Timesaving Tips for Digital Debugging with a Logic Analyzer

Application Note

New Designs, New Headaches

New digital devices have become progressively more powerful by incorporating faster microprocessors and higher-speed memory systems. They have also become more complex, more sensitive to signal quality, and more time consuming to troubleshoot. Tight schedules do not allow extra time for debugging. This note explains how to speed up troubleshooting by using more of the features in your logic analyzer and oscilloscope.

Introduction

Today's designs are harder to debug because there are more factors that can go wrong. Consider the emergence of DDR2, and even DDR3, memory systems. Their high-speed digital edges are very sensitive to signal integrity.

Fast edges can also increase crosstalk. On older designs, you could take the stability of circuit board traces for granted. High edge rates, however, can make them act as transmission lines, sending and receiving interference. Faster edges also create larger transient currents. Dynamic currents from these transients can induce ground bounce and power distribution artifacts. Fast edge faults usually appear in your signals as intermittent glitches. Before you can solve such problems, you have to find the effects, characterize them, and work back to determine their cause.

Other project-stopping culprits include timing violations, software driver errors, and race conditions. All of these can create faults in your design and they are exceptionally difficult to resolve.

In this application note we will discuss several timesaving tips to help improve your productivity in the digital debugging stage as you address fast edge effects and intermittent challenges.

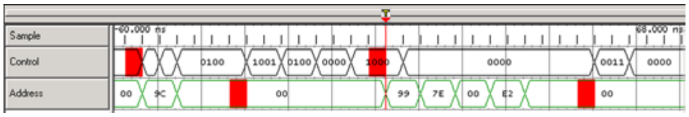


Figure 1. Sample ticks, Control bus and Address bus showing red glitch flags.

Tip #1: Look for Glitches

What is a glitch?

If your device is malfunctioning, a good way to begin troubleshooting is to check for glitches. Glitches are very narrow pulses that your system may, or may not, interpret as logic changes. Most problems will appear as glitches in one or more of your signals. The effect of glitches on system operation is unpredictable. They can be your first sign of a wide variety of device faults, including race conditions, termination errors, driver errors, timing violations, and crosstalk.

Locating the problem

Since the problems that glitches cause are often intermittent, they can be very hard to resolve. A solid approach is to combine classic top/down troubleshooting with the specific advantages of your test instruments. Start wide, with a macro view of device operation, and then begin focusing in on problems.

Glitch hunting is a good illustration of this approach. On the macro level, your Tektronix logic analyzer allows you to perform glitch triggering on buses that are hundreds of signals wide. The logic analyzer checks every signal for glitches. Red bars on the bus timing diagrams show glitch locations for further analysis. On the micro level, your Tektronix oscilloscope can help characterize the problem by revealing exactly what the glitch looks like. Using the iView™ measurements on the TLA5000 or TLA700 Series logic analyzers, you can combine your logic analyzer and your oscilloscope into a single system and progressively “zoom in” on the problem.

Using the top/down methodology, we can step through the debugging process to more easily find glitches and troubleshoot the problem. In the following four steps we will identify two different glitches and their likely source.

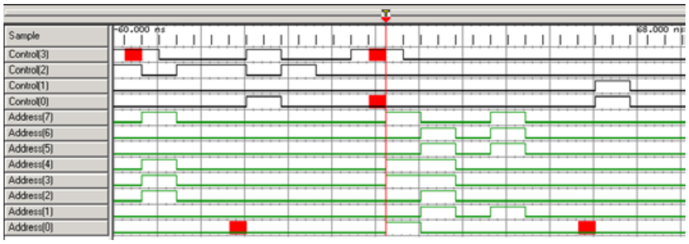


Figure 2. 4-bit Control bus and 8-bit Address bus expanded showing red glitch flags on individual signals.

Step I: Examine the bus

Begin by focusing on what works and look globally for faults. Your logic analyzer’s bus timing waveform will flag any glitches that occur.

When it comes to looking for intermittent effects, such as glitches, use a logic analyzer with a long record length. Tektronix logic analyzers can have up to 256 Mb of deep timing capability. The logic analyzer’s bus timing waveform can examine all the signal lines of the bus at once. If the logic analyzer detects a glitch on any of the lines, it will flag the bus and the time location.

In Figure 1, the top waveform is Sample, which shows the sample ticks that represent the logic analyzer’s deep timing sampling rate of up to 2 GHz (500ps). The next two lines are the bus waveforms – the 4-bit Control bus and the 8-bit Address bus. The red glitch flags that appear on both bus waveforms signify that there was more than one transition between the sample points at those locations.

Step II: Examine the lines

Now focus on where the problems are. Use the logic analyzer’s timing signal waveform to display the individual lines of the bus and flag where glitches occur. Expand the logic analyzer’s deep timing signal waveform, which can also be up to 512 Mb deep.

In Figure 2, the analyzer has expanded the Control bus into its four individual signals and the Address bus into its eight individual signals. The red glitch flags from the bus waveform in Figure 1 are now shown as glitch flags on signal lines Control (3) and Control (0), and as two glitches on the Address (0) signal line.

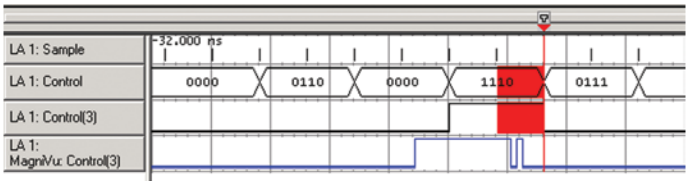


Figure 3. MagniVu™ waveform of Control (3) showing glitch.

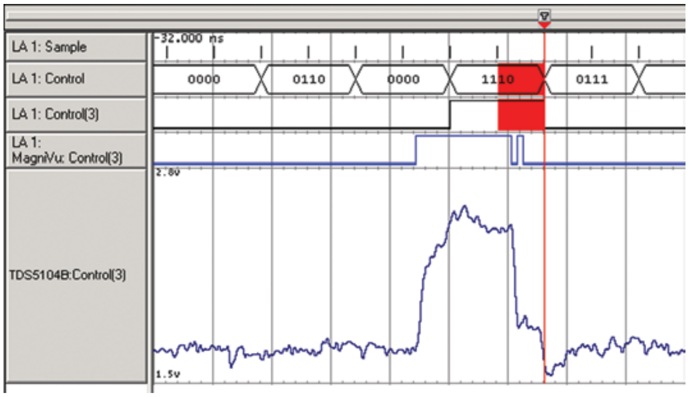


Figure 5. iView oscilloscope trace showing analog representation of Control (3) signal.

Step III: Take a closer look

Use a high-resolution timing view to examine the faults in fine detail. See how they relate to other events or faults. In addition to deep timing, Tektronix logic analyzers have high-resolution MagniVu™ 20 ps (50 GHz) timing capability that runs simultaneously with the deep timing capability. MagniVu waveforms can display all channels in high-resolution up to a 128 Kb memory depth. It is like having two logic analyzers in one: a deep timing logic analyzer and a high-resolution timing logic analyzer, both using the same probes.

In this example, it appears there may be two different problems causing glitches. First, focus on the Control (3) signal line and show the MagniVu trace for the Control (3) signal. Figure 3 shows that because of its higher resolution, MagniVu waveforms can reveal that the glitch only appears at the end of a digital pulse – not at the beginning of the pulse and not by itself. This is an important clue to the cause of the fault. Proceeding to Step IV will reveal the likely cause.

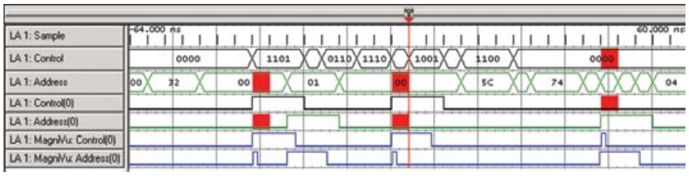


Figure 4. Control (0) and Address (0) lines with MagniVu traces showing glitches caused by crosstalk.

Now focus on the second glitch identified in Control (0). Use MagniVu high-resolution timing to examine the two remaining flagged lines, Control (0) and Address (0). Figure 4 shows that since MagniVu waveforms are examining the signals at a much higher resolution, 125 ps, it is able to discern far narrower glitches on both lines. Note that the glitch and a pulse occur at the same time on both signal lines. That often indicates crosstalk between the two signals, but you need to take a different kind of close-up look to be sure. Proceeding to Step IV will also reveal more about this.

Step IV: Examine the analog waveform

Discover what the glitch really looks like by comparing the analog and digital perspectives using your oscilloscope and the logic analyzer's iView™ capability. iView capability allows the logic analyzer to trigger the oscilloscope at exactly the right time to capture the glitch. With iView measurements, the logic analyzer also time-correlates the data and displays both the analog and digital waveforms on the logic analyzer's display.

Looking at signal line Control (3), Figure 5 shows an analog iView display of the glitch. Considering both domains, it is obvious that something is distorting both the rising and falling edges of the pulse. The rising edge does not droop enough to trigger a logic transition and therefore has not appeared as a glitch. The falling edge, however, bounces high enough to pass through the logic threshold and sometimes act as a logic transition. Although the bus clock is not particularly fast, the LVPECL logic family that the circuit uses still introduces fast edges. The bouncing on the pulse edges suggests a termination problem on the circuit board magnified by the greater sensitivity of the fast edges.

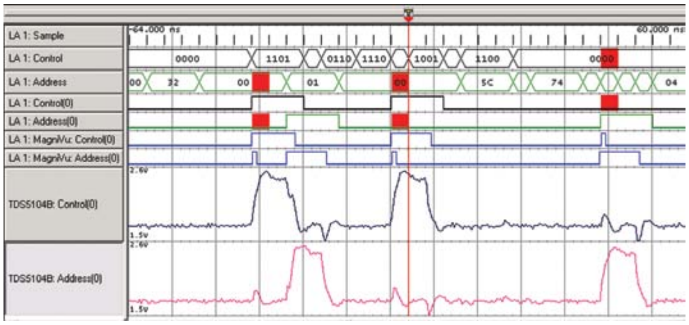


Figure 6. Crosstalk between Control (0) and Address (0) shown using iView measurement.

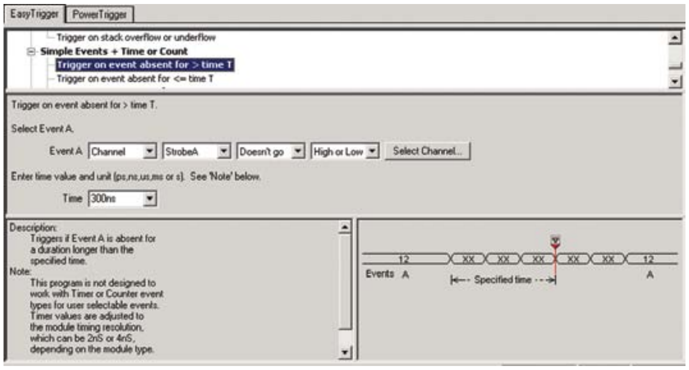
Testing the earlier crosstalk hypothesis on Control (0) and Address (0), Figure 6 shows that for every leading edge of one signal there is a corresponding positive voltage pulse on the other. This makes crosstalk between Control (0) and Address (0) the obvious diagnosis. Crosstalk can easily occur on adjacent runs or pins within the package. High frequency signals and clock edges have a greater susceptibility to crosstalk effects than lower frequency signals. This implies that even design practices that were consistently successful at slower frequencies can be a contributor to failures at higher frequencies.

Although the buses in these two examples were quite narrow, logic analyzer glitch triggering can be used on buses with hundreds of signals. The analyzer checks every signal line for glitches. If it flags a glitch, start focusing in on the problem until you determine the source of the glitch.

Tip #2: Use a Timeout Trigger to See What Isn't Happening

If your prototype's errors continue to elude you, go back to what is working. Consider the design's total operation. For example, you may have a signal that is supposed to do something periodically. Is it? This can be your key to capturing the problem, but only if your logic analyzer trigger can trigger on "nothing."

For example, you could have a strobe that provides a "local clock" function for a group of data lines. If the strobe does not act, or does not act often enough, then the device is not functioning as planned. Alternately, you could have embedded a "watchdog" or "heartbeat" pulse right into your system. As long as the heartbeat is pulsing then you know that the section is working. If the heartbeat stops, then you know when the failure became critical.


EasyTrigger
PowerTrigger
Trigger on stack overflow or underflow
Simple Events + Time or Count
Trigger on event absent for > time T
Trigger on event absent for <= time T
Select Channel...
12
XX
XX
XX
XX
XX
12
Events A
Specified time
A

Trigger on event absent for > time T.

Select Event A.

Event A: Channel | StrobeA | Doesn't go | High or Low

Enter time value and unit (ps,ns,us,ms or s). See 'Note' below.

Time: 300ns

Description:
Triggers if Event A is absent for a duration longer than the specified time.

Note:
This program is not designed to work with Timer or Counter event types for user selectable events. Timer values are adjusted to the module timing resolution, which can be 2nS or 4nS, depending on the module type.

Figure 7. EasyTrigger timeout trigger definition screen.

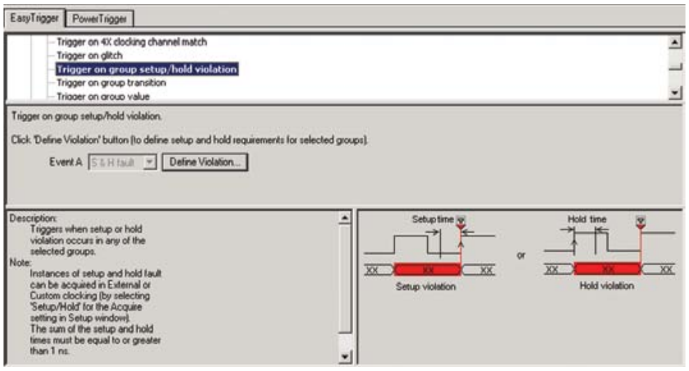
Fortunately, it is very easy to set your logic analyzer to trigger on "nothing" and to give you a detailed display of the state of the system.

Triggering on the absence of activity is called Timeout Triggering. You can set the analyzer to watch a line or group of lines; if nothing happens, if there are no logic changes in the period of time you specify, then the logic analyzer will trigger. You can also decide how deep your record of activity will be. Figure 7 shows the Timeout Trigger screen from the EasyTrigger menu in the logic analyzer. You can set it up in seconds.

The cause of the failure may take place well before the heartbeat actually stops. The system could have continued on for some time before the effect of the fault became critical. By setting the trigger deep in the capture memory, you can acquire up to 512 Mb of pre-trigger information. Then you can analyze the record for possible causes.

Tip #3: Look for Setup/Hold Violations

Meeting setup and hold requirements is crucial for having a reliable product. Inadequate setup/hold margin is a common source of intermittent problems in designs. Searching for setup/hold violations can be very time consuming using the traditional approach of probing a clock and data line using an oscilloscope. The TLA logic analyzer can automate searching for setup/hold violations for you by triggering on and displaying any user-defined setup/hold violation on all your signals at once. Use the power of the TLA setup/hold violation trigger to watch all the signals in your system at the same time. The TLA will trigger on any violation and display all of the setup/hold violations in your system.



EasyTrigger | PowerTrigger

- Trigger on 4X clocking channel match
- Trigger on glitch
- **Trigger on group setup/hold violation**
- Trigger on group transition
- Trigger on group value

Trigger on group setup/hold violation.

Click 'Define Violation' button (to define setup and hold requirements for selected groups).

Event A: S & H fault | Define Violation...

Description:
Triggers when setup or hold violation occurs in any of the selected groups.

Note:
Instances of setup and hold fault can be acquired in External or Custom clocking (by selecting 'Setup/Hold' for the Acquire setting in Setup window).
The sum of the setup and hold times must be equal to or greater than 1 ns.

Figure 8. Setup/hold violation trigger parameter selection dialog.

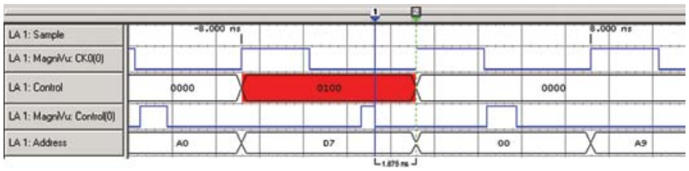
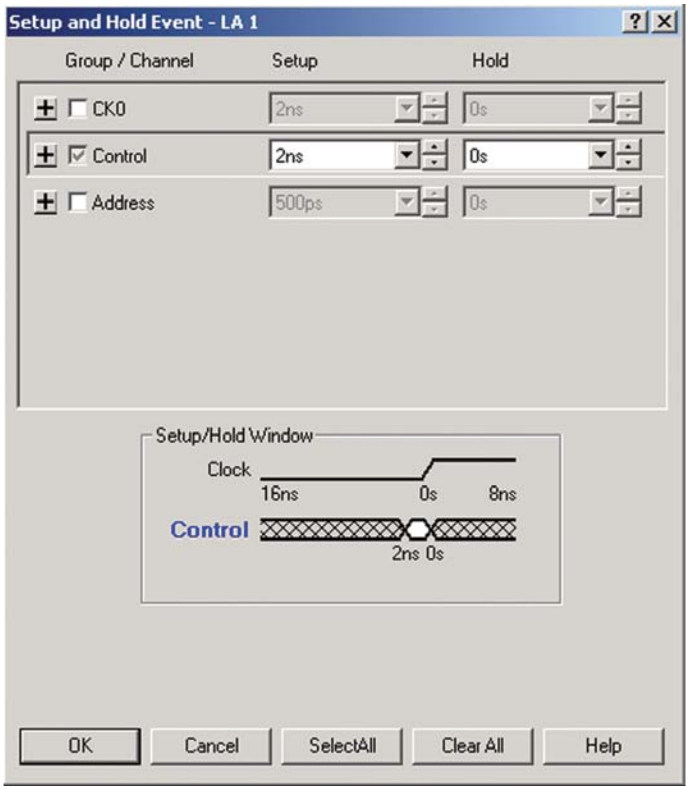


Figure 10. Trigger display of set-up/hold violation on Control (0).

You can test for setup/hold violations directly by using your logic analyzer's Setup and Hold Violation trigger. Figure 8 shows the EasyTrigger setup/hold violation triggering setup menu.

Figure 9 shows the dialog where you can specify the setup/hold violation parameters for the signals you would like to monitor. The TLA can monitor every signal in your system for setup/hold violations simultaneously.

Figure 10 shows the result of the logic analyzer triggering on a setup violation. In this case, the signal only had a set-up time of 1.875 ns which is less than the required 2 ns. Now that you have identified the problem, you can work toward a solution.

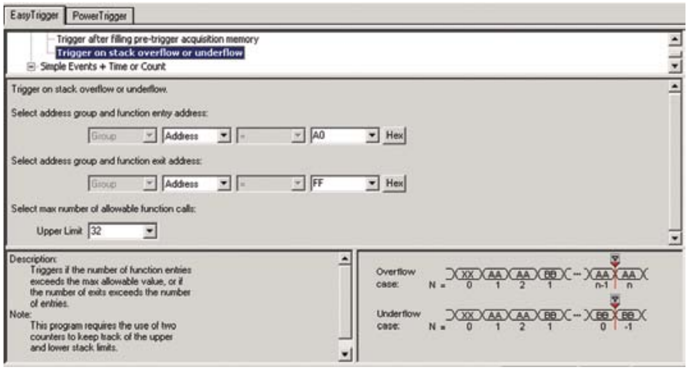


Setup and Hold Event - LA 1

Group / Channel	Setup	Hold
☐ CK0	2ns	0s
☑ Control	2ns	0s
☐ Address	500ps	0s

Figure 9. EasyTrigger setup/hold violation trigger definition screen.

All synchronous digital circuits have setup/hold requirements. Confirming setup/hold compliance should be part of your troubleshooting routine. Your logic analyzer's EasyTrigger Menu makes it a simple test to set up.



Trigger after filling pre-trigger acquisition memory
Trigger on stack overflow or underflow
Simple Events + Time or Count

Trigger on stack overflow or underflow.

Select address group and function entry address:
Group | Address | = | A0 | Hex

Select address group and function exit address:
Group | Address | = | FF | Hex

Select max number of allowable function calls:
Upper Limit 32

Description:
Triggers if the number of function entries exceeds the max allowable value, or if the number of exits exceeds the number of entries.
Note:
This program requires the use of two counters to keep track of the upper and lower stack limits.

Figure 11. EasyTrigger stack overflow or underflow trigger definition screen.

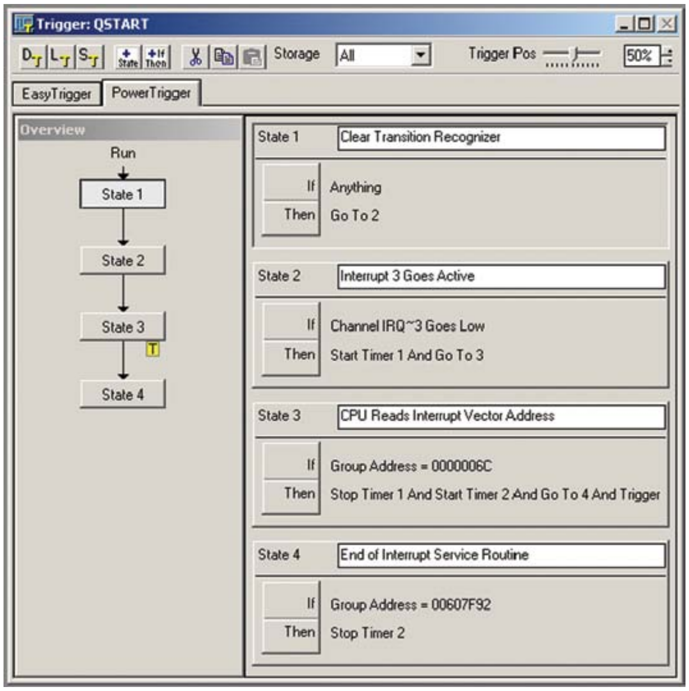
Tip #4: Look for Overflow and Underflow Errors

FIFOs can used to transfer data from one sub-system to another - especially those running in different clock domains. Common FIFO problems include overrun and underrun. Overrun occurs when the system writes data faster than it can be readout. Underrun occurs when data is read from the FIFO faster than it can be written. Diagnosing overrun and underrun related errors can be if the right type of triggering is not used.

Luckily, triggering on overrun and underrun errors can be very simple to set up. Tektronix EasyTrigger includes “Trigger on stack overflow or underflow” as one of its many predefined triggers.

For example, consider the interrupt handler of a microprocessor. Interrupts are requests designed to take a processor out of its normal assignment and to address something from the periphery. The list of interrupts forms a stack in memory that waits for the processor to deal with them. If interrupts come faster than the processor can handle them, then the requests “overrun” the stack and are lost. The processor may end up in an unknown state or simply not do what it is supposed to.

Figure 11 shows the setup screen for overflow or underflow trigger. It lets you specify what kind of event or events the logic analyzer should track. The trigger uses counters to track the number of events that are incrementing and decrementing the stack. Again, having a sufficiently long record length is important. The condition that caused the error may have occurred well before symptoms appear.



Storage All
Trigger Pos 50%

State 1 Clear Transition Recognizer
If Anything
Then Go To 2

State 2 Interrupt 3 Goes Active
If Channel IRQ~3 Goes Low
Then Start Timer 1 And Go To 3

State 3 CPU Reads Interrupt Vector Address
If Group Address = 0000006C
Then Stop Timer 1 And Start Timer 2 And Go To 4 And Trigger

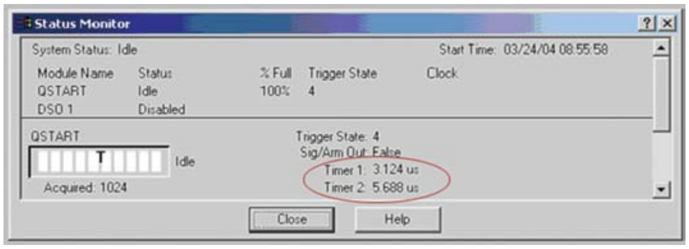
State 4 End of Interrupt Service Routine
If Group Address = 00607F92
Then Stop Timer 2

Figure 12. Logic analyzer PowerTrigger set up for two timers measuring interrupt latency.

Tip #5 Checking Interrupt Latency

Interrupts are used by the microprocessor to respond to external events in a timely manner. Errors can occur if the interrupt latency is too long. Interrupt latency is the time it takes a microprocessor to respond and to process the interrupt. Interrupt latency is divided into the time from the initial request to the start of the interrupt software routine and the length of time to complete the interrupt software routine handling the interrupt.

You can measure interrupt latency by using two logic analyzer trigger timers. As shown in Figure 12, the first timer, in states 2 and 3, measures the length of time from when the hardware requests interrupt service to the start of the software interrupt routine. The second timer, in states 3 and 4, measures the length of time it takes for the software interrupt routine to complete. The logic analyzer uses two timers, two comparators, one edge detector and four trigger states to implement this PowerTrigger setup.



```
System Status: Idle                              Start Time: 03/24/04 08:55:58
 Module Name   Status          % Full  Trigger State    Clock
 QSTART        Idle            100%    4
 DSO 1         Disabled

QSTART                                Trigger State: 4
                  Idle                 Sig/Arm Out: False
                                          Timer 1: 3.124 us
  Acquired: 1024                          Timer 2: 5.688 us
```

Figure 13. Two logic analyzer trigger timers measuring interrupt latency.

In Figure 13, the first timer measured 3.124 µs from the time the hardware requested interrupt service to the start of the interrupt service routine. The second timer measured 5.688 µs for the service routine to complete serving the interrupt.

Examples of interrupt errors are: interrupts not being serviced because higher priority interrupts or non-maskable interrupts are inhibiting lower priority interrupts from being serviced, incorrect interrupt priorities assigned to the interrupting tasks, interrupt service routine with non-reentrant interrupt code that responds a second interrupt before it completes servicing the first interrupt, interrupts occurring too fast, and interrupt software routines taking too long to complete their operation.

Summary

Your logic analyzer can make your debugging process faster and more direct. Use a top/down methodology that takes advantage of your logic analyzer's special features. Work from a broad picture of how your prototype functions to a close-up view focused on the faults. Special triggers enable logic analyzers to quickly test for common problems. When you combine your logic analyzer's digital perspective with your oscilloscope's analog view, you make it much easier to characterize the source of the fault.

Better Tools

Although logic analyzers and oscilloscopes have long been the tools of choice for digital troubleshooting, not every designer knows how capable they can be. Logic analyzers speed up debugging and verification by wading through the information stream to trigger on circuit faults and capture related events. Oscilloscopes reveal how signal integrity problems can create false logic transformations by peering behind idealized digital timing diagrams and showing the raw, analog waveforms. Today's instruments, such as Tektronix logic analyzers, are even more powerful. They offer enhanced capabilities, including record lengths up to 512 Mb deep, MagniVu 50 ps resolution, triggers designed for troubleshooting, and the ability to work together with Tektronix oscilloscopes. With the TLA Series' iView™ integrated digital-analog view, you can see time-correlated digital and analog signals on your logic analyzer display.

To fully integrate your Tektronix TLA Series logic analyzer and oscilloscope into a single troubleshooting system, use the iLink™ Tool Set. This Tektronix-exclusive capability provides comprehensive digital and analog insight to help you quickly find and characterize faults. The iLink Tool Set— available on the TLA6000 and TLA7000 Series – includes:

- iCapture™ simultaneous digital and analog acquisition through a single logic analyzer probe
- iView™ time-correlated, integrated logic analyzer and oscilloscope measurements on one display
- iVerify™ multi-channel bus analysis and validation testing using oscilloscope-generated eye diagrams

Contact Tektronix:

ASEAN / Australasia	(65) 6356 3900
Austria*	00800 2255 4835
Balkans, Israel, South Africa and other ISE Countries	+41 52 675 3777
Belgium*	00800 2255 4835
Brazil	+55 (11) 3759 7600
Canada	1 (800) 833-9200
Central East Europe, Ukraine and the Baltics	+41 52 675 3777
Central Europe & Greece	+41 52 675 3777
Denmark	+45 80 88 1401
Finland	+41 52 675 3777
France*	00800 2255 4835
Germany*	00800 2255 4835
Hong Kong	400-820-5835
India	000-800-650-1835
Italy*	00800 2255 4835
Japan	81 (3) 6714-3010
Luxembourg	+41 52 675 3777
Mexico, Central/South America & Caribbean	52 (55) 56 04 50 90
Middle East, Asia and North Africa	+41 52 675 3777
The Netherlands*	00800 2255 4835
Norway	800 16098
People's Republic of China	400-820-5835
Poland	+41 52 675 3777
Portugal	80 08 12370
Republic of Korea	001-800-8255-2835
Russia & CIS	+7 (495) 7484900
South Africa	+27 11 206 8360
Spain*	00800 2255 4835
Sweden*	00800 2255 4835
Switzerland*	00800 2255 4835
Taiwan	886 (2) 2722-9622
United Kingdom & Ireland*	00800 2255 4835
USA	1 (800) 833-9200

*** If the European phone number above is not accessible, please call +41 52 675 3777**

Contact List Updated 25 May 2010

For Further Information
Tektronix maintains a comprehensive, constantly expanding collection of application notes, technical briefs and other resources to help engineers working on the cutting edge of technology. Please visit **www.tektronix.com**

11/10 EA/WWW 57W-17683-3

Tektronix®