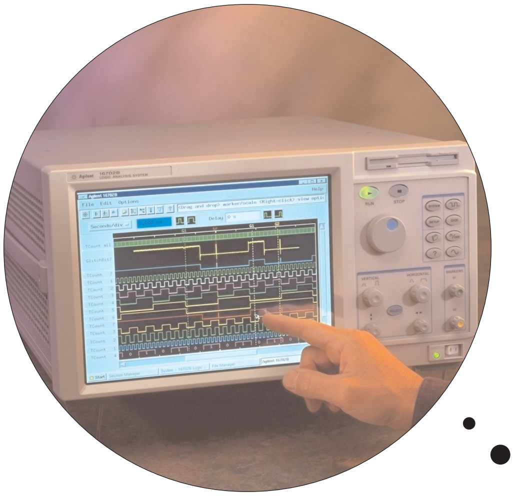
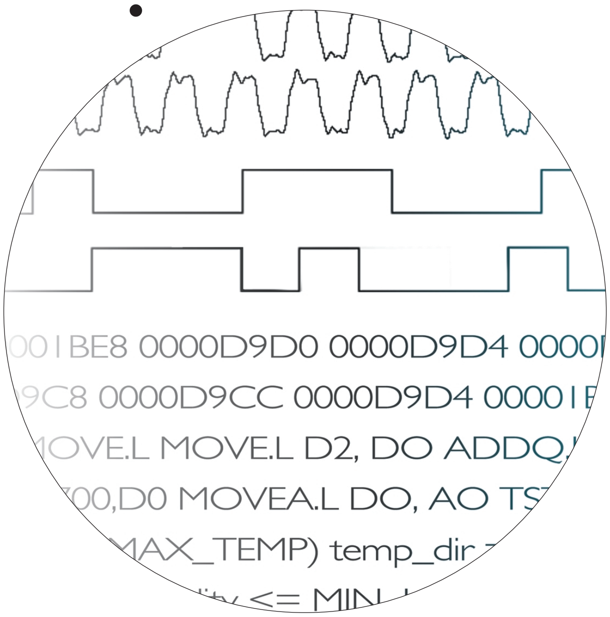
8 Hints For Solving Common Debugging Problems With Your Logic Analyzer

Application Note 1326




001BE8 0000D9D0 0000D9D4 0000
9C8 0000D9CC 0000D9D4 00001
OVE.L MOVE.L D2, DO ADDQ.
00,D0 MOVEA.L DO, AO TS
MAX_TEMP) temp_dir
<= MIN

Agilent Technologies
Innovating the HP Way

Reducing the Complexity in Your Job

Logic analyzers are complex instruments. They have to be in order to handle the capabilities of today's advanced electronic devices. Unfortunately, the complexity of logic analyzers can cause you headaches when you need critical information about your digital designs.

Yet using a logic analyzer is frequently the best way, and sometimes the only way, to understand how your device is working, or why it's not. So, if you need to look at your logic in state mode, for example, or examine timing relationships on a large number of channels, you might reluctantly pull out your logic analyzer. Sound familiar?

We'd like to help you overcome your reluctance by helping you build the measurement expertise you need to do your job. That's why we've gathered and published these hints. They are not intended to be comprehensive tutorials. They are suggestions intended to help you understand how you can use logic analyzers most effectively and how they can help you save time in getting your job done.

Table of Contents

Acquiring Data From a Multiplexed Address/Data Bus

HINT 1

Many engineers who design with modern microprocessors and micro-controllers use multiplexed buses in order to conserve pins and reduce cost. Engineers used this technique in early processors such as the Intel 8088, and they're using it in current processors such as custom-cored ASICs. In order to effectively capture the information you need in this complex design arena, you need a logic analyzer with specialized features.

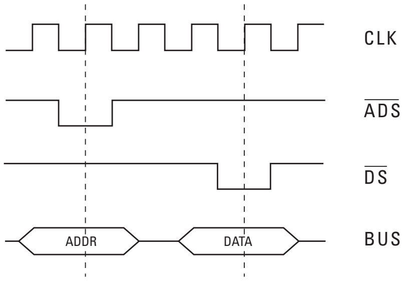
Today's logic analyzers offer specific clocking capabilities to handle the acquisition of address and data from multiplexed buses. You can capture information on two different clock events, one when the address is valid and another when the data is valid. Because the bus is multiplexed, you only need one set of logic analyzer probes. You can connect the logic analyzer probes to the multiplexed address/data bus either with individual probes or through a probing adapter.

Since the sample clocks are coming from the device under test, set the analyzer to "State" or "Synchronous" mode.

The key to making this measurement is to determine the clock or clock combination for the address phase and for the data phase. If these are not different, the bus is not really multiplexed.

An example of a multiplexed bus:

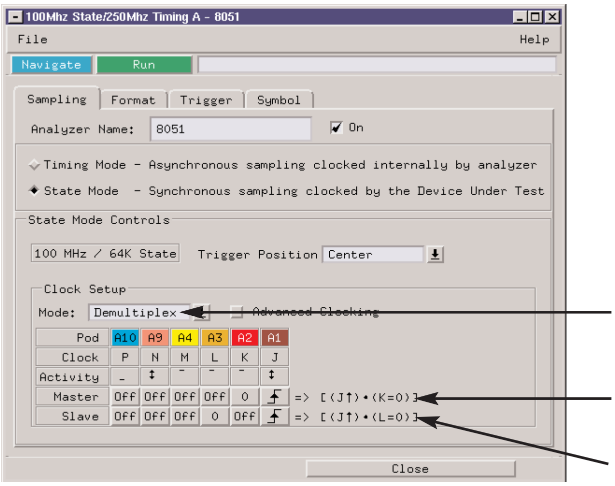
address clock specification:
CLK rising AND ADS# = low (true)
data clock specification:
CLK rising AND DS# = low (true)



As an example, in the Agilent Technologies 16700B Series logic analyzers, you can go to the Logic Analysis System window and click on the Logic Analysis card. From the pop-up menu, select 'Setup', then click on the Sampling tab. On this screen, you can set your logic analyzer to "Demultiplex" clocking mode, as shown in Figure 1.1. This screen also allows you to specify the logic for the master and slave clocks. Then go to the Format tab, shown in Figure 1.2, and assign the pins for your logic analyzer probe.

Note that your logic analyzer's setup and hold requirements in demultiplex clocking mode may be different from those in normal clocking mode. Check your logic analyzer's specifications for details.

Figure 1.1 Sampling Clocking Mode



100Mhz State/250Mhz Timing A - 8051

File Help

Navigate Run

Sampling | Format | Trigger | Symbol

Analyzer Name: 8051 ☑ On

◇ Timing Mode - Asynchronous sampling clocked internally by analyzer

◆ State Mode - Synchronous sampling clocked by the Device Under Test

State Mode Controls

100 MHz / 64K State Trigger Position Center

Clock Setup

Mode: Demultiplex □ ~~Advanced Clocking~~

Pod	A10	A9	A4	A3	A2	A1	
Clock	P	N	M	L	K	J	
Activity	_	↕	‾	‾	‾	↕	
Master	Off	Off	Off	Off	0	↑	=> [(J↑)•(K=0)]
Slave	Off	Off	Off	0	Off	↑	=> [(J↑)•(L=0)]

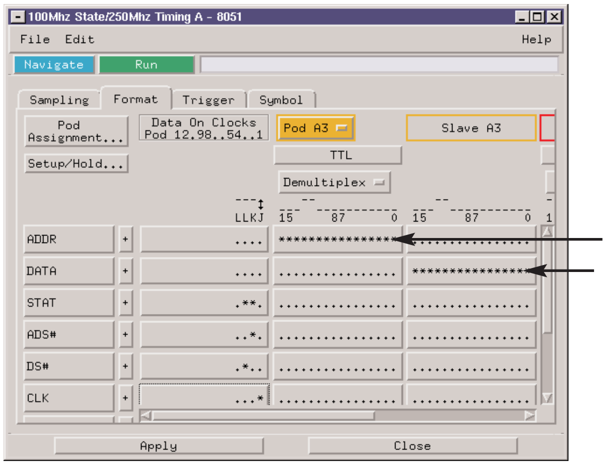
Close

Select Demultiplex mode

Read address here

Read data here

Figure 1.2 Formatting Clock Mode



100Mhz State/250Mhz Timing A - 8051

File Edit Help

Navigate Run

Sampling | Format | Trigger | Symbol

Pod Assignment... Data On Clocks Pod 12.98..54..1 Pod A3 Slave A3

Setup/Hold... TTL

Demultiplex

		LLKJ	15 87 0	15 87 0
ADDR	+	....	***************	....
DATA	+	....	....	***************
STAT	+	.**.	....	....
ADS#	+	..*.	....	....
DS#	+	.*..	....	....
CLK	+	...*	....	....

Apply Close

Assign pins here

What to do When Your Target System is Functioning Normally, but the Data You Capture Does Not Appear to be Valid

HINT 2

This situation occurs in several circumstances:

1. When your logic analyzer sampling thresholds do not match your target system's switching characteristics
2. When you're in state sampling mode and your target system does not meet the setup and hold times required by your logic analyzer
3. When the target system generates excessive noise (ground bounce, simultaneous switching noise or crosstalk), causing the analyzer to sample incorrectly.

Adjusting various logic analyzer settings often solves the first two conditions. The third condition, noise, is most often only solved by design changes to your board.

Let's look at the first two situations.

Different logic families have different switching thresholds.

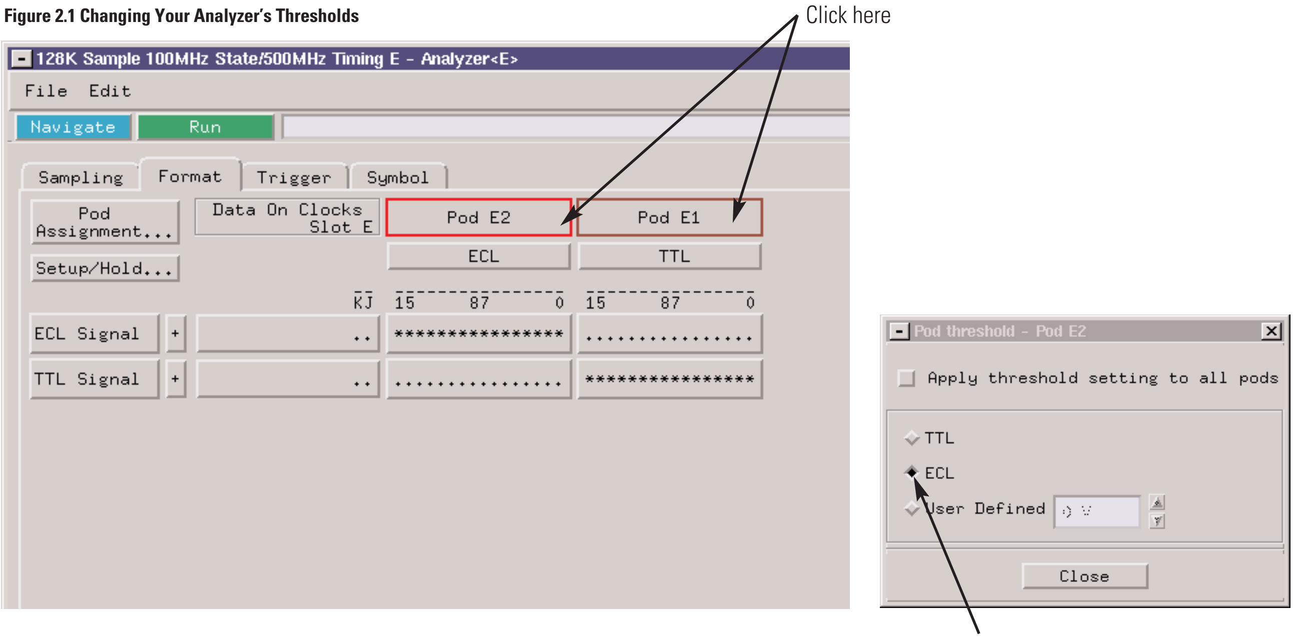
Some examples are:

TTL	=>	1.5V
5V CMOS	=>	2.5V
3.3V CMOS	=>	1.6V

If you suspect that sampling thresholds may be the cause of your problem, then the solution is straightforward. First, identify the logic families (and therefore, the switching voltage) of the signals you are probing. Next, change the threshold settings on the logic analyzer. Some analyzers allow you to set different thresholds for different groups of signals. The Agilent Technologies 16700B Series logic analyzers allow independent settings on 16 channel (1 pod) boundaries.
In addition to the standard switching thresholds, these logic analyzers allow you to specify any arbitrary threshold.

From the Logic Analysis System window, click on the Logic Analysis card. From the pop-up menu, select 'Setup' to get the Setup screen and go to the Format tab. This screen allows you to change both the sampling threshold and the setup and hold times. Figure 2.1 shows the threshold setup dialog box.

Figure 2.1 Changing Your Analyzer's Thresholds



Click here

Change threshold here

The second situation involves setup and hold times and only occurs when you sample based on the clock supplied by the target system.

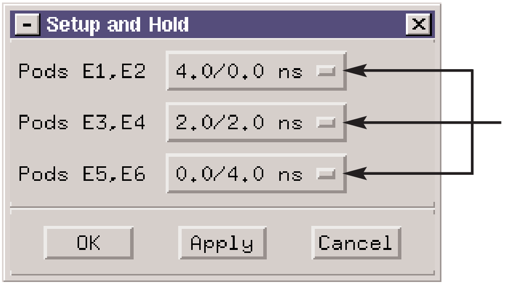
The solution here is also relatively straightforward. First, determine the setup and hold requirements of the logic you are probing. Hopefully, you considered this in your design because the interconnected IC's must satisfy each other's setup and hold requirements. Next, determine the setup and hold time requirements for your logic analyzer. Does your target meet the logic analyzer's required time? If the "window" (setup+hold) supplied is smaller than that required by the logic analyzer, you may need to upgrade your logic analyzer. If the window is larger, but either the setup or hold time is violated, you may be in luck. Some logic analyzers allow you to adjust the window as needed.

The example given shows that you can adjust from 4.0 ns setup and 0.0 ns hold time all the way to 0.0 ns setup and 4.0 ns hold time, in 0.5 ns increments. It also allows you to set different channel groups differently. Figure 2.2 demonstrates this.

The techniques mentioned above of adjusting your thresholds and setup/hold windows can be valuable ways to squeeze more mileage out of your analyzer, especially if your target system is heavily taxing the capabilities of your current analyzer.

If your logic analyzer meets the specifications required by your target system, and neither of these techniques solve your problems, you most likely have a noise problem. That means it's time to get your oscilloscope back out and put on your analog designer hat.

Figure 2.2 Adjusting Setup and Hold Times



Pods E1,E2 4.0/0.0 ns

Pods E3,E4 2.0/2.0 ns

Pods E5,E6 0.0/4.0 ns

Select the setup/hold time from the popup window

Using a "Golden Trace" to Troubleshoot Unexpected System Changes

HINT 3

When you are developing your target system and it is working properly, life is good. This is the time you should connect your target to a logic analyzer and capture a 'golden trace' of the signals. Performing this small task might save the day if your target system starts behaving unexpectedly.

If you have saved a 'golden trace,' you can use your logic analyzer's Compare tool to zero in on the problem. The Compare tool takes data from a stored 'golden trace' file and compares it to a real-time trace so that you can find the differences. You can set up the Compare tool to perform the comparison on specific signals or buses, or on all signals or buses that are present in both data sources. You can also set it up in a repetitive run mode in which each new run is compared to a data file. The analyzer will stop capturing and comparing data when it finds a difference.

In the example shown in Figure 3.1, the Compare tool has an analyzer as one data source and a data file as the other. The data file represents your stored 'golden trace.' You can display the results of the comparison using a Listing tool.

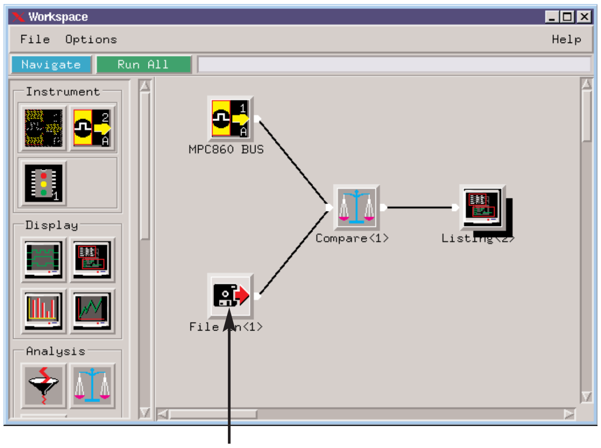
Right click on the Compare icon and select Setup from the pop-up menu. In the Setup dialog, right click on the Run button and select 'Repetitive' from the pop-up menu. Then click on the Run button to run the Compare tool repetitively. When the measurements stop, use the listing window to examine the exact state on which the difference occurred.

In the listing window, you can search on the "DiffFlag" label to find the exact state on which the comparison failed, as shown in Figure 3.2. The data that is different in each source is highlighted.

This technique shows you what changed between your 'golden trace' and the current trace and helps you quickly identify what caused the problem.

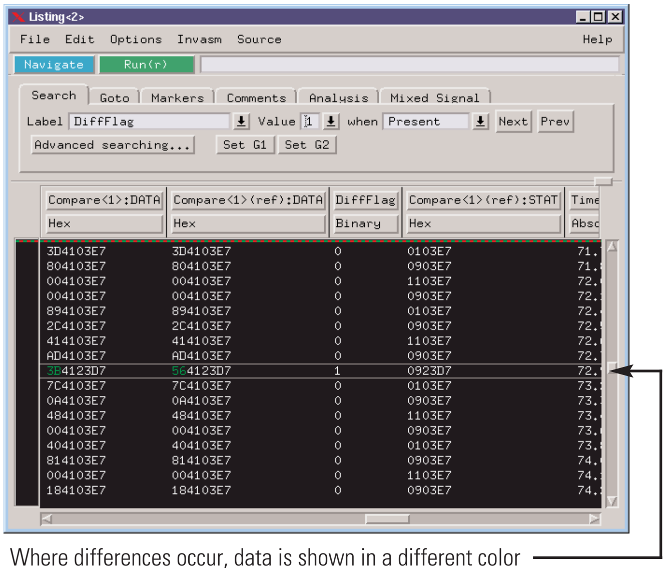
Often, simulators are used to test software when hardware is not yet available. When the first versions of the hardware become available, the software that was tested and working on the simulation system may not work on the actual system hardware. You can use the Compare tool to compare simulation data (that has been run through a conversion program so that it can be read by the analyzer) with real acquired data. This helps you with hardware/software integration by letting you quickly find signals that are not behaving as they did in the simulations.

Figure 3.1 Using the Compare Tool on Signals or Buses



Your stored "golden trace" file

Figure 3.2 Using the DiffFlag Function to Identify Changes



Label DiffFlag Value 1 when Present

Compare<1>:DATA	Compare<1>(ref):DATA	DiffFlag	Compare<1>(ref):STAT	Time
Hex	Hex	Binary	Hex	Absc
3D4103E7	3D4103E7	0	0103E7	71.
804103E7	804103E7	0	0903E7	71.
004103E7	004103E7	0	1103E7	72.
004103E7	004103E7	0	0903E7	72.
894103E7	894103E7	0	0103E7	72.
2C4103E7	2C4103E7	0	0903E7	72.
414103E7	414103E7	0	1103E7	72.
AD4103E7	AD4103E7	0	0903E7	72.
3B4123D7	564123D7	1	0923D7	72.
7C4103E7	7C4103E7	0	0103E7	73.
0A4103E7	0A4103E7	0	0903E7	73.
484103E7	484103E7	0	1103E7	73.
004103E7	004103E7	0	0903E7	73.
404103E7	404103E7	0	0103E7	73.
814103E7	814103E7	0	0903E7	74.
004103E7	004103E7	0	1103E7	74.
184103E7	184103E7	0	0903E7	74.

Where differences occur, data is shown in a different color

Using Offsets to Avoid False Triggers

HINT 4

Because today's processors have very complex and deep pipelines and prefetch queues, many instructions can be fetched and brought across the bus and yet never be executed. This makes it difficult to accurately set the trigger on your logic analyzer. When the logic analyzer triggers on one of these unexecuted fetches a false trigger results.

This problem typically occurs when you're trying to trigger on a line of code that is directly after a function call or the end of a loop. Setting a trigger on lines 25 or 57 in the following example code may result in false triggers (see Figure 4.1).

One way to avoid false triggers in these situations is to use a trigger offset. A trigger offset indicates that the logic analyzer should trigger on a given address only if the address value itself is captured and the address value that is the trigger address plus the offset value is also captured. This tells the analyzer to trigger only if the address of interest is captured AND an address that is some number of bytes away is also captured.

Typically, the offset value should be the number of bytes that the prefetch queue and pipeline can hold. For example, the processor being used has a four-instruction queue and each instruction is 32 bits (or 4 bytes). The processor also has a one-instruction prefetch queue (another 4 bytes). The offset for this processor should be 20 bytes (5 instructions total).

Note that this is not a foolproof scheme for avoiding these types of false triggers. Using an offset of the full prefetch queue depth and pipelining queue depth could result in a missed trigger if a branch takes place between the base address and the offset address. In this case, the queue depth is too large of an offset. Reduce the offset to avoid completely missing the trigger.

Figure 4.1 C Source Code

Line #	C source code
11	for(int i=0;i<MAX_LOOP;i++)
12	{
.....	
24	}
25	count++;
..	...
56	storeCount(count);
57	count++;

Reducing Security Risks on Networked Logic Analyzers

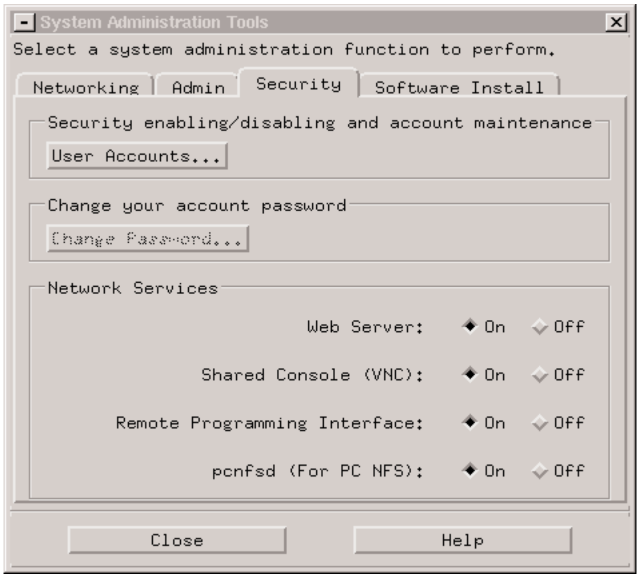
HINT 5

Without special precautions, having your logic analyzer on the network might pose a security risk. People using the analyzer might have access to files on other networked machines that they are not supposed to access. And, can inversely, people on the network might have access to data on the logic analyzer that they are not supposed to see. The overall responsibility for security is on the shoulders of your network administrators, but if the logic analyzer does not provide some basic security features, such as user accounts (logins), passwords, file permissions, etc., you may be out of luck.

Most modern logic analyzers provide the features needed to make themselves "good network citizens." These analyzers allow you to set up user accounts and passwords to control who can use the analyzer. In addition, they allow you to specify file permissions for individual users so that you can control access to the data on the logic analyzer. They also allow you to share files on a network in a controlled fashion.

The screen shot in Figure 5.1 shows the "Secure Mode" setup dialog, and indicates how the various features provided are set up. You can access this dialog box through the Main Window "System Admin" dialog, and then select the "Security" tab. Click on "User Accounts" to manage user information.

Figure 5.1 Secure Mode Setup



System Administration Tools

Select a system administration function to perform.

Security enabling/disabling and account maintenance

Change your account password

Network Services

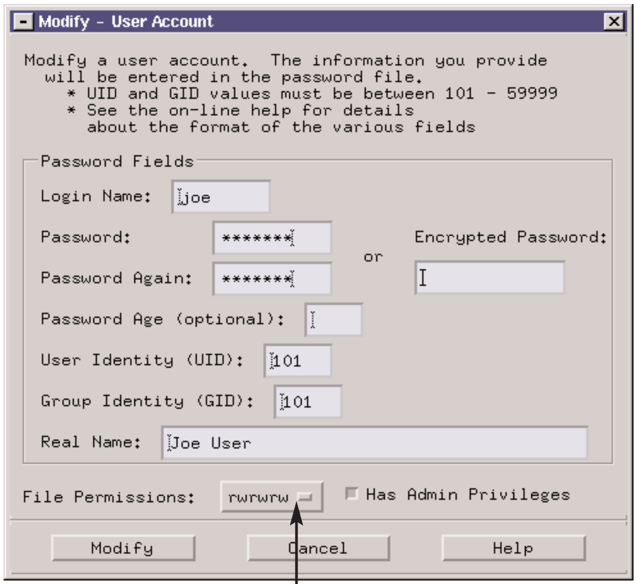
Web Server: ● On ○ Off

Shared Console (VNC): ● On ○ Off

Remote Programming Interface: ● On ○ Off

pcnfsd (For PC NFS): ● On ○ Off

Figure 5.2 Assigning Passwords



Modify - User Account

Modify a user account. The information you provide will be entered in the password file.
* UID and GID values must be between 101 - 59999
* See the on-line help for details about the format of the various fields

Password Fields

Login Name: joe

Password: *******

Password Again: *******

or

Encrypted Password:

Password Age (optional):

User Identity (UID): 101

Group Identity (GID): 101

Real Name: Joe User

File Permissions: rwrwrw

□ Has Admin Privileges

Click here to assign permissions.

How to Capture Data Before a System Crash

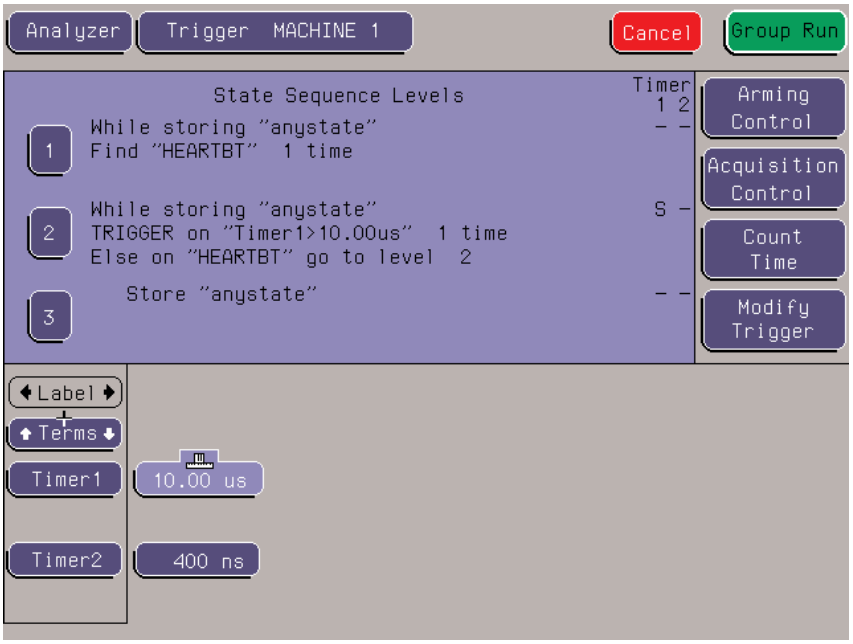
HINT 6

Capturing the cause of a system crash can be tricky with a logic analyzer. This is true because you have to get the instrument to trigger when nothing happens. How do you describe "nothing" to a logic analyzer's trigger menu? One tried-and-true technique is to set up the logic analyzer to store only the data of interest and never trigger. Then you can stop the instrument manually when the system under test crashes. When this technique works, the logic analyzer has a history stored in its pretrigger trace buffer. But this technique only works if you can set up your logic analyzer to capture meaningful information, then stop when the system under test crashes.

Here's one way around the problem: use a timer in the logic analyzer to get a trigger soon after the system crashes. To use a timer in the trigger, first figure out what event should happen regularly in order for you to consider your system under test alive and working fine. Let's call this event the system's "heartbeat". This could be an address-strobe, a periodic interrupt, or anything that occurs consistently at some minimum frequency. Then set a timer equal to the longest period between successive heartbeats that you would expect to see before you would consider the system nonfunctional. In the trigger sequence, if you start the timer when a heartbeat occurs, and restart the timer whenever another heartbeat occurs, you can then test if the timer reaches the maximum value that you set. If the timer reaches the maximum value, then the logic analyzer should trigger, since the "heartbeat" isn't occurring as often as it should. You can also experiment with the timer value to determine just how long a period between "heartbeats" is normal.

Here's an example of this trigger timer idea using an Agilent Technologies 1670G, a benchtop logic analyzer and an optional integrated oscilloscope. Set up the logic analyzer trigger like you see in Figure 6.1.

Figure 6.1 Configuring a "Heartbeat" Trigger

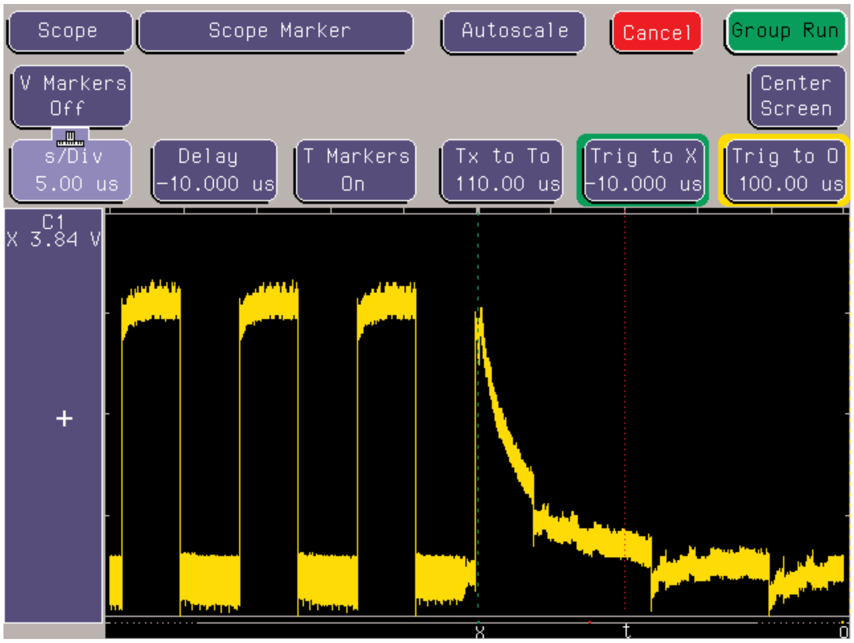


```
State Sequence Levels                                   Timer
                                                         1 2
1  While storing "anystate"                              - -
   Find "HEARTBT"   1 time

2  While storing "anystate"                              S -
   TRIGGER on "Timer1>10.00us"   1 time
   Else on "HEARTBT" go to level  2

3     Store "anystate"                                   - -

Timer1   10.00 us
Timer2   400 ns
```

The "heartbeat" in this example is a signal that consistently occurs at 8 µs intervals. This trigger setup will force a trigger 10 µs after the last "heartbeat". With the Agilent Technologies 1670G, you can send the trigger to the oscilloscope, which is also probing the device under test, to get a better understanding of the cause of the crash. Figure 6.2 shows what the oscilloscope caught 10 µs prior to the trigger that it received from the logic analyzer.

In this case, the oscilloscope can help get to the cause of the system crash. Without an actual trigger event, there would be no trigger signal to send to the oscilloscope. Using timers in a logic analyzer's trigger system can allow you to trigger on "nothing" since you can describe the condition as too much time between successive "heartbeats".

Figure 6.2 Oscilloscope Display of Results



V Markers Off

s/Div 5.00 us

Delay -10.000 us

T Markers On

Tx to To 110.00 us

Trig to X -10.000 us

Trig to O 100.00 us

C1 X 3.84 V

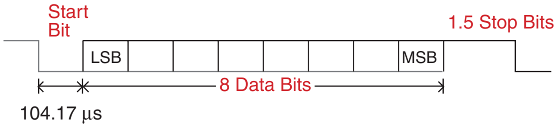
Analyzing Serial Data with a Logic Analyzer

HINT 7

While logic analyzers are commonly used to analyze parallel data (e.g. microprocessor address and data), some modern logic analyzers also allow you to analyze serial data. This feature comes in handy when you are trying to debug simple serial protocols such as RS-232C, CAN, LAN, USB or your proprietary bus. A logic analyzer is not a tool for analyzing higher-level protocols such as TCP-IP. You will need a protocol analyzer for that task.

Here's an example of how to use an Agilent Technologies 16700B Series logic analyzer to analyze and troubleshoot data from an RS-232C output port. Once you have connected the serial port on your target system to the logic analyzer through one of its data pods, you will need to put the logic analyzer in the "Timing" or "Asynchronous" mode because the sampling clock is being supplied by the logic analyzer. Even though you are in timing analysis mode, you will be able to view the data as parallel information in a listing form. Next you will need to assign a label (Serial) to the input data bit. The RS-232C protocol has a data format with 1 start bit, 8 data bits, and 1.5 stop bits, as shown in Figure 7.1.

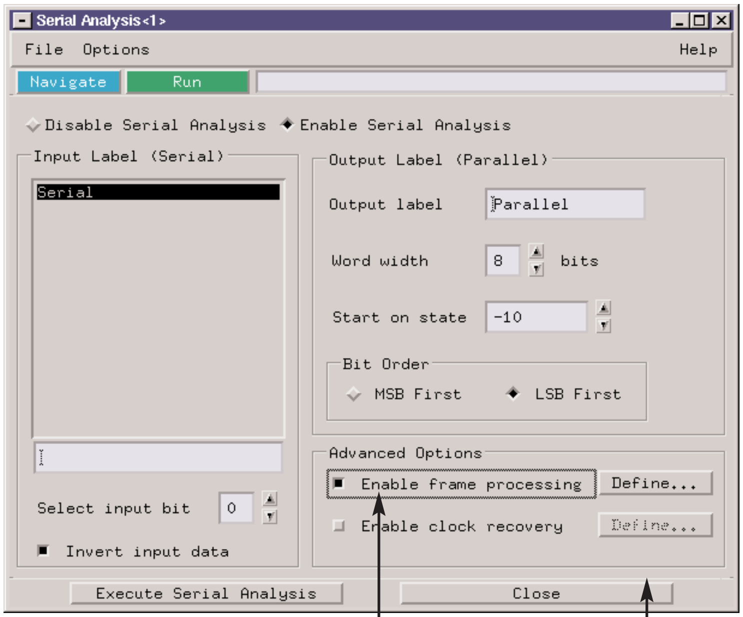
Figure 7.1 Timing for RS-232C Before it is Inverted.



To capture the data, you can set the trigger on any data value. Here we have set the trigger on the start bit. Once you have done this, you will need to create the Serial Analysis tool from the Analysis tab on the Listing display window. This will bring up the Serial Analysis window, as shown in Figure 7.2.

Select the Serial input label and create an output Parallel label with a word width of 8 bits, with the LSB as the first bit. This box will also allow you to invert the input data so that it becomes easy to analyze. Now comes the really neat part! You will have to specify to the logic analyzer how you would like the input frame to be processed. Select the box for frame processing on this dialog box and click on the define button.

Figure 7.2 Serial Analysis Window



◇ Disable Serial Analysis ◆ Enable Serial Analysis

Input Label (Serial)

Serial

Select input bit 0

■ Invert input data

Output Label (Parallel)

Output label Parallel

Word width 8 bits

Start on state -10

Bit Order

◇ MSB First ◆ LSB First

Advanced Options

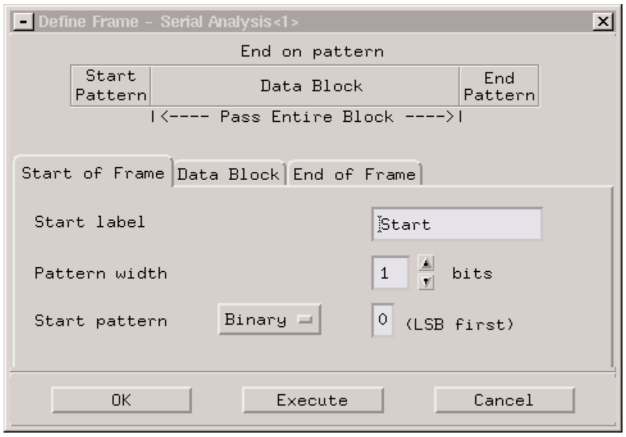
■ Enable frame processing Define...

□ Enable clock recovery Define...

Make sure this box is selected.

Then click on the Define button to bring up the Frame Parameters dialog boxes.

Figure 7.3 Start of Frame Tab

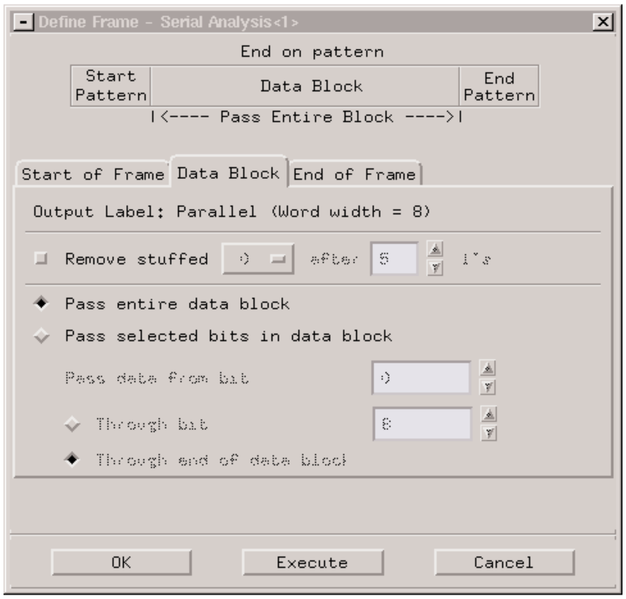


Start label: Start

Pattern width: 1 bits

Start pattern: Binary 0 (LSB first)

Figure 7.4 Data Block Tab



Output Label: Parallel (Word width = 8)

□ Remove stuffed 0 after 5 1's

● Pass entire data block

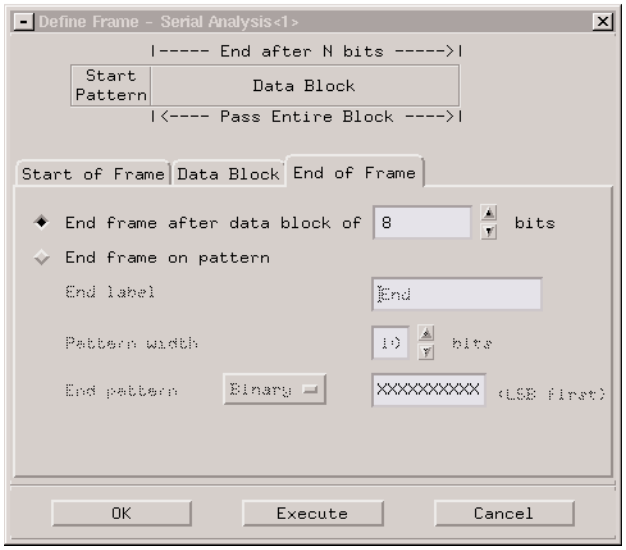
○ Pass selected bits in data block

Pass data from bit 0

○ Through bit 8

● Through end of data block

Figure 7.5 End of Frame Tab


Define Frame - Serial Analysis<1>
|----- End after N bits ---->|
Start Pattern
Data Block
|<---- Pass Entire Block ---->|
Start of Frame
Data Block
End of Frame
OK
Execute
Cancel

● End frame after data block of 8 bits

○ End frame on pattern

End label: End

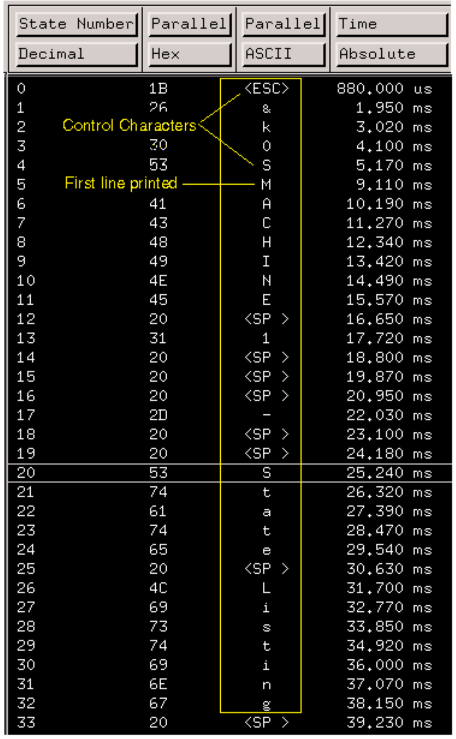
Pattern width: 10 bits

End pattern: Binary XXXXXXXXXX (LSB first)

In the dialog box for specifying the frame parameters, you can specify the start of the frame to have the label "Start", as shown in Figure 7.3. It is binary and has a width of 1 bit. Under the Data Block tab, uncheck the box for stuffed 0s because the RS-232C protocol does not do any bit stuffing. Since you do not need to perform any pattern manipulation, pass the entire data block through. In a similar fashion, under the End of Frame tab you will need to specify that the frame ends after 8 bits, as shown in Figure 7.5. Having done all this work, you are now ready to look at the serial data. Figure 7.6 shows a listing of the data, looking at just the Parallel labels. It shows the serial data coming down the output port.

As you can see, the Serial Analysis tool in the logic analyzer gives you the versatility to analyze and debug serial protocols.

Figure 7.6 Data Listing



State Number Decimal	Parallel Hex	Parallel ASCII	Time Absolute
0	1B	<ESC>	880.000 us
1	26	&	1.950 ms
2		k	3.020 ms
3	30	0	4.100 ms
4	53	S	5.170 ms
5		M	9.110 ms
6	41	A	10.190 ms
7	43	C	11.270 ms
8	48	H	12.340 ms
9	49	I	13.420 ms
10	4E	N	14.490 ms
11	45	E	15.570 ms
12	20	<SP >	16.650 ms
13	31	1	17.720 ms
14	20	<SP >	18.800 ms
15	20	<SP >	19.870 ms
16	20	<SP >	20.950 ms
17	2D	-	22.030 ms
18	20	<SP >	23.100 ms
19	20	<SP >	24.180 ms
20	53	S	25.240 ms
21	74	t	26.320 ms
22	61	a	27.390 ms
23	74	t	28.470 ms
24	65	e	29.540 ms
25	20	<SP >	30.630 ms
26	4C	L	31.700 ms
27	69	i	32.770 ms
28	73	s	33.850 ms
29	74	t	34.920 ms
30	69	i	36.000 ms
31	6E	n	37.070 ms
32	67	g	38.150 ms
33	20	<SP >	39.230 ms

Generating Files for Agilent Technologies Pattern Generator Using Third-Party EDA Tools

HINT 8

Electronic Design Automation (EDA) tools have become an integral part of most engineers' tool sets. You can use EDA tools for drawing, analyzing, simulating and documenting the circuits that you are working on. Wouldn't it be nice if you could leverage the work that you did in your EDA environment while debugging your prototype? The pattern generators that are integrated into the Agilent Technologies logic analyzers allow you to do this without much trouble.

With shrinking times-to-market, you typically cannot wait for your prototype to be complete before you start testing. Suppose your circuit board is ready but your third-party partner has not yet delivered your ASIC. The pattern generators available in the Agilent Technologies logic analyzers allow you to debug your circuit by replacing the missing components. All you need to do is program the pattern generator and hook it to the circuit where the ASIC belongs. The integrated pattern generator will stimulate your circuit while you analyze it using the logic analyzer.

SynaptiCAD, the creators of WaveFormer Pro, an EDA tool used for drawing, analyzing, simulating, and documenting timing diagrams, has worked closely with Agilent Technologies to make it easy for you to export your timing diagrams to Agilent's pattern generators. Once you have created your timing diagram using WaveFormer Pro, make sure that it includes a sampling clock and user-created signals because the signals are sampled using the first clock in the timing diagram. In addition, the diagram should include at least one documentation marker. The first documentation marker found in the timing diagram denotes the beginning of the main sequence (separating the initialization from the main section). If no documentation marker is present, only main sequences will appear (i.e., no initial sequences).

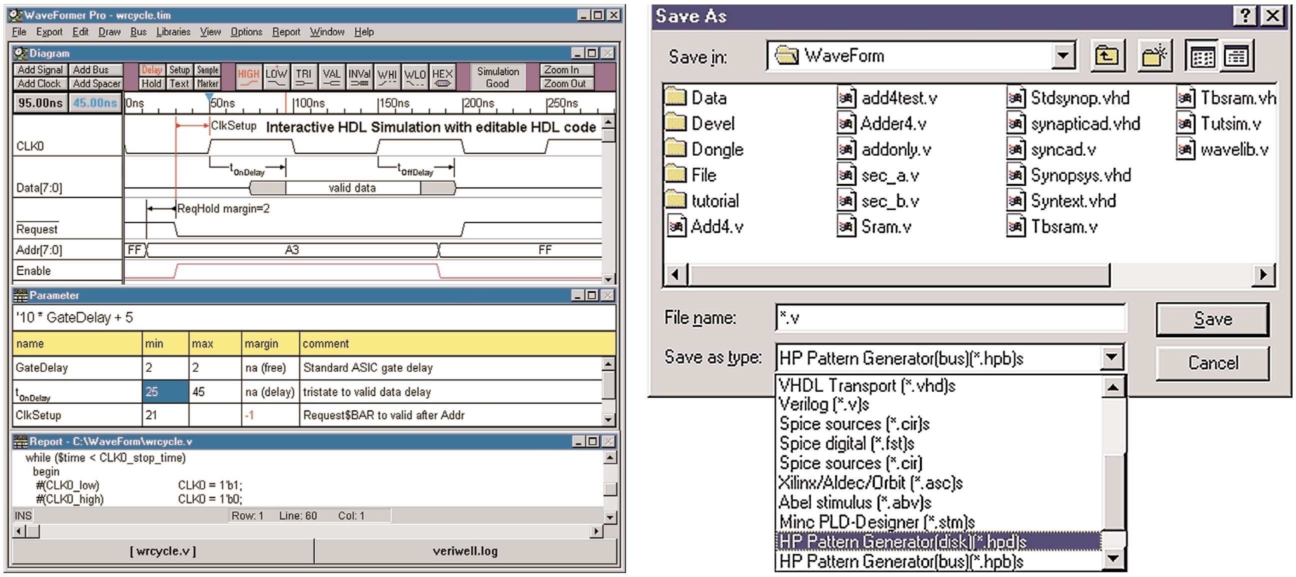
As shown in Figure 8.1, in order to export a timing diagram to a pattern generator, simply go the Export menu in WaveFormer Pro and select 'Export Signals As'. This will bring up a dialog box that will allow you to save your file either as HP pattern generator disk format (disk *.hpd) or HP pattern generator bus format (bus *.hpb). The *.hpd format is used for file transfer via network or diskette. The *.hpb format uses the GP-IB bus for file transfer.

It's also possible to generate input files for the pattern generator without using an EDA tool. You can either program the pattern generator using the system interface or you can type the pattern generator commands in an ASCII file and import them into the pattern generator.

Here is an example of a file that you can load into the pattern generator.

```
ASCII      000000
ASCDOWN
FORM: MODE FULL
LABEL 'LAB1',8
LABEL 'DATA',8
LABEL 'TEST',9
LABEL 'CLK',3
VECT #800000092
12 34 56 7
0 22 7 0
A0 33 00 1
*M
92 6F 00 1
CA CA 00 1
00 10 11 0
```

Figure 8.1 Exporting a File From WaveFormer Pro



'10 * GateDelay + 5

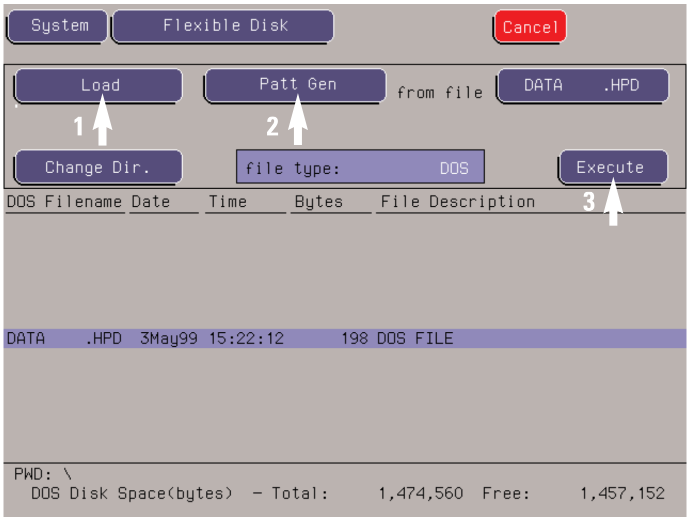
name	min	max	margin	comment
GateDelay	2	2	na (free)	Standard ASIC gate delay
$t_{OnDelay}$	25	45	na (delay)	tristate to valid data delay
ClkSetup	21		-1	Request$BAR to valid after Addr

```
  while ($time < CLK0_stop_time)
    begin
     #(CLK0_low)                CLK0 = 1'b1;
     #(CLK0_high)               CLK0 = 1'b0;
```

File name: *.v

Save as type: HP Pattern Generator(bus)(*.hpb)s

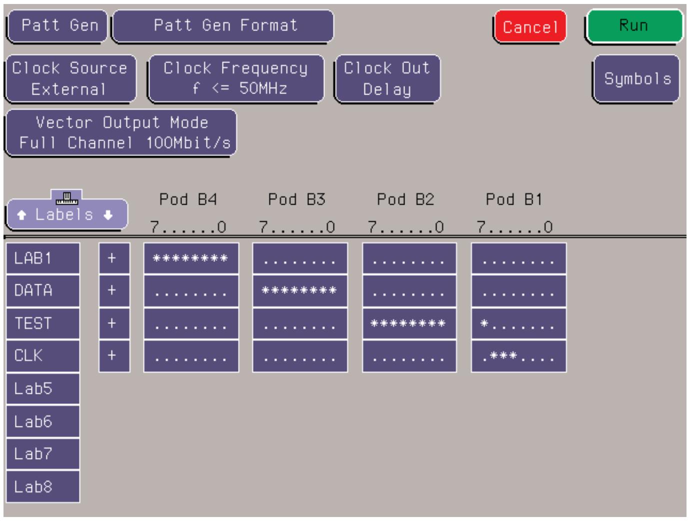
Figure 8.2 Selecting the File to Load Into the Pattern Generator



file type: DOS

```
DOS Filename Date      Time      Bytes   File Description
DATA    .HPD  3May99  15:22:12      198 DOS FILE

 PWD: \
   DOS Disk Space(bytes)  - Total:      1,474,560  Free:      1,457,152
```

Figure 8.3 Pattern Generator Pin Assignments



Clock Source External

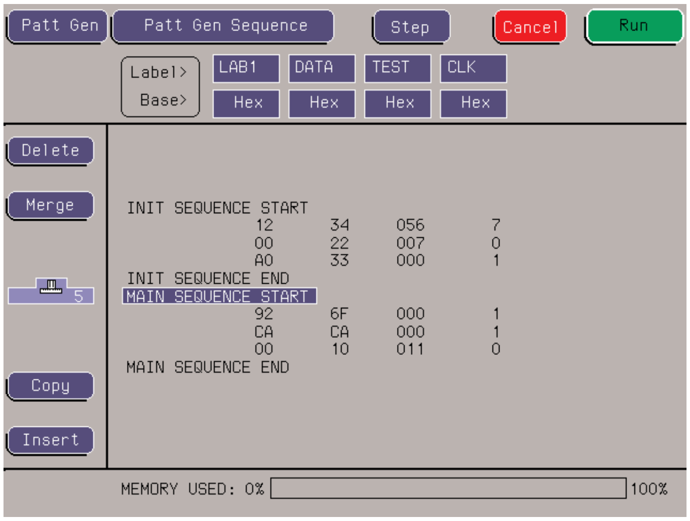
Clock Frequency f <= 50MHz

Clock Out Delay

Vector Output Mode Full Channel 100Mbit/s

		Pod B4 7......0	Pod B3 7......0	Pod B2 7......0	Pod B1 7......0
LAB1	+	********	....	....	....
DATA	+	....	********	....	....
TEST	+	....	....	********	*.......
CLK	+	....	....	....	.***....
Lab5					
Lab6					
Lab7					
Lab8					

Figure 8.4 Pattern Generator Sequence



Label>	LAB1	DATA	TEST	CLK
Base>	Hex	Hex	Hex	Hex

```
INIT SEQUENCE START
            12      34      056       7
            00      22      007       0
            A0      33      000       1
INIT SEQUENCE END
MAIN SEQUENCE START
            92      6F      000       1
            CA      CA      000       1
            00      10      011       0
MAIN SEQUENCE END
```

MEMORY USED: 0% 100%

In the Agilent Technologies 1670G logic analyzer, go to the System screen and select the file that you created above. As shown in Figure 8.2, select the Load button, then select the Pattern Generator button and click on the Execute button.

That is all you have to do. The data in the input file has been imported into your pattern generator! Figure 8.3 shows the pin assignments in the pattern generator and Figure 8.4 shows the pattern generator sequence that will result from your actions.

The combination of WaveFormer Pro and an Agilent Technologies logic analyzer makes a very powerful design and debugging tool. The above hint shows how to get data from WaveFormer Pro into Agilent's pattern generators. In addition, data captured by your logic analyzer can be imported into WaveFormer Pro. The circuit operations captured by the logic analyzers can be translated by WaveFormer Pro into stimulus files for driving the simulations to test your circuit design. For more information, see your logic analyzer user manual or go to **www.syncad.com**.

Logic Analyzer Families to Help You Get Your Job Done

Agilent Technologies 16700B Series Logic Analysis Systems

With the power of the latest technology and the familiarity of windows, the Agilent Technologies 16700B Series logic analysis systems offer a single solution for hardware, software and system debugging.

Hardware designers get measurement power plus processor execution control, register access and other tools to explore software-dependent hardware problems such as interrupt handling.

Software designers get debugging and analysis tools that overcome the drawbacks of traditional emulation, while providing an easier way to solve hardware-dependent software problems that only a logic analyzer can uncover.

System designers get time-correlated views showing system activity from analog signals all the way to source code. The Agilent Technologies logic analysis systems' cross-domain displays minimize the mysteries of hardware-software interaction, helping your team track symptoms back to root causes quickly and confidently.

The 16700B and 16702B are high-performance platforms for applications that use 32- or 64-bit microprocessors in multiprocessor systems; core-based ASICs; or systems on silicon.

For more information on Agilent Technologies logic analysis systems, visit **http://www.agilent.com/find/LAsystems**

Cost-Effective Solutions That Match Your Specific Application Needs

The Agilent Technologies 1670 Series benchtop analyzers offer cost-effective 150 MHz state analysis and 500 MHz timing analysis, and a color, flat-panel display with built-in VGA resolution. Oscilloscope, pattern generator, and deep memory features in the 1670 Series give you the capability to configure a solution that meets your demanding troubleshooting challenges.

Navigating through the user interface is made simple via your choice of either mouse or front-panel operation. An optional keyboard is also available.

Graphical trigger macros assist in making powerful measurements. Trigger set-ups can be selected from a categorized list of trigger macros. Each macro is shown in graphical form and has a written description. Macros can be chained together to create a custom trigger sequence.

For more information on Agilent benchtop logic analyzers, visit **http://www.agilent.com/find/LAbenchtop**

Figure 8.5 Agilent Technologies 1670G Series Benchtop Logic Analyzers

Agilent Technologies'
Test and Measurement Support,
Services, and Assistance

Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlay Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise

Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage

Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

www.agilent.com

By internet, phone, or fax, get assistance with all your test & measurement needs

Online assistance:
www.agilent.com/find/assist

Phone or Fax
United States:
(tel) 1 800 452 4844

Canada:
(tel) 1 877 894 4414
(fax) (905) 206 4120

Europe:
(tel) (31 20) 547 2000

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Latin America:
(tel) (305) 267 4245
(fax) (305) 267 4286

Australia:
(tel) 1 800 629 485
(fax) (61 3) 9272 0749

New Zealand:
(tel) 0 800 738 378
(fax) 64 4 495 8950

Asia Pacific:
(tel) (852) 3197 7777
(fax) (852) 2506 9284

Product specifications and descriptions in this document subject to change without notice.

Printed in USA May, 2000
5968-5700E

Agilent Technologies
Innovating the HP Way