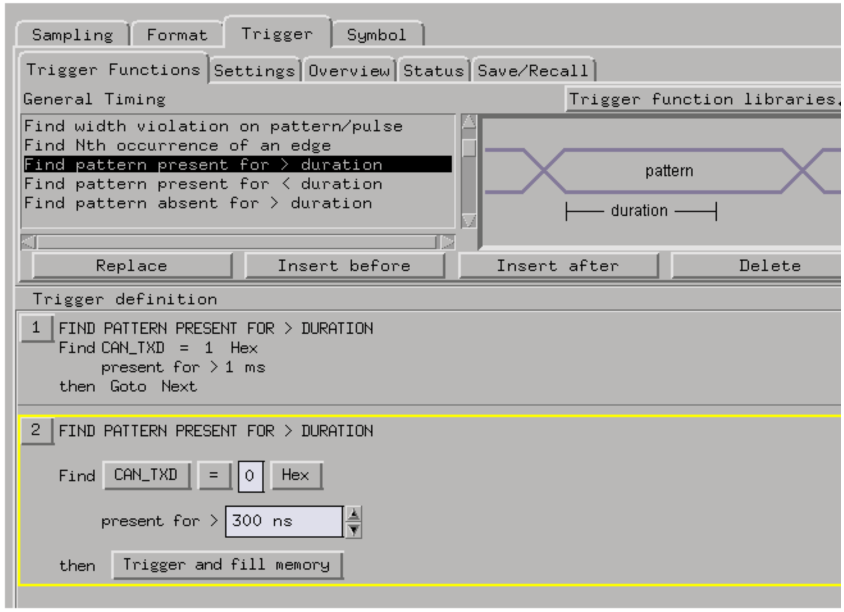


3. Format a label for the serial data channel.



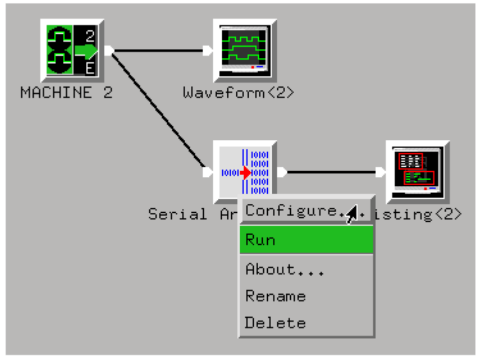
Capturing the Data

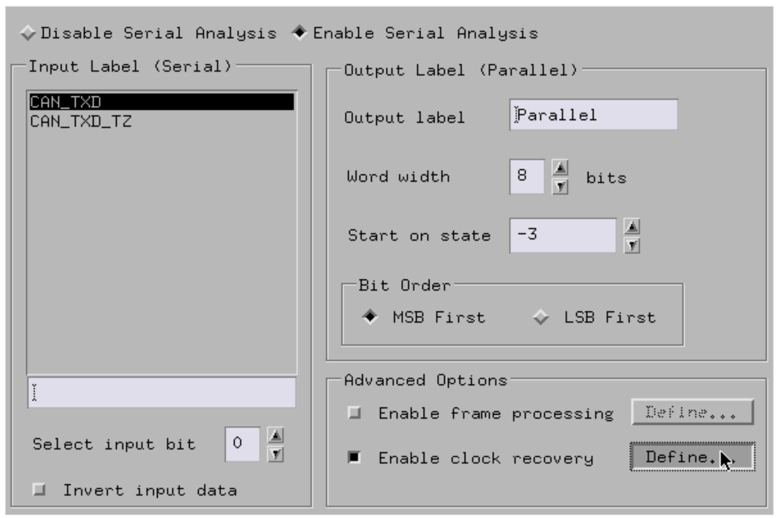
1. Set up a trigger specification to capture data on the serial channel. You may want to trigger on a particular pattern and capture data that occurs after that pattern.

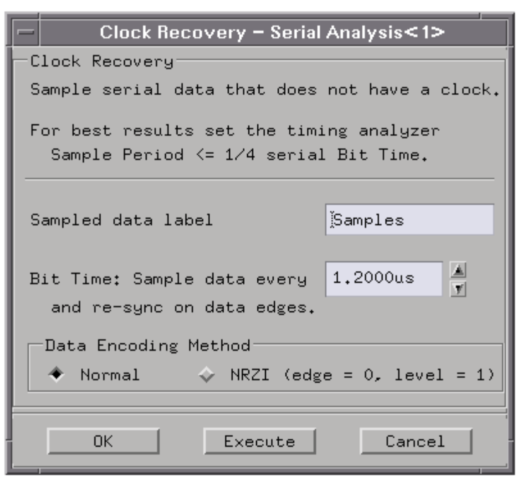


2. Configure the measurement workspace so that the logic analyzer feeds the Serial To Parallel tool, which in turn feeds the Listing display.

3. Open the Serial To Parallel tool and set up the type of conversion.


MACHINE 2
Waveform<2>
Serial Ar
isting<2>
Configure...
Run
About...
Rename
Delete

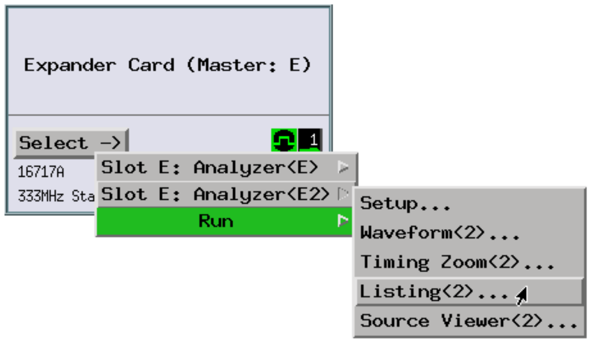

Disable Serial Analysis
Enable Serial Analysis
Input Label (Serial)
CAN_TXD
CAN_TXD_TZ
Select input bit
0
Invert input data
Output Label (Parallel)
Output label
Parallel
Word width
8
bits
Start on state
-3
Bit Order
MSB First
LSB First
Advanced Options
Enable frame processing
Define...
Enable clock recovery
Define...

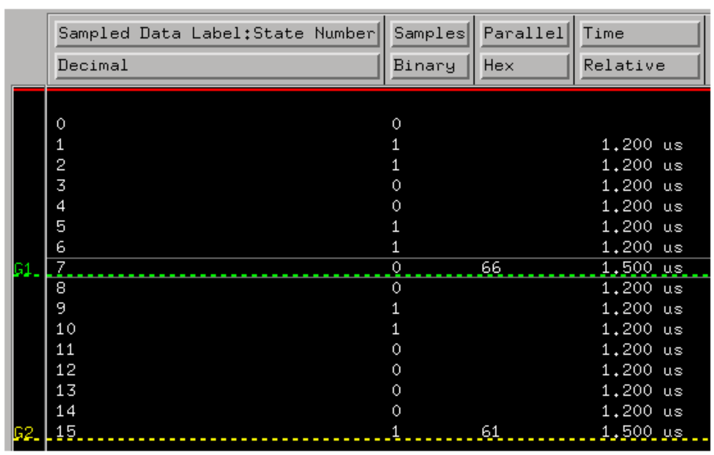

Clock Recovery – Serial Analysis<1>
Clock Recovery
Sample serial data that does not have a clock.
For best results set the timing analyzer
Sample Period <= 1/4 serial Bit Time.
Sampled data label
Samples
Bit Time: Sample data every
1.2000us
and re-sync on data edges.
Data Encoding Method
Normal
NRZI (edge = 0, level = 1)
OK
Execute
Cancel

4. Select the Run button to start the measurement.

Displaying the Data

1. Open the Listing display window to see the results of the serial to parallel conversion.

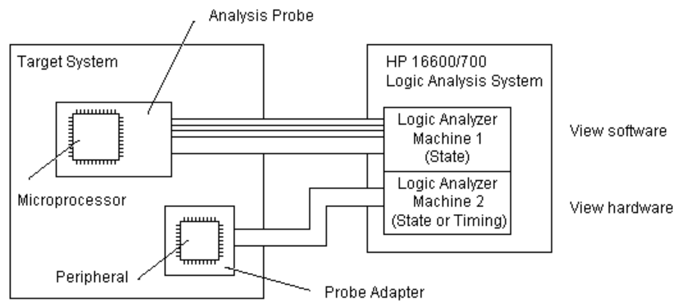




See Also

"To trigger on an 8-bit serial pattern" on page 155

To capture driver execution (& view HW and SW)

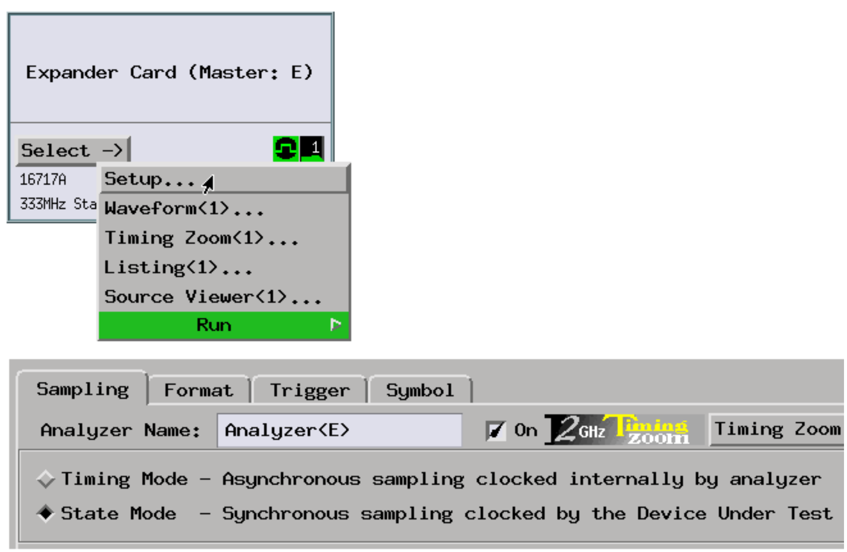


Possible uses:

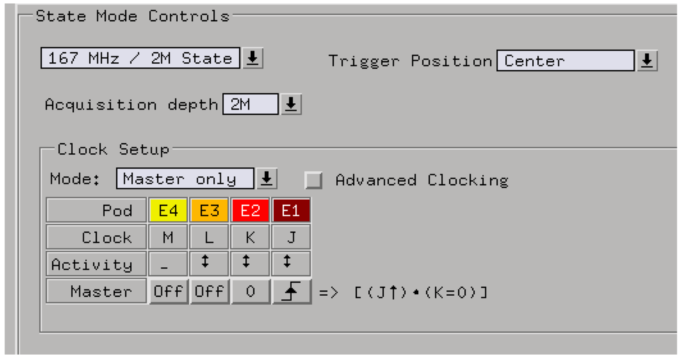
- To view and correlate driver software execution with hardware signals.

Probing the Target System

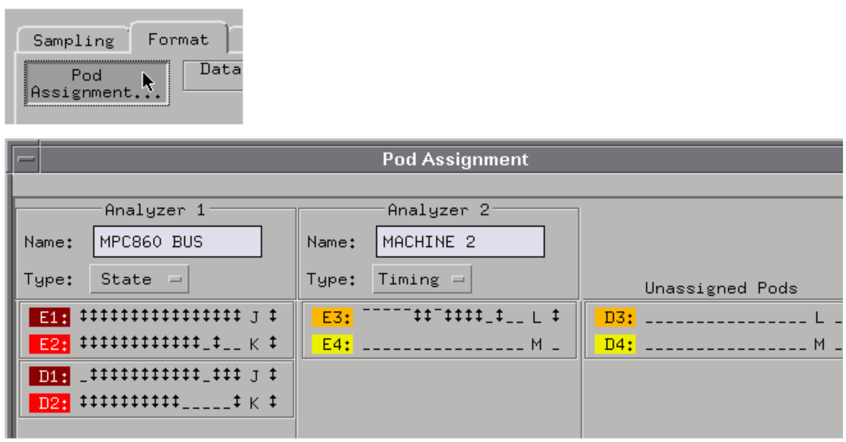
1. Probe the microprocessor (using an analysis probe if possible).
2. Probe the peripheral with other logic analysis channels (and possibly a probe adapter).
3. Configure a state analysis machine.



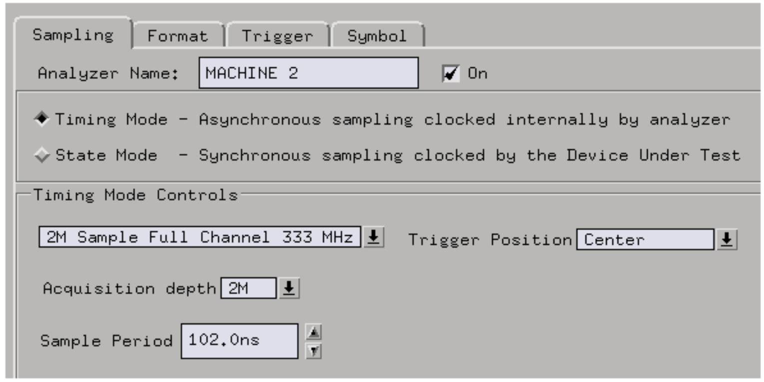
4. Select the state analyzer's clock input.



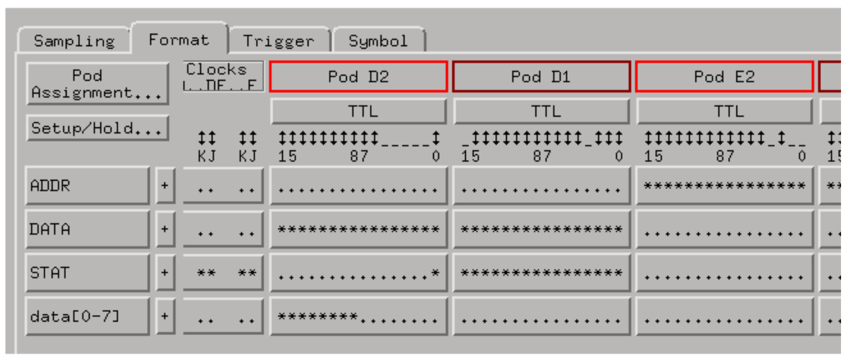
5. Assign pods. Use one logic analyzer machine for analyzing the microprocessor. Create another logic analysis machine for analyzing the peripheral by specifying the Analyzer 2 type.



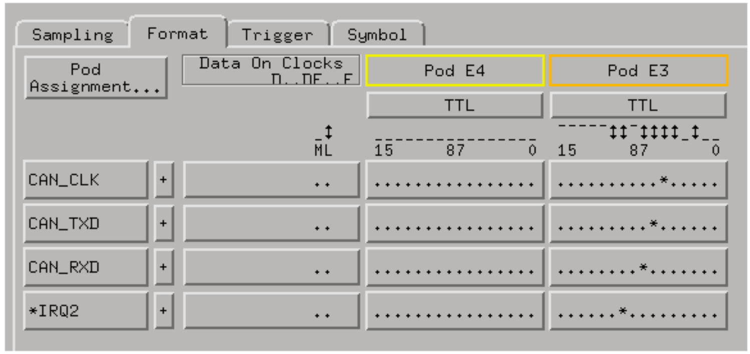
6. Specify the sampling options for the second logic analyzer machine.



7. Format labels for the signals that are probing the microprocessor.

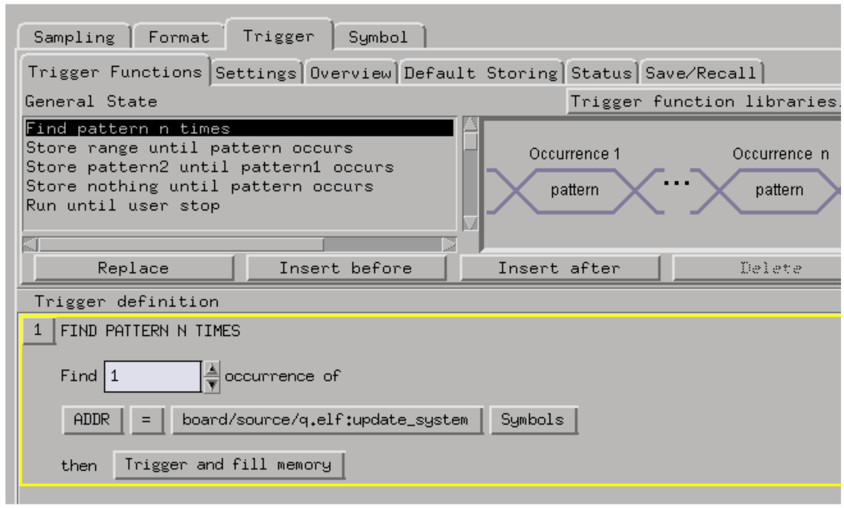


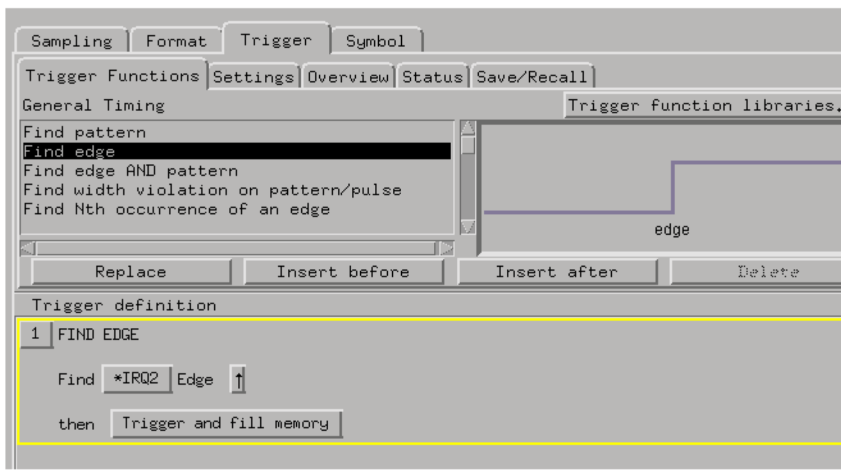
8. Format labels for the signals that are probing the peripheral.



Capturing the Data

1. In each machine, set up the trigger and resources for the data you want to capture.

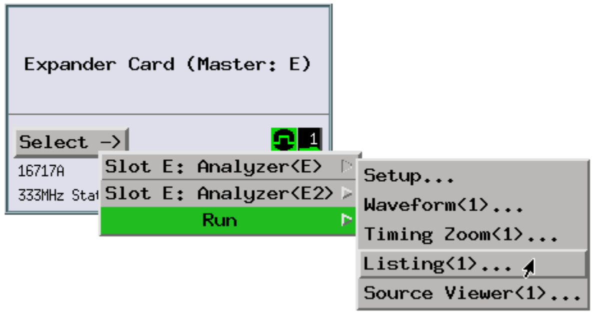


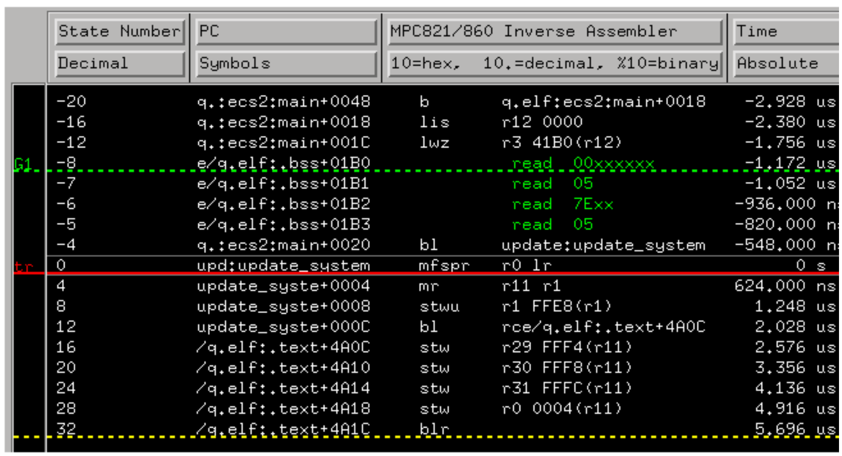


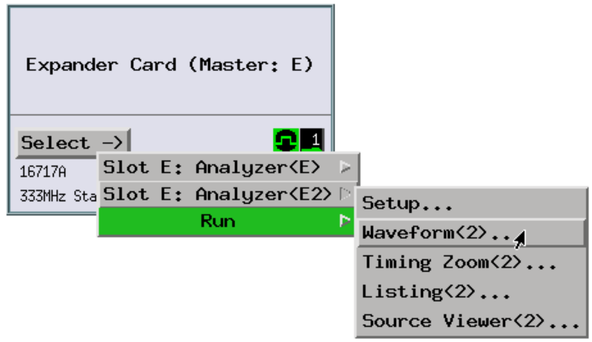
2. Select the Run button to start measurements in each logic analyzer.

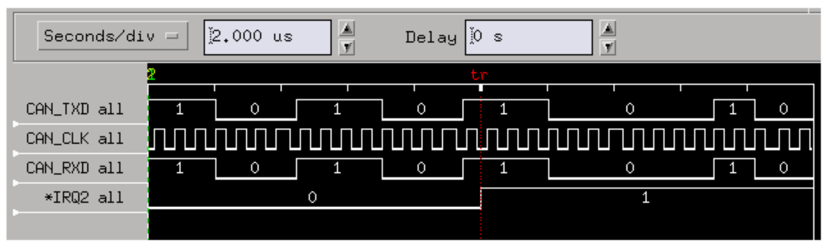
Displaying the Data

1. Use the Listing display to show the captured software execution and use the Waveform display to show the captured hardware signals.


Expander Card (Master: E)
Select ->
1
16717A
333MHz Stat
Slot E: Analyzer<E>
Slot E: Analyzer<E2>
Run
Setup...
Waveform<1>...
Timing Zoom<1>...
Listing<1>...
Source Viewer<1>...


State Number
PC
MPC821/860 Inverse Assembler
Time
Decimal
Symbols
10=hex, 10.=decimal, %10=binary
Absolute
-20 q.:ecs2:main+0048 b q.elf:ecs2:main+0018 -2.928 us
-16 q.:ecs2:main+0018 lis r12 0000 -2.380 us
-12 q.:ecs2:main+001C lwz r3 41B0(r12) -1.756 us
G1 -8 e/q.elf:.bss+01B0 read 00xxxxxx -1.172 us
-7 e/q.elf:.bss+01B1 read 05 -1.052 us
-6 e/q.elf:.bss+01B2 read 7Exx -936.000 n
-5 e/q.elf:.bss+01B3 read 05 -820.000 n
-4 q.:ecs2:main+0020 bl update:update_system -548.000 n
tr 0 upd:update_system mfspr r0 lr 0 s
4 update_syste+0004 mr r11 r1 624.000 ns
8 update_syste+0008 stwu r1 FFE8(r1) 1.248 us
12 update_syste+000C bl rce/q.elf:.text+4A0C 2.028 us
16 /q.elf:.text+4A0C stw r29 FFF4(r11) 2.576 us
20 /q.elf:.text+4A10 stw r30 FFF8(r11) 3.356 us
24 /q.elf:.text+4A14 stw r31 FFFC(r11) 4.136 us
28 /q.elf:.text+4A18 stw r0 0004(r11) 4.916 us
32 /q.elf:.text+4A1C blr 5.696 us



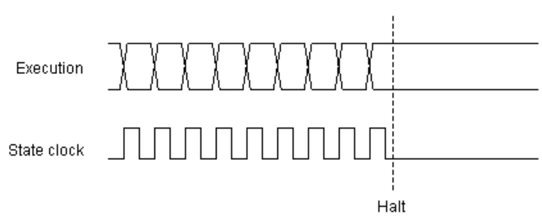


Each analyzer triggers according to its own setup. You can change this by setting up one analyzer to be armed by another analyzer.

See Also

"System Integration" on page 257 for information on coordinating the collection of data with a group run or arming a measurement in one machine by a trigger in the other.

To capture execution up to a failure or halt



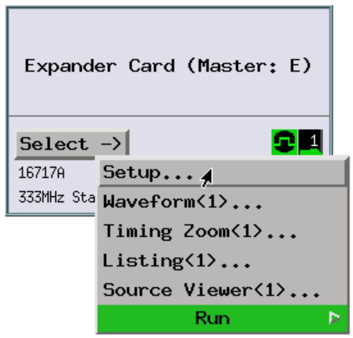
Possible uses:

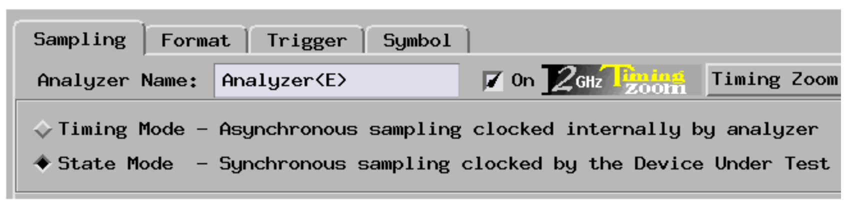
- To store and display all activity leading up to a system crash.

- To run the logic analyzer indefinitely until the Stop button is selected so that you can observe system activity at your discretion.

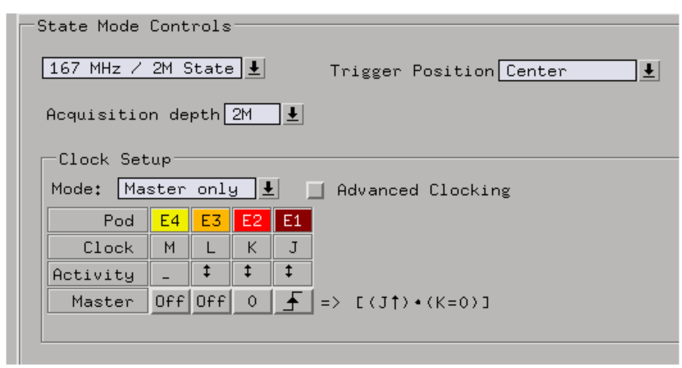
Probing the Target System

1. Configure a state analysis machine.

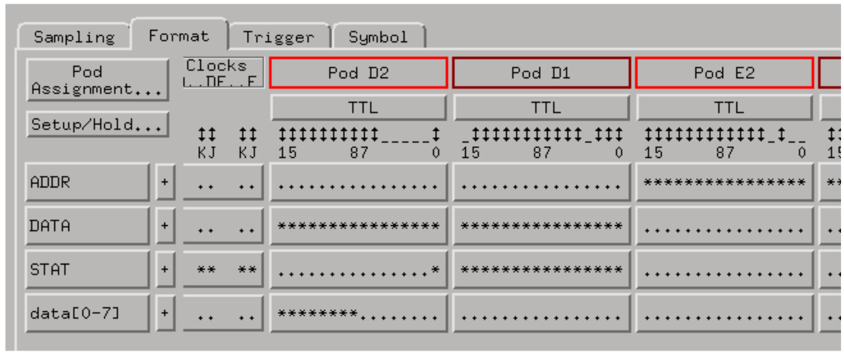




2. Select the state analyzer's clock input.

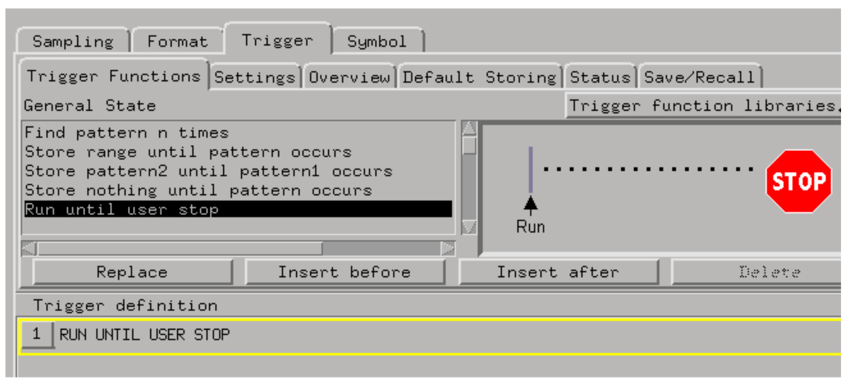


3. Format labels for the signals on which you will look for the event.



Capturing the Data

1. Use the "Run until user stop" trigger function.



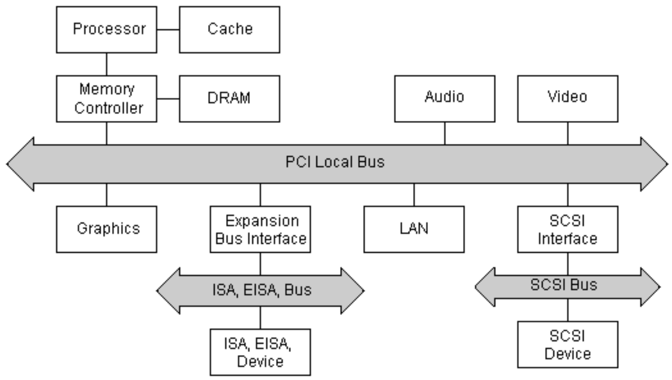
2. Select the Run button to start the measurement.

3. When the system fails, crashes, or halts, select the Stop  button to see the states that were captured before the failure.

Displaying the Data

1. Use the Listing display to show the states that led up to the failure.

To view bus activity



Target system buses are good locations to view system activity and may be the first place you look when isolating problems (especially in multi-processor systems).

Requirements:

- This measurement requires an analysis probe for the standard bus you wish to view.

Possible uses:

- To isolate system problems.
- To view and correlate activity on multiple buses.
- To view and correlate bus activity to microprocessor execution.

Probing the Target System

1. Use a standard bus analysis probe to make the physical connection between the logic analyzer and the bus.
2. Use the the configuration files included with the analysis probe to configure the analyzer and format labels.

Capturing the Data

1. Set up a trigger specification using the labels defined by the analysis probe configuration files.

Displaying the Data

1. Use the Listing display to view captured bus activity. If the standard bus analysis probe provides an inverse assembler, you will see mnemonics for

bus commands, status, etc.

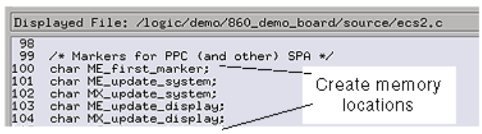
See Also

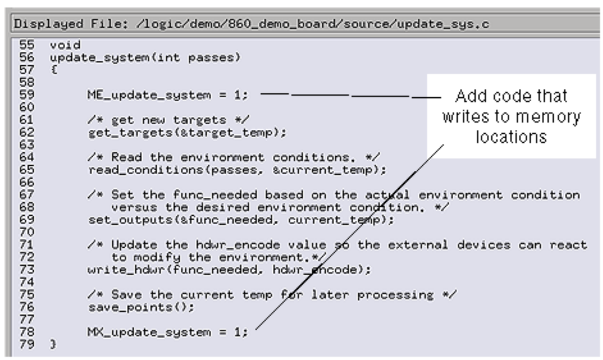
"To analyze bus stability (with SPA)" on page 88

"To analyze bus occupation & bandwidth (with SPA)" on page 131

"To simulate bus occupation and measure SW performance" on page 287

To capture simple program messages





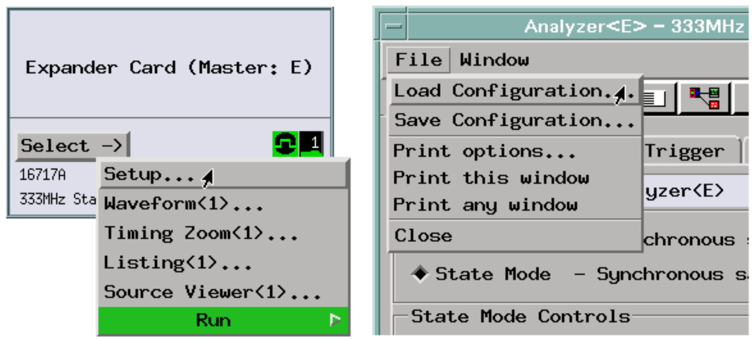
By adding program code that causes activity external to the processor (also known as "instrumenting your code"), you can create specific program message events that can be captured by the analyzer.

Possible uses:

- To view processor execution when the instruction cache is turned ON.
- To view higher-level program activity (like Real-Time OS function calls or OS calls).

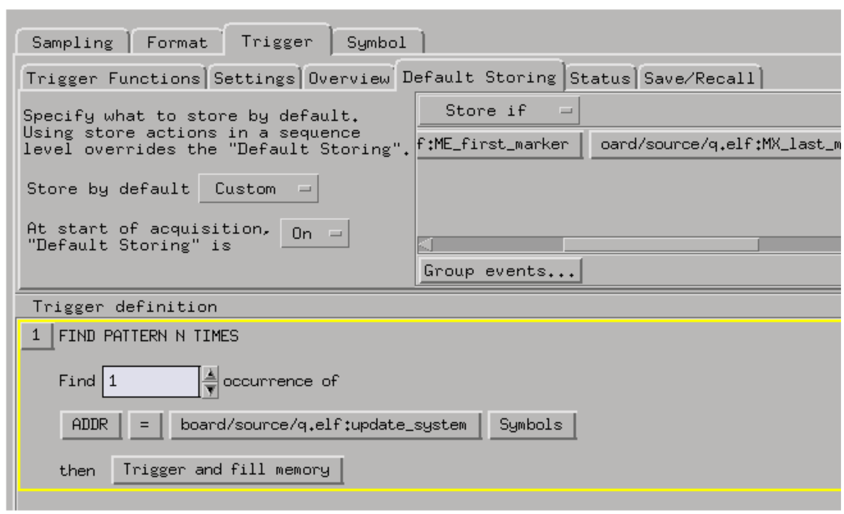
Probing the Target System

1. Typically, you will use an analysis probe to connect the logic analyzer to the microprocessor, and you will use the provided configuration files to configure the analyzer and define labels.



Capturing the Data

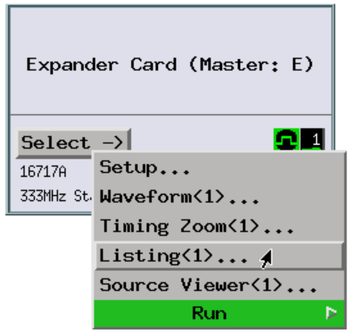
1. Set up a trigger specification and use storage qualifiers that capture the program messages you have coded into your programs.

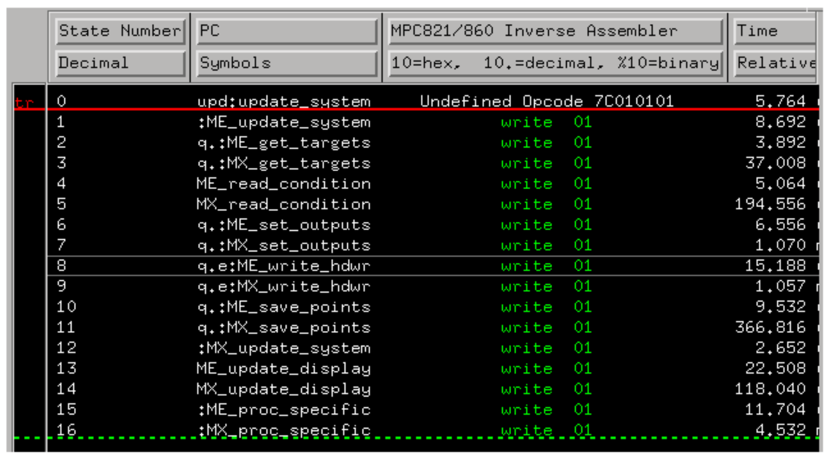


2. Select the Run button to start the measurement.

Displaying the Data

1. Use the Listing display to view the captured program messages.





To trigger on packet data (with DataComm Analysis)

Requirements:

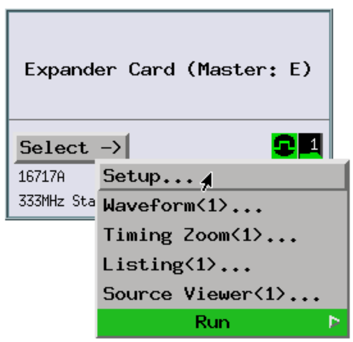
- The DataComm Analysis tool set.
- An Agilent Technologies 16715/16/17/18/19A logic analyzer module (and its VisiTrigger capabilities).

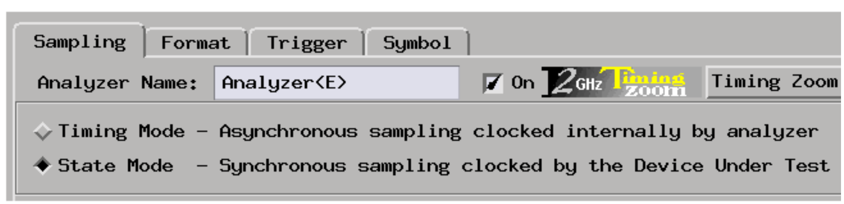
Possible uses:

- To look at data traveling across parallel buses inside network switching systems.

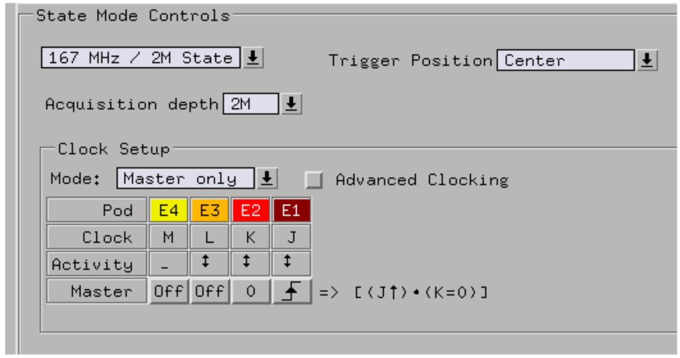
Probing the Target System

1. Connect logic analyzer probe channels to:
   - A communication data bus.
   - Control signals that identify the start of packet, the packet data, and the end of packet.
2. You also need to connect a logic analyzer CLK input channel to:
   - A clock signal that identifies when the data bus and control signals are valid and should be sampled by the logic analyzer.
3. Configure a (synchronous sampling) state analysis machine.

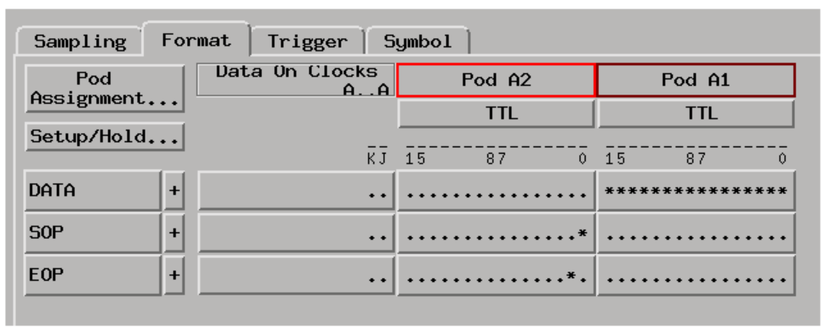




4. Select the state analyzer's clock input.

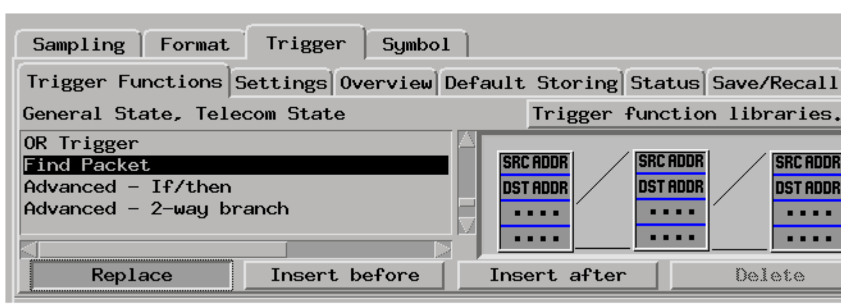


5. Format a DATA label for the logic analyzer channels that are probing the data bus. Format 1-bit labels for the channels that are probing the start of packet, packet data, and end of packet signals.

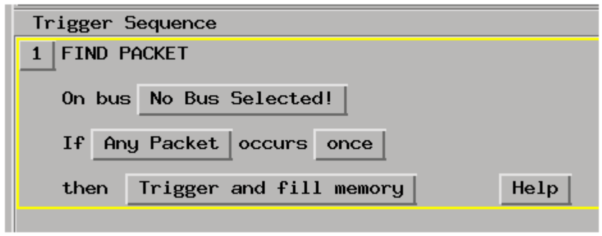


Capturing the Data

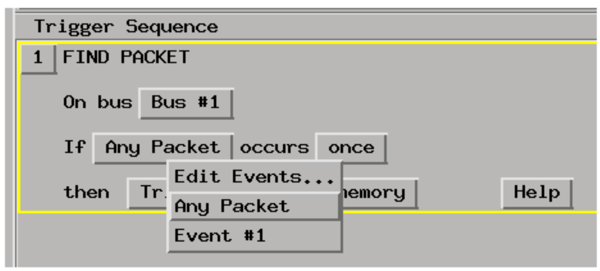
1. Select the "Find Packet" trigger function.



2. In the "Find Packet" trigger sequence level, select the bus button.
3. In the Bus Selector dialog, select the bus definition you want to use and select the OK button.



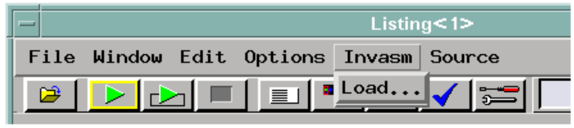
4. Specify the packet event to find.

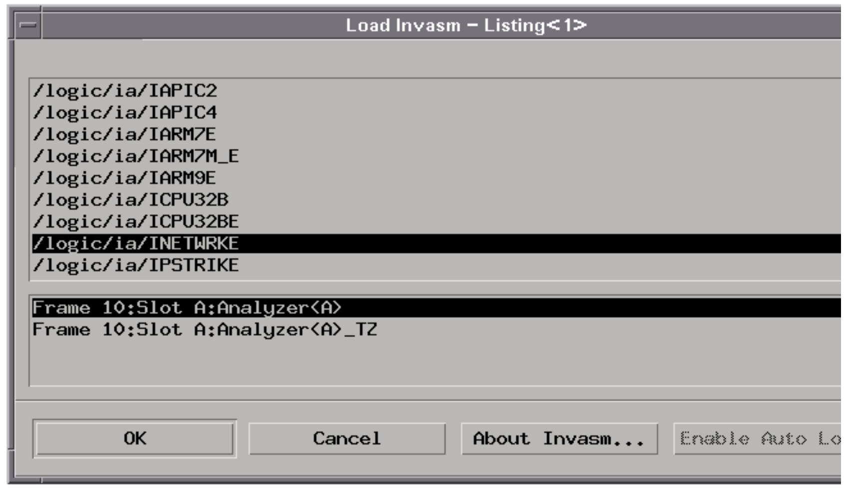


5. Select the Run button to start the measurement.

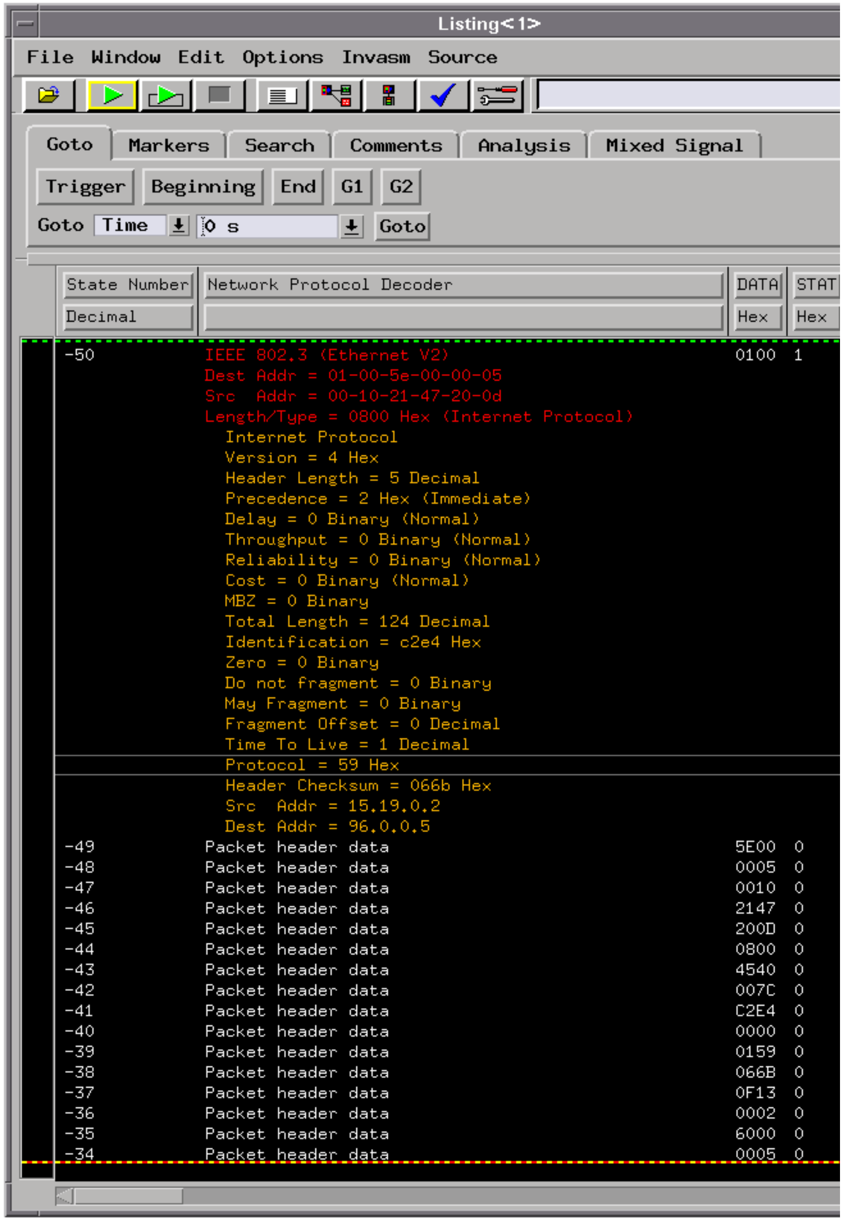
Displaying the Data

1. In the logic analyzer’s Listing window, load the INETWRKE network protocol decoder inverse assembler.





2. View the captured data.



See Also

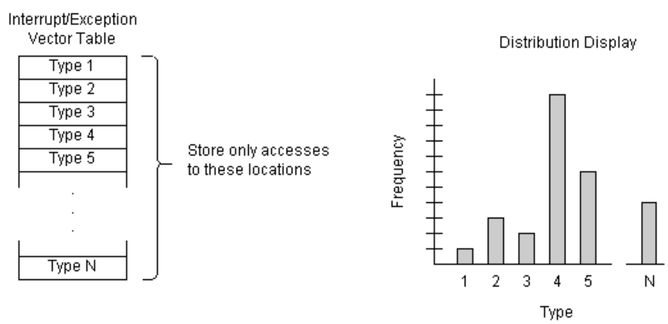
Using the DataComm Analysis Toolset (see the *DataComm Analysis*

Toolset help volume)

Making Interrupt Service Routine Measurements

- "To capture interrupt frequency and type" on page 183
- "To measure interrupt latency and execution time" on page 186
- "To simulate particular interrupt sequences" on page 191
- "To view the occurrence rate of an event (with SPA)" on page 192

To capture interrupt frequency and type

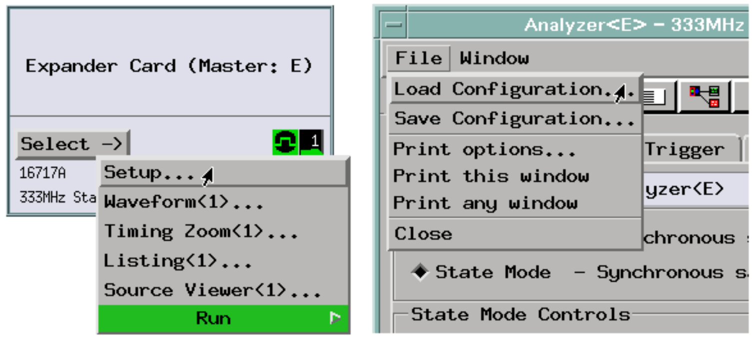


Possible uses:

- To analyze interrupt processing.

Probing the Target System

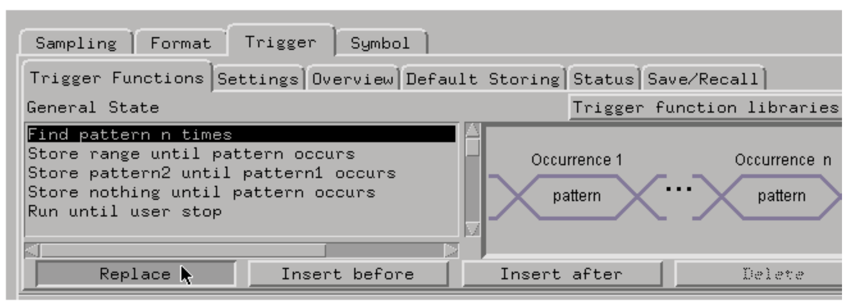
1. Typically, you will use an analysis probe to connect the logic analyzer to the microprocessor, and you will use the provided configuration files to configure the analyzer and define labels.

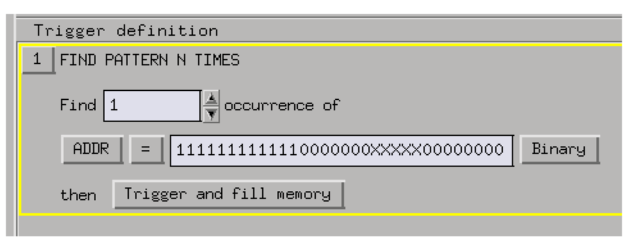


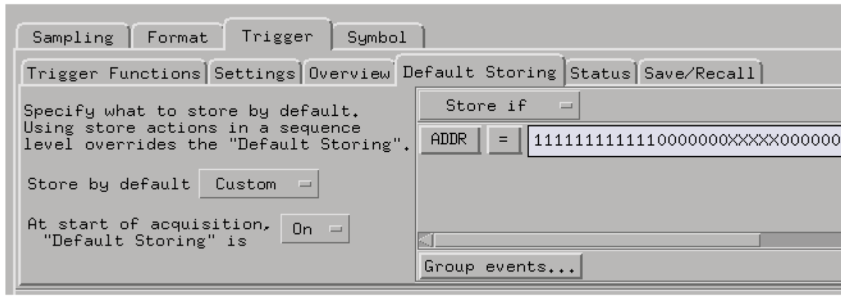
Capturing the Data

1. The trigger specification will depend on the interrupt mechanism of your microprocessor.

   If your processor uses an interrupt vector table, set up a trigger specification that only stores accesses to the interrupt vector table locations.



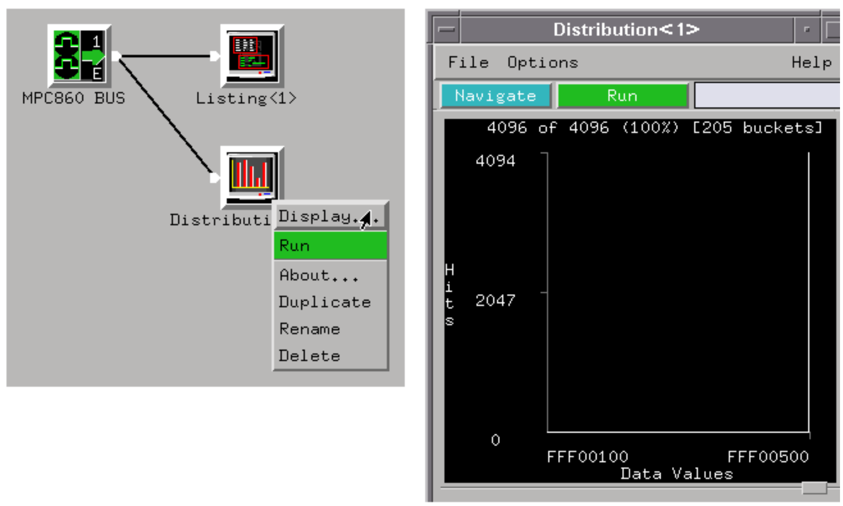




2. Select the Run button to start the measurement.

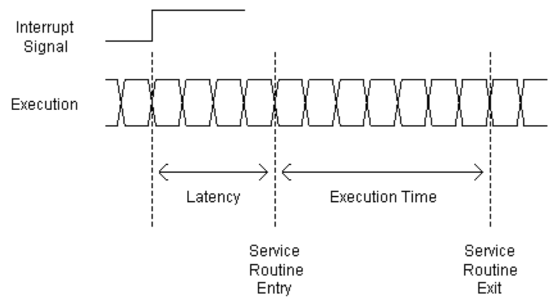
Displaying the Data

1. In the Workspace window, add the Distribution display to view the captured data.



If your trigger specification stored interrupt vector table accesses, the captured table locations will show the interrupt types.

To measure interrupt latency and execution time

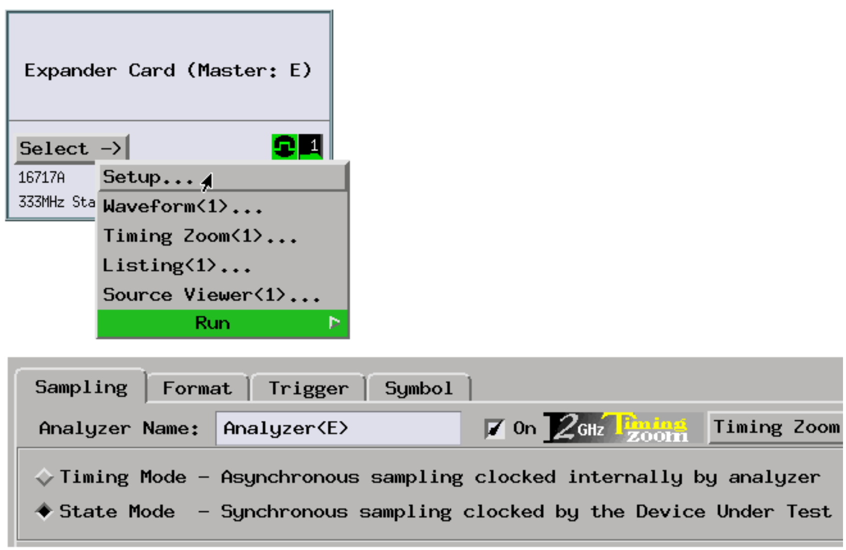


Possible uses:

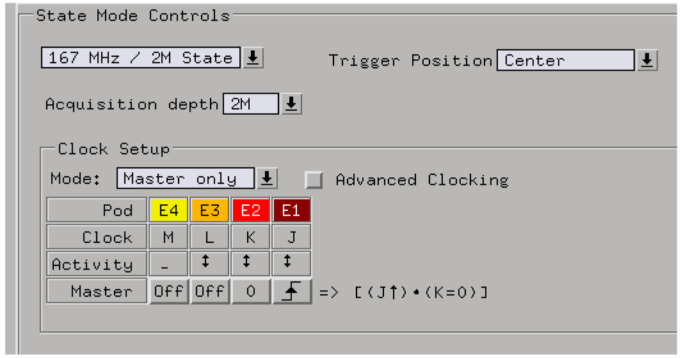
- To see if interrupt processing meets specifications.

Probing the Target System

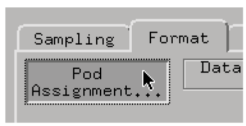
1. Configure a state analysis machine.

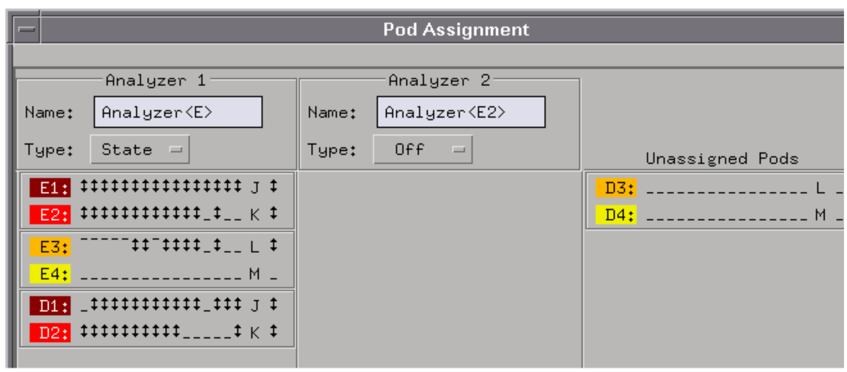


2. Select the state analyzer's clock input.

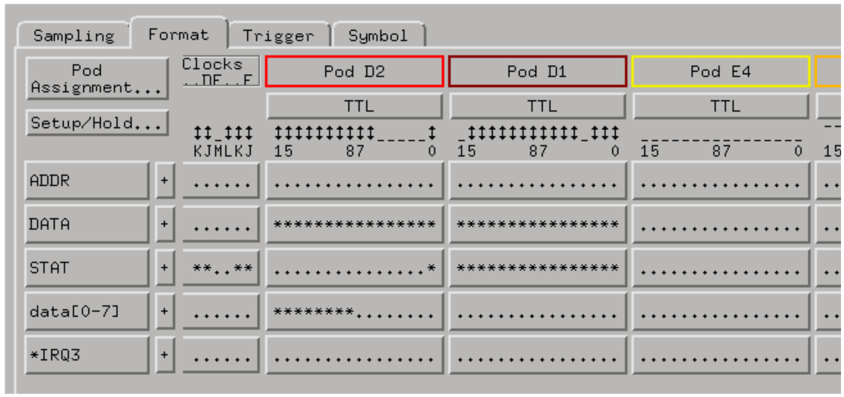


3. Assign pods if necessary.



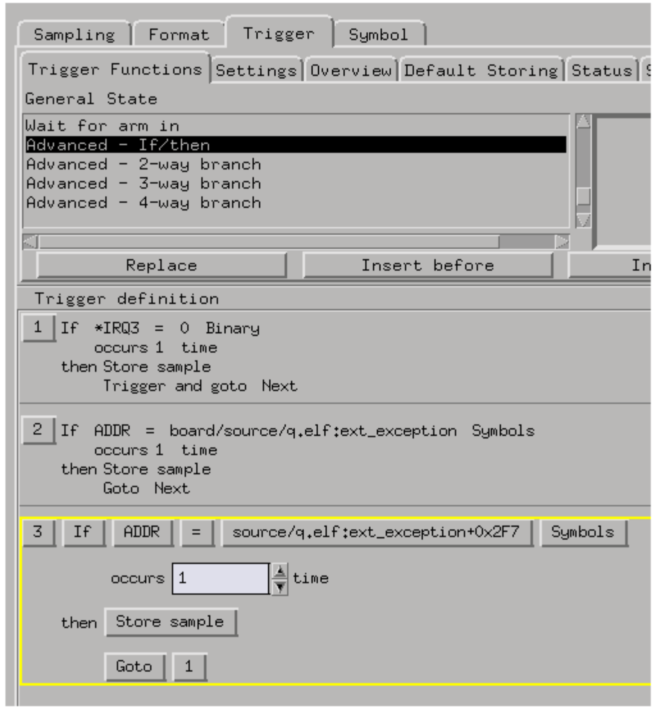


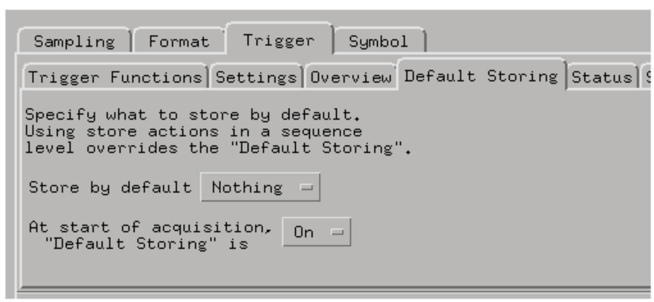
4. Format labels for the signals on which you will look for the event.



Capturing the Data

1. Set up a trigger specification that stores the state when the interrupt signal becomes active, the interrupt service routine entry point, and the interrupt routine exit point.



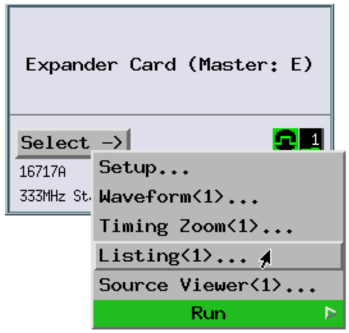


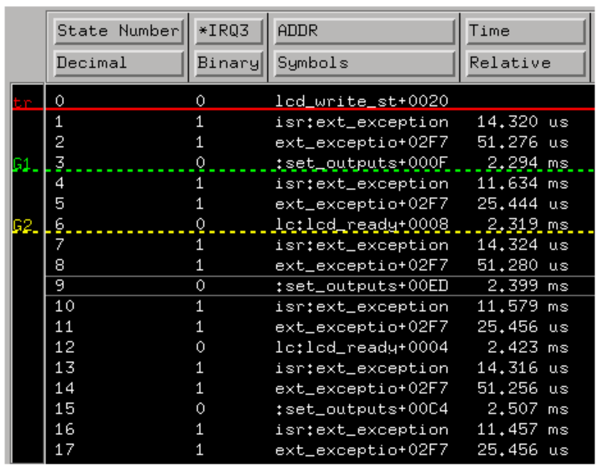
2. Make sure the analyzer is counting time.

3. Select the Run button to start the measurement.

Displaying the Data

1. Use the Listing display to view the captured states.





	State Number	*IRQ3	ADDR	Time
	Decimal	Binary	Symbols	Relative
tr	0	0	lcd_write_st+0020	
	1	1	isr:ext_exception	14.320 us
	2	1	ext_exceptio+02F7	51.276 us
G1	3	0	:set_outputs+000F	2.294 ms
	4	1	isr:ext_exception	11.634 ms
	5	1	ext_exceptio+02F7	25.444 us
G2	6	0	lc:lcd_ready+0008	2.319 ms
	7	1	isr:ext_exception	14.324 us
	8	1	ext_exceptio+02F7	51.280 us
	9	0	:set_outputs+00ED	2.399 ms
	10	1	isr:ext_exception	11.579 ms
	11	1	ext_exceptio+02F7	25.456 us
	12	0	lc:lcd_ready+0004	2.423 ms
	13	1	isr:ext_exception	14.316 us
	14	1	ext_exceptio+02F7	51.256 us
	15	0	:set_outputs+00C4	2.507 ms
	16	1	isr:ext_exception	11.457 ms
	17	1	ext_exceptio+02F7	25.456 us

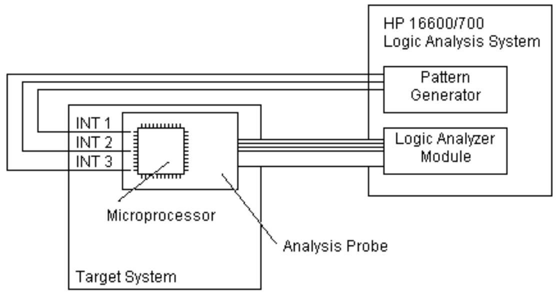
The relative time between the state where the interrupt signal became active and the service routine entry shows the interrupt latency.

The relative time between the service routine entry and exit shows the interrupt execution time.

See Also

"To measure function execution time (with SPA)" on page 218

To simulate particular interrupt sequences



Requirements:

- This measurement requires a pattern generator module (Agilent Technologies 16522A).

Possible uses:

- To test the processing of multiple interrupts.

Probing the Target System

1. Connect pattern generator outputs to microprocessor interrupt inputs.
2. Configure the pattern generator to output the desired sequence of interrupt signals.
3. Typically, you will use an analysis probe to connect the logic analyzer to the microprocessor, and you will use the provided configuration files to configure the analyzer and define labels.

Capturing the Data

1. Set up the trigger specification to capture the interrupt processing.

Displaying the Data

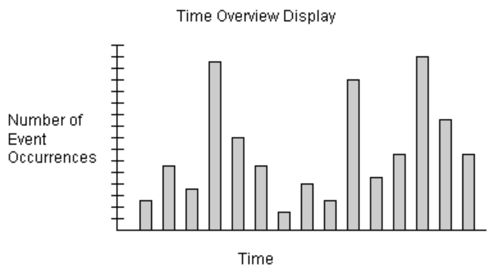
1. Use the Listing display to view the captured interrupt processing.

See Also

"To generate pattern stimulus on devices" on page 75

"To generate patterns when a source line executes" on page 262

To view the occurrence rate of an event (with SPA)



Requirements:

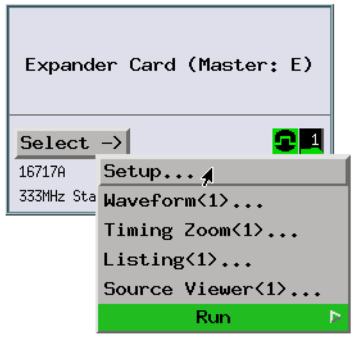
- This measurement requires the system performance analyzer (SPA) tool set.

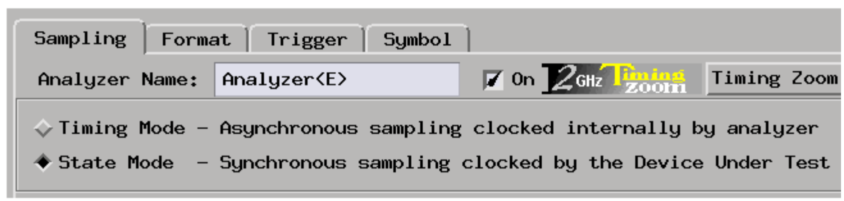
Possible uses:

- To display interrupt loading.
- To measure the frequency at which data is acquired from sensors.

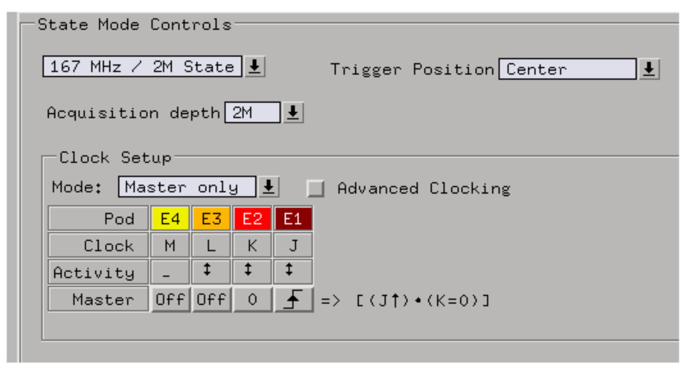
Probing the Target System

1. Connect the logic analyzer to the target system signals on which you will look for the event. You can use an analysis probe to connect the logic analyzer to a microprocessor or standard bus.
2. Configure a state analysis machine. (If you are using an analysis probe, use the provided configuration files to configure the analyzer and define labels.)

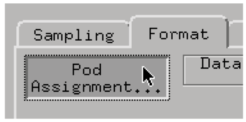


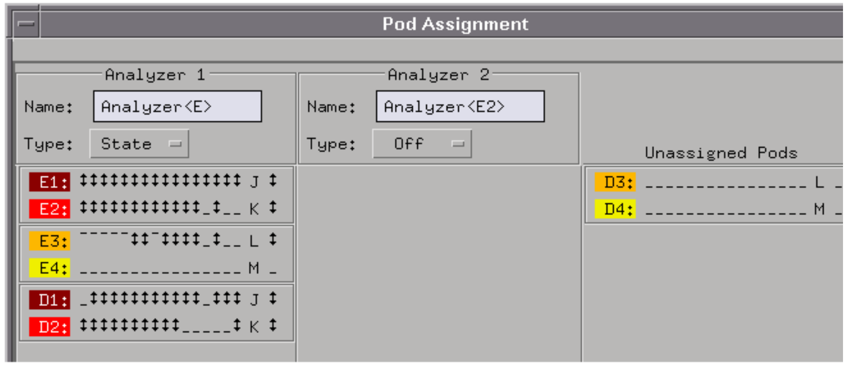


3. Select the state analyzer’s clock input.

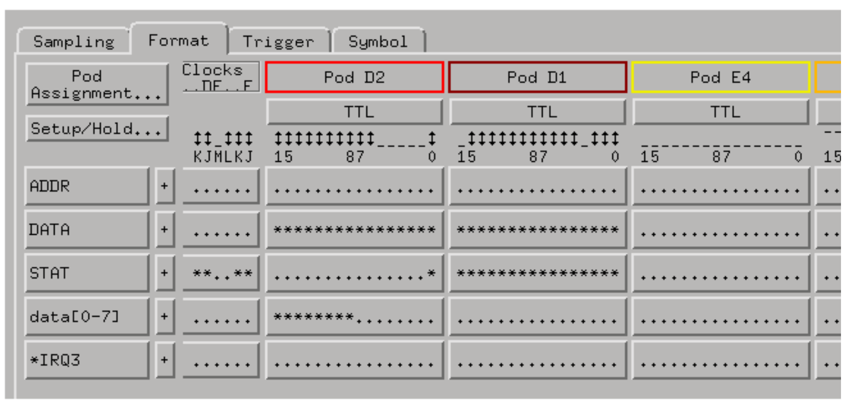


4. Assign pods if necessary.



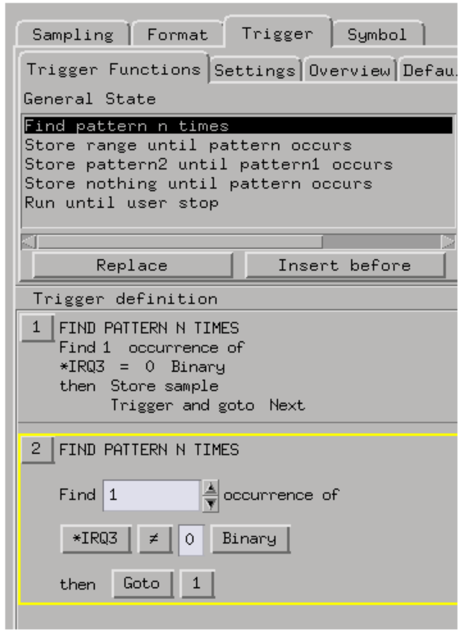


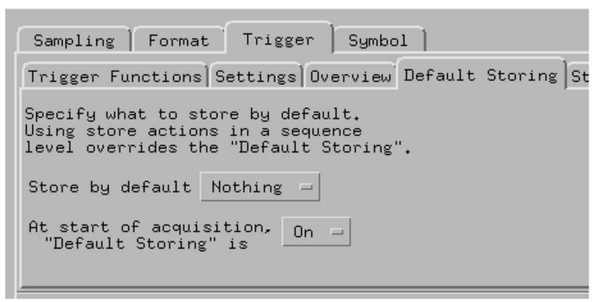
5. Format labels for the signals on which you will look for the event.



Capturing the Data

1. Set up the logic analyzer trigger specification to capture the events you're interested in.

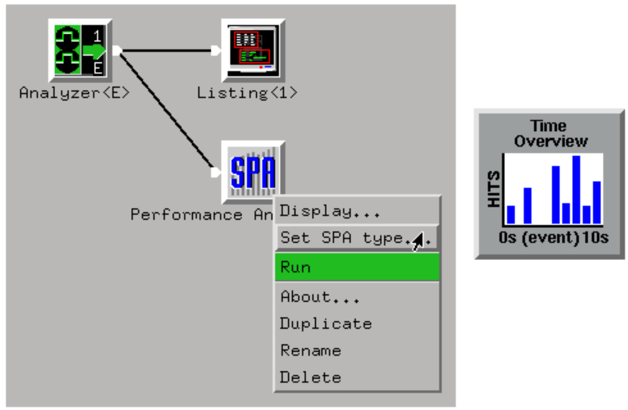




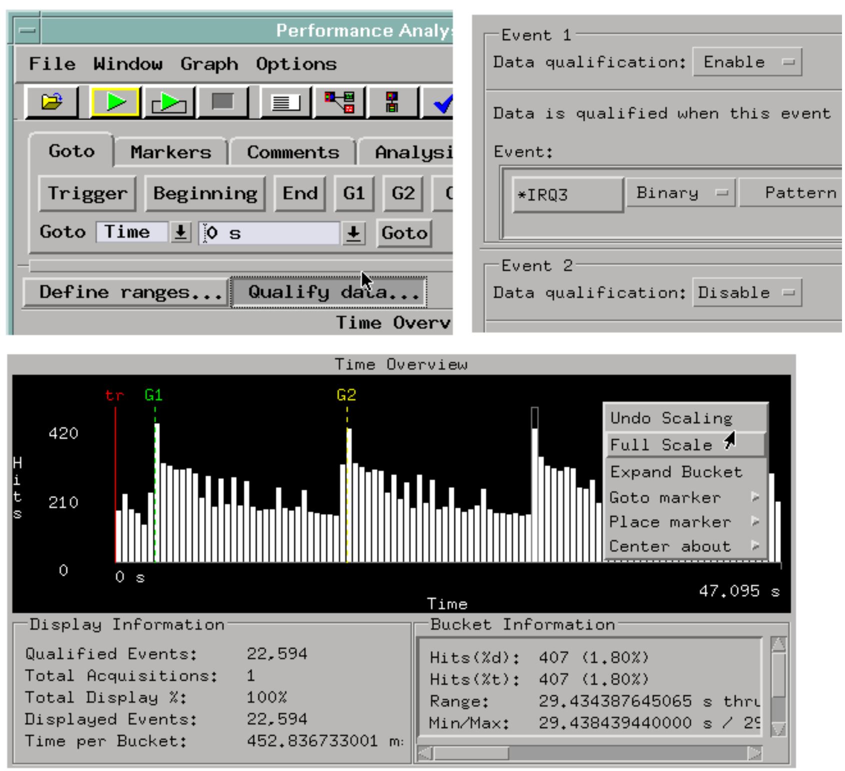
2. Select the Run button to start the measurement.

Displaying the Data

1. Use the system performance analyzer’s Time Overview display to show the results of the measurement.



2. Define the event whose occurrence rate you wish to measure.



You can define both the event and the time period in which the events are counted.

Software Development

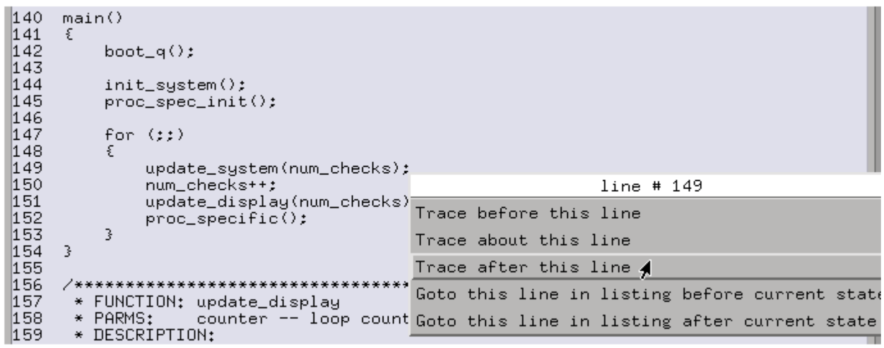
Analyzing Real-Time Software Execution

Analyzing Real-Time Variable Access

Analyzing Real-Time Memory Usage

Analyzing Real-Time Software Execution

To trace about a source line



Requirements:

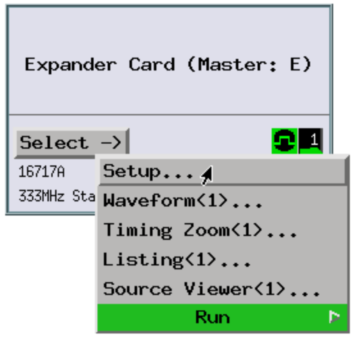
- This measurement requires the source correlation tool set product. When this product is installed, you can view high-level source files in a special listing window.

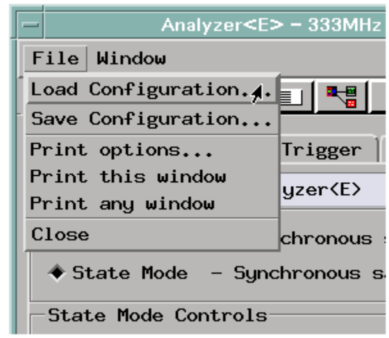
Possible uses:

- To quickly capture and view execution around a particular high-level source line.

Probing the Target System

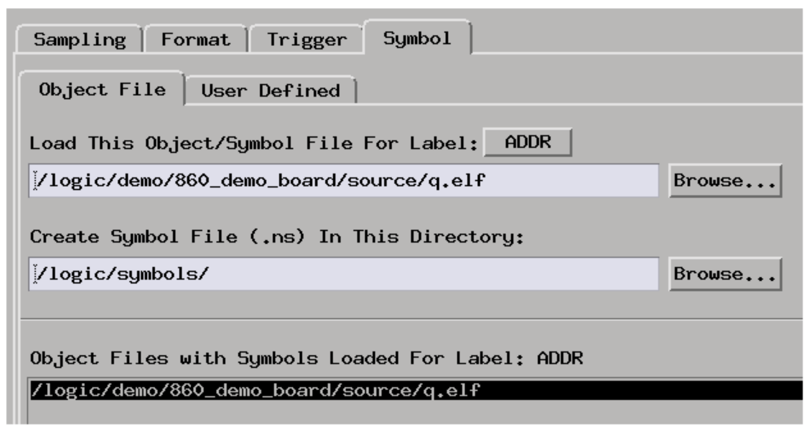
1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



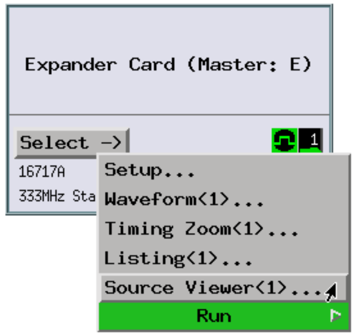


Capturing the Data

1. Download symbols from your target system program's object module file.

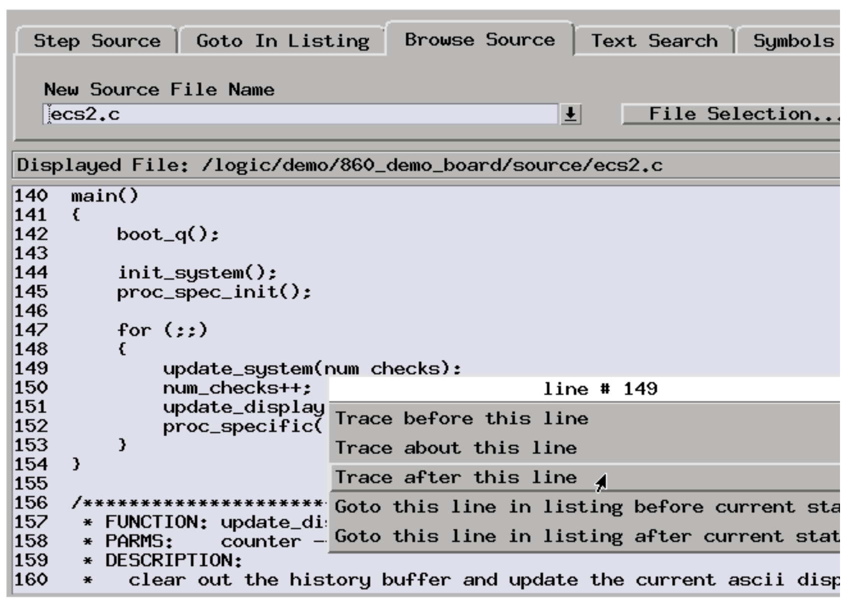


2. Open the Source Viewer window.



3. Browse the source file that contains the line you want to trigger on.

4. Select the line you want to trigger on and choose the "Trigger after this line" menu item.

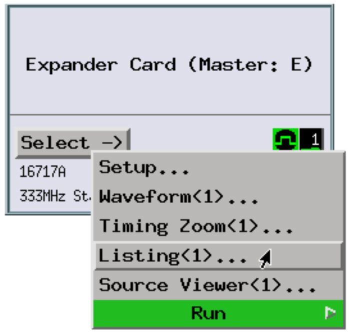


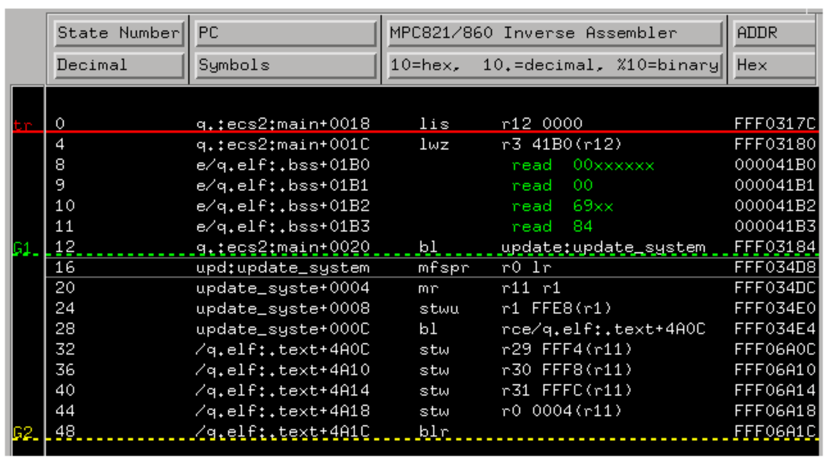
NOTE: Source Viewer commands that set up triggers only modify the trigger condition. They do not modify the trigger position, storage qualifiers, `else` branch conditions, or other levels in the trigger sequence.

5. Select the Run button to start the measurement.

Displaying the Data

1. Open the Listing window to display the captured data. You may want to load an inverse assembler and display symbols in the address label column.





2. You can use the Step Source Previous and Next buttons in the Source Viewer window to browse the captured data by associated source lines.

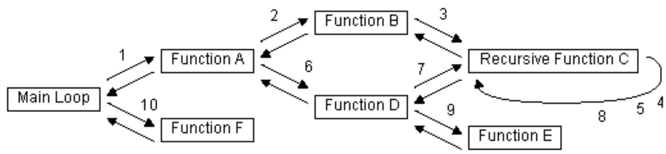
See Also

"To stop execution at a source line (in ROM)" on page 226

"To generate patterns when a source line executes" on page 262

"To trigger an oscilloscope when a source line executes" on page 294

To trace function flow



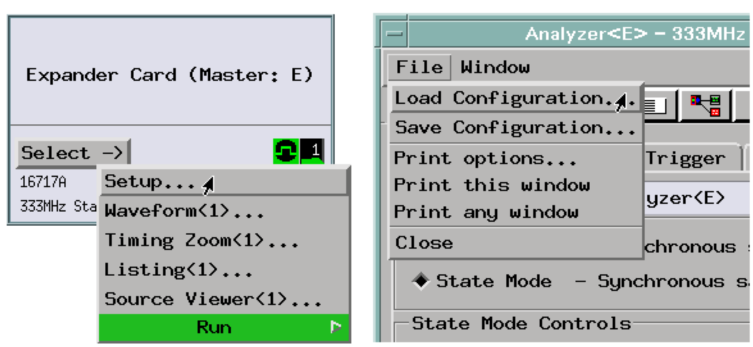
Function Calling Sequence:		
		Main Loop
	1	Function A
	2	Function B
	3	Recursive Function C
	4	Recursive Function C
	5	Recursive Function C
	6	Function D
	7	Recursive Function C
	8	Recursive Function C
	9	Function E
	10	Function F
	1	Function A
	2	. . .

Possible uses:

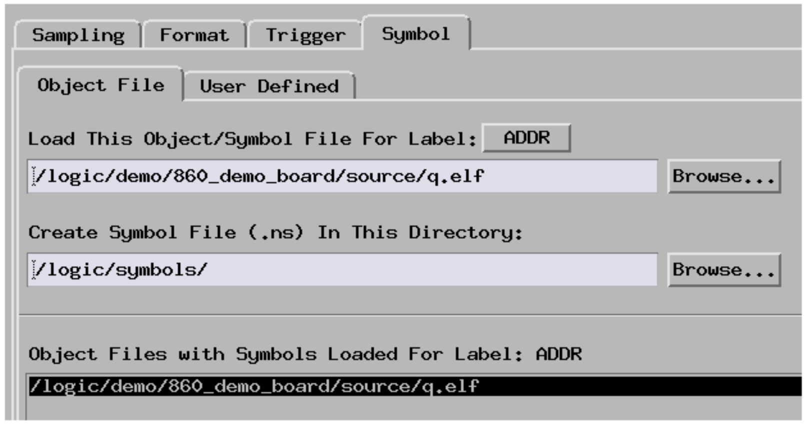
- To view the execution order of routines.

Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



2. Load symbols from your program's object module file.

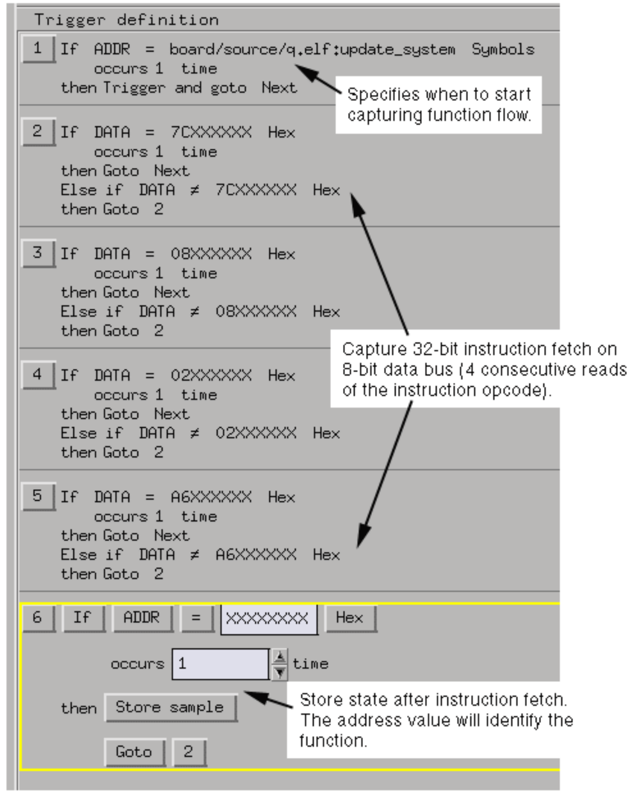


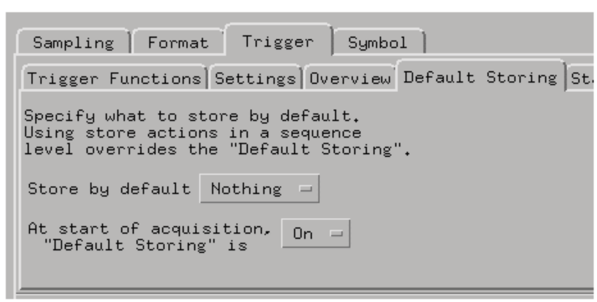
Capturing the Data

1. Set up a trigger specification that stores only the type of execution that occurs when a function is entered.

   For example, in the Motorola 68XXX microprocessors, the LINK instruction is commonly used on function entry to set up a new stack frame. In the PowerPC microprocessors, the "mfspr r0,lr" instruction is commonly used at function entry.

   Another way to identify function entry points is to add statements at the beginning of functions that write to particular memory locations (also known as instrumenting your code). This is the best way to identify function entry points when instruction caches are turned ON.

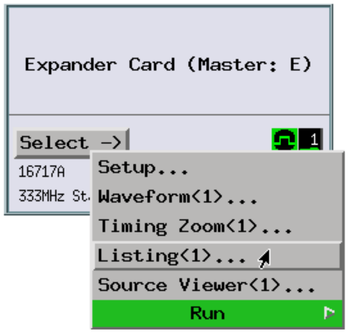


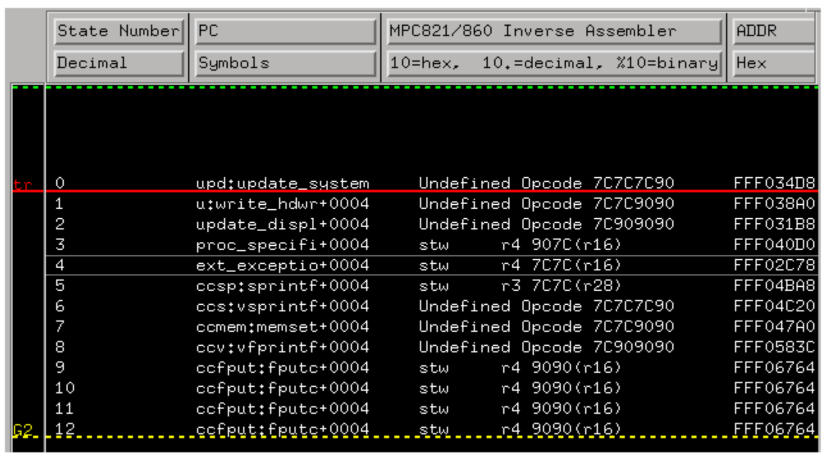


2. Select the Run button to start the measurement.

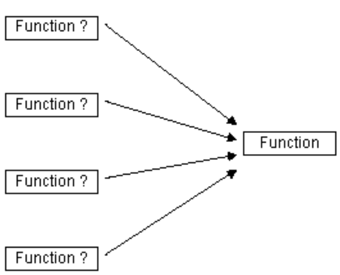
Displaying the Data

1. Open the Listing display to view the captured execution. By viewing the symbolic information associated with the captured states, you will see the function execution sequence.





To trace callers of a function

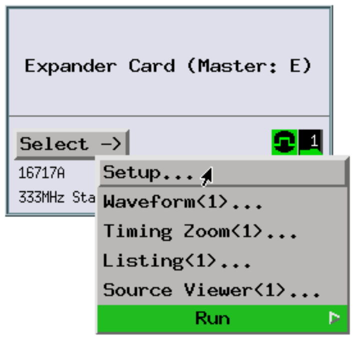


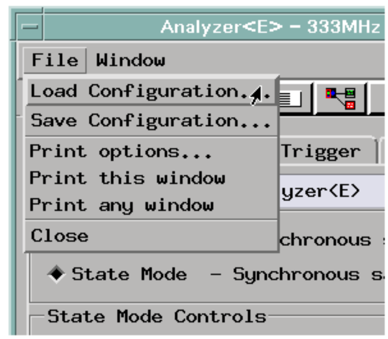
Possible uses:

- To show the callers of a particular function.
- To find out from where an exception or task call originates.

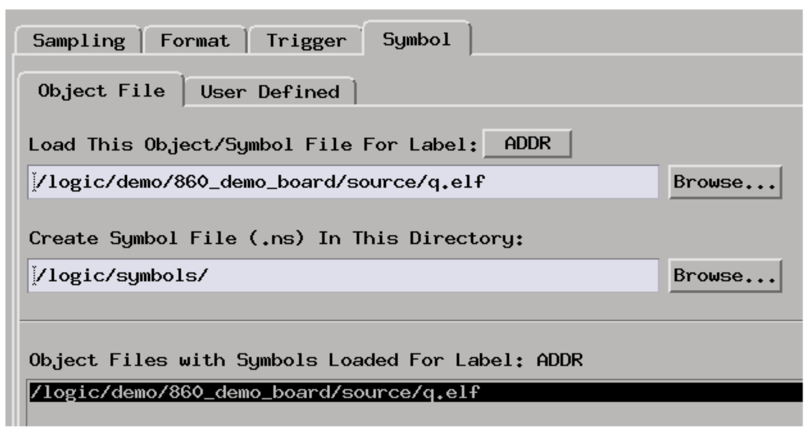
Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.





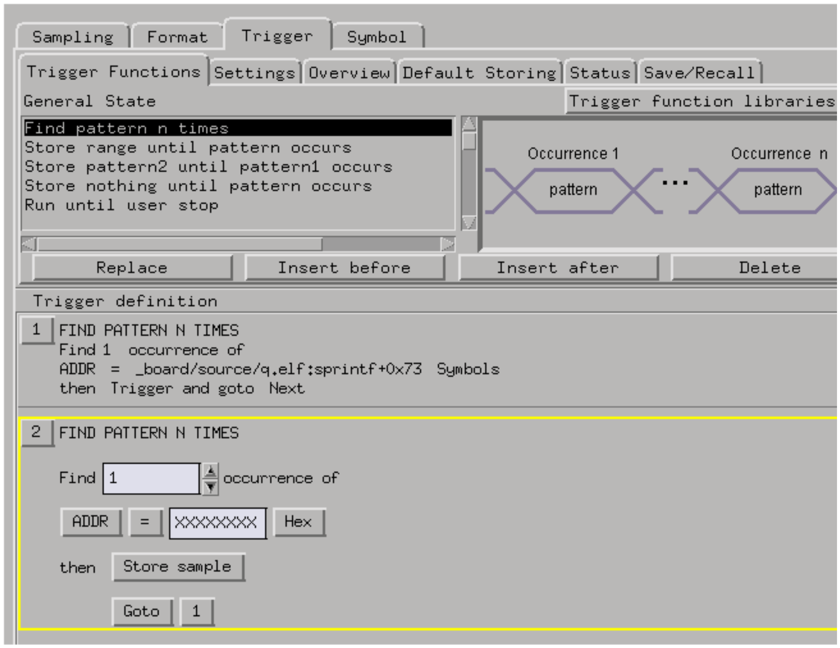
2. Load symbols from your program's object module file.

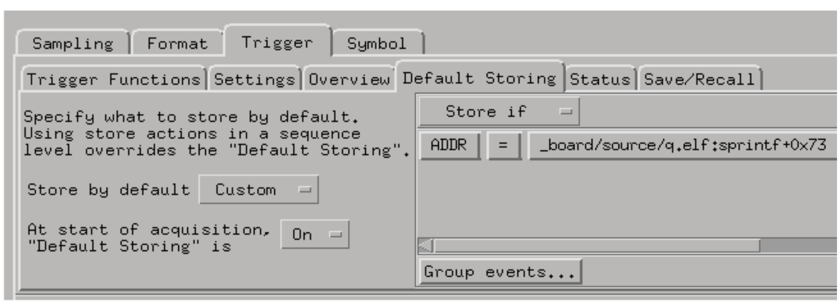


Capturing the Data

1. Set up a trace that captures and stores only entry into a particular function and uses context store to store the states that occurred before entry into the function.

   If your logic analyzer doesn't have the context store feature, you can set up a trigger sequence that stores a function's exit and where the execution returns to (which should identify the calling function).

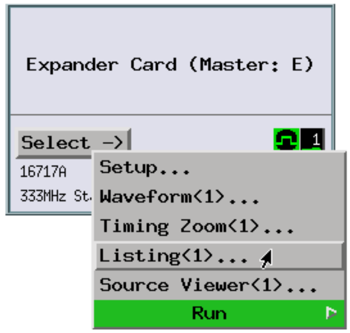


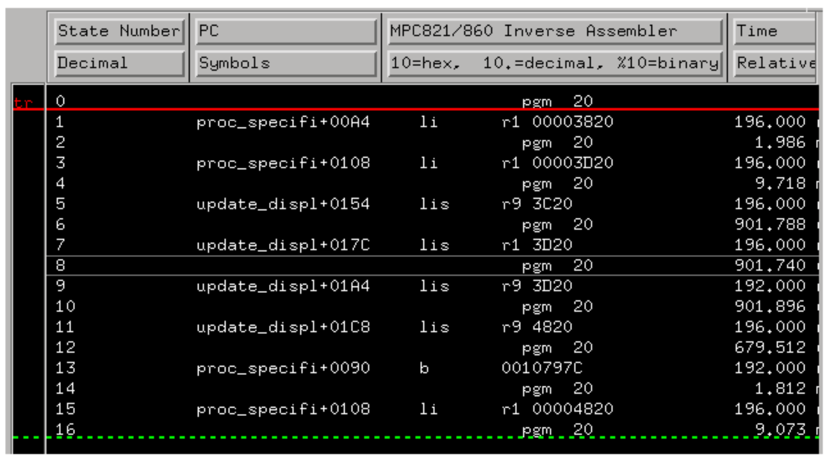


2. Select the Run button to start the measurement.

Displaying the Data

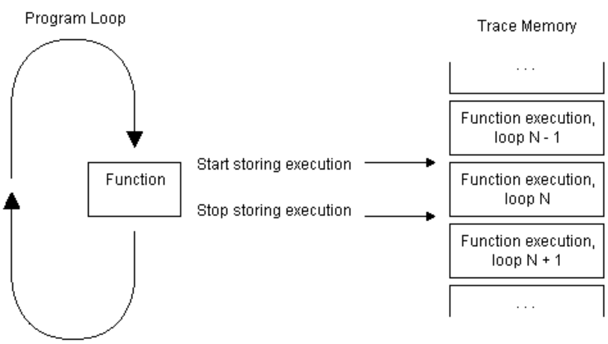
1. Open the Listing window to view the captured execution. Include symbols in the listing.





You can also open the Source Viewer window and use the Step Source Previous and Next buttons to browse the captured data by associated source lines.

To trace execution within a function

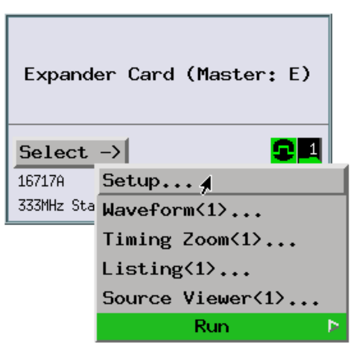


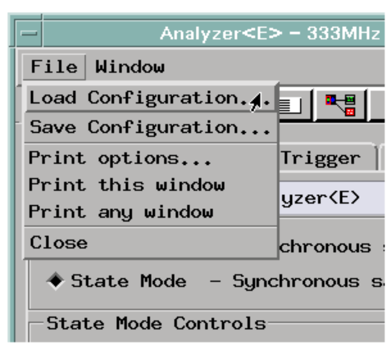
Possible uses:

- To capture more function execution data (because only function execution states are stored in trace memory).
- To capture a "window" of program execution or look at consecutive executions of a function.
- To store (and time) the execution of a memory management subroutine.
- To store (and time) an access to a disk drive.

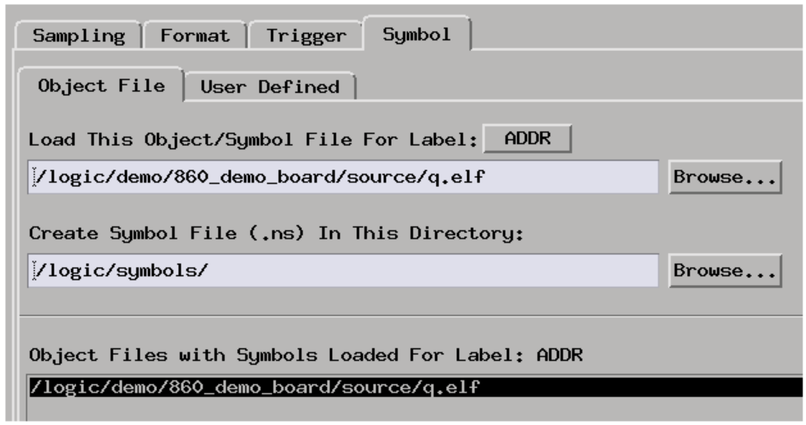
Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and format labels.





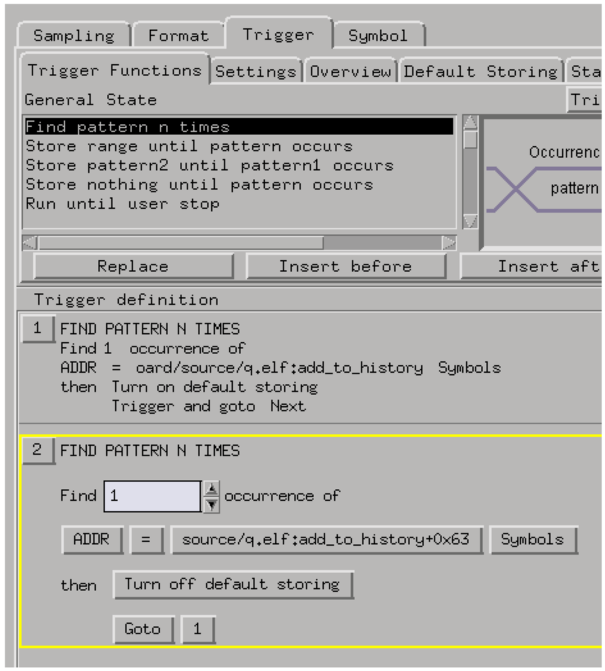
2. Load symbols from your program's object module file.

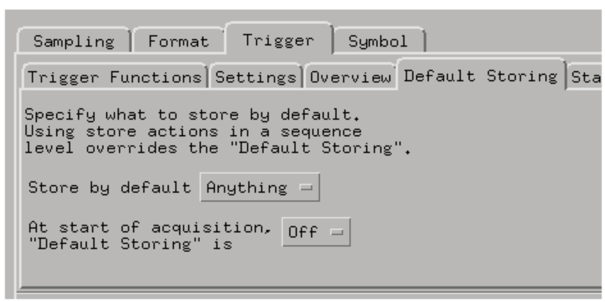


Capturing the Data

1. Set up pattern resources that define the window start and window end events.

2. Set up a trigger sequence where level 1, while storing no states, looks for the window start event; when it's found, the analyzer triggers. Level 2, while storing all states, looks for the window end event. At this point, the next sequence level can store no states (or you can branch back to the first level and store consecutive windows of program execution).



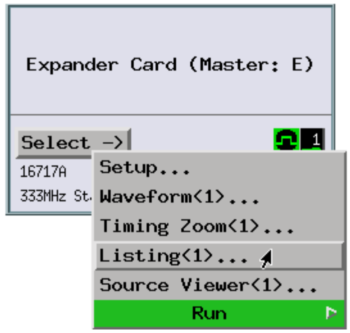


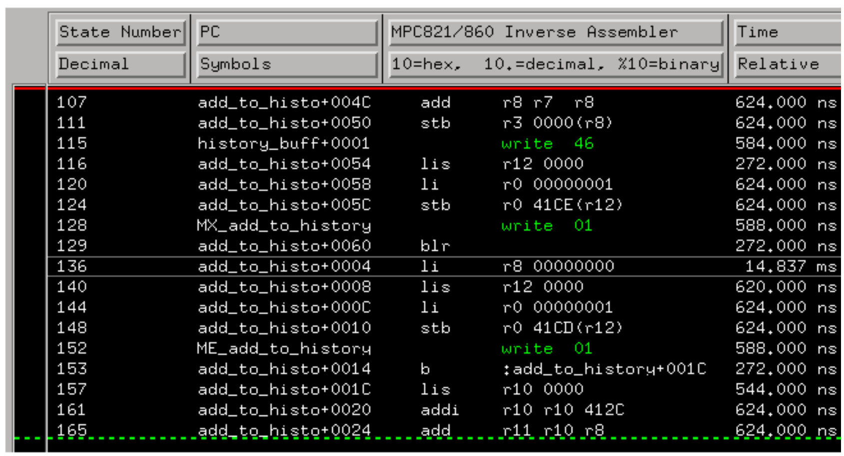
3. Select the Run button to start the measurement.

   Unless the window of program execution fills trace memory, you may have to select the Stop button in order to display the captured states.

Displaying the Data

1. When the analyzer triggers, open the Listing window to show that the window of program execution was captured.





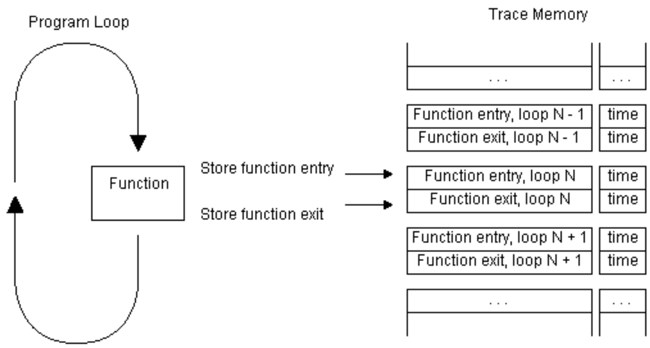
If the analyzer never triggers, the window start event never occurs.

You can also open the Source Viewer window and use the Step Source Previous and Next buttons to browse the captured data by associated source lines.

See Also

“If the trigger doesn't occur as expected” on page 309

To measure function execution time

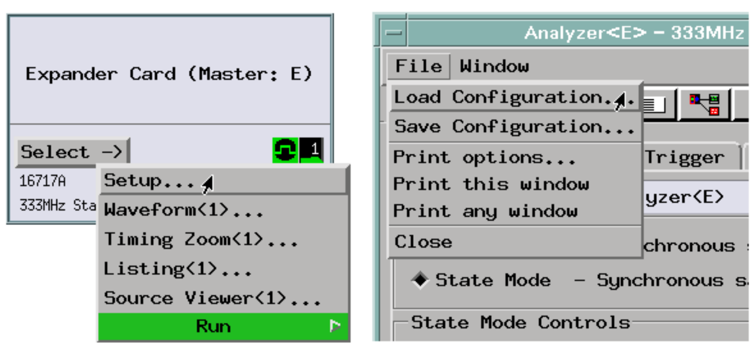


Possible uses:

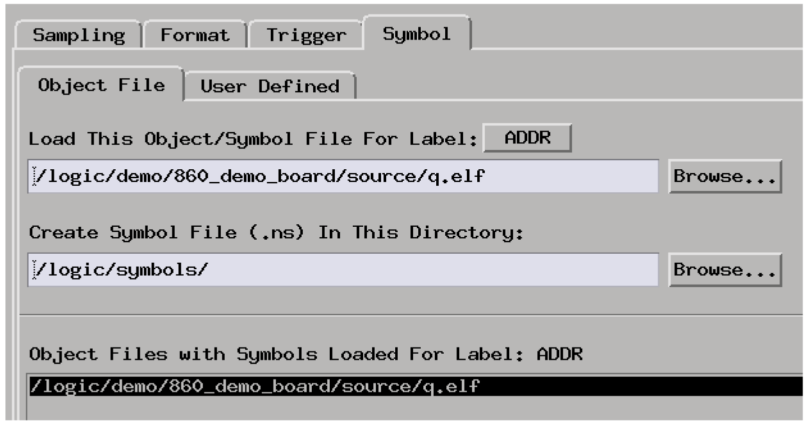
- To see if function execution times fall within specifications.

Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

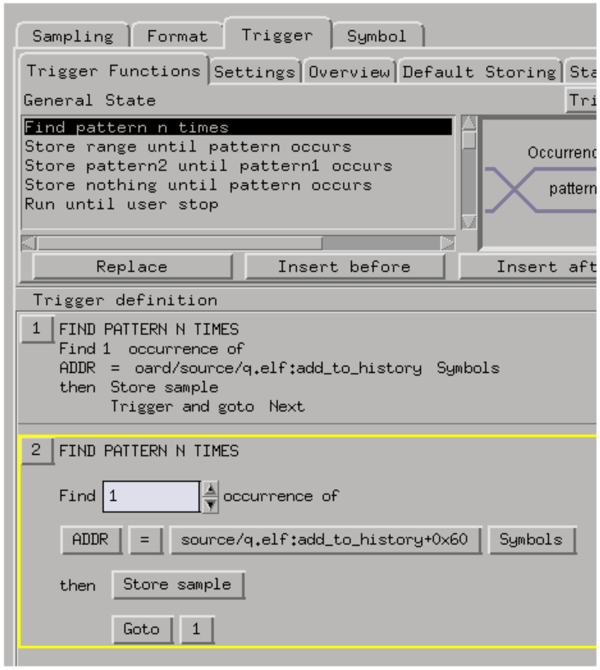


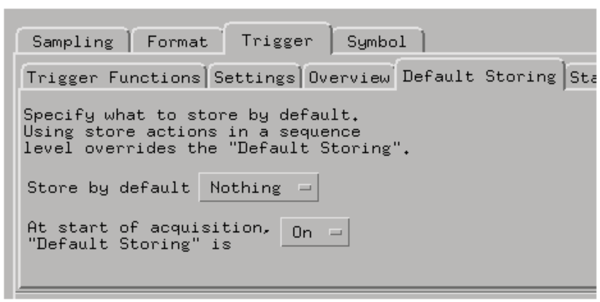
2. Load symbols from your program's object module file.



Capturing the Data

1. Set up pattern resources that define function entry and exit events.
2. Set up a trigger specification that stores only the entry and exit states of the function you're interested in. (This is the same as looking at the execution of a particular function, except only the entry and exit states are stored.) Be sure to turn ON the time count.

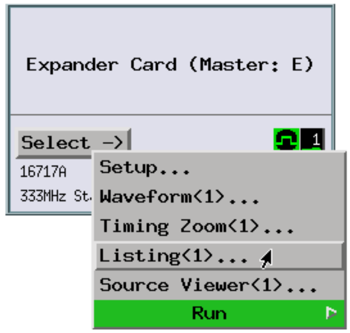


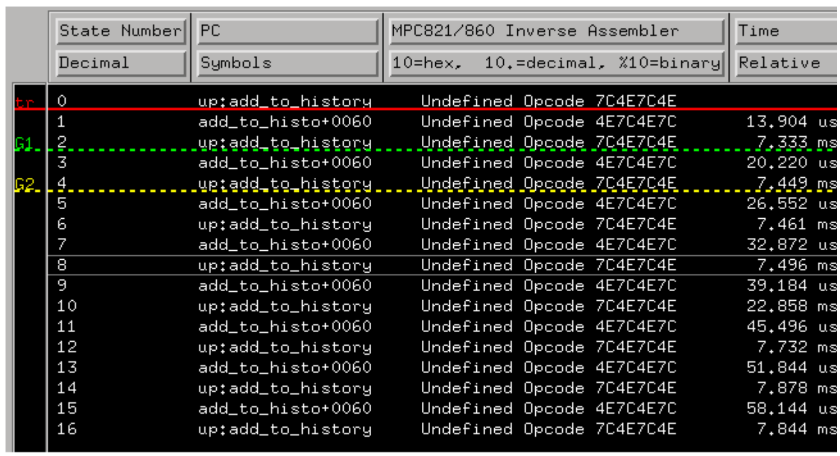


3. Select the Run button to start the measurement.

Displaying the Data

1. Use the Listing display to show the captured function entry and exit points. Count relative time to show relative function execution times.





	State Number	PC	MPC821/860 Inverse Assembler	Time
	Decimal	Symbols	10=hex, 10.=decimal, %10=binary	Relative
tr	0	up:add_to_history	Undefined Opcode 7C4E7C4E	
	1	add_to_histo+0060	Undefined Opcode 4E7C4E7C	13.904 us
G1	2	up:add_to_history	Undefined Opcode 7C4E7C4E	7.333 ms
	3	add_to_histo+0060	Undefined Opcode 4E7C4E7C	20.220 us
G2	4	up:add_to_history	Undefined Opcode 7C4E7C4E	7.449 ms
	5	add_to_histo+0060	Undefined Opcode 4E7C4E7C	26.552 us
	6	up:add_to_history	Undefined Opcode 7C4E7C4E	7.461 ms
	7	add_to_histo+0060	Undefined Opcode 4E7C4E7C	32.872 us
	8	up:add_to_history	Undefined Opcode 7C4E7C4E	7.496 ms
	9	add_to_histo+0060	Undefined Opcode 4E7C4E7C	39.184 us
	10	up:add_to_history	Undefined Opcode 7C4E7C4E	22.858 ms
	11	add_to_histo+0060	Undefined Opcode 4E7C4E7C	45.496 us
	12	up:add_to_history	Undefined Opcode 7C4E7C4E	7.732 ms
	13	add_to_histo+0060	Undefined Opcode 4E7C4E7C	51.844 us
	14	up:add_to_history	Undefined Opcode 7C4E7C4E	7.878 ms
	15	add_to_histo+0060	Undefined Opcode 4E7C4E7C	58.144 us
	16	up:add_to_history	Undefined Opcode 7C4E7C4E	7.844 ms

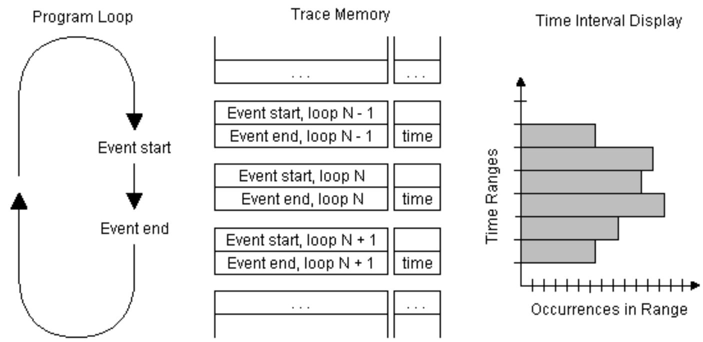
You can use a global marker to search for the function exit states whose relative time values show the function execution time.

See Also

"To measure function execution time (with SPA)" on page 218

"To trace execution within a function" on page 210

To measure function execution time (with SPA)



The system performance analyzer's Time Interval display gives you a histogram of (and statistics on) event execution times.

Requirements:

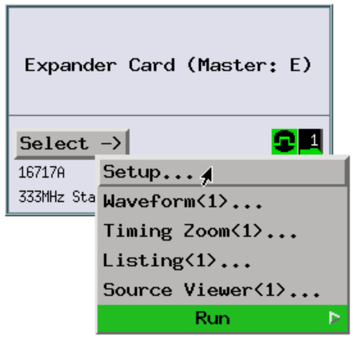
- This measurement requires the system performance analyzer (SPA) tool set.

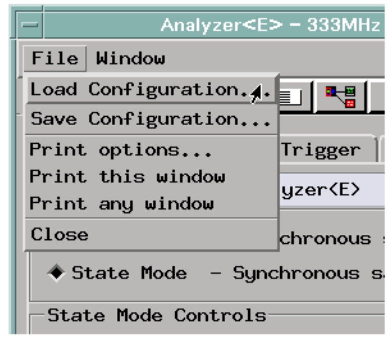
Possible uses:

- To check how the execution time of a particular function varies.
- To determine if optimization of a function is needed.

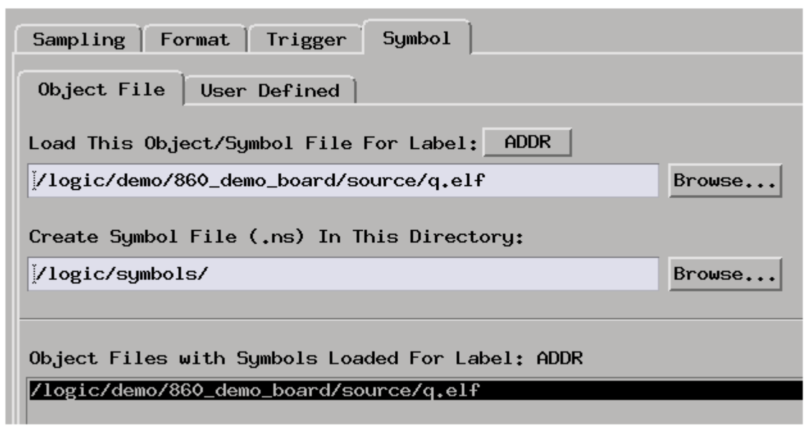
Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



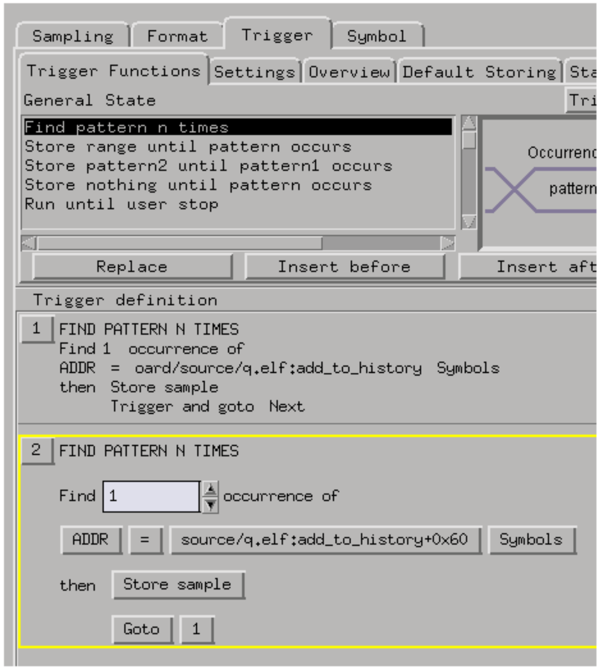


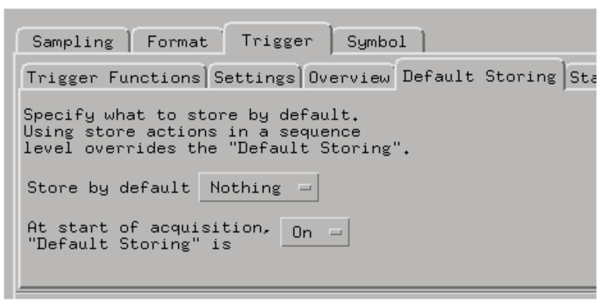
2. Load symbols from your program's object module file.



Capturing the Data

1. Set up the logic analyzer trigger specification to capture only the entry and exit points of a function.

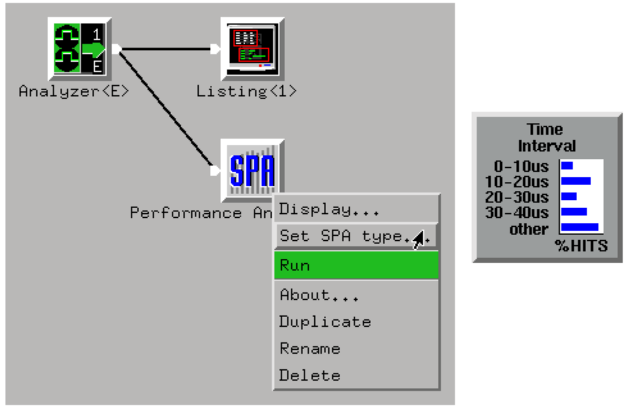




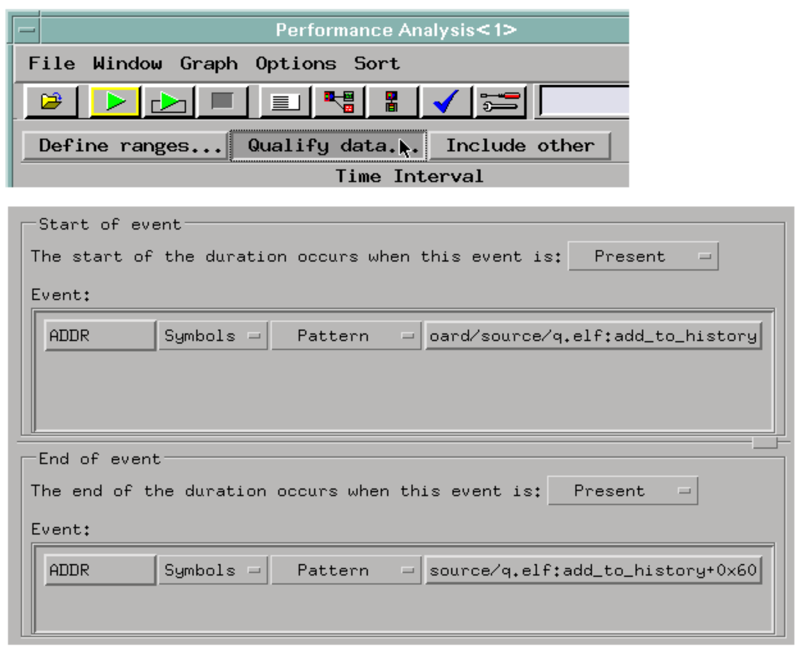
2. Select the Run button to start the measurement.

Displaying the Data

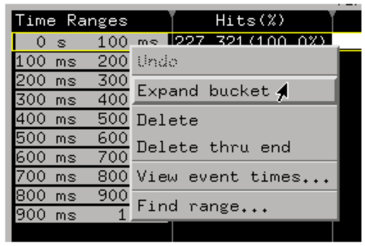
1. In the Workspace window, use the system performance analyzer's Time Interval display to view the captured data.

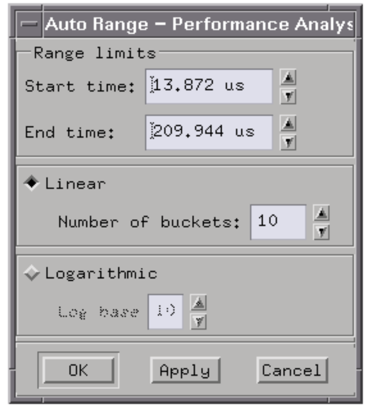


2. Define the start and end of the event (that is, the function) whose time variations you wish to measure.

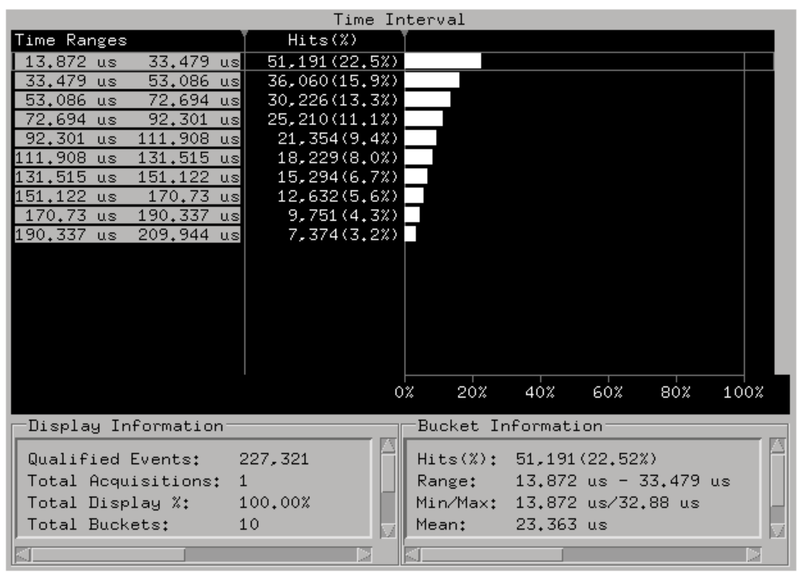


3. Define *buckets* for captured time ranges.





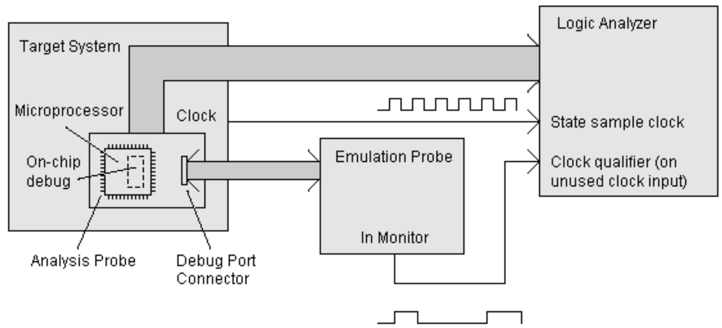
4. View the measurement results.



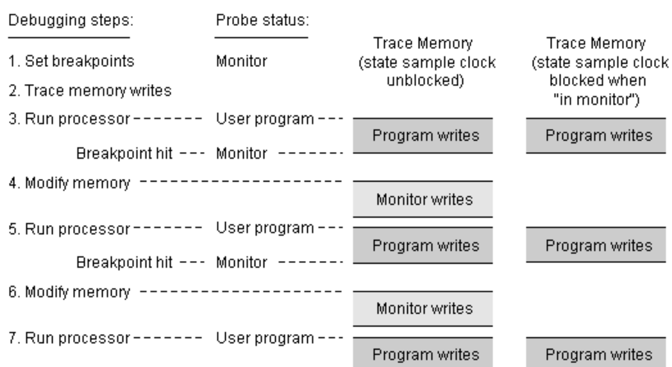
See Also

"To measure function execution time" on page 214

To omit monitor cycles from the trace



Use the emulator probe’s "in monitor" signal to block the logic analyzer’s state clock.



Requirements:

- You need a signal to tell the logic analyzer when the processor is executing in the monitor. The emulation probe provides such a signal.

Possible uses:

- To capture execution between a debugger's breakpoints.

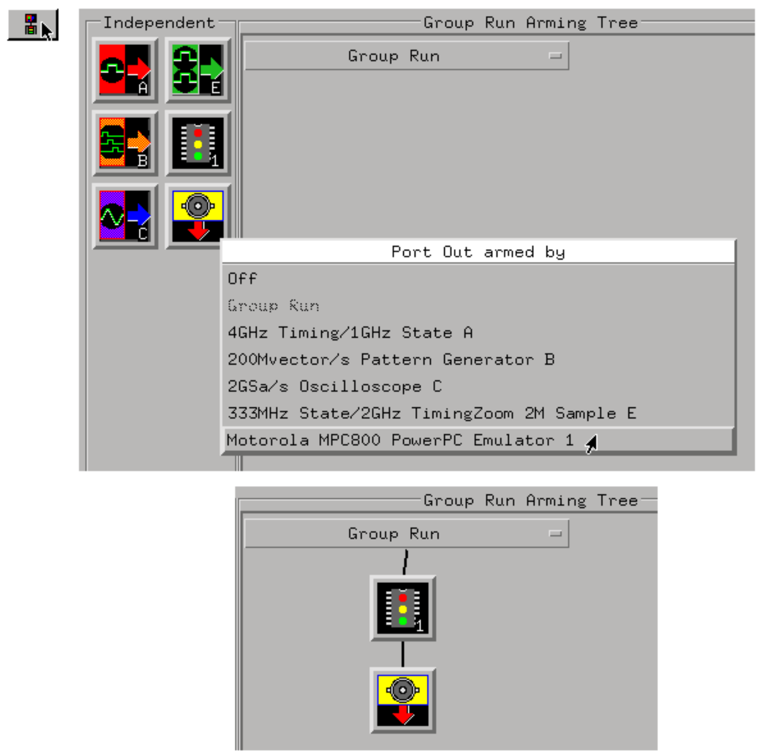
Probing the Target System

1. If you're using the emulation probe to provide the "in monitor" signal, set up the emulation probe connection to the target system processor.

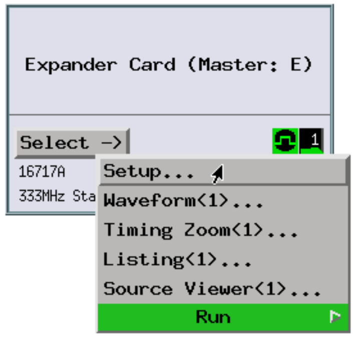
2. Also, configure the emulation probe to output its "in monitor" signal.
3. Connect the logic analyzer Port Out signal to one of the unused clock inputs on the logic analyzer pods. (An Intermodule setup will be used to route the emulation module's "in monitor" signal to the logic analyzer's Port Out BNC connector.)
4. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

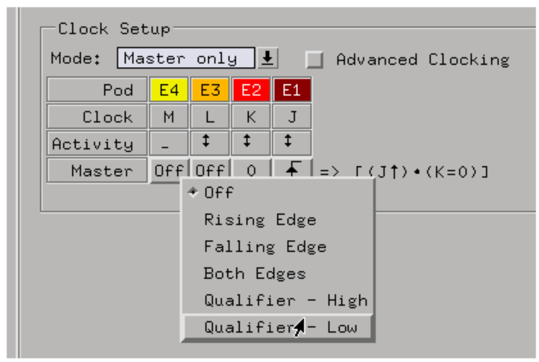
Capturing the Data

1. Set up the Intermodule window so that Port Out is armed by the emulation module.



2. Set up the trigger specification as you would normally, but also set up the logic analyzer state clock to only occur when the "in monitor" signal is false.





3. Select the Group Run button to start the trace measurement.

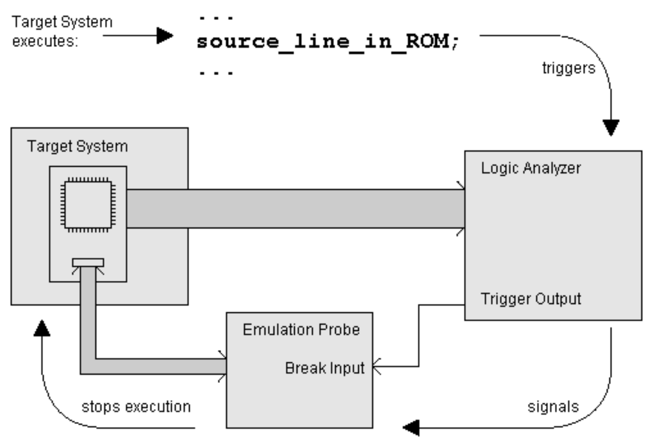
   You can freely set breakpoints and examine the state of the microprocessor knowing that the logic analyzer will not capture any of the monitor cycles.

   If your logic analyzer measurements count time, you'll see large time values in the trace when the microprocessor is executing in it's debug monitor mode.

Displaying the Data

1. Use the Listing display to the captured data. You may have to stop the measurement to view captured data (if the events you're capturing are infrequent enough to allow breaks and monitor cycles to be captured without clock qualification).

To stop execution at a source line (in ROM)



Normally, you would use a debugger to stop microprocessor execution at a particular source line. However, if the debugger implements breakpoints by replacing code (or some other mechanism that requires code to be in RAM), you will not be able to set breakpoints on source code that exists in ROM. Luckily, the logic analyzer can tell the emulation probe to stop processor execution when it captures a particular event.

Requirements:

- To make this measurement, you need a microprocessor run control mechanism, like an emulation probe, that can stop microprocessor execution when a trigger signal is received from the logic analyzer.

Possible uses:

- To stop microprocessor execution when debugger breakpoints cannot.

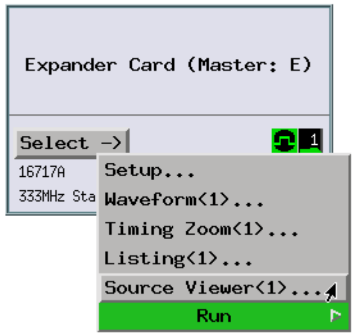
Probing the Target System

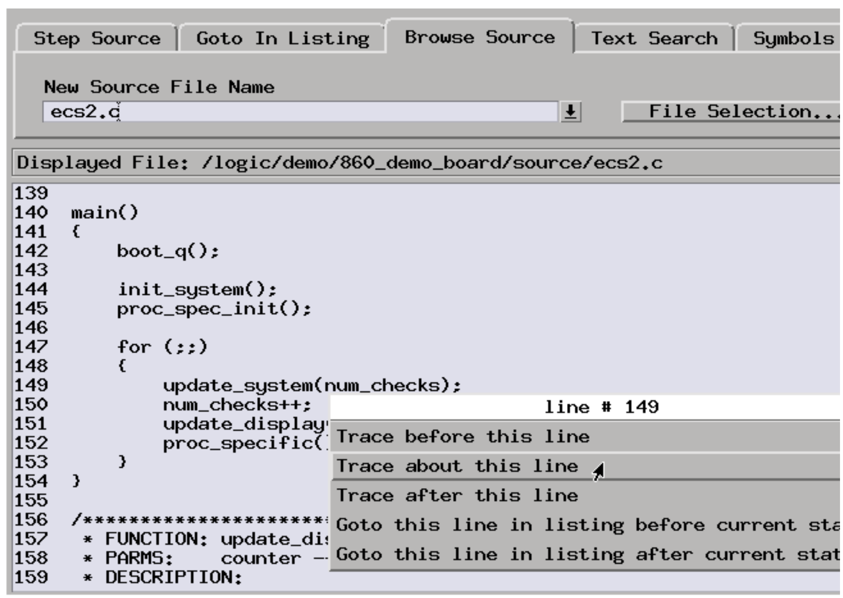
1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.
2. Download program symbols to the logic analyzer and set up access to the program source files.
3. Make sure the emulation probe (or emulation module and emulation

adapter) has been connected to the target system.

Capturing the Data

1. Set up the logic analyzer to trigger on the source line you're interested in.

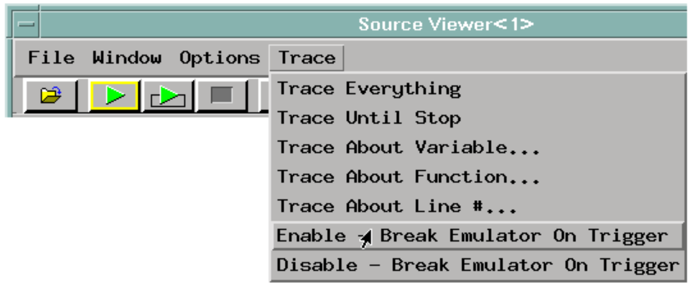




NOTE: Source Viewer commands that set up triggers only modify the trigger condition. They do not modify the trigger position, storage qualifiers, `else` branch conditions, or other levels in the trigger sequence.

2. In the Source Viewer window, choose the Trace->Enable - Break Emulator

On Trigger command.

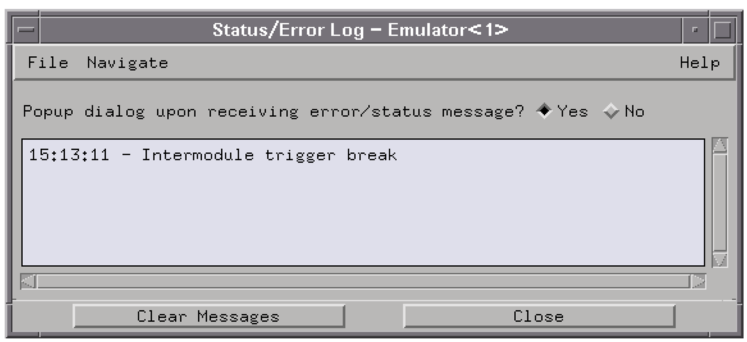


This command will automatically set up the Intermodule window to specify that the emulation module's break input be armed by the logic analyzer's trigger.

3. Select the Group Run button to start the measurement.

Displaying the Data

1. When the logic analyzer trigger is found, microprocessor execution stops.



Note that microprocessor execution does not stop immediately after the logic analyzer trigger because of delay in the intermodule signals and the speed of the processor.

See Also

"To trace about a source line" on page 199

"To stop execution on a corrupt variable" on page 245

Analyzing Real-Time Variable Access

To find NULL pointer de-references

```
main() {
   int a = 7, b, *x, *y;

   x = &a;
   y = NULL;
   b = *x;   /* b = 7 */
   b = *y;   /* NULL pointer de-reference,
                b = contents of NULL address
             */
}
```

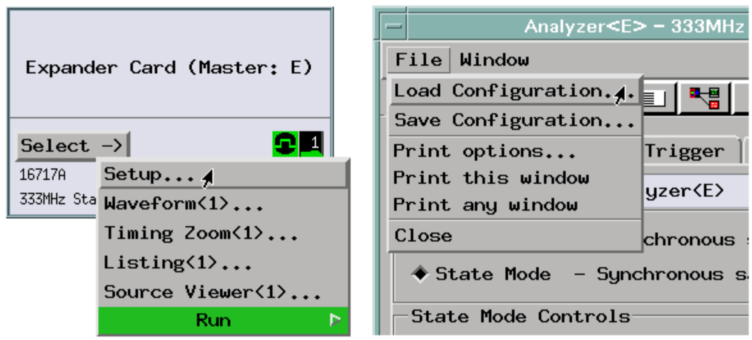
Because a NULL pointer has particular address, commonly 0, you can trace accesses of the NULL address to find NULL pointer de-references.

Possible uses:

- To check for the possible cause of NULL pointer de-references.

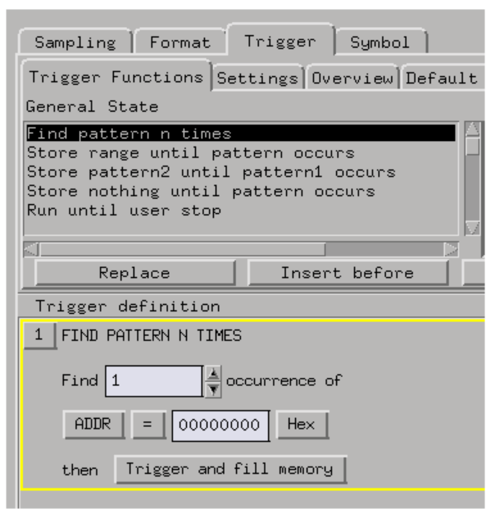
Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



Capturing the Data

1. Set up to trigger on an access of address 0.



The states that are stored before the trigger may show a NULL pointer de-reference.

2. Select the Run button to start the measurement.

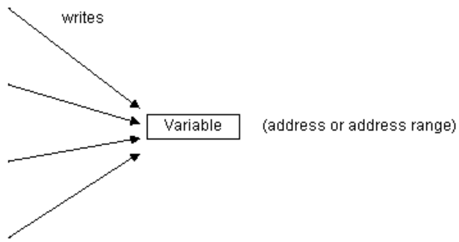
Displaying the Data

1. If the analyzer triggers, open the Listing window to display the access of address 0. You may want to load an inverse assembler, load symbols, and display symbols in the address label column.

2. You can also open the Source Viewer window and use the Step Source

Previous and Next buttons to browse the captured data by associated source lines.

To trace a variable's values



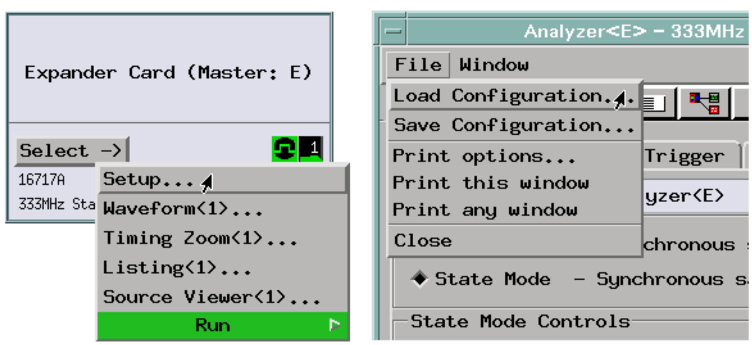
Store only variable write accesses.

Possible uses:

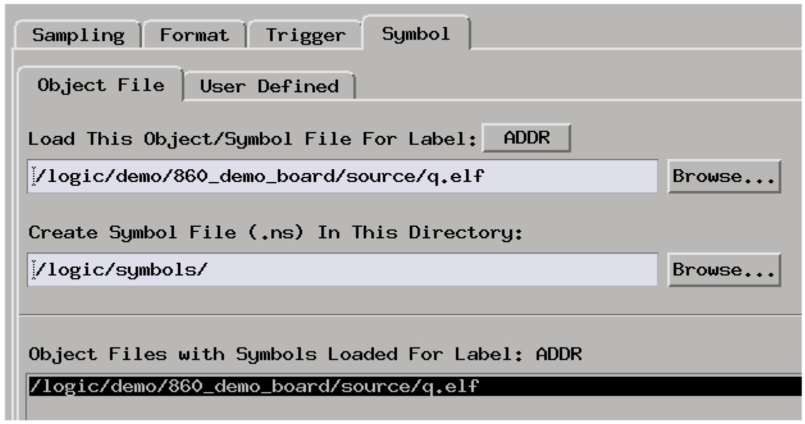
- To look for inappropriate variable values (in memory locations, not local variables on the stack or in microprocessor registers).

Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

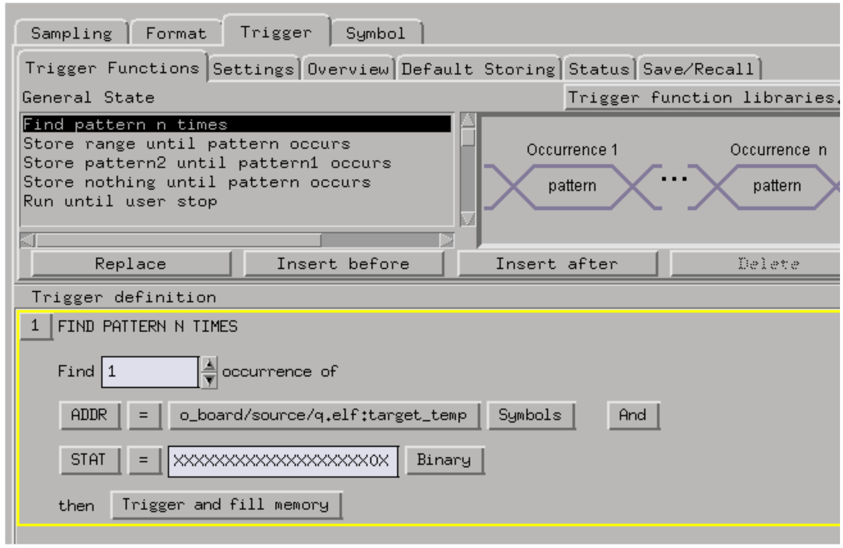


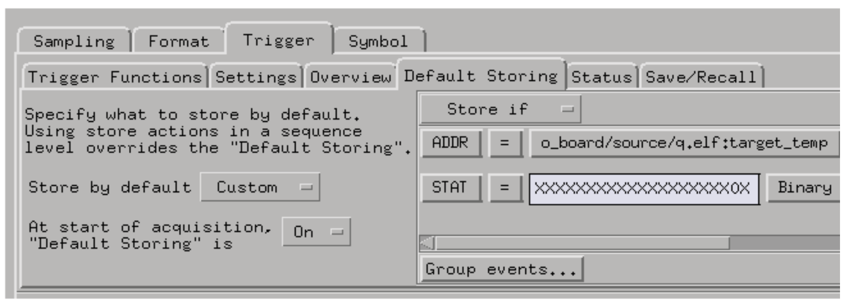
2. Load symbols from your program's object module file.



Capturing the Data

1. Set up a trigger specification that stores only writes to the variable addresses.

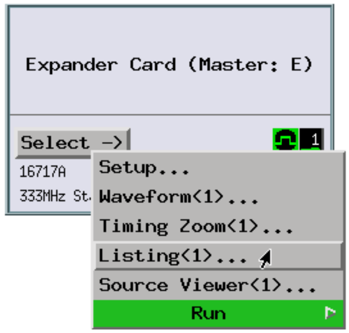


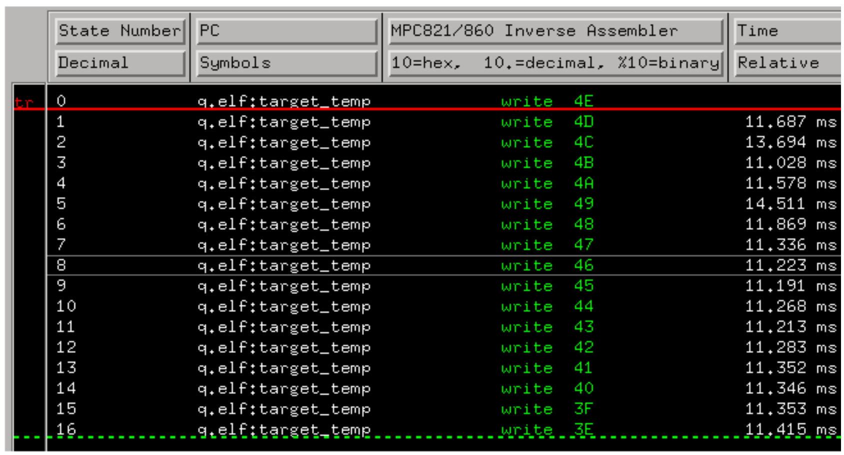


2. Select the Run button to start the measurement.

Displaying the Data

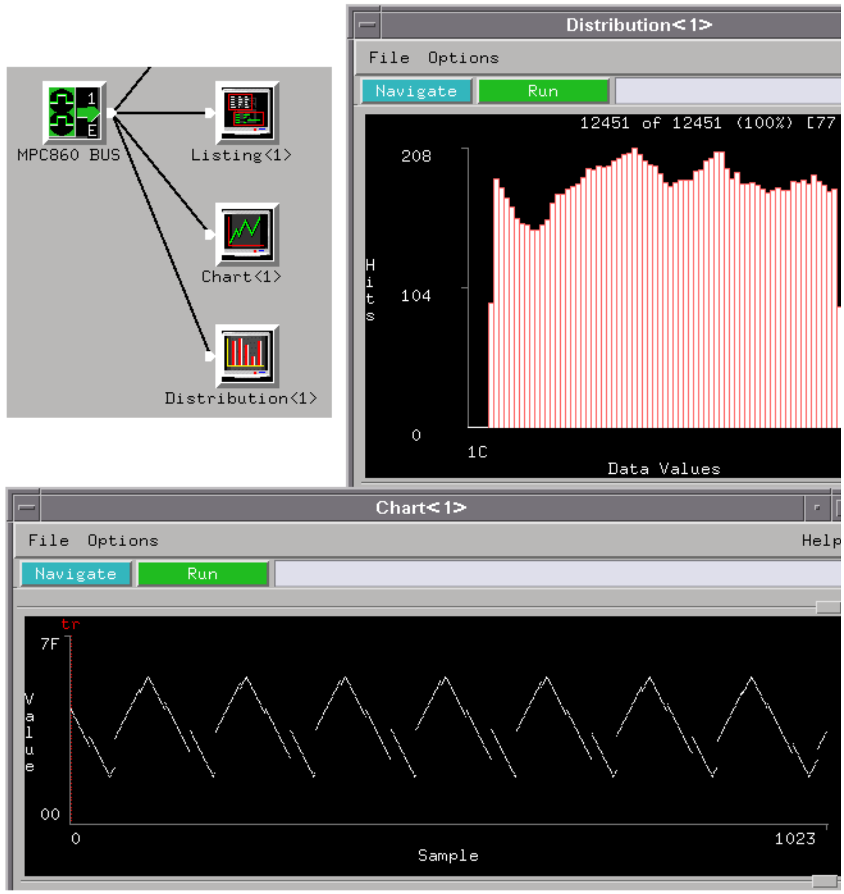
1. Open the Listing window to view the captured variable values.



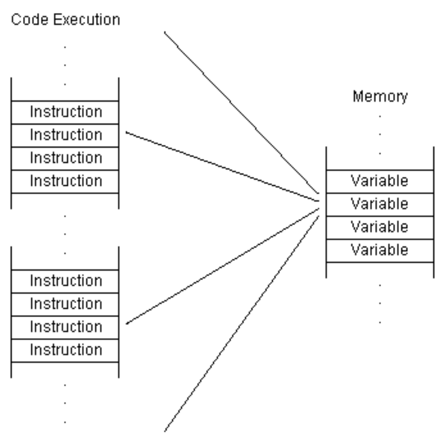


State Number	PC	MPC821/860 Inverse Assembler	Time
Decimal	Symbols	10=hex, 10.=decimal, %10=binary	Relative
0	q.elf:target_temp	write 4E	
1	q.elf:target_temp	write 4D	11.687 ms
2	q.elf:target_temp	write 4C	13.694 ms
3	q.elf:target_temp	write 4B	11.028 ms
4	q.elf:target_temp	write 4A	11.578 ms
5	q.elf:target_temp	write 49	14.511 ms
6	q.elf:target_temp	write 48	11.869 ms
7	q.elf:target_temp	write 47	11.336 ms
8	q.elf:target_temp	write 46	11.223 ms
9	q.elf:target_temp	write 45	11.191 ms
10	q.elf:target_temp	write 44	11.268 ms
11	q.elf:target_temp	write 43	11.213 ms
12	q.elf:target_temp	write 42	11.283 ms
13	q.elf:target_temp	write 41	11.352 ms
14	q.elf:target_temp	write 40	11.346 ms
15	q.elf:target_temp	write 3F	11.353 ms
16	q.elf:target_temp	write 3E	11.415 ms

In the Workspace window, you can also set up the Chart, Distribution, or system performance analyzer displays for different views of the variable values.


MPC860 BUS
Listing<1>
Chart<1>
Distribution<1>
Distribution<1>
File Options
Navigate
Run
12451 of 12451 (100%)
208
104
0
Hits
1C
Data Values
Chart<1>
File Options
Help
Navigate
Run
tr
7F
Value
00
0
1023
Sample

To find where variables are accessed from



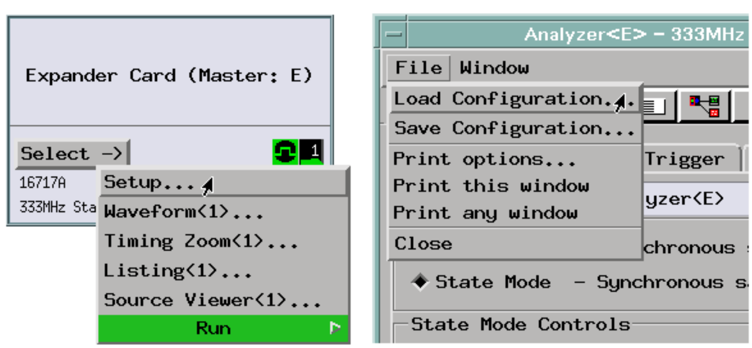
By storing only writes to a variable and context storing instructions, you will see the code that writes to the variable.

Possible uses:

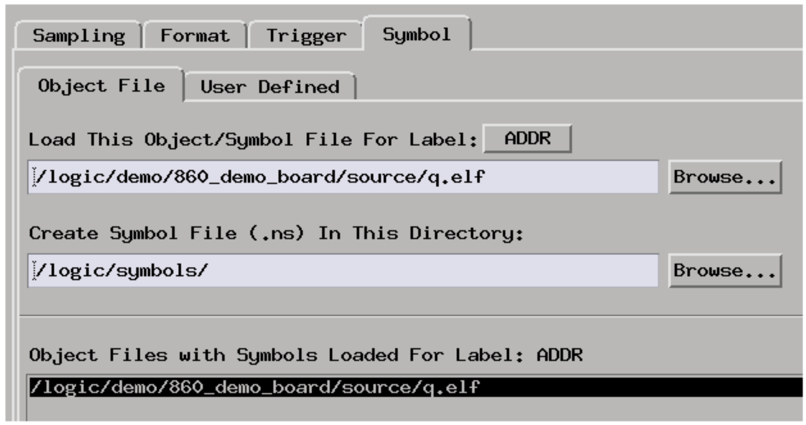
- To find which functions access a global variable.
- To trace writers of a variable.

Probing the Target System

1. Typically, this measurement is made with a state analyzer and an analysis probe capturing software execution. Configure the analyzer and format labels by loading the configuration files provided with the analysis probe.



2. Load symbols from your program's object module file.

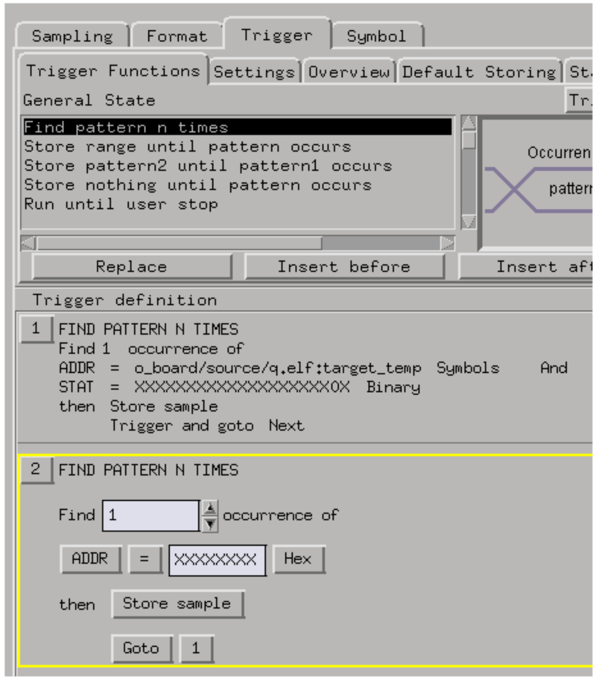


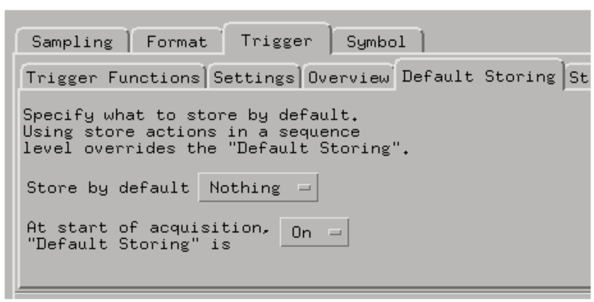
Capturing the Data

1. Set up a trigger specification to store only variable accesses and, for each variable access, another state that indicates where the variable was accessed from.

   If the analyzer has context store capability, use it to store any instruction fetch that occurs before the variable access.

   If the analyzer doesn't have context store capability, you can look for the variable access and store any instruction fetch that occurs after the access.

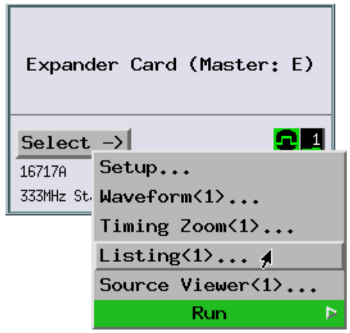


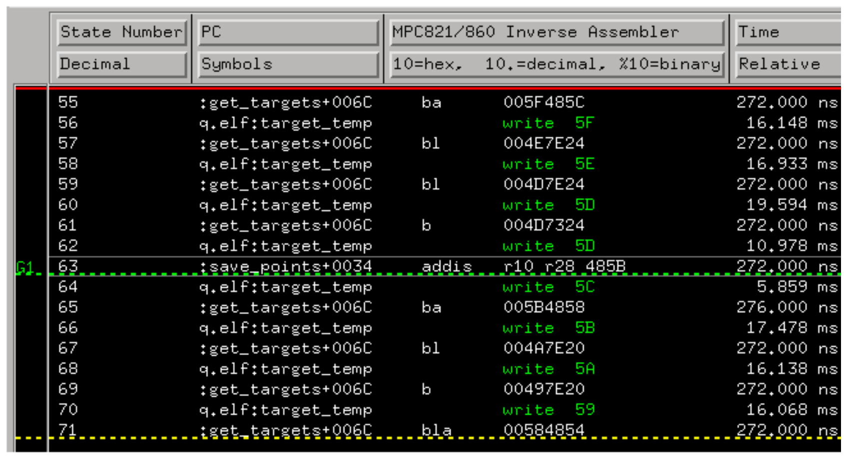


2. Select the Run button to start the measurement.

Displaying the Data

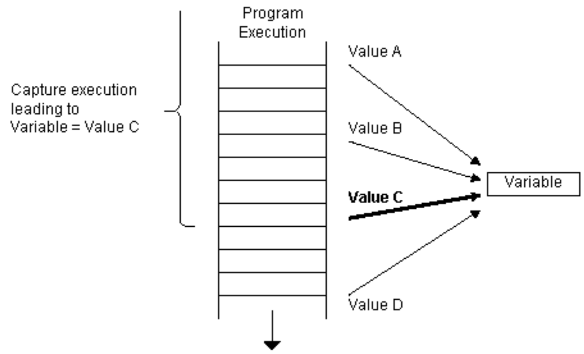
1. Open the Listing window to display the captured data. You may want to load an inverse assembler and display symbols in the address label column.





	State Number	PC	MPC821/860 Inverse Assembler		Time
	Decimal	Symbols	10=hex, 10.=decimal, %10=binary		Relative
	55	:get_targets+006C	ba	005F485C	272.000 ns
	56	q.elf:target_temp		write 5F	16.148 ms
	57	:get_targets+006C	bl	004E7E24	272.000 ns
	58	q.elf:target_temp		write 5E	16.933 ms
	59	:get_targets+006C	bl	004D7E24	272.000 ns
	60	q.elf:target_temp		write 5D	19.594 ms
	61	:get_targets+006C	b	004D7324	272.000 ns
	62	q.elf:target_temp		write 5D	10.978 ms
G1	63	:save_points+0034	addis	r10 r28 485B	272.000 ns
	64	q.elf:target_temp		write 5C	5.859 ms
	65	:get_targets+006C	ba	005B4858	276.000 ns
	66	q.elf:target_temp		write 5B	17.478 ms
	67	:get_targets+006C	bl	004A7E20	272.000 ns
	68	q.elf:target_temp		write 5A	16.138 ms
	69	:get_targets+006C	b	00497E20	272.000 ns
	70	q.elf:target_temp		write 59	16.068 ms
	71	:get_targets+006C	bla	00584854	272.000 ns

To trace before a variable value

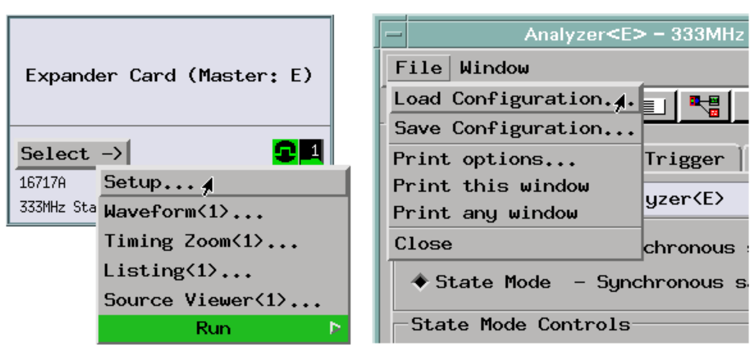


Possible uses:

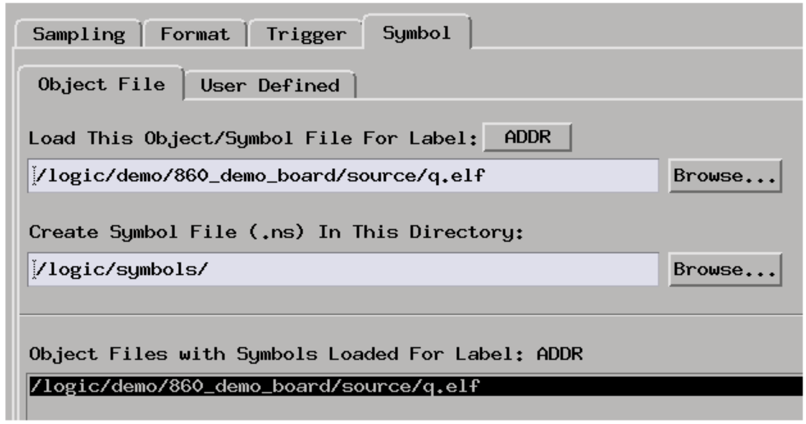
- To find the cause of a variable corruption.
- To find a variable value outside a specified range.
- To find when a variable equals a particular value.

Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

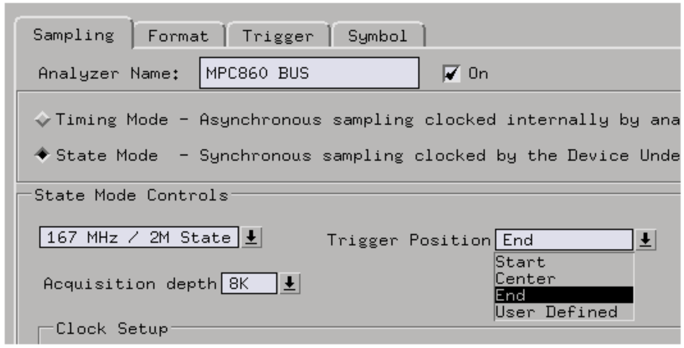


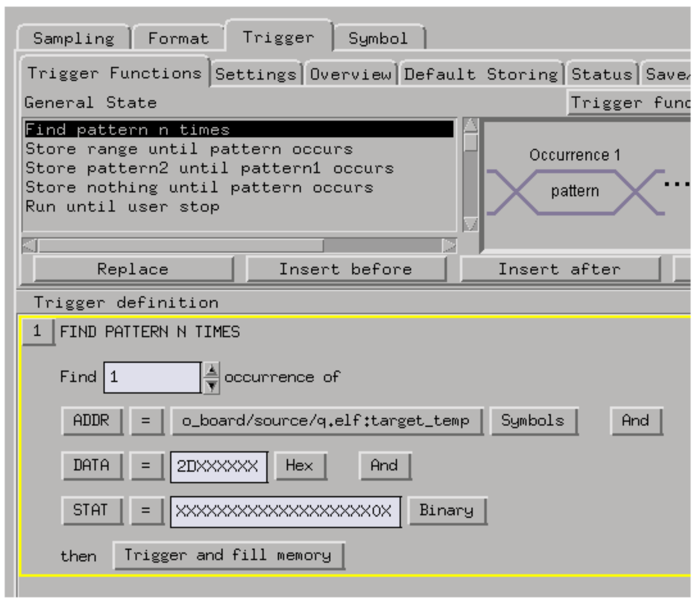
2. Load symbols from your program's object module file.



Capturing the Data

1. Set up a trigger specification that, while storing all states, looks for a particular value being written to a variable. Make sure the trigger point appears at the end of the trace.

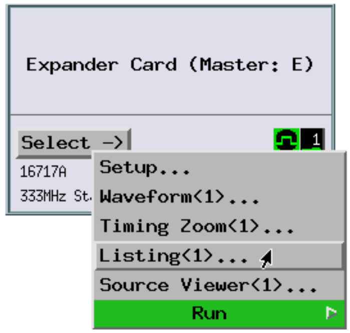


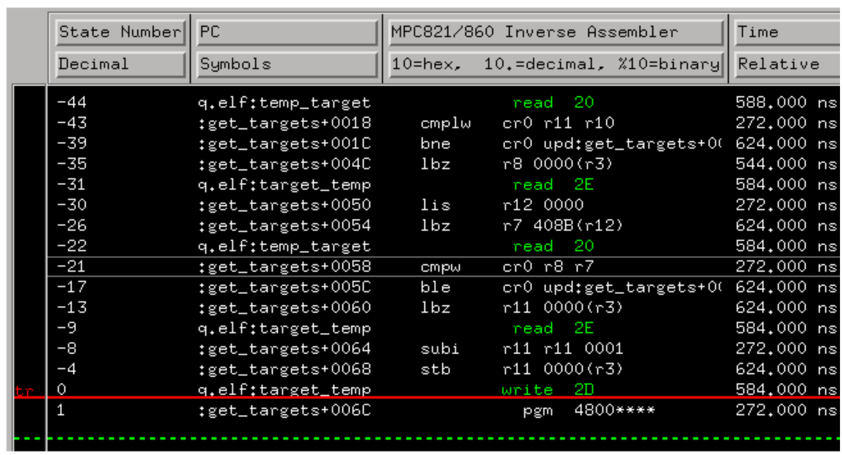


2. Select the Run button to start the measurement.

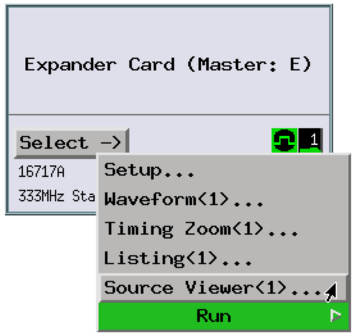
Displaying the Data

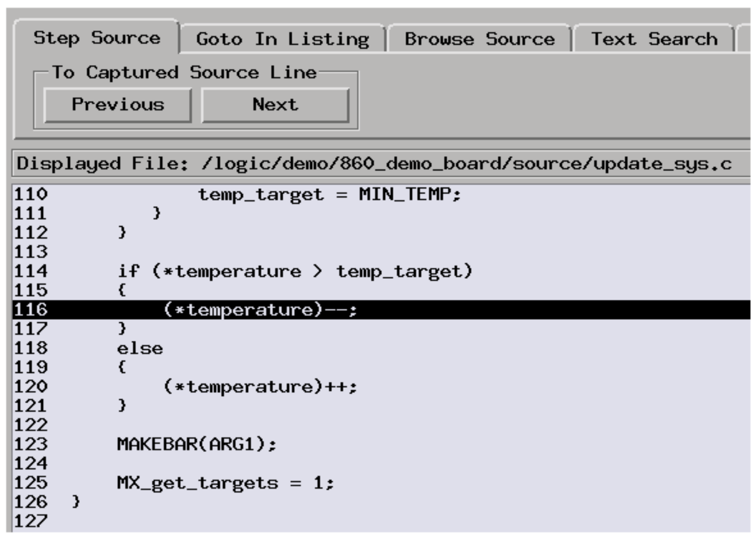
1. Open the Listing window to view the execution that led to the write of the particular value to the variable.



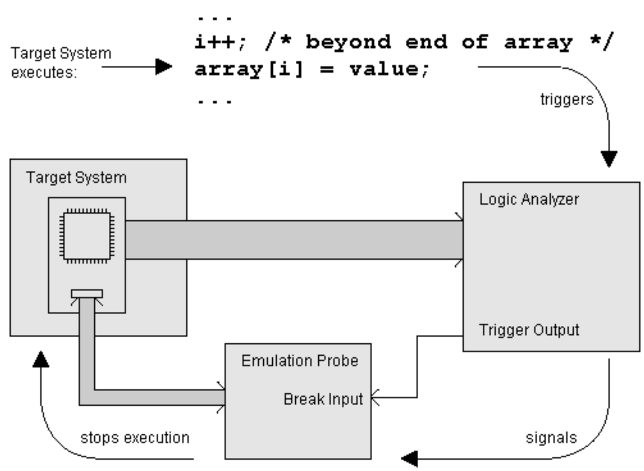


A correlated source viewer may be helpful in relating the execution to your high-level program.


Expander Card (Master: E)
Select ->
16717A
333MHz Sta
Setup...
Waveform<1>...
Timing Zoom<1>...
Listing<1>...
Source Viewer<1>...
Run


Step Source
Goto In Listing
Browse Source
Text Search
To Captured Source Line
Previous
Next
Displayed File: /logic/demo/860_demo_board/source/update_sys.c
110 temp_target = MIN_TEMP;
111 }
112 }
113
114 if (*temperature > temp_target)
115 {
116 (*temperature)--;
117 }
118 else
119 {
120 (*temperature)++;
121 }
122
123 MAKEBAR(ARG1);
124
125 MX_get_targets = 1;
126 }
127

To stop execution on a corrupt variable

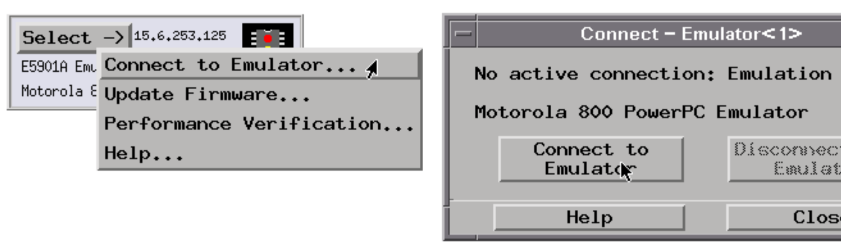


Possible uses:

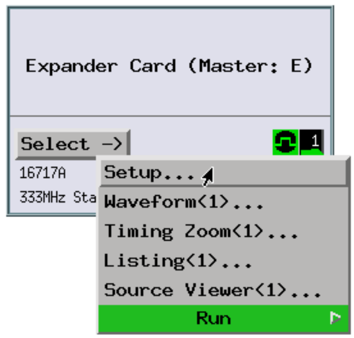
- To inspect the state of the microprocessor at some point while capturing the real-time execution that leads up to it.

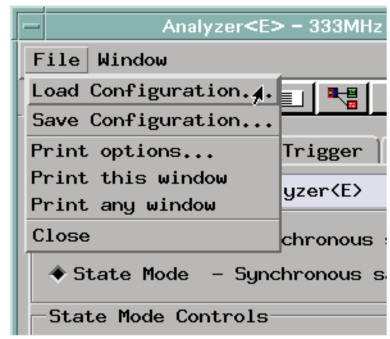
Probing the Target System

1. Use an emulation probe (either connected to a debug port in target system or connected to an analysis probe) and some kind of debugging interface (3rd party debugger, emulation control tool set, etc.).

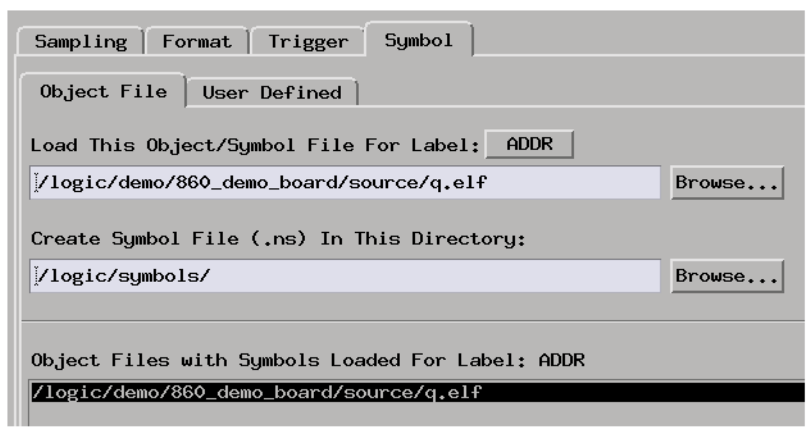


2. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

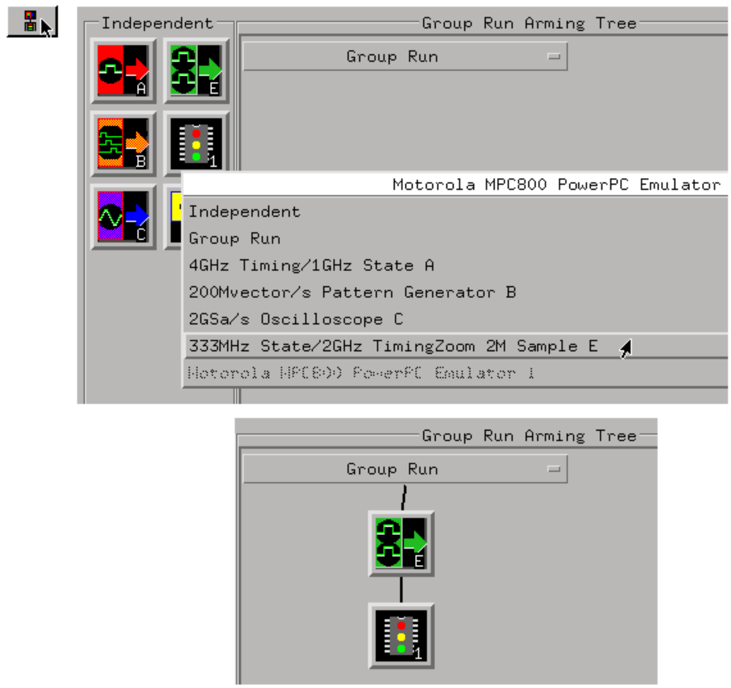




3. Load symbols from your program's object module file.

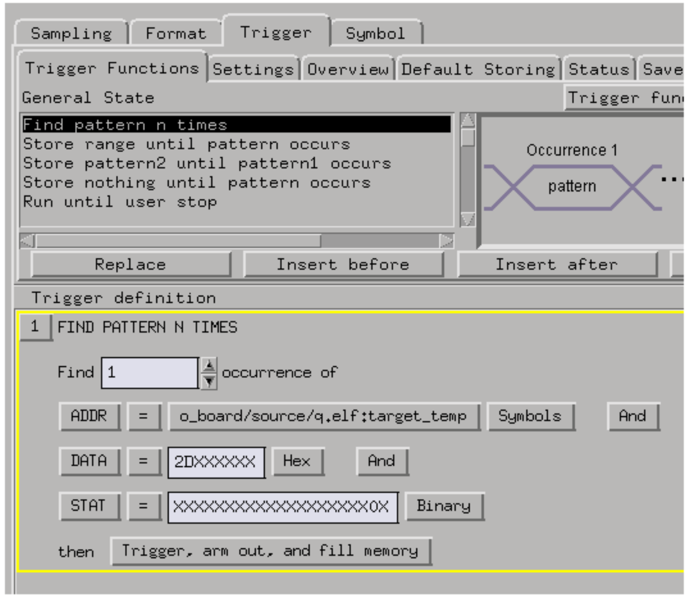


4. Open the Intermodule window and set up the emulation probe to be armed by the logic analyzer.



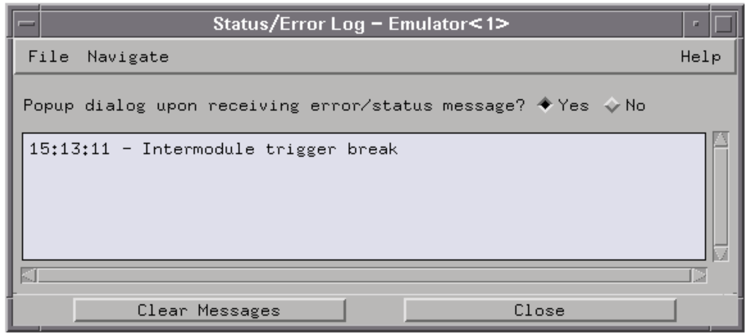
Capturing the Data

1. Set up the logic analyzer to trigger on variable address and a particular data value write.



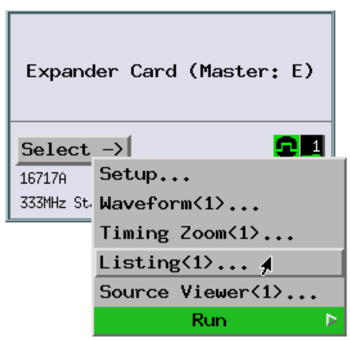
2. Select the Group Run button to start the measurement.

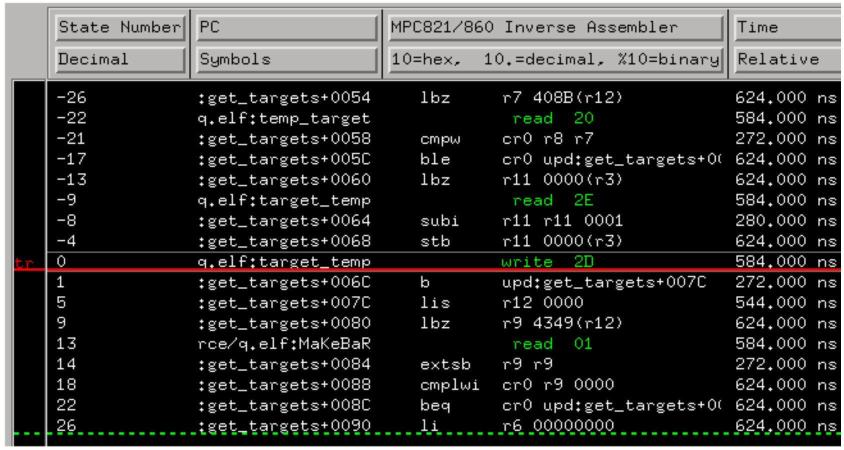
   When the logic analyzer triggers, the emulation probe stops user program execution and the processor continues to run in its background mode.



Displaying the Data

1. Use the Listing display to view the execution that leads up to the variable write. (You may have to stop the measurement before you can display the captured data.)





State Number	PC	MPC821/860 Inverse Assembler		Time
Decimal	Symbols	10=hex, 10.=decimal, %10=binary		Relative
-26	:get_targets+0054	lbz	r7 408B(r12)	624.000 ns
-22	q.elf:temp_target		read 20	584.000 ns
-21	:get_targets+0058	cmpw	cr0 r8 r7	272.000 ns
-17	:get_targets+005C	ble	cr0 upd:get_targets+0(	624.000 ns
-13	:get_targets+0060	lbz	r11 0000(r3)	624.000 ns
-9	q.elf:target_temp		read 2E	584.000 ns
-8	:get_targets+0064	subi	r11 r11 0001	280.000 ns
-4	:get_targets+0068	stb	r11 0000(r3)	624.000 ns
0	q.elf:target_temp		write 2D	584.000 ns
1	:get_targets+006C	b	upd:get_targets+007C	272.000 ns
5	:get_targets+007C	lis	r12 0000	544.000 ns
9	:get_targets+0080	lbz	r9 4349(r12)	624.000 ns
13	rce/q.elf:MaKeBaR		read 01	584.000 ns
14	:get_targets+0084	extsb	r9 r9	272.000 ns
18	:get_targets+0088	cmplwi	cr0 r9 0000	624.000 ns
22	:get_targets+008C	beq	cr0 upd:get_targets+0(	624.000 ns
26	:get_targets+0090	li	r6 00000000	624.000 ns

Note that processor execution doesn't stop immediately after the variable write and that additional cycles are executed before the processor is halted.

See Also

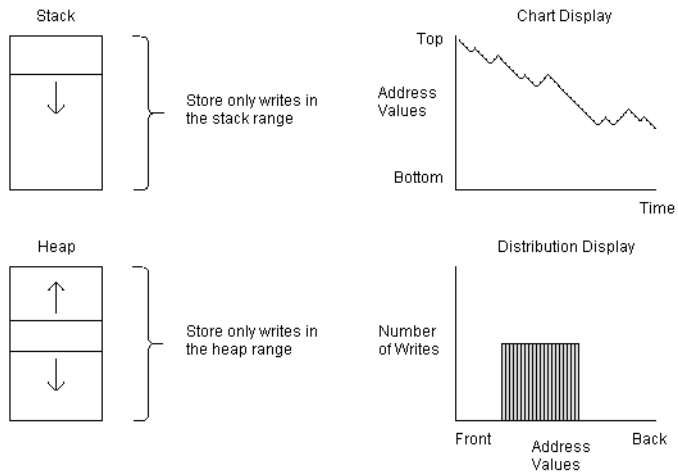
"To trace before a variable value" on page 240

"To stop execution at a source line (in ROM)" on page 226

Analyzing Real-Time Memory Usage

- "To monitor stack or heap usage" on page 251
- "To find stack overflow or guarded memory access" on page 255

To monitor stack or heap usage

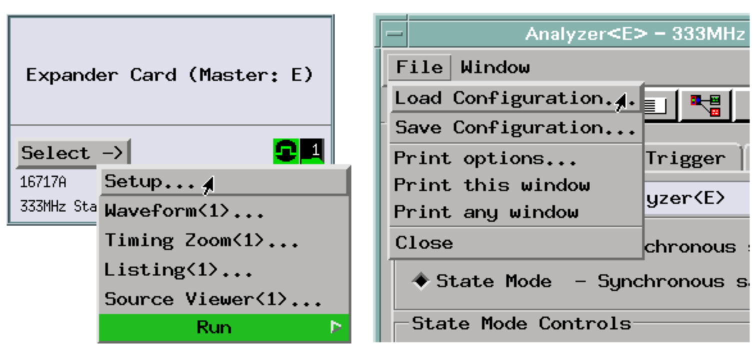


Possible uses:

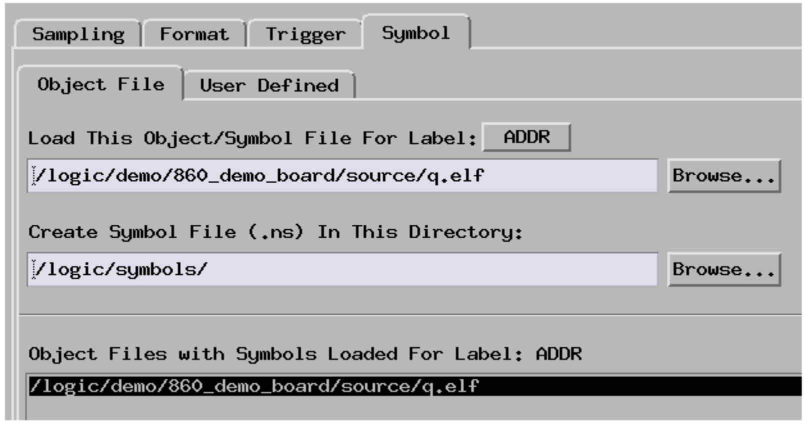
- To determine the necessary stack or heap size or view stack or heap usage.
- To look for stack or heap corruption.

Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

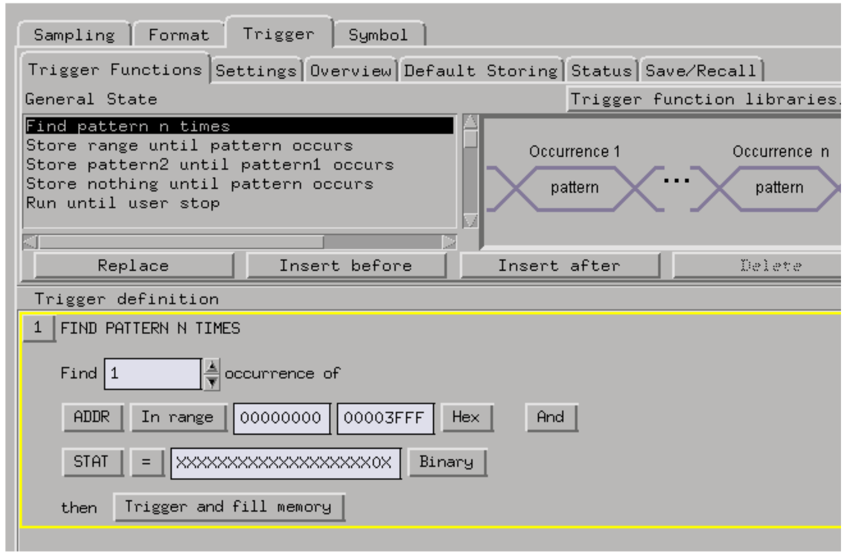


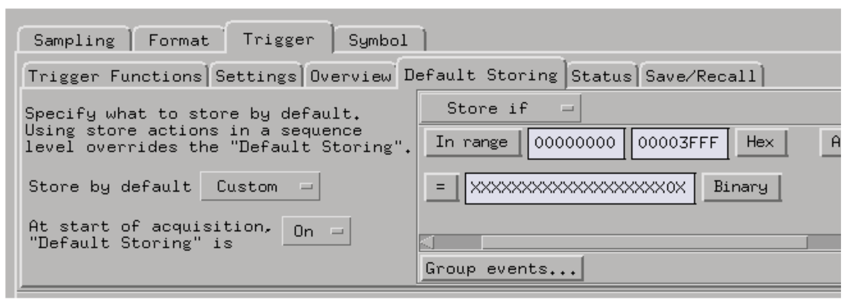
2. Load symbols from your program's object module file.



Capturing the Data

1. Set up a trigger specification that stores only writes into the range of memory addresses reserved for the stack or heap.



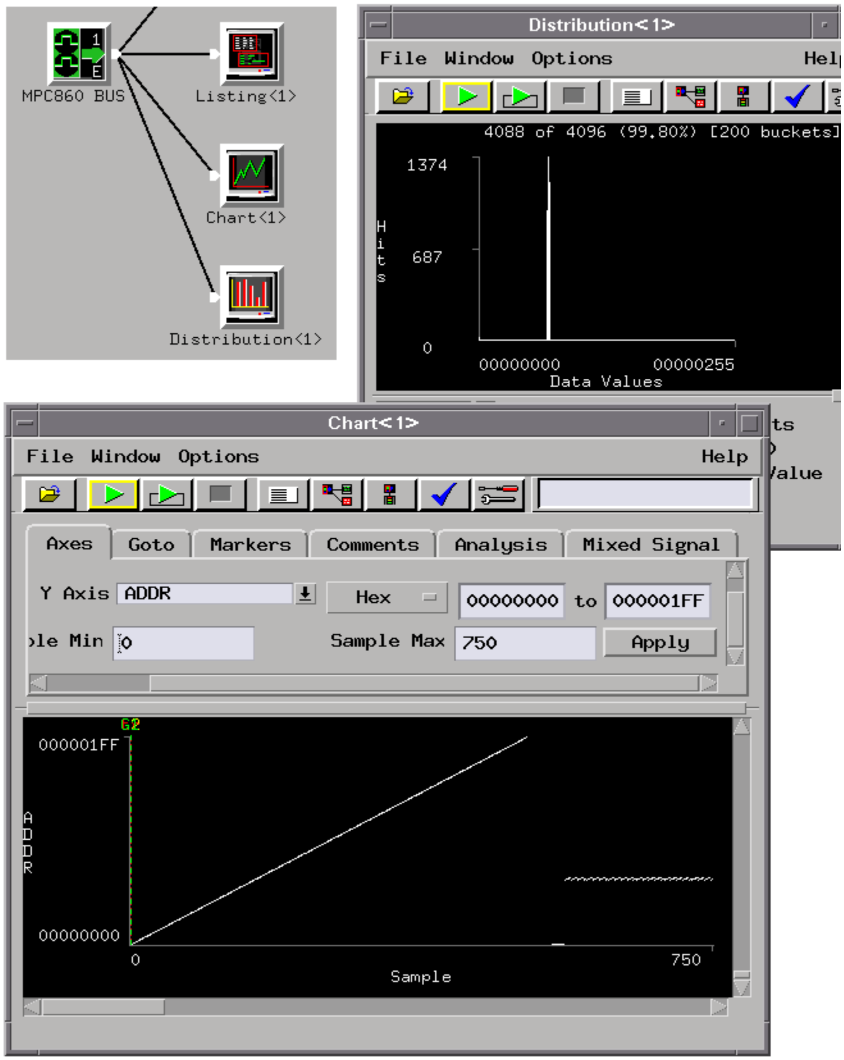


2. Select the Run button to start the measurement.

Displaying the Data

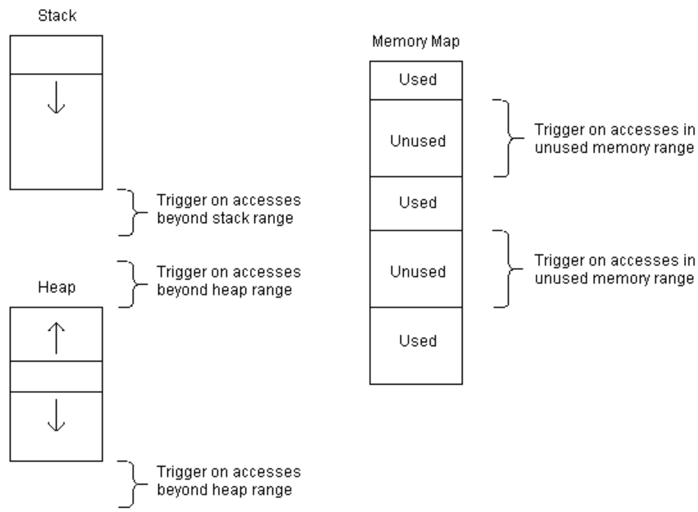
1. In the Workspace window, you can use the Chart display to view the addresses in the stack or heap range that were written to.

   You can also use the Distribution display to view the address locations that were written to and the number of number of writes that were made to each location.



See Also "To find stack overflow or guarded memory access" on page 255

To find stack overflow or guarded memory access

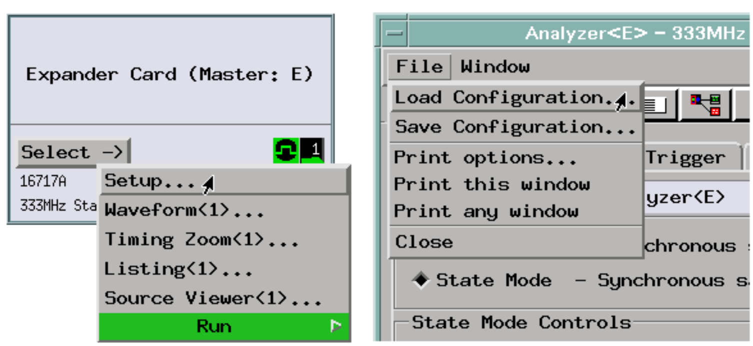


Possible uses:

- To trace execution that leads to stack or heap overflow.
- To find accesses to non-existent memory or memory that should not be accessed (in other words, guarded memory).

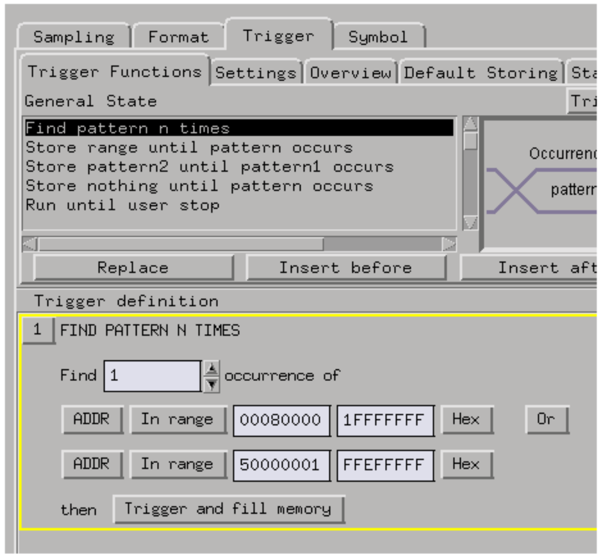
Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



Capturing the Data

1. Set up the logic analyzer to trigger on accesses outside the range of the stack or heap, or accesses of memory that does not exist, and store execution that leads up to the bad access.



2. Select the Run button to start the measurement.

Displaying the Data

1. Open the Listing display and correlated Source Viewer window to view the execution that led to the stack or heap overflow or the guarded memory access.

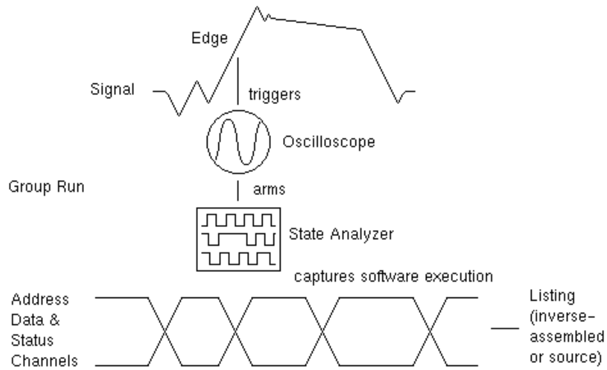
See Also

"To monitor stack or heap usage" on page 251

System Integration

Making Cross-Domain Measurements

To capture software execution when a scope triggers



Possible uses:

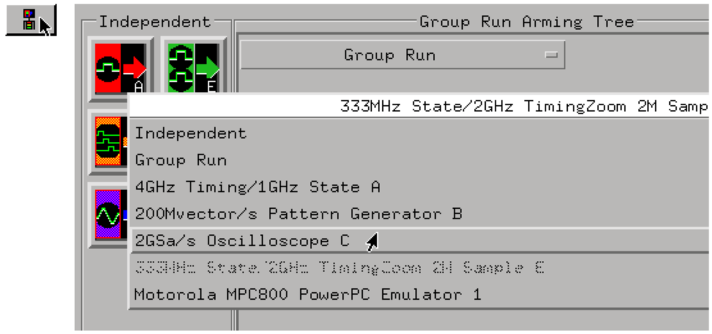
- To check whether a signal anomaly is related to software execution.

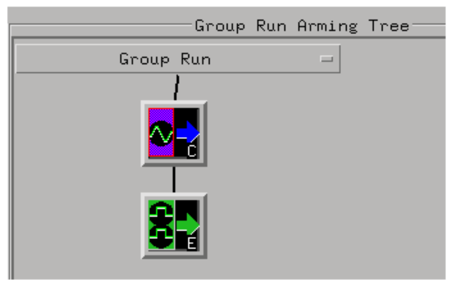
Probing the Target System

1. Connect the oscilloscope channel probe to the signal of interest in the target system.
2. Open the oscilloscope display, select the Channels tab, and set up the oscilloscope channel.
3. Configure a state analysis machine (with an analysis probe) to capture software execution (pre-defined format is included with the analysis probe).

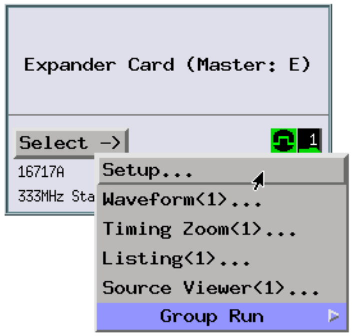
Capturing the Data

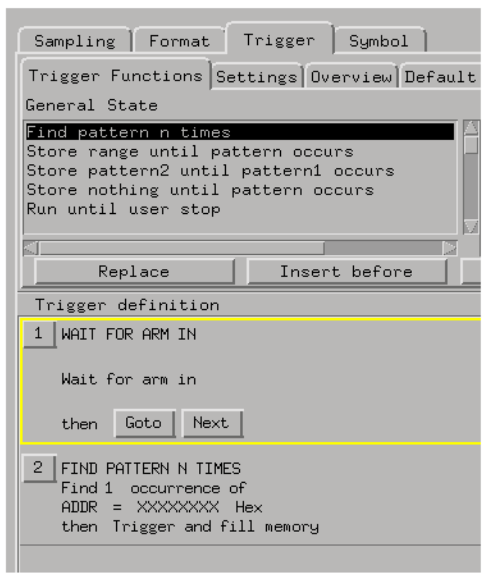
1. Set up the oscilloscope to trigger on the signal edge of interest.
2. Open the Intermodule window, and set up the logic analyzer to be armed by the oscilloscope trigger.





3. Set up the state analyzer to trigger on anything (after the arm).



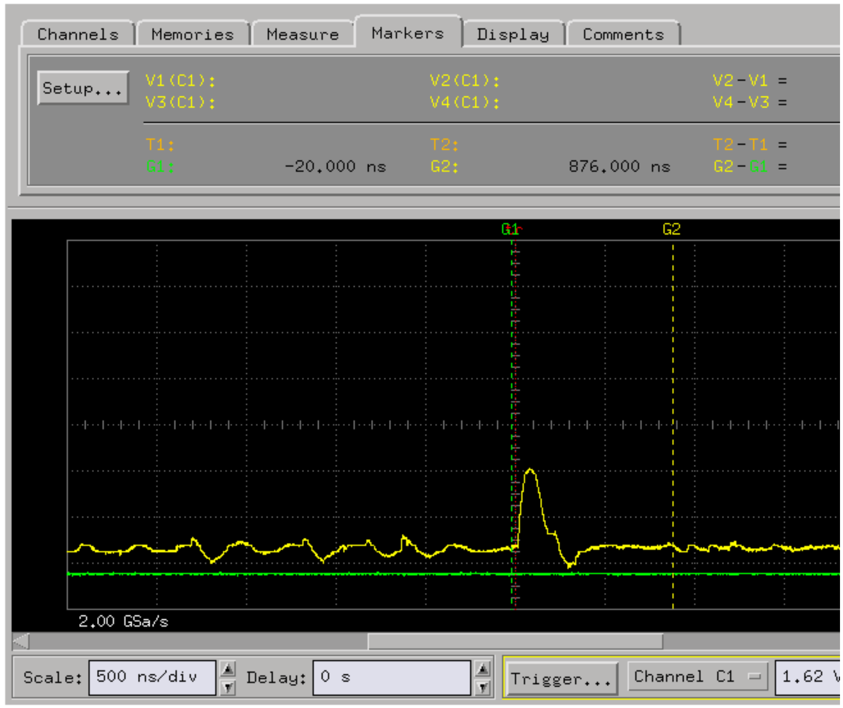


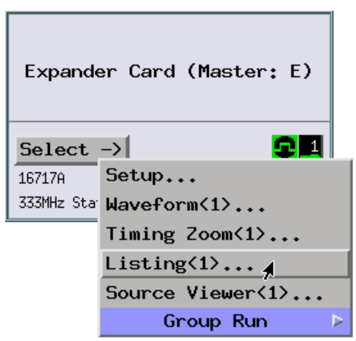
Count time in the logic analyzer so that the captured data may be correlated.

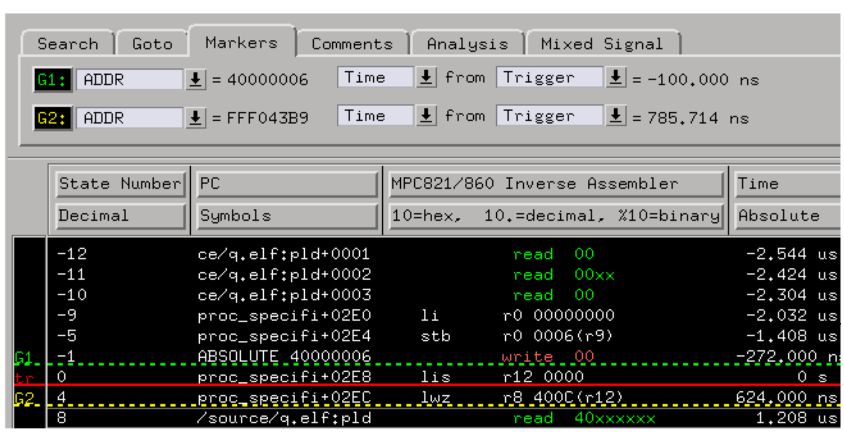
4. Select the Group Run button to start the measurement.

Displaying the Data

1. Use global markers to show the correlation between the oscilloscope trigger and the captured software execution.


Channels
Memories
Measure
Markers
Display
Comments
Setup...
V1(C1):
V3(C1):
V2(C1):
V4(C1):
V2-V1 =
V4-V3 =
T1:
G1: -20.000 ns
T2:
G2: 876.000 ns
T2-T1 =
G2-G1 =
G1
G2
2.00 GSa/s
Scale: 500 ns/div
Delay: 0 s
Trigger...
Channel C1


Expander Card (Master: E)
Select ->
16717A
333MHz Sta
Setup...
Waveform<1>...
Timing Zoom<1>...
Listing<1>...
Source Viewer<1>...
Group Run


Search
Goto
Markers
Comments
Analysis
Mixed Signal
G1: ADDR = 40000006 Time from Trigger = -100.000 ns
G2: ADDR = FFF043B9 Time from Trigger = 785.714 ns
State Number
PC
MPC821/860 Inverse Assembler
Time
Decimal
Symbols
10=hex, 10.=decimal, %10=binary
Absolute
-12 ce/q.elf:pld+0001 read 00 -2.544 us
-11 ce/q.elf:pld+0002 read 00xx -2.424 us
-10 ce/q.elf:pld+0003 read 00 -2.304 us
-9 proc_specifi+02E0 li r0 00000000 -2.032 us
-5 proc_specifi+02E4 stb r0 0006(r9) -1.408 us
G1 -1 ABSOLUTE 40000006 write 00 -272.000 ns
tr 0 proc_specifi+02E8 lis r12 0000 0 s
G2 4 proc_specifi+02EC lwz r8 400C(r12) 624.000 ns
8 /source/q.elf:pld read 40xxxxxx 1.208 us

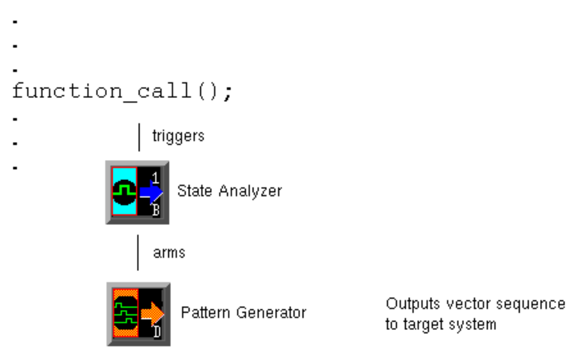
You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

You may want to open the Source Viewer window to view the source code associated with the oscilloscope trigger.

See Also

"To make basic oscilloscope measurements" on page 11

To generate patterns when a source line executes



Possible uses:

- To synchronize a sequence of target system test vectors.

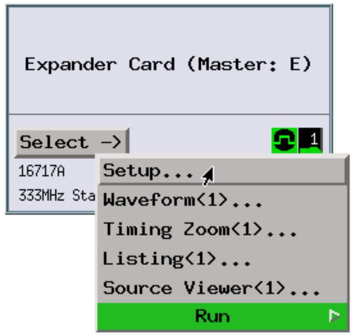
Probing the Target System

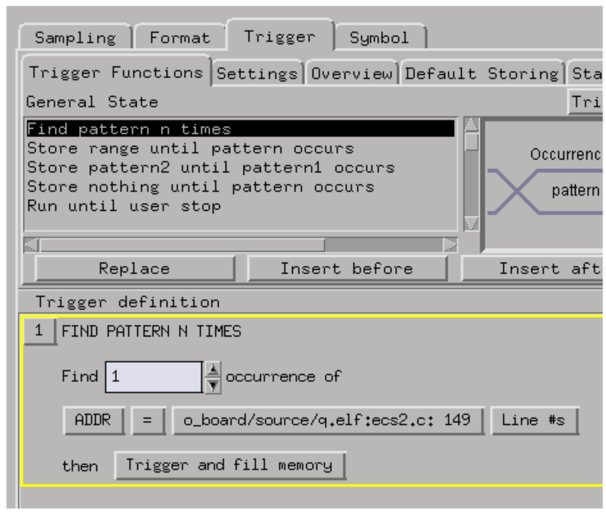
1. Configure a logic analyzer (with an analysis probe) for capturing software execution by loading the configuration file that is included with the analysis probe.

2. Select the pattern generator probing, connect the probes, map probe channels to labels, configure the vector output mode and clock source, and build a sequence of test vectors.

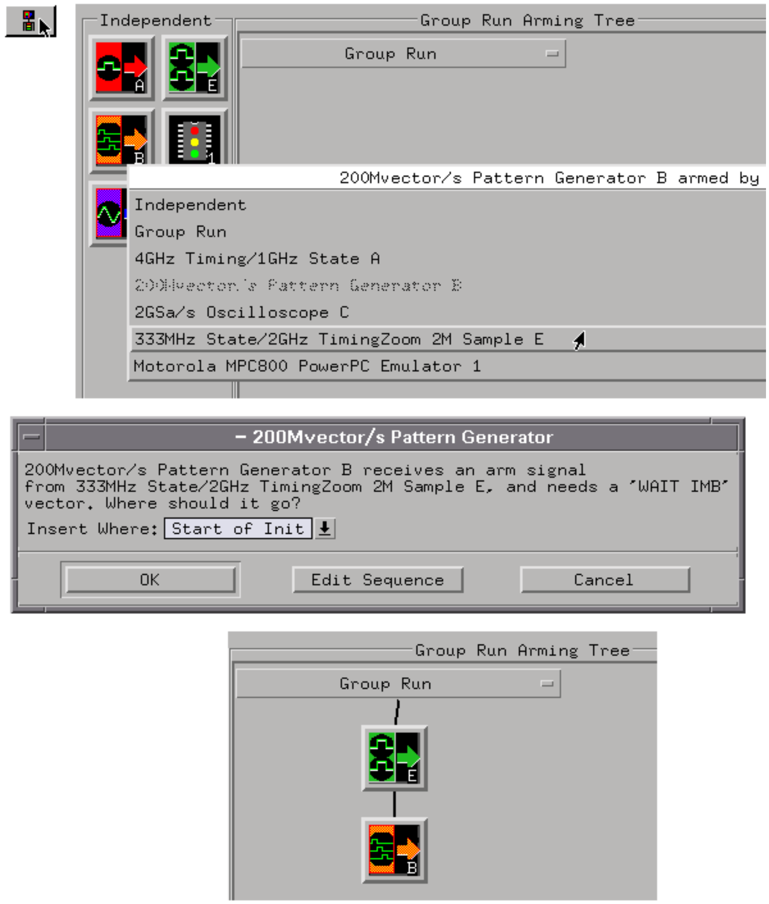
Capturing the Data

1. Set up the logic analyzer to trigger on the source line of interest.

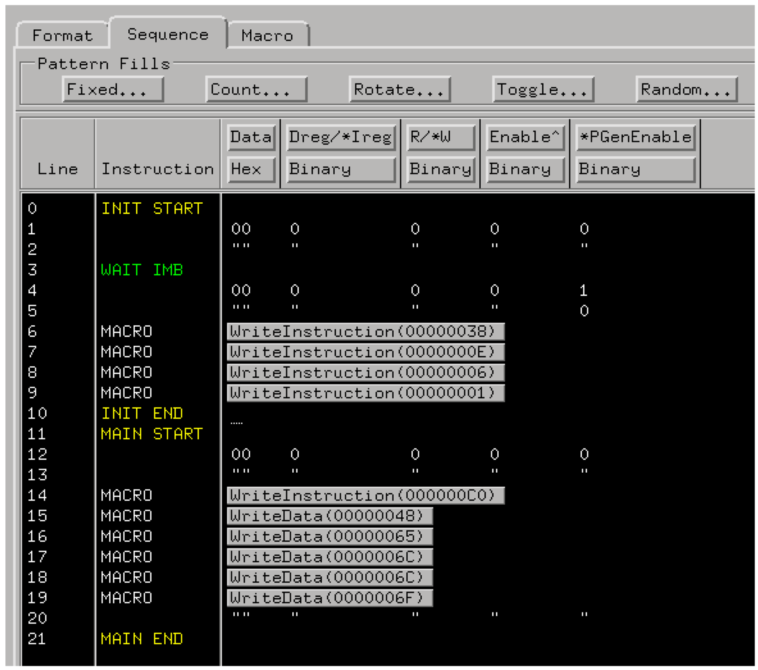




2. Open the Intermodule window, and set up the pattern generator to be armed by the logic analyzer trigger.



3. Set up the pattern generator to wait for the logic analyzer trigger before outputting its test vectors.



4. Select the Group Run button to start the measurement.

Displaying the Data

1. You can use the Listing display to show the logic analyzer states captured after the source line trigger.

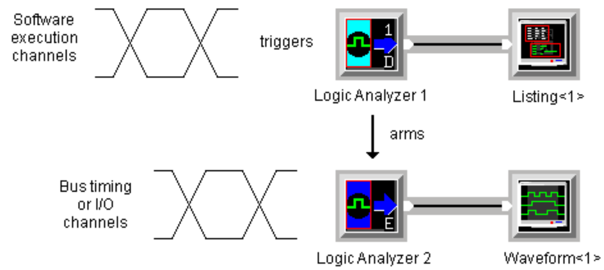
   The states captured after the trigger will show the target system's response to the pattern generator stimulus.

See Also

"To generate pattern stimulus on devices" on page 75

"To simulate particular interrupt sequences" on page 191

To arm one logic analyzer with another’s trigger



Possible uses:

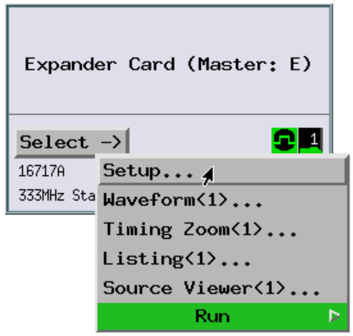
- To correlate execution in different parts of the target, for example, software execution and standard bus execution, or software execution in different parts of a multi-processor system.

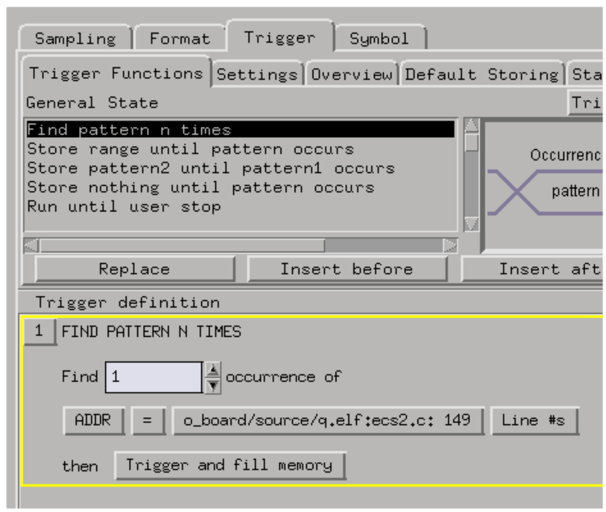
Probing the Target System

1. Configure each logic analyzer, set up the analyzer's sample clock input if it's a state analyzer, and format labels for the logic analysis channels used.

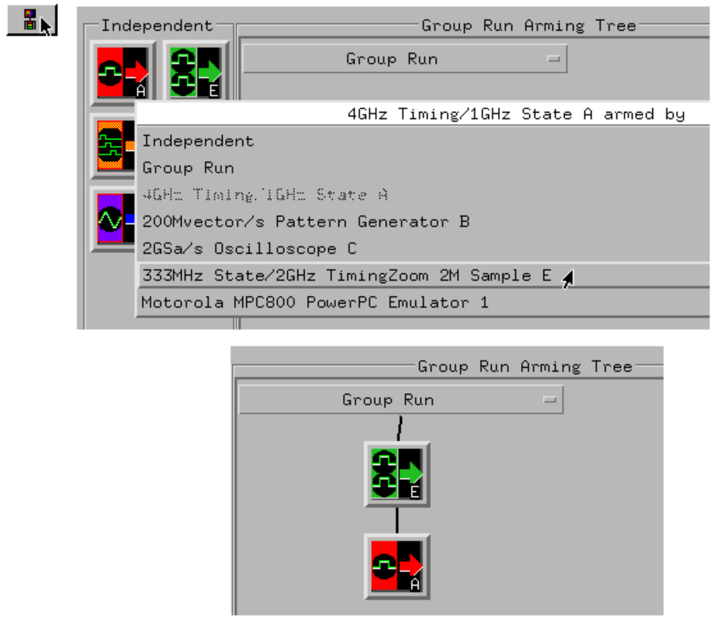
Capturing the Data

1. Set up one logic analyzer to trigger on the event of interest.

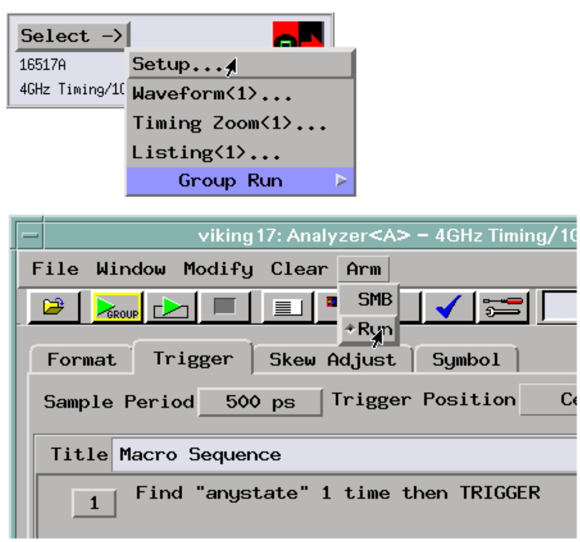




2. Open the Intermodule window, and set up one logic analyzer to be armed by the trigger of the other logic analyzer.



3. Set up the other logic analyzer's trigger in terms of the arming signal it receives from the first analyzer.

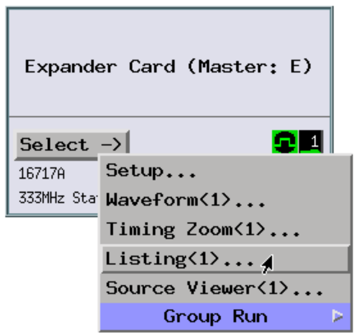


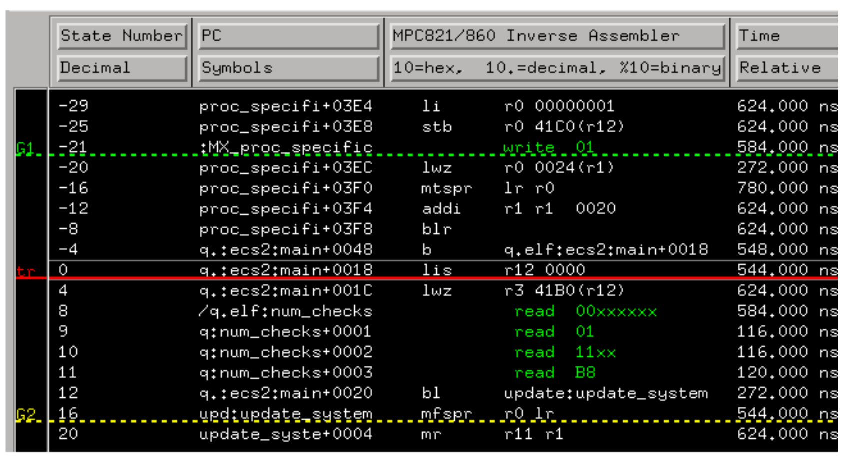
Count time in each logic analyzer so that the captured data displays may be correlated.

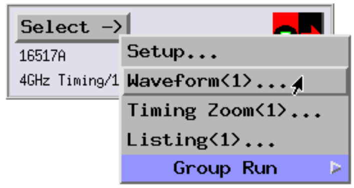
4. Select the Group Run button to start the measurement.

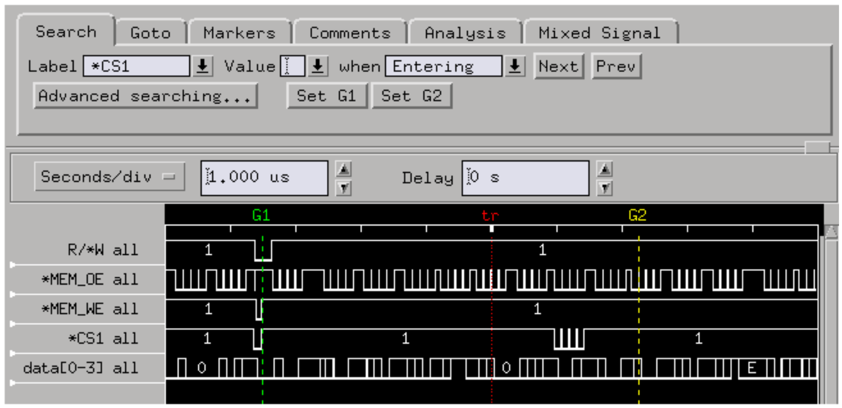
Displaying the Data

1. Use the Listing or Waveform display tools to view the data captured by each analyzer.





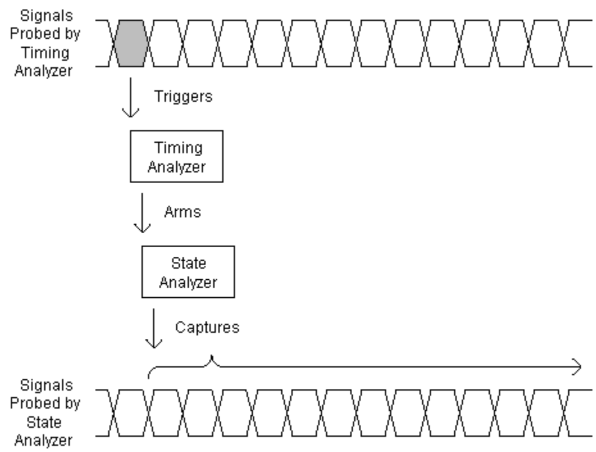




Use markers to correlate the data that was captured.

You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in both logic analyzers.

To arm a state machine with a timing machine trigger

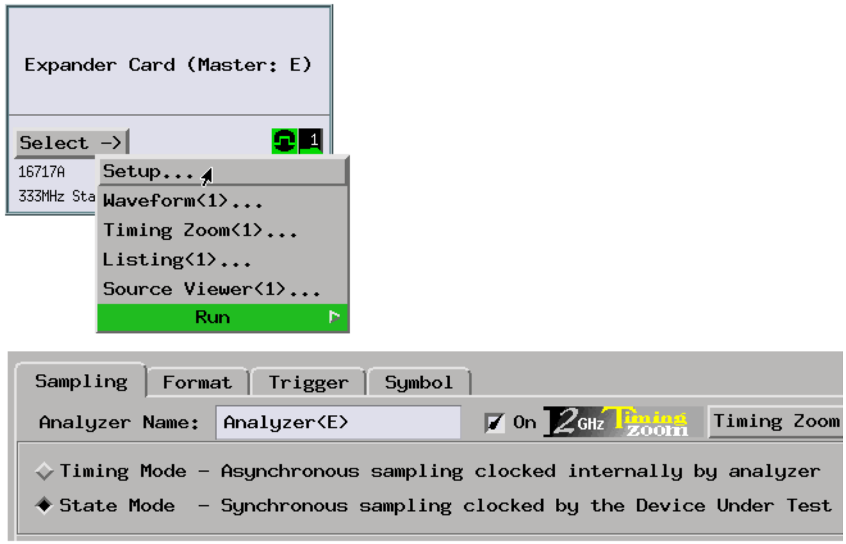


Possible uses:

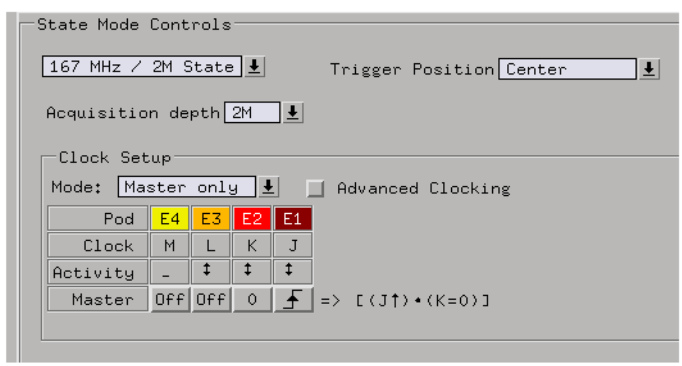
- To examine software execution when a timing violation occurs.
- To determine whether an incorrectly timed pulse is the result of a hardware defect or an incorrectly programmed counter.
- To capture software execution and correlate it with separate bus timing data or an I/O data stream.

Probing the Target System

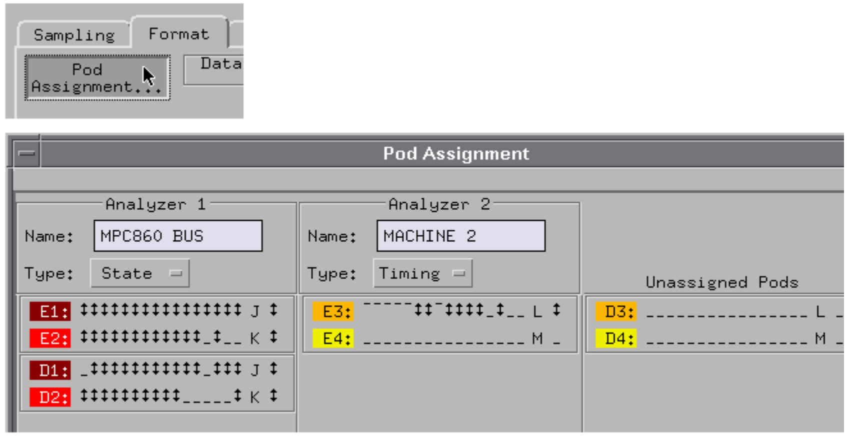
1. Configure a state analysis machine (with an analysis probe) to capture software execution.



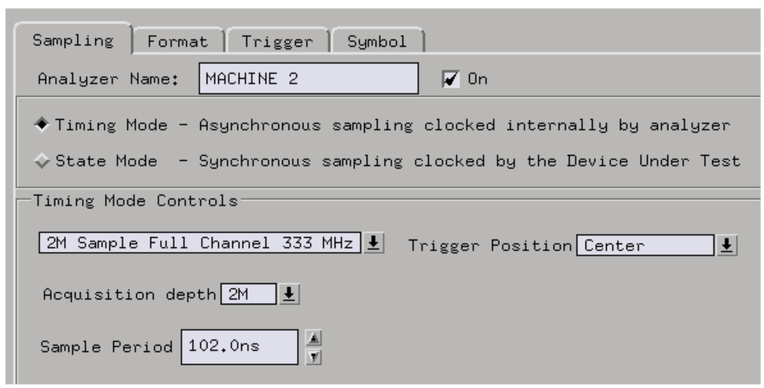
2. Select the state analyzer's clock input.



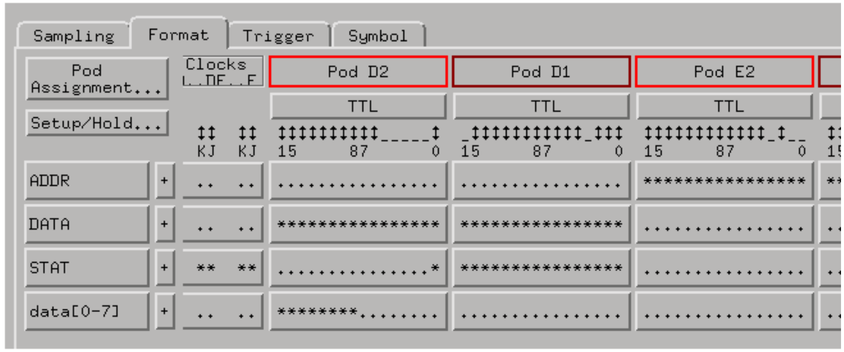
3. Assign pods. Use one logic analyzer machine for analyzing the software execution. Create another logic analysis machine for analyzing bus timing data by specifying the Analyzer 2 type.



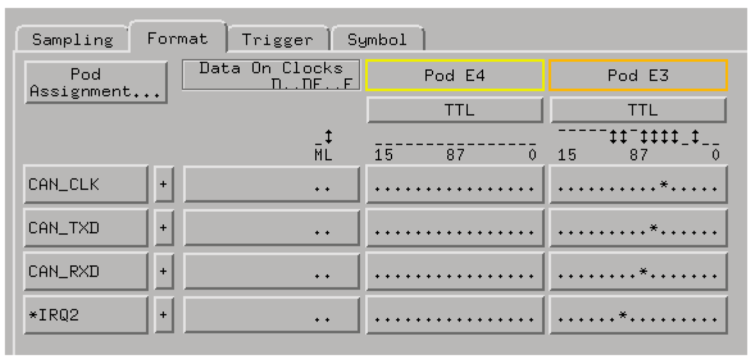
4. Specify the sampling options for the second logic analyzer machine.



5. Format state analyzer labels for the signals that capture software execution.

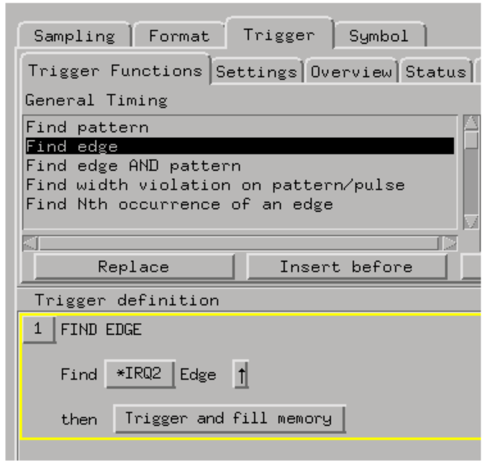


6. Format timing analyzer labels for the signals that capture bus timing data.

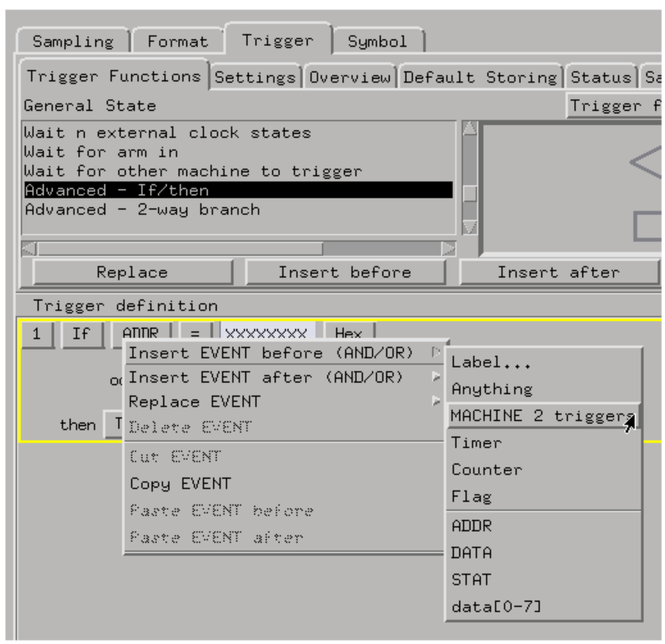


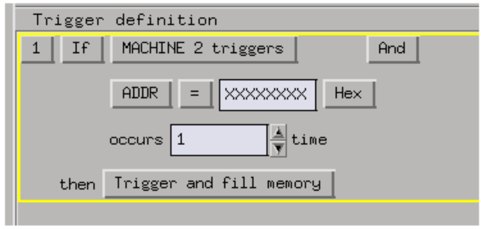
Capturing the Data

1. Set up the timing analyzer to trigger on the timing event of interest.



2. Set up the state analyzer to trigger on anything, immediately after it is armed.



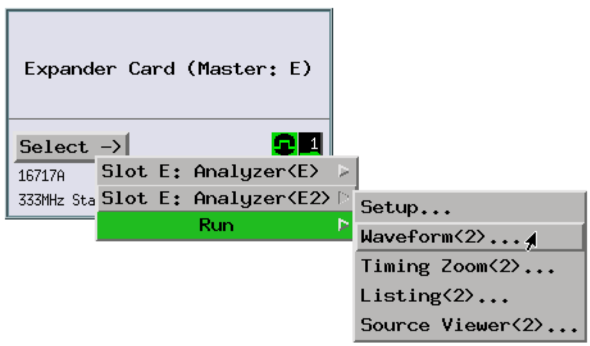


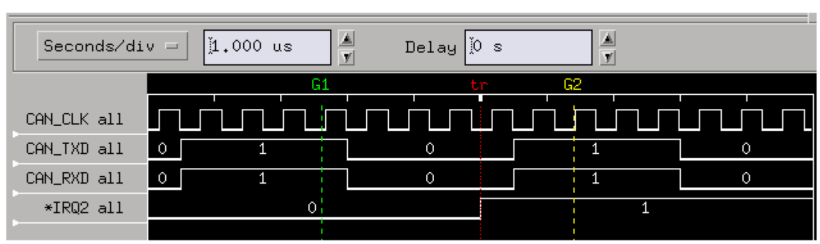
Count time in the state analyzer so that the displays may be correlated.

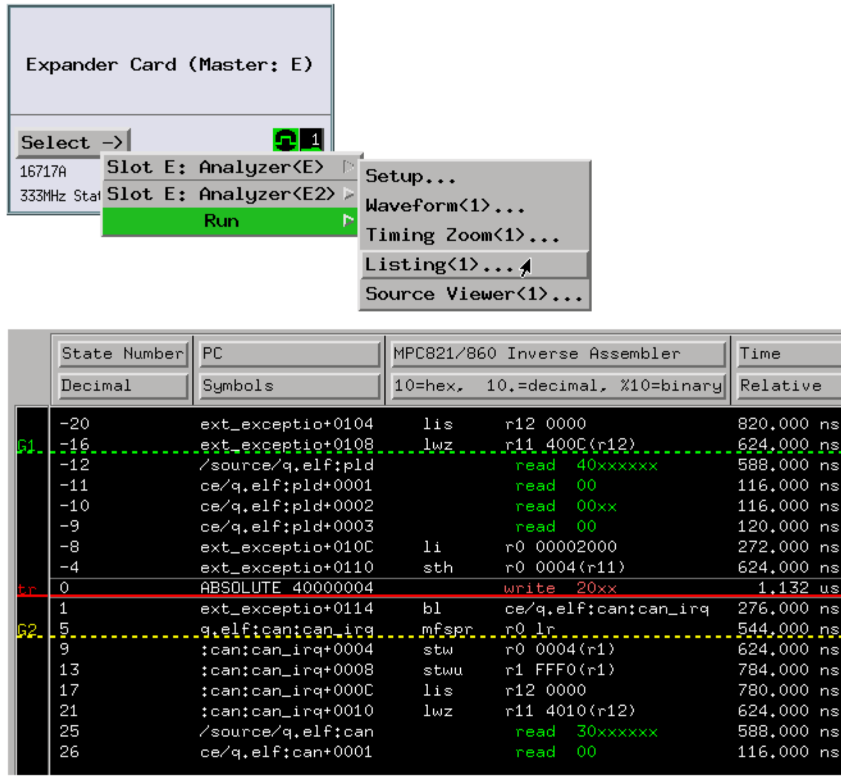
3. Select the Run button to start the measurement.

Displaying the Data

1. Use the Waveform display to show the captured bus timing and use the Listing display to show the captured software execution.

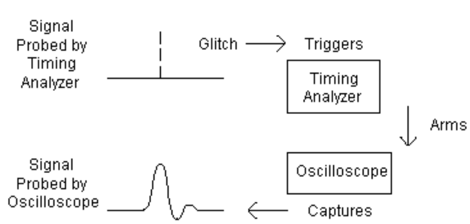






You can use markers to correlate software execution to what was captured with the timing analyzer.

To arm an oscilloscope when the analyzer triggers



Possible uses:

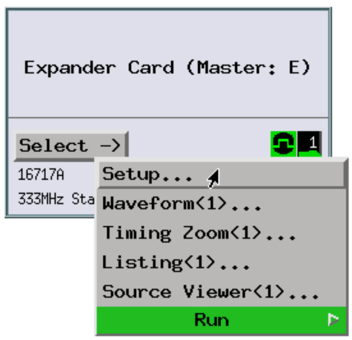
- To view the analog parameters of a glitch captured by the logic analyzer.

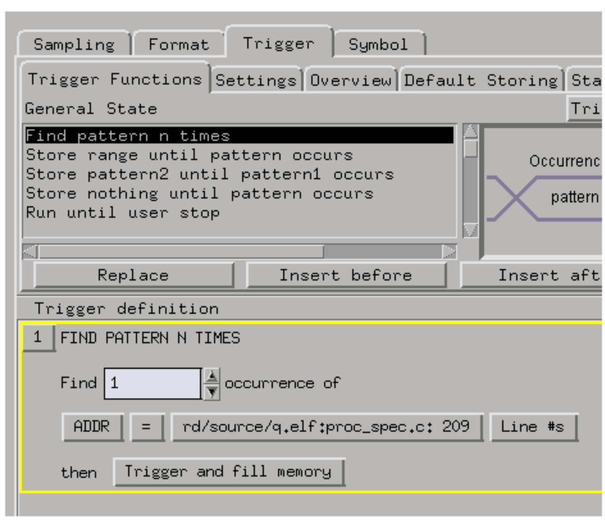
Probing the Target System

1. Configure the logic analyzer.
2. Format labels for the logic analyzer channels.
3. Connect the oscilloscope channel probes to the signals of interest in the target system.
4. Open the oscilloscope display, select the Channels tab, and set up the oscilloscope channels.

Capturing the Data

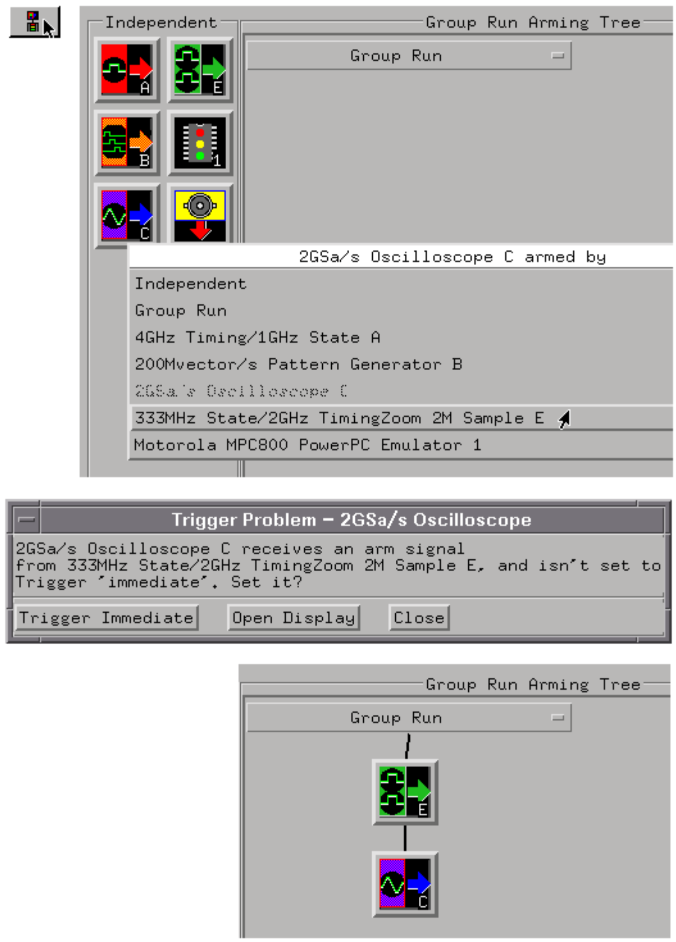
1. Set up the logic analyzer to trigger on the event of interest.



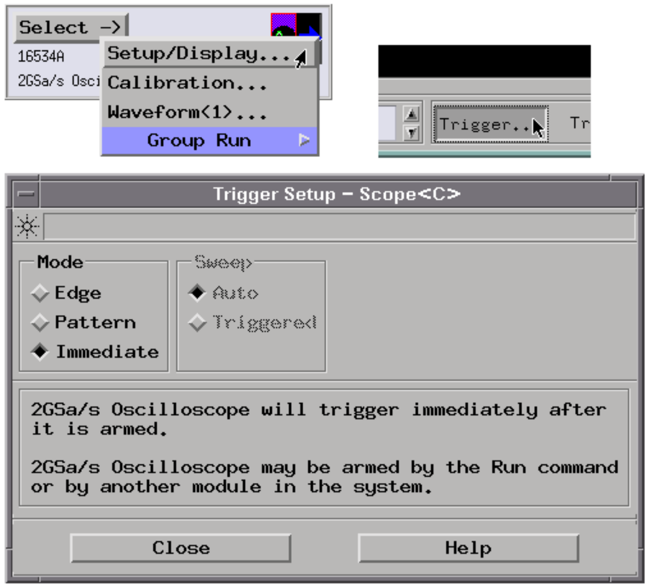


If the logic analyzer is configured as a state analyzer, be sure to count time so that the measurement displays can be correlated.

2. Open the Intermodule window, and set up the oscilloscope to be armed by the logic analyzer trigger.



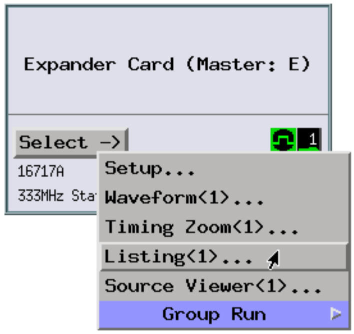
3. Set up the oscilloscope to trigger immediately (after the arm).

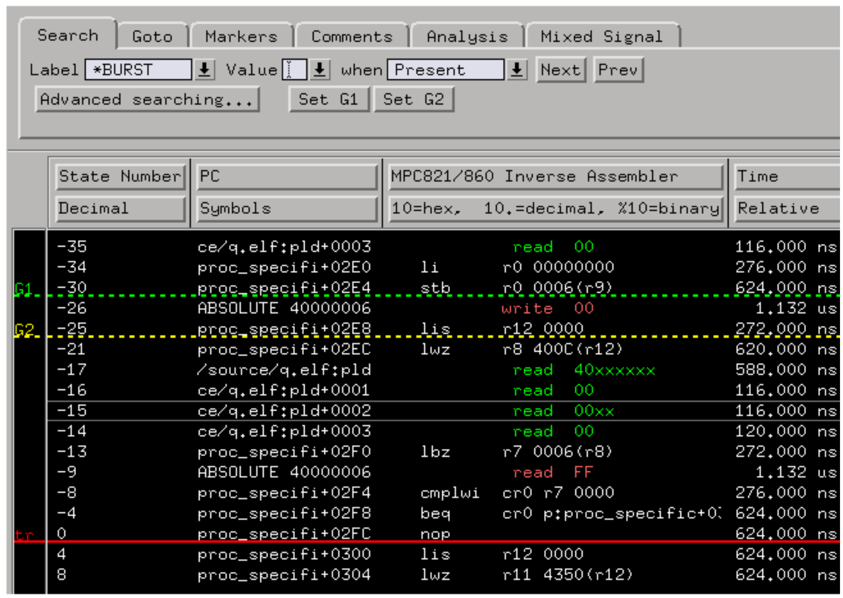


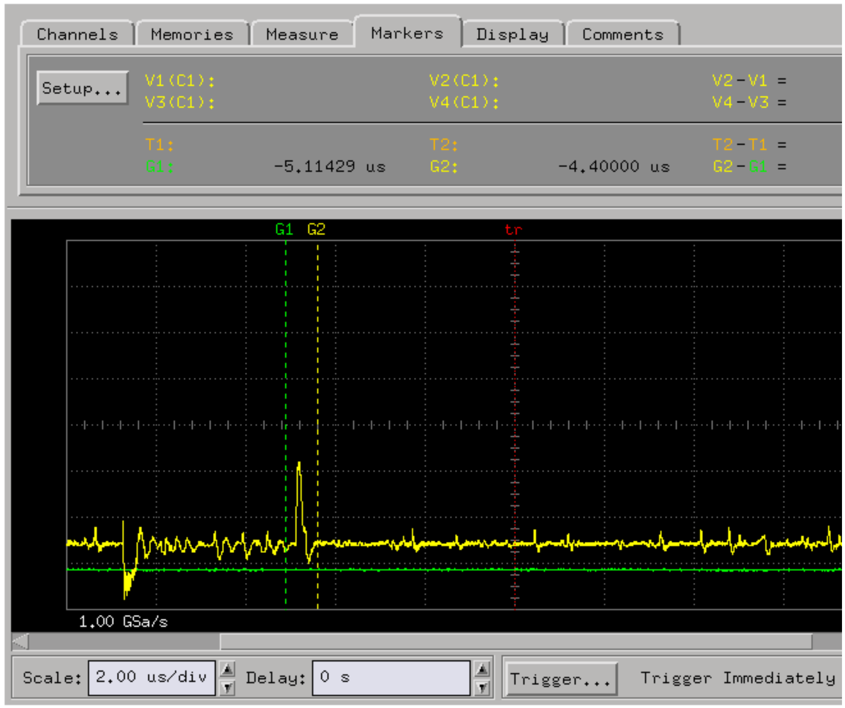
4. Select the Group Run button to start the measurement.

Displaying the Data

1. Use global markers to show the correlation between the logic analyzer trigger and the captured oscilloscope data.


Expander Card (Master: E)
Select ->
16717A
333MHz Sta
Setup...
Waveform<1>...
Timing Zoom<1>...
Listing<1>...
Source Viewer<1>...
Group Run


Search
Goto
Markers
Comments
Analysis
Mixed Signal
Label *BURST
Value
when Present
Next
Prev
Advanced searching...
Set G1
Set G2
State Number
PC
MPC821/860 Inverse Assembler
Time
Decimal
Symbols
10=hex, 10.=decimal, %10=binary
Relative
-35 ce/q.elf:pld+0003 read 00 116.000 ns
-34 proc_specifi+02E0 li r0 00000000 276.000 ns
G1 -30 proc_specifi+02E4 stb r0 0006(r9) 624.000 ns
-26 ABSOLUTE 40000006 write 00 1.132 us
G2 -25 proc_specifi+02E8 lis r12 0000 272.000 ns
-21 proc_specifi+02EC lwz r8 400C(r12) 620.000 ns
-17 /source/q.elf:pld read 40xxxxxx 588.000 ns
-16 ce/q.elf:pld+0001 read 00 116.000 ns
-15 ce/q.elf:pld+0002 read 00xx 116.000 ns
-14 ce/q.elf:pld+0003 read 00 120.000 ns
-13 proc_specifi+02F0 lbz r7 0006(r8) 272.000 ns
-9 ABSOLUTE 40000006 read FF 1.132 us
-8 proc_specifi+02F4 cmplwi cr0 r7 0000 276.000 ns
-4 proc_specifi+02F8 beq cr0 p:proc_specific+0: 624.000 ns
tr 0 proc_specifi+02FC nop 624.000 ns
4 proc_specifi+0300 lis r12 0000 624.000 ns
8 proc_specifi+0304 lwz r11 4350(r12) 624.000 ns



You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

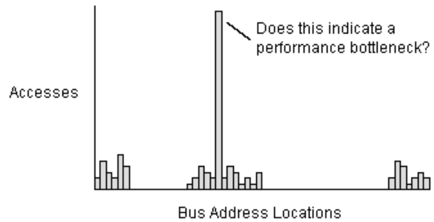
See Also

"To make basic oscilloscope measurements" on page 11

Making System Profile Measurements

- "To isolate the root cause of a performance bottleneck" on page 283
- "To simulate bus occupation and measure SW performance" on page 287

To isolate the root cause of a performance bottleneck



Requirements:

- This measurement requires the system performance analyzer (SPA) tool set.

Possible uses:

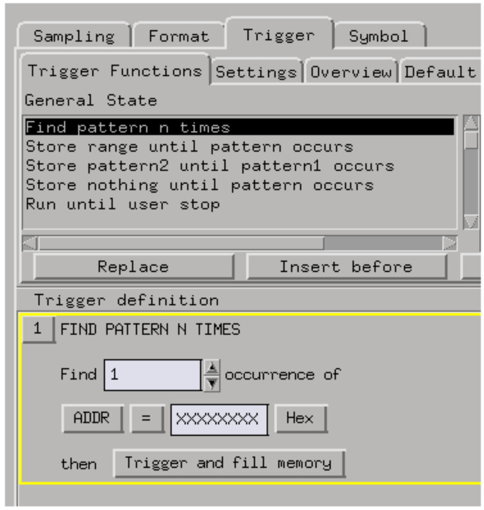
- To determine what's responsible for throughput bottlenecks.
- To detect which peripherals are most frequently used.
- To identify areas for performance improvements.
- To pinpoint regions of high memory activity.
- To measure program coverage.
- To measure stack usage.
- To isolate defects like invalid pointers.

Probing the Target System

1. Use an analysis probe to probe the microprocessor or standard bus whose performance you wish to analyze.
2. Load the configuration file included with the analysis probe to configure a state analysis machine.

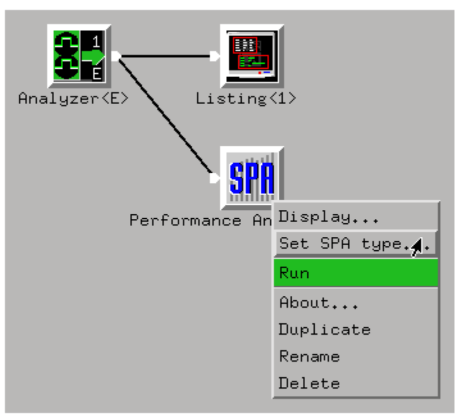
Capturing the Data

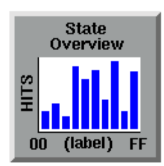
1. Set up a trigger specification to capture all bus cycles.



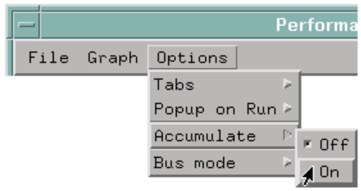
2. In the Workspace window, add the system performance analyzer to the measurement set up.

3. Use the system performance analyzer's State Overview display.





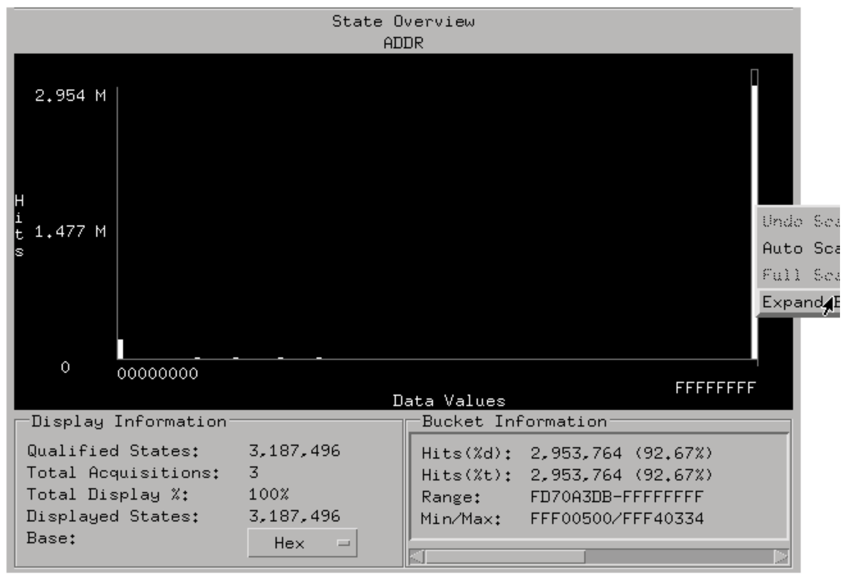
4. Set up the system performance analyzer to accumulate data and choose a repetitive run.



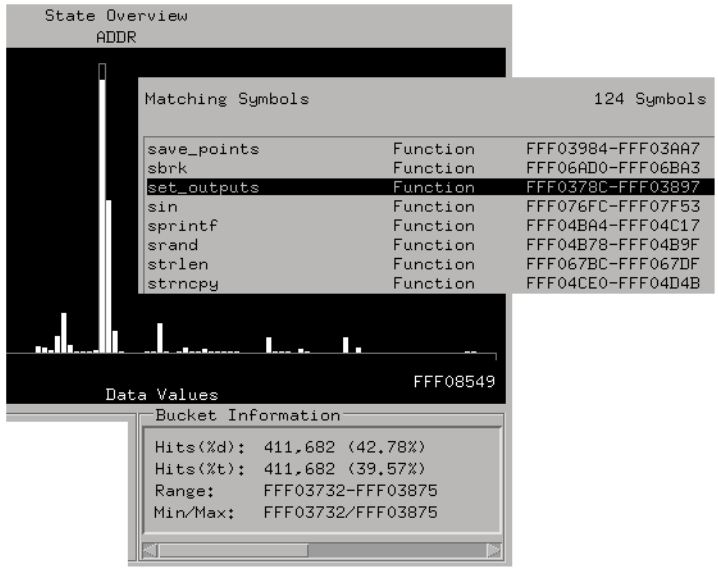
5. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.

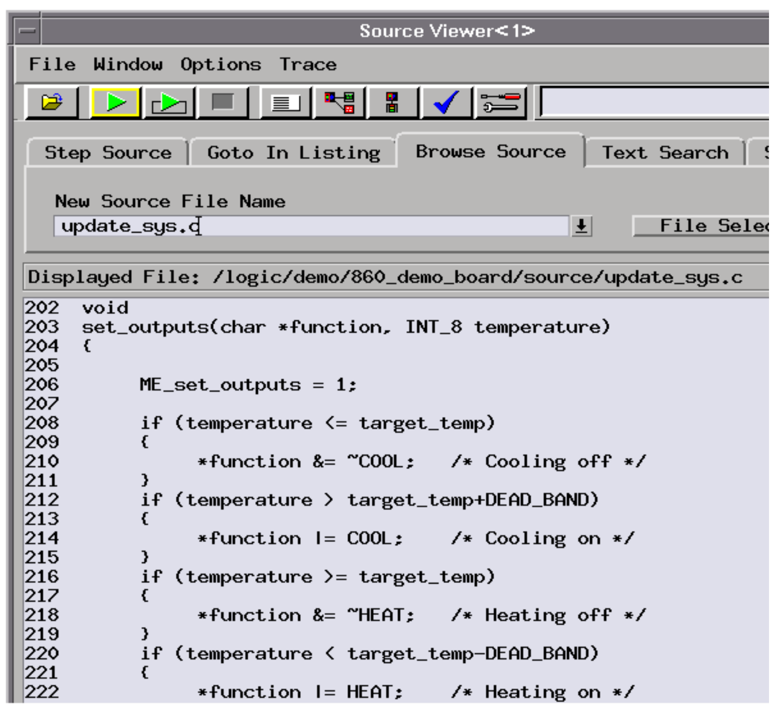
Displaying the Data

1. Expand buckets that show the most activity.

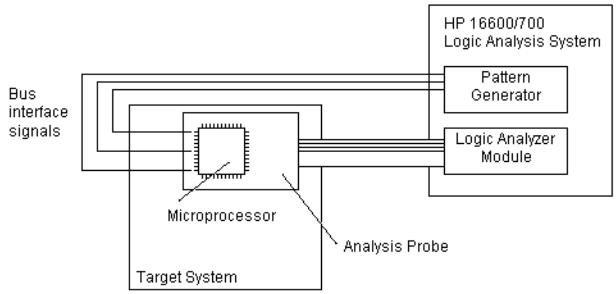


2. Select a bucket and use its information to correlate high activity to source code.


State Overview
ADDR
Matching Symbols
124 Symbols
save_points Function FFF03984-FFF03AA7
sbrk Function FFF06AD0-FFF06BA3
set_outputs Function FFF0378C-FFF03897
sin Function FFF076FC-FFF07F53
sprintf Function FFF04BA4-FFF04C17
srand Function FFF04B78-FFF04B9F
strlen Function FFF067BC-FFF067DF
strncpy Function FFF04CE0-FFF04D4B
FFF08549
Data Values
Bucket Information
Hits(%d): 411,682 (42.78%)
Hits(%t): 411,682 (39.57%)
Range: FFF03732-FFF03875
Min/Max: FFF03732/FFF03875


Source Viewer<1>
File Window Options Trace
Step Source
Goto In Listing
Browse Source
Text Search
New Source File Name
update_sys.c
File Selec
Displayed File: /logic/demo/860_demo_board/source/update_sys.c
void
set_outputs(char *function, INT_8 temperature)
{
ME_set_outputs = 1;
if (temperature <= target_temp)
{
function &= ~COOL; / Cooling off */
}
if (temperature > target_temp+DEAD_BAND)
{
function |= COOL; / Cooling on */
}
if (temperature >= target_temp)
{
function &= ~HEAT; / Heating off */
}
if (temperature < target_temp-DEAD_BAND)
{
function |= HEAT; / Heating on */

To simulate bus occupation and measure SW performance



Requirements:

- This measurement requires a pattern generator module (Agilent Technologies 16522A).
- This measurement requires the system performance analyzer (SPA) tool set.

Possible uses:

- To test how potential bus arbitration sequences can affect software performance.

Probing the Target System

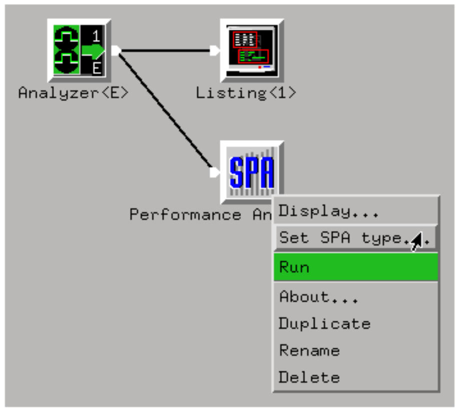
1. Connect pattern generator outputs to the appropriate bus interface signals.
2. Configure the pattern generator to output the desired sequence of bus arbitration signals.
3. Typically, you will use an analysis probe to connect the logic analyzer to the microprocessor or standard bus, and you will use the provided configuration files to configure the analyzer and define labels.

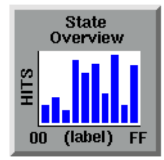
Capturing the Data

1. Set up the logic analyzer to capture all software execution as the pattern generator simulates bus arbitration sequences.

   You may want to set up an intermodule measurement to coordinate the pattern generator stimulus and the logic analyzer's capture of the response.
2. In the Workspace window, add the system performance analyzer to the measurement set up.

3. Use the system performance analyzer's State Overview display.





4. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.

Displaying the Data

1. Use the system performance analyzer's State Overview display to show which addresses have the most activity.

   You may want to expand the buckets that have the most activity and look at the bucket information to see the source code that's responsible for the activity.

See Also

"To generate pattern stimulus on devices" on page 75

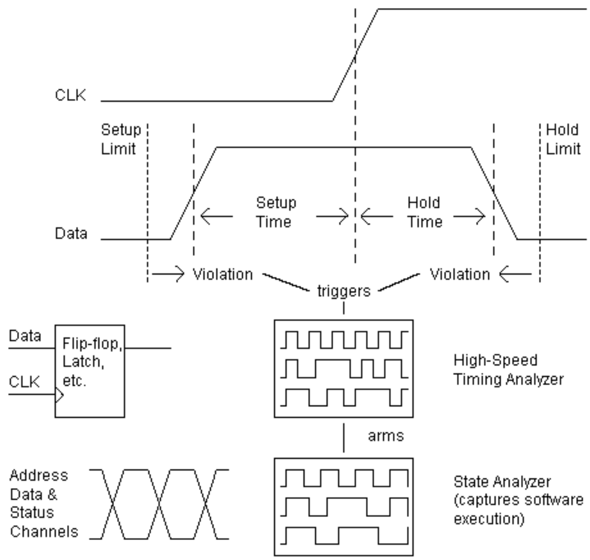
"To isolate the root cause of a performance bottleneck" on page 283

"To generate patterns when a source line executes" on page 262

Isolating Critical Defects

- "To capture SW execution on a setup or hold violation" on page 289
- "To trigger an oscilloscope when a source line executes" on page 294

To capture SW execution on a setup or hold violation



Possible uses:

- To see how setup or hold violations affect software execution.

Requirements:

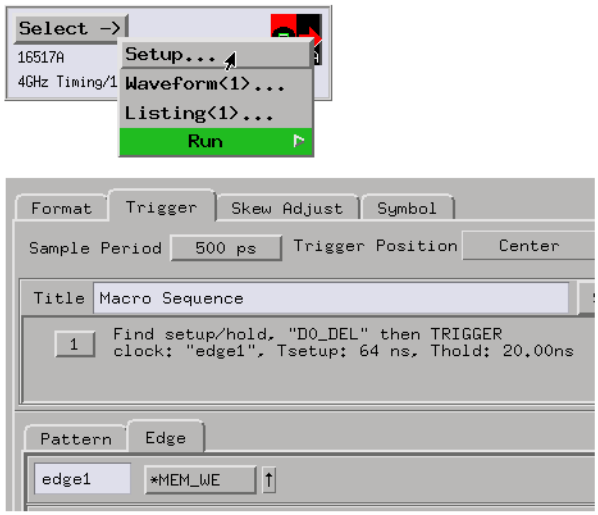
- The Agilent Technologies 16517A 4GHz Timing/1GHz State Logic Analyzer can look for setup and hold violations on multiple channels (for example, a data bus).

Probing the Target System

1. Connect the timing analyzer probes to the signals on which you are looking for a setup or hold violation.
2. Connect the state analyzer probes to the processor whose software execution you wish to capture. (Typically, you use an analysis probe to probe a processor.)
3. Configure the timing analyzer and format labels for the signals of interest.
4. Configure the state analyzer to capture software execution. (Typically, you use configuration files included with the analysis probe to configure and format labels.)

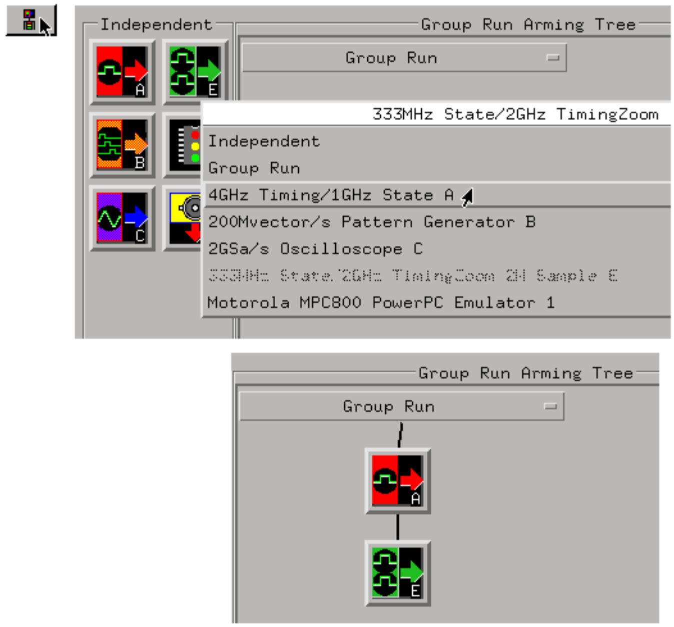
Capturing the Data

1. Set up the timing analyzer to trigger on a setup or hold violation. (The Agilent Technologies 16517A 4GHz Timing/1GHz State Logic Analyzer includes a trigger function for capturing setup or hold violations.)

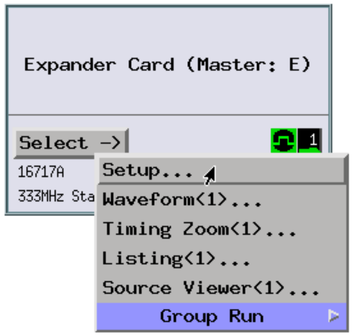


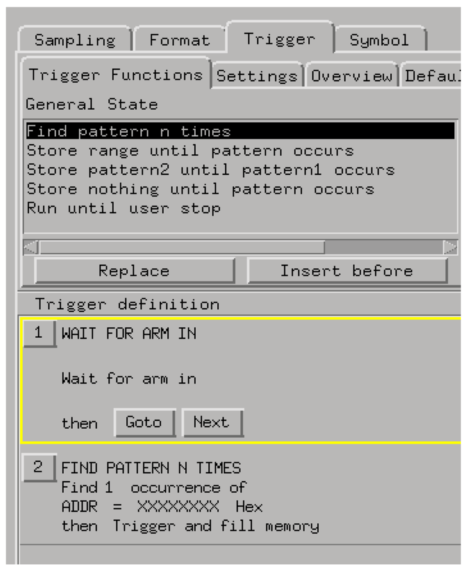
Count time in the logic analyzers so that the captured data may be correlated.

2. Open the Intermodule window, and set up the logic analyzer to be armed by the oscilloscope trigger.



3. Set up the state analyzer to trigger on anything (after the arm).

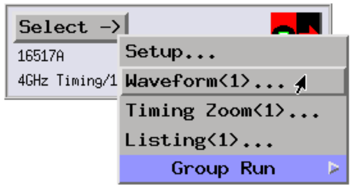


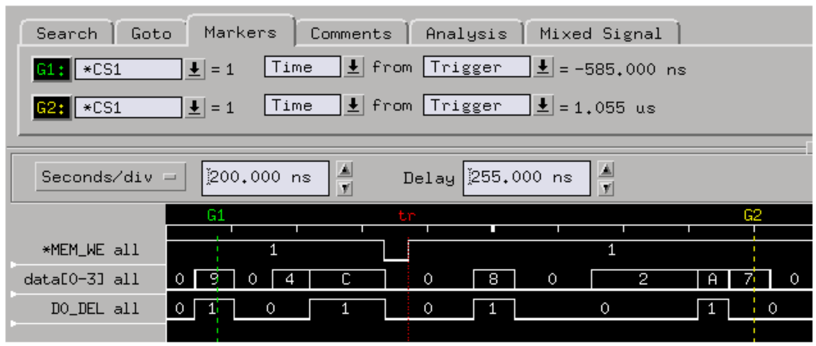


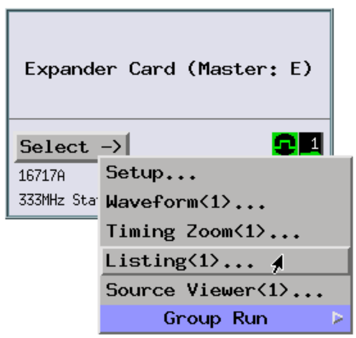
4. Select the Group Run button to start the measurement.

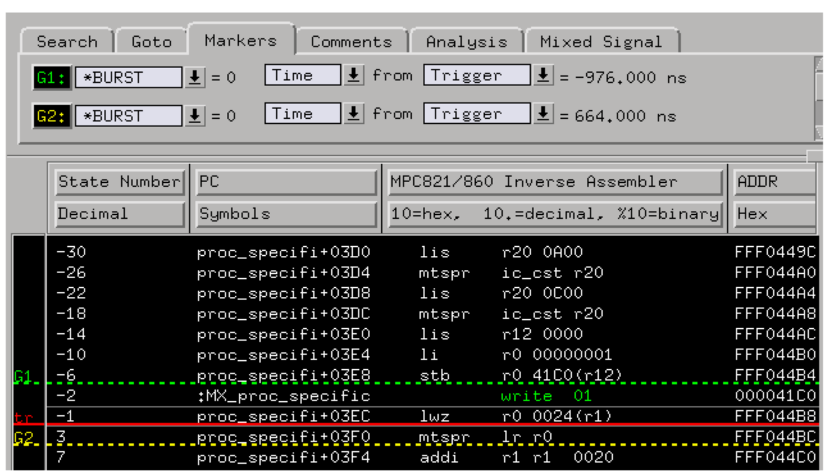
Displaying the Data

1. Use global markers to show the correlation between the timing analyzer trigger and the captured software execution.


Select ->
16517A
4GHz Timing/1
Setup...
Waveform<1>...
Timing Zoom<1>...
Listing<1>...
Group Run


Search
Goto
Markers
Comments
Analysis
Mixed Signal
G1: *CS1 = 1 Time from Trigger = -585.000 ns
G2: *CS1 = 1 Time from Trigger = 1.055 us
Seconds/div
200.000 ns
Delay
255.000 ns
G1
tr
G2
*MEM_WE all
data[0-3] all
DO_DEL all


Expander Card (Master: E)
Select ->
16717A
333MHz Sta
Setup...
Waveform<1>...
Timing Zoom<1>...
Listing<1>...
Source Viewer<1>...
Group Run


Search
Goto
Markers
Comments
Analysis
Mixed Signal
G1: *BURST = 0 Time from Trigger = -976.000 ns
G2: *BURST = 0 Time from Trigger = 664.000 ns
State Number
Decimal
PC
Symbols
MPC821/860 Inverse Assembler
10=hex, 10.=decimal, %10=binary
ADDR
Hex
-30 proc_specifi+03D0 lis r20 0A00 FFF0449C
-26 proc_specifi+03D4 mtspr ic_cst r20 FFF044A0
-22 proc_specifi+03D8 lis r20 0C00 FFF044A4
-18 proc_specifi+03DC mtspr ic_cst r20 FFF044A8
-14 proc_specifi+03E0 lis r12 0000 FFF044AC
-10 proc_specifi+03E4 li r0 00000001 FFF044B0
G1 -6 proc_specifi+03E8 stb r0 41C0(r12) FFF044B4
-2 :MX_proc_specific write 01 000041C0
tr -1 proc_specifi+03EC lwz r0 0024(r1) FFF044B8
G2 3 proc_specifi+03F0 mtspr lr r0 FFF044BC
7 proc_specifi+03F4 addi r1 r1 0020 FFF044C0

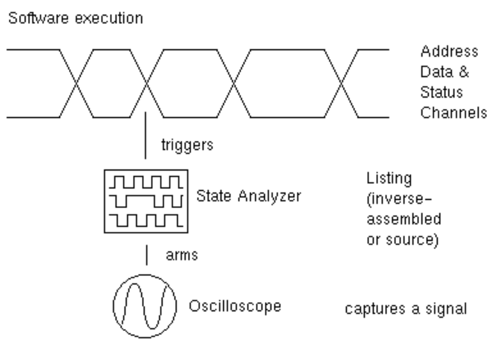
You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

You may want to open the Source Viewer window to view the source code associated with the timing analyzer trigger.

See Also

"To find setup and hold violations" on page 56

To trigger an oscilloscope when a source line executes



Possible uses:

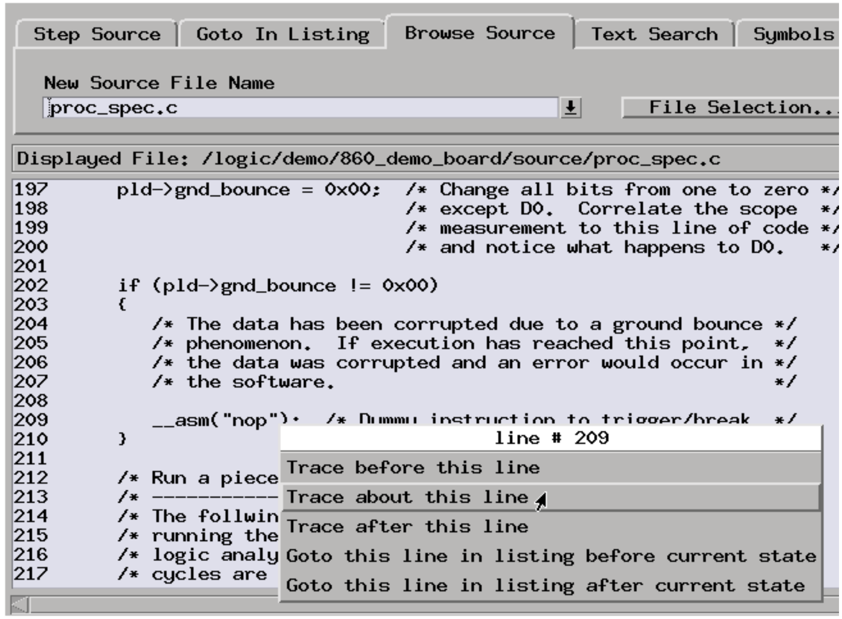
- To see the effect of a certain type of software execution on signals. (For example, do certain bus value changes cause ground bounce?)

Probing the Target System

1. Configure a state analysis machine (with an analysis probe) to capture software execution (pre-defined format is included with the analysis probe).
2. Connect the oscilloscope channel probes to the signals of interest in the target system.
3. Open the oscilloscope display, select the Channels tab, and set up the oscilloscope channels.

Capturing the Data

1. Use the Source Viewer to set up a trigger on the software execution of interest.

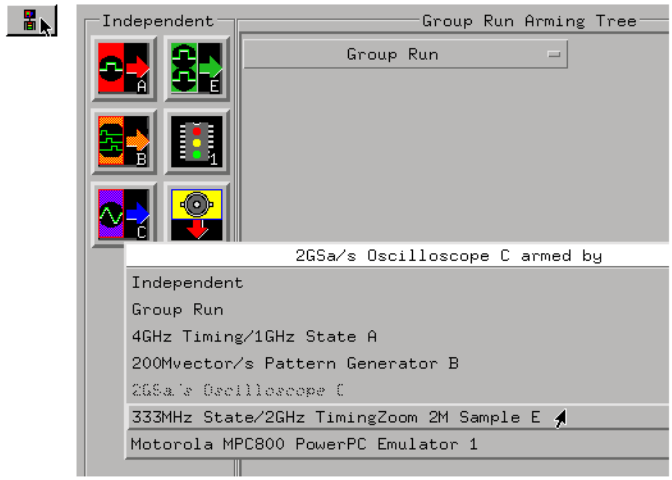


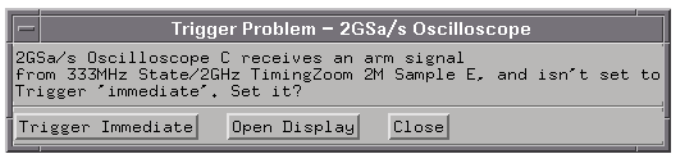
NOTE: Source Viewer commands that set up triggers only modify the trigger condition. They do not modify the trigger position, storage qualifiers, `else` branch conditions, or other levels in the trigger sequence.

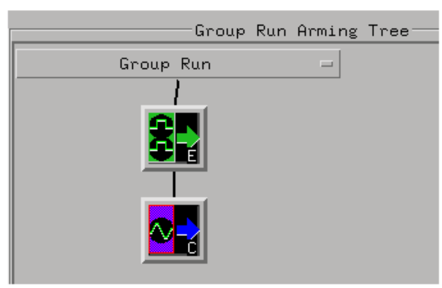
Count time in the logic analyzer so that the captured data may be correlated.

A dialog will inform you when the trigger has been set.

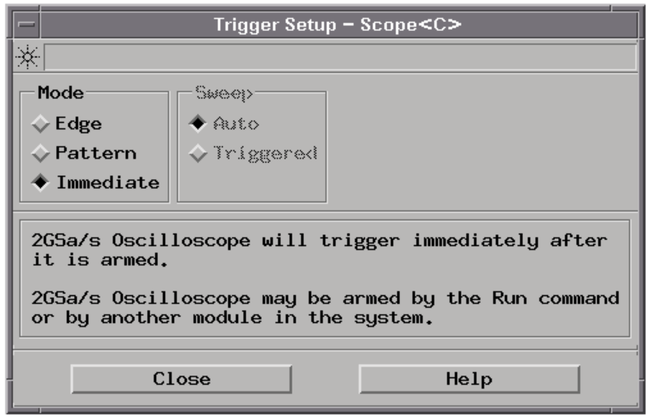
2. Open the Intermodule window, and set up the oscilloscope to be armed by the logic analyzer trigger.







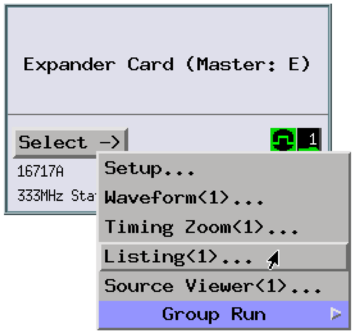
3. Set up the oscilloscope to trigger immediately (after the arm).

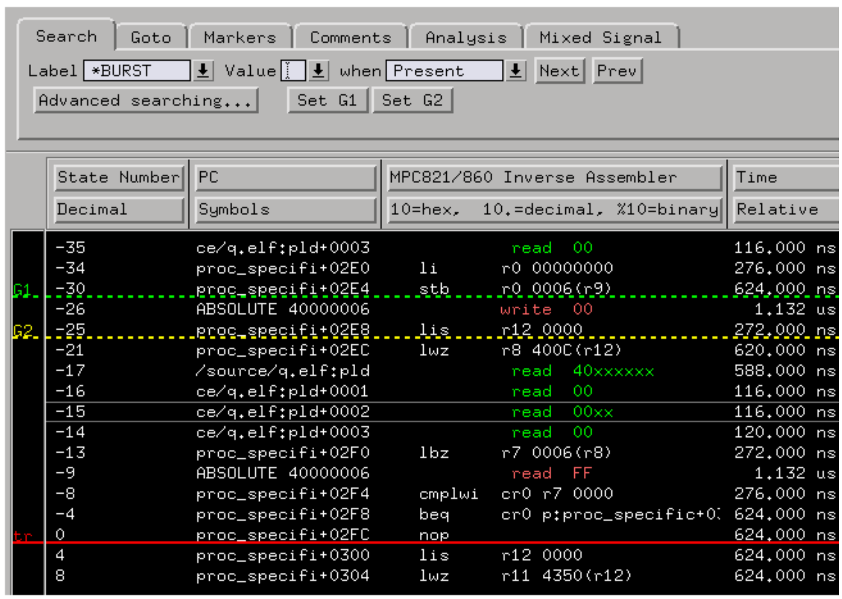


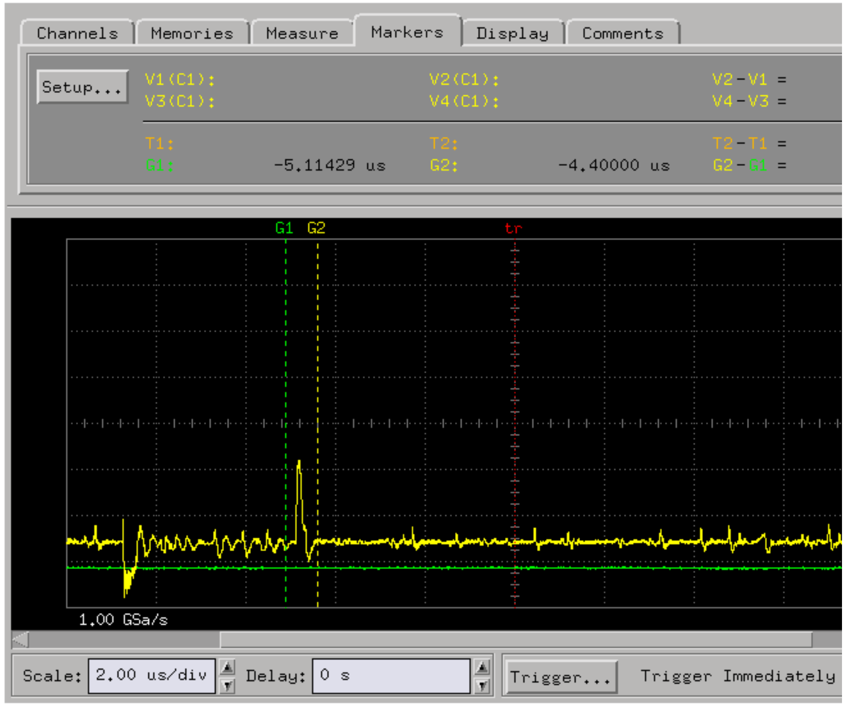
4. Select the Group Run button to start the measurement.

Displaying the Data

1. Use global markers to show the correlation between the logic analyzer trigger and the captured oscilloscope data.


Expander Card (Master: E)
Select ->
16717A
333MHz Sta
Setup...
Waveform<1>...
Timing Zoom<1>...
Listing<1>...
Source Viewer<1>...
Group Run


Search
Goto
Markers
Comments
Analysis
Mixed Signal
Label *BURST
Value
when Present
Next
Prev
Advanced searching...
Set G1
Set G2
State Number
Decimal
PC
Symbols
MPC821/860 Inverse Assembler
10=hex, 10.=decimal, %10=binary
Time
Relative
-35 ce/q.elf:pld+0003 read 00 116.000 ns
-34 proc_specifi+02E0 li r0 00000000 276.000 ns
G1 -30 proc_specifi+02E4 stb r0 0006(r9) 624.000 ns
-26 ABSOLUTE 40000006 write 00 1.132 us
G2 -25 proc_specifi+02E8 lis r12 0000 272.000 ns
-21 proc_specifi+02EC lwz r8 400C(r12) 620.000 ns
-17 /source/q.elf:pld read 40xxxxxx 588.000 ns
-16 ce/q.elf:pld+0001 read 00 116.000 ns
-15 ce/q.elf:pld+0002 read 00xx 116.000 ns
-14 ce/q.elf:pld+0003 read 00 120.000 ns
-13 proc_specifi+02F0 lbz r7 0006(r8) 272.000 ns
-9 ABSOLUTE 40000006 read FF 1.132 us
-8 proc_specifi+02F4 cmplwi cr0 r7 0000 276.000 ns
-4 proc_specifi+02F8 beq cr0 p:proc_specific+0 624.000 ns
tr 0 proc_specifi+02FC nop 624.000 ns
4 proc_specifi+0300 lis r12 0000 624.000 ns
8 proc_specifi+0304 lwz r11 4350(r12) 624.000 ns



You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

See Also

"To make basic oscilloscope measurements" on page 11

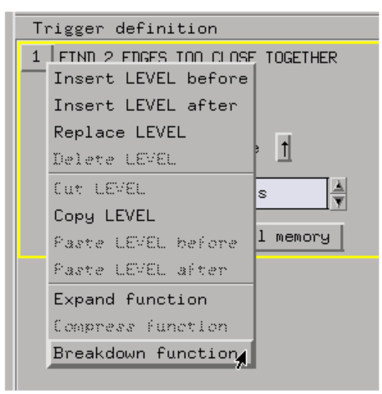
Measurement Tips & Tricks

- "Setting up 16715/16/17/18/19A triggers" on page 300
- "Setting up triggers in other logic analyzers" on page 302
- "Use trigger functions for easy measurement set up" on page 305
- "Modify trigger functions to build new measurements" on page 307
- "Know how processor execution affects measurements" on page 308
- "Getting the most out of trace memory" on page 309
- "If the trigger doesn't occur as expected" on page 309

Setting up 16715/16/17/18/19A triggers

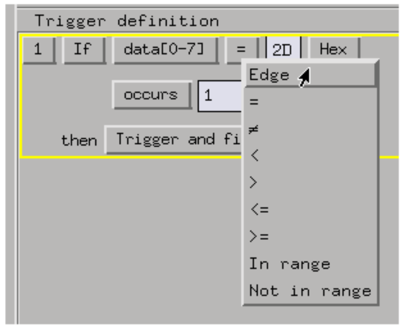
In General...

- Use trigger functions for basic measurements.
- For more complicated measurements, where no trigger function exists, start with a trigger function that's similar to the measurement you want to make. Then, break down the trigger function and edit the advanced trigger sequence levels.



Timing Analyzer Triggers

- Everything that looks like a button in the trigger definition gives you a way to modify the trigger setup.



For example, to look for a edge instead of a pattern, select the button that equates a label with a pattern and choose an edge comparison instead.

State Analyzer Triggers

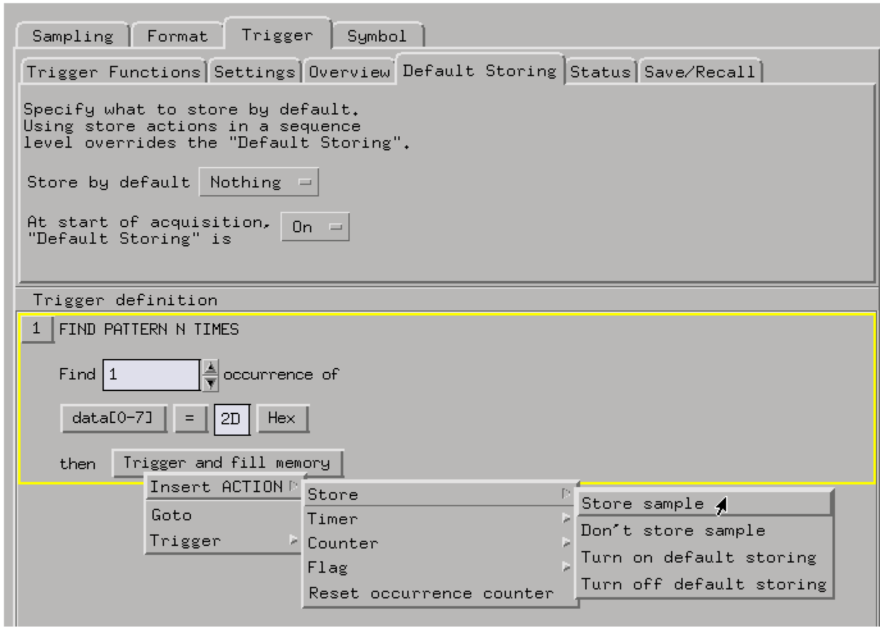
For every state analysis sample, a logic analyzer needs to know two things:

1. Should some action (like a trigger) be taken as a result of this sample?

2. What should be done with this sample? That is, should it be stored in logic analyzer memory or should it be discarded? (This question doesn't need to be asked when using a timing analyzer because all samples are stored.)

State analysis trigger definitions are made simpler with a *default storage* qualifier. This makes it possible to ignore, at all trigger sequence levels, the question about what to do with the sample.

Of course, sometimes it’s useful to specify storage qualifiers at certain levels in the trigger sequence. For this, you can insert storage *actions* in trigger definitions everywhere there is a trigger or goto action. Storage actions in the trigger definition override the default storage qualifier. Storage actions can also be used to turn on or off the default storing.



See also

"Modify trigger functions to build new measurements" on page 307

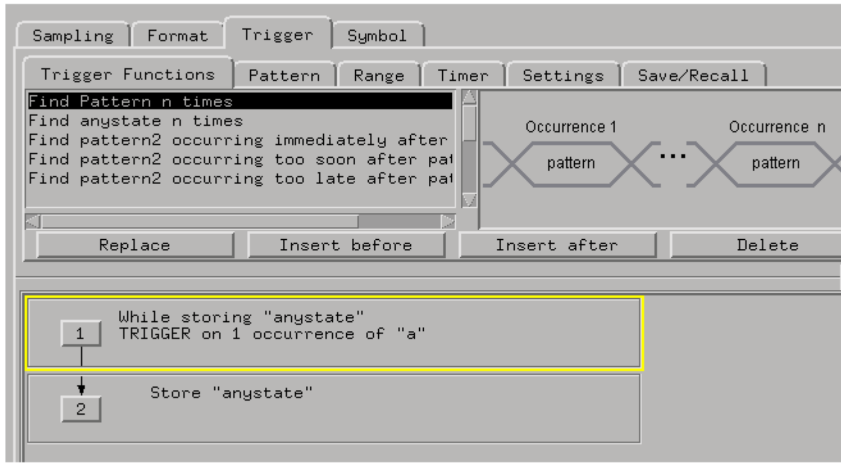
"Setting up triggers in other logic analyzers" on page 302

Setting up triggers in other logic analyzers

There are differences in the way that triggers are set up between the Agilent Technologies 16715/16/17/18/19A logic analyzers and other logic analyzers.

Similarities

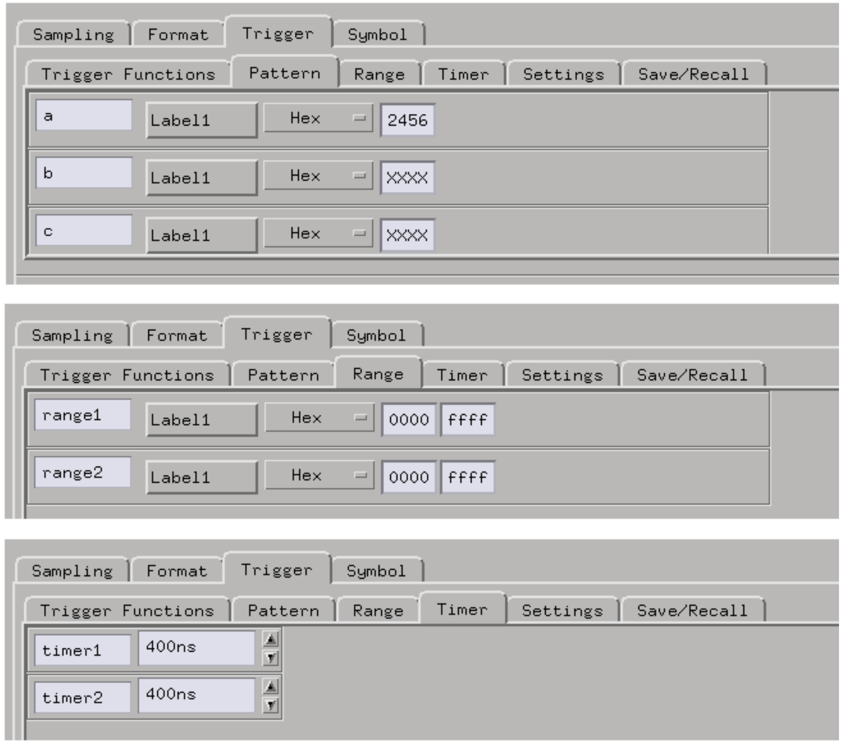
- In both types of logic analyzers, you are first given the choice of using trigger functions for trigger setup.



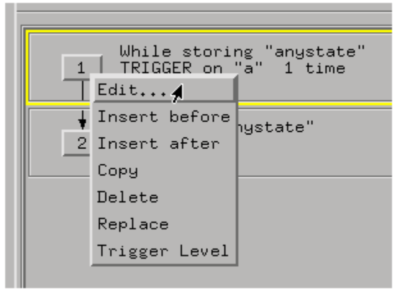
- Both types of logic analyzers also have a Settings tab for changing logic analyzer options and a Save/Recall tab for saving trigger setups.

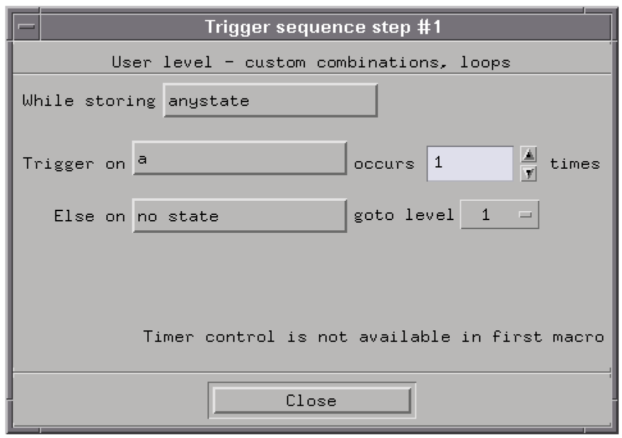
Differences

- Patterns, edges, ranges, and timers are set up under their own tabs in the older logic analyzers.



All editing of the trigger setup happens in these tabs and in the trigger sequence level button menus.





See also

"Setting up 16715/16/17/18/19A triggers" on page 300

Use trigger functions for easy measurement set up

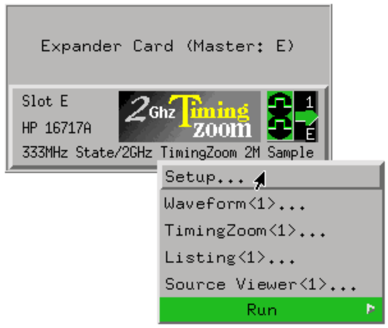
Many common measurement setups are provided with logic analyzers. These setups are called *trigger functions*, and you can use them for quick measurement setup.

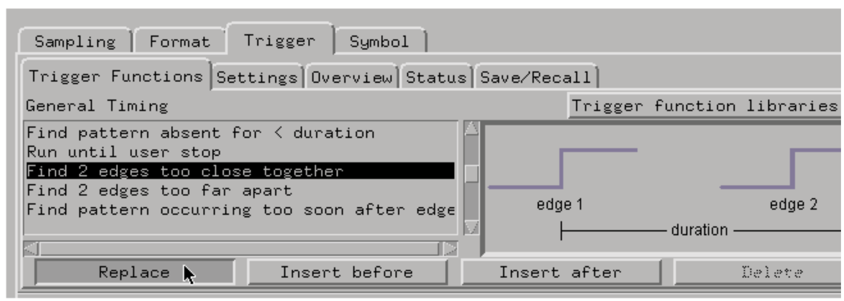
You can use different trigger functions at different sequence step levels to combine them into a single measurement.

To access trigger functions

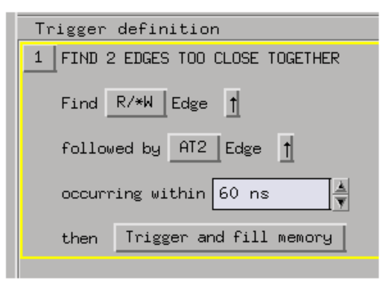
1. In the Trigger tab of a logic analyzer's Setup window, select the Trigger Functions tab.

2. Select a trigger function, and select the Replace, Insert Before, or Insert After button to move it to the trigger definition below.





3. In the trigger definition, specify the appropriate values and options, and select the Close button.

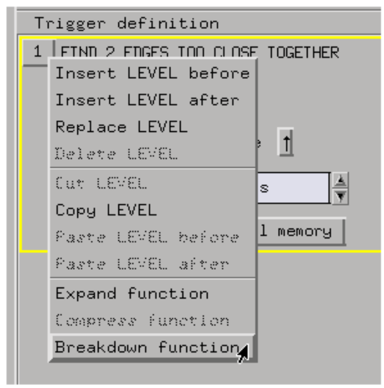


See also "Modify trigger functions to build new measurements" on page 307

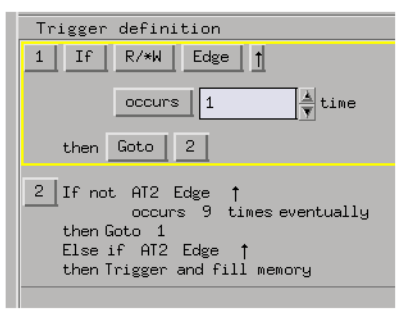
Modify trigger functions to build new measurements

Break the trigger function down to see the measurement in terms of the logic analyzer resources:

1. In Trigger window, break down the function from the Modify menu



Note that in a timing analyzer, the sample period and occurrence counts can be used to measure time.



Once broken-down, you can modify a trigger function.

Know how processor execution affects measurements

Instruction Cache

When instruction caches are turned ON, a complete view of processor execution cannot be viewed at the pins of the processor.

You can solve this problem by turning OFF instruction caches or by instrumenting your code (adding instructions that cause activity that can be viewed at the pins of the processor).

Chip Selects, MMUs, Paged Memory

When MMUs or paged memory moves code to different locations in memory, or when chip selects or reassigned address lines appear to change code addresses, symbol values are no longer accurate.

The symbol tools require that a one-to-one mapping exists between physical and logical addresses.

When chip selects reassign address lines, you may be able include them in the ADDR label specification to re-create a one-to-one mapping.

Word Alignment

When processors fetch multiple instructions (for example, byte instructions in a word fetch, or burst mode fetches), not all instruction addresses appear on the address bus. So, if you wanted to trigger on an instruction address, it might never be seen on the address bus.

You can work around this problem by aligning symbols to the word (or burst) boundary or by manually setting the lower address bits to 0 or X (don’t care) in the trigger specification.

Note that the workaround could result in a trigger on an unexecuted instruction (for example, if the previous instruction causes an execution branch or jump).

Unexecuted Prefetches

In processors that have prefetch queues and/or instruction pipelines, some fetched instructions are not executed. And, you could trigger on an instruction that isn’t executed.

Most inverse assemblers will flag unexecuted instructions with "nu" or "-". Enhanced inverse assemblers let you filter away unexecuted instructions from view.

To prevent false triggering, you can add an offset to addresses in the trigger specification (where a capture of the offset address indicates execution at the previous address). Another (less practical) way to prevent false triggering is to add NOPs to code to account for prefetch depth and pipelines.

Getting the most out of trace memory

Your strategy for capturing the right amount of data depends on the amount of trace memory your logic analyzer has.

Using deep memory analyzers

If trace memory is deep, you can capture all execution and use the filtering tools to only display relevant data.

Using filter tools

One strategy when using deep memory analyzers is to use the filtering and display tools to look at different aspects of the captured data.

For example, if you use the deep memory analyzer to capture all execution, you can use the Filter tool to isolate writes to a certain variable and the Chart display to track the values that were written. Or, you can use the Distribution display to see how the values vary.

Using storage qualifiers in the measurement

If the data you're interested in appears infrequently and you are not able to capture enough of it when all states are stored, you can use storage qualifiers to store only the data you're interested in.

Using the context store feature found in some analyzers

The context store feature stores events of interest plus the context data before and after these events. These events plus their context may occur far apart in time. There might be no way to capture a series of these events, even with very deep analyzers, unless they have context store.

If the trigger doesn't occur as expected

You've set up a trigger specification that you believe will lead to a trigger, and when you run the measurement, the trigger doesn't occur.

Or, you’ve set up a trigger specification that will only lead to a trigger in a error condition.

How do you tell what the logic analyzer is doing when the trigger doesn’t occur? How do you know which parts of your trigger specification the logic analyzer has or hasn’t seen?

There are a few strategies you can use when verifying or debugging sequence level programming:

- Look at the run status message line or open the Run Status window. It will tell you what level of the sequence the logic analyzer is in.
- Stop the measurement and look at the data that was captured. This is particularly useful when you use storage qualifiers to store "no states" (or only the states you're interested in) and the branches taken are stored.
- Save the trigger setup; then, simplify it to see what part of the sequence does get captured. When you learn what needs to be modified, you can recall the original trigger setup and make changes to it.

Glossary

absolute Denotes the time period or count of states between a captured state and the trigger state. An absolute count of -10 indicates the state was captured ten states before the trigger state was captured.

acquisition Denotes one complete cycle of data gathering by a measurement module. For example, if you are using an analyzer with 128K memory depth, one complete acquisition will capture and store 128K states in acquisition memory.

analysis probe A probe connected to a microprocessor or standard bus in the device under test. An analysis probe provides an interface between the signals of the microprocessor or standard bus and the inputs of the logic analyzer. Also called a *preprocessor*.

analyzer 1 In a logic analyzer with two *machines*, refers to the machine that is on by default. The default name is *Analyzer<N>*, where N is the slot letter.

analyzer 2 In a logic analyzer with two *machines*, refers to the machine that is off by default. The default name is *Analyzer<N2>*, where N is the slot letter.

arming An instrument tool must be armed before it can search for its trigger condition. Typically, instruments are armed immediately when *Run* or *Group Run* is selected. You can set up one instrument to arm another using the *Intermodule Window*. In these setups, the second instrument cannot search for its trigger condition until it receives the arming signal from the first instrument. In some analyzer instruments, you can set up one analyzer *machine* to arm the other analyzer machine in the *Trigger Window*.

asterisk (*) See *edge terms*, *glitch*, and *labels*.

bits Bits represent the physical logic analyzer channels. A bit is a *channel* that has or can be assigned to a *label*. A bit is also a position in a label.

card This refers to a single instrument intended for use in the Agilent Technologies 16600A-series or 16700A/B-series mainframes. One card fills one slot in the mainframe. A module may comprise a single card or multiple cards cabled together.

channel The entire signal path from the probe tip, through the cable and module, up to the label grouping.

click When using a mouse as the

pointing device, to click an item, position the cursor over the item. Then quickly press and release the *left mouse button*.

clock channel A logic analyzer *channel* that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a *data channel*, except in the Agilent Technologies 16517A.

context record A context record is a small segment of analyzer memory that stores an event of interest along with the states that immediately preceded it and the states that immediately followed it.

context store If your analyzer can perform context store measurements, you will see a button labeled *Context Store* under the Trigger tab. Typical context store measurements are used to capture writes to a variable or calls to a subroutine, along with the activity preceding and following the events. A context store measurement divides analyzer memory into a series of context records. If you have a 64K analyzer memory and select a 16-state context, the analyzer memory is divided into 4K 16-state context records. If you have a 64K analyzer memory and select a 64-state context, the analyzer memory will be divided into 1K 64-state records.

count The count function records periods of time or numbers of state transactions between states stored in memory. You can set up the analyzer count function to count occurrences of a selected event during the trace, such as counting how many times a variable is read between each of the writes to the variable. The analyzer can also be set up to count elapsed time, such as counting the time spent executing within a particular function during a run of your target program.

cross triggering Using intermodule capabilities to have measurement modules trigger each other. For example, you can have an external instrument arm a logic analyzer, which subsequently triggers an oscilloscope when it finds the trigger state.

data channel A *channel* that carries data. Data channels cannot be used to clock logic analyzers.

data field A data field in the pattern generator is the data value associated with a single label within a particular data vector.

data set A data set is made up of all labels and data stored in memory of any single analyzer machine or instrument tool. Multiple data sets can be displayed together when sourced into a single display tool. The Filter tool is used to pass on partial data sets to analysis or display tools.

debug mode See *monitor*.

delay The delay function sets the horizontal position of the waveform on the screen for the oscilloscope and timing analyzer. Delay time is measured from the trigger point in seconds or states.

demo mode An emulation control session which is not connected to a real target system. All windows can be viewed, but the data displayed is simulated. To start demo mode, select *Start User Session* from the Emulation Control Interface and enter the demo name in the *Processor Probe LAN Name* field. Select the *Help* button in the *Start User Session* window for details.

deskewing To cancel or nullify the effects of differences between two different internal delay paths for a signal. Deskewing is normally done by routing a single test signal to the inputs of two different modules, then adjusting the Intermodule Skew so that both modules recognize the signal at the same time.

device under test The system under test, which contains the circuitry you are probing. Also known as a *target system*.

don't care For *terms*, a "don't care" means that the state of the signal (high or low) is not relevant to the measurement. The analyzer ignores the state of this signal when determining whether a match occurs on an input label. "Don't care" signals are still sampled and their values can be displayed with the rest of the data. Don't cares are represented by the *X* character in numeric values and the dot (.) in timing edge specifications.

dot (.) See *edge terms*, *glitch*, *labels*, and *don't care*.

double-click When using a mouse as the pointing device, to double-click an item, position the cursor over the item, and then quickly press and release the *left mouse button* twice.

drag and drop Using a Mouse: Position the cursor over the item, and then press and hold the *left mouse button*. While holding the left mouse button down, move the mouse to drag the item to a new location. When the item is positioned where you want it, release the mouse button.

Using the Touchscreen:
Position your finger over the item, then press and hold finger to the screen. While holding the finger down, slide the finger along the screen dragging the item to a new location. When the item is positioned where you want it, release your finger.

edge mode In an oscilloscope, this is the trigger mode that causes a trigger based on a single channel edge, either rising or falling.

edge terms Logic analyzer trigger resources that allow detection of transitions on a signal. An edge term can be set to detect a rising edge, falling edge, or either edge. Some logic analyzers can also detect no edge or a *glitch* on an input signal. Edges are specified by selecting arrows. The dot (.) ignores the bit. The asterisk (*) specifies a glitch on the bit.

emulation module A module within the logic analysis system mainframe that provides an emulation connection to the debug port of a microprocessor. An E5901A emulation module is used with a target interface module (TIM) or an analysis probe. An E5901B emulation module is used with an E5900A emulation probe.

emulation probe The stand-alone equivalent of an *emulation module*. Most of the tasks which can be performed using an emulation module can also be performed using an emulation probe connected to your logic analysis system via a LAN.

emulator An *emulation module* or an *emulation probe*.

Ethernet address See *link-level address*.

events Events are the things you are looking for in your target system. In the logic analyzer interface, they take a single line. Examples of events are *Label1 = XX* and *Timer 1 > 400 ns*.

filter expression The filter expression is the logical *OR* combination of all of the filter terms. States in your data that match the filter expression can be filtered out or passed through the Pattern Filter.

filter term A variable that you define in order to specify which states to filter out or pass through. Filter terms are logically OR'ed together to create the filter expression.

Format The selections under the logic analyzer *Format* tab tell the logic analyzer what data you want to collect, such as which channels represent buses (labels) and what logic threshold your signals use.

frame The Agilent Technologies 16600A-series or 16700A/B-series logic analysis system mainframe. See also *logic analysis system*.

gateway address An IP address entered in integer dot notation. The default gateway address is 0.0.0.0, which allows all connections on the local network or subnet. If connections are to be made across networks or subnets, this address must be set to the address of the gateway machine.

glitch A glitch occurs when two or more transitions cross the logic threshold between consecutive timing analyzer samples. You can specify glitch detection by choosing the asterisk (*) for *edge terms* under the timing analyzer Trigger tab.

grouped event A grouped event is a list of *events* that you have grouped, and optionally named. It can be reused in other trigger sequence levels. Only available in Agilent Technologies 16715A, 16716A, and 16717A logic analyzers.

held value A value that is held until the next sample. A held value can exist in multiple data sets.

immediate mode In an oscilloscope, the trigger mode that does not require a specific trigger condition such as an edge or a pattern. Use immediate mode when the oscilloscope is armed by another instrument.

interconnect cable Short name for *module/probe interconnect cable*.

intermodule bus The intermodule bus (IMB) is a bus in the frame that allows the measurement modules to communicate with each other. Using the IMB, you can set up one instrument to *arm* another. Data acquired by instruments using the IMB is time-correlated.

intermodule Intermodule is a term used when multiple instrument tools are connected together for the purpose of one instrument arming another. In such a configuration, an arming tree is developed and the group run function is designated to start all instrument tools. Multiple instrument configurations are done in the Intermodule window.

internet address Also called Internet Protocol address or IP address. A 32-bit network address. It is usually represented as decimal numbers separated by periods; for example, 192.35.12.6. Ask your LAN administrator if you need an internet address.

labels Labels are used to group and identify logic analyzer channels. A label consists of a name and an associated bit or group of bits. Labels are created in the Format tab.

line numbers A line number (Line #s) is a special use of *symbols*. Line numbers represent lines in your source file, typically lines that have no unique symbols defined to represent them.

link-level address Also referred to as the Ethernet address, this is the unique address of the LAN interface. This value is set at the factory and cannot be changed. The link-level address of a particular piece of equipment is often printed on a label above the LAN connector. An example of a link-level address in hexadecimal: 0800090012AB.

local session A local session is when you run the logic analysis system using the local display connected to the product hardware.

logic analysis system The Agilent Technologies 16600A-series or 16700A/B-series mainframes, and all tools designed to work with it. Usually used to mean the specific system and tools you are working with right now.

machine Some logic analyzers allow you to set up two measurements at the same time. Each measurement is handled by a different machine. This is represented in the Workspace window by two icons, differentiated by a *1* and a *2* in the upper right-hand corner of the icon. Logic analyzer resources such as pods and trigger terms cannot be shared by the machines.

markers Markers are the green and yellow lines in the display that are labeled *x*, *o*, *G1*, and *G2*. Use them to measure time intervals or sample intervals. Markers are assigned to patterns in order to find patterns or track sequences of states in the data. The x and o markers are local to the immediate display, while G1 and G2 are global between time correlated displays.

master card In a module, the master card controls the data acquisition or output. The logic analysis system references the module by the slot in which the master card is plugged. For example, a 5-card Agilent Technologies 16555D would be referred to as *Slot C: machine* because the master card is in slot C of the mainframe. The other cards of the module are called *expansion cards*.

menu bar The menu bar is located at the top of all windows. Use it to select *File* operations, tool or system *Options*, and tool or system level *Help*.

message bar The message bar displays mouse button functions for the window area or field directly beneath the mouse cursor. Use the mouse and message bar together to prompt yourself to functions and shortcuts.

module/probe interconnect cable

The module/probe interconnect cable connects an E5901B emulation module to an E5900B emulation probe. It provides power and a serial connection. A LAN connection is also required to use the emulation probe.

module An instrument that uses a single timebase in its operation. Modules can have from one to five cards functioning as a single instrument. When a module has more than one card, system window will show the instrument icon in the slot of the *master card*.

monitor When using the Emulation Control Interface, running the monitor means the processor is in debug mode (that is, executing the debug exception) instead of executing the user program.

panning The action of moving the waveform along the timebase by varying the delay value in the Delay field. This action allows you to control the portion of acquisition memory that will be displayed on the screen.

pattern mode In an oscilloscope, the trigger mode that allows you to set the oscilloscope to trigger on a specified combination of input signal levels.

pattern terms Logic analyzer resources that represent single states to be found on labeled sets of bits; for example, an address on the address bus or a status on the status lines.

period (.) See *edge terms*, *glitch*, *labels*, and *don't care*.

pod pair A group of two pods containing 16 channels each, used to physically connect data and clock signals from the unit under test to the analyzer. Pods are assigned by pairs in the analyzer interface. The number of pod pairs avalaible is determined by the channel width of the instrument.

pod See *pod pair*

point To point to an item, move the mouse cursor over the item, or position your finger over the item.

preprocessor See *analysis probe*.

primary branch The primary branch is indicated in the *Trigger sequence step* dialog box as either the *Then find* or *Trigger on* selection. The destination of the primary branch is always the next state in the sequence, except for the Agilent Technologies 16517A. The primary branch has an optional occurrence count field that can be used to count a number of occurrences of the branch condition. See also *secondary branch*.

probe A device to connect the various instruments of the logic analysis system to the target system. There are many types of probes and the one you should use depends on the instrument and your data requirements. As a verb, "to probe" means to attach a probe to the target system.

processor probe See *emulation probe*.

range terms Logic analyzer resources that represent ranges of values to be found on labeled sets of bits. For example, range terms could identify a range of addresses to be found on the address bus or a range of data values to be found on the data bus. In the trigger sequence, range terms are considered to be true when any value within the range occurs.

relative Denotes time period or count of states between the current state and the previous state.

remote display A remote display is a display other than the one connected to the product hardware. Remote displays must be identified to the network through an address location.

remote session A remote session is when you run the logic analyzer using a display that is located away from the product hardware.

right-click When using a mouse for a pointing device, to right-click an item, position the cursor over the item, and then quickly press and release the *right mouse button*.

sample A data sample is a portion of a *data set*, sometimes just one point. When an instrument samples the target system, it is taking a single measurement as part of its data acquisition cycle.

Sampling Use the selections under the logic analyzer Sampling tab to tell the logic analyzer how you want to make measurements, such as State vs. Timing.

secondary branch The secondary branch is indicated in the *Trigger sequence step* dialog box as the *Else on* selection. The destination of the secondary branch can be specified as any other active sequence state. See also *primary branch*.

session A session begins when you start a *local session* or *remote session* from the session manager, and ends when you select *Exit* from the main window. Exiting a session returns all tools to their initial configurations.

skew Skew is the difference in channel delays between measurement channels. Typically, skew between modules is caused by differences in designs of measurement channels, and differences in characteristics of the electronic components within those channels. You should adjust measurement modules to eliminate as much skew as possible so that it does not affect the accuracy of your measurements.

state measurement In a state measurement, the logic analyzer is clocked by a signal from the system under test. Each time the clock signal becomes valid, the analyzer samples data from the system under test. Since the analyzer is clocked by the system, state measurements are *synchronous* with the test system.

store qualification Store qualification is only available in a *state measurement*, not *timing measurements*. Store qualification allows you to specify the type of information (all samples, no samples, or selected states) to be stored in memory. Use store qualification to prevent memory from being filled with unwanted activity such as no-ops or wait-loops. To set up store qualification, use the *While storing* field in a logic analyzer trigger sequence dialog.

subnet mask A subnet mask blocks out part of an IP address so that the networking software can determine whether the destination host is on a local or remote network. It is usually represented as decimal numbers separated by periods; for example, 255.255.255.0. Ask your LAN administrator if you need a the subnet mask for your network.

symbols Symbols represent patterns and ranges of values found on labeled sets of bits. Two kinds of symbols are available:

- Object file symbols - Symbols from your source code, and symbols generated by your compiler. Object file symbols may represent global variables, functions, labels, and source line numbers.
- User-defined symbols - Symbols you create.

Symbols can be used as *pattern* and *range* terms for:

- Searches in the listing display.
- Triggering in logic analyzers and in the source correlation trigger setup.
- Qualifying data in the filter tool and system performance analysis tool set.

system administrator The system administrator is a person who manages your system, taking care of such tasks as adding peripheral devices, adding new users, and doing system backup. In general, the system administrator is the person you go to with questions about implementing your software.

target system The system under test, which contains the microprocessor you are probing.

terms Terms are variables that can be used in trigger sequences. A term can be a single value on a label or set of labels, any value within a range of values on a label or set of labels, or a glitch or edge transition on bits within a label or set of labels.

TIM A TIM (Target Interface Module) makes connections between the cable from the emulation module or emulation probe and the cable to the debug port on the system under test.

time-correlated Time correlated measurements are measurements involving more than one instrument in which all instruments have a common time or trigger reference.

timer terms Logic analyzer resources that are used to measure the time the trigger sequence remains within one sequence step, or a set of sequence steps. Timers can be used to detect when a condition lasts too long or not long enough. They can be used to measure pulse duration, or duration of a wait loop. A single timer term can be used to delay trigger until a period of time after detection of a significant event.

timing measurement In a timing measurement, the logic analyzer samples data at regular intervals according to a clock signal internal to the timing analyzer. Since the analyzer is clocked by a signal that is not related to the system under test, timing measurements capture traces of electrical activity over time. These measurements are *asynchronous* with the test system.

tool icon Tool icons that appear in the workspace are representations of the hardware and software tools selected from the toolbox. If they are placed directly over a current measurement, the tools automatically connect to that measurement. If they are placed on an open area of the main window, you must connect them to a measurement using the mouse.

toolbox The Toolbox is located on the left side of the main window. It is used to display the available hardware and software tools. As you add new tools to your system, their icons will appear in the Toolbox.

tools A tool is a stand-alone piece of functionality. A tool can be an instrument that acquires data, a display for viewing data, or a post-processing analysis helper. Tools are represented as icons in the main window of the interface.

trace See *acquisition*.

trigger sequence A trigger sequence is a sequence of events that you specify. The logic analyzer compares this sequence with the samples it is collecting to determine when to *trigger*.

trigger specification A trigger specification is a set of conditions that must be true before the instrument triggers.

trigger Trigger is an event that occurs immediately after the instrument recognizes a match between the incoming data and the trigger specification. Once trigger occurs, the instrument completes its *acquisition*, including any store qualification that may be specified.

workspace The workspace is the large area under the message bar and to the right of the toolbox. The workspace is where you place the different instrument, display, and analysis tools. Once in the workspace, the tool icons graphically represent a complete picture of the measurements.

zooming In the oscilloscope or timing analyzer, to expand and contract the waveform along the time base by varying the value in the s/Div field. This action allows you to select specific portions of a particular waveform in acquisition memory that will be displayed on the screen. You can view any portion of the waveform record in acquisition memory.

Index

Index