

## Using an Oscilloscope for Debugging Serial Buses

by Doug Beck, Senior Usability Engineer, Agilent Technologies

Oscilloscopes now provide many of the features of protocol analyzers such as decode listings and protocol specific triggering. However, a good question is “If protocol analyzers already exist, why is this capability needed?”

First of all, having a single instrument that provides the capabilities of an oscilloscope, logic analyzer and protocol analyzer provides lots of advantages. There is also the convenience of bench space: why crowd your bench with three instruments when one will do? And even if you happen to have all three instruments handy, it is a hassle to turn on another instrument and set it up. A single *All-in-one* instrument saves time.

Secondly, having a single instrument that provides both serial and analog capabilities means that serial problems can be traced back to their root cause. For example, if there is inter-symbol interference on a PCI-Express bus, the problem will show up as an invalid packet (and invalid symbol) in the decode listing. Just point at the problem in the listing and the corresponding waveform is shown on the screen. Without both serial and analog capabilities, it is difficult to tell what caused the invalid symbol.

Another benefit of using oscilloscopes to debug serial buses is easy activity. There’s no need for a standard port or special IO for connectivity. In addition, oscilloscope probes are passive while protocol analyzers typically provide re-transmit and re-timing. If physical layer problems exist, connecting the protocol analyzer can mask them.

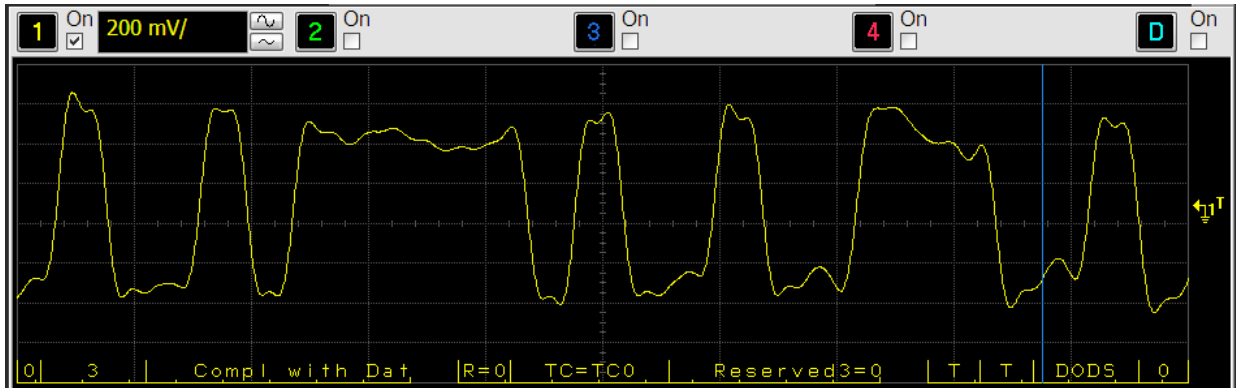
Finally, oscilloscopes are able to support a wide variety of protocols such as I<sup>2</sup>C, SPI, RS232, USB and PCI-Express. So the oscilloscope doesn’t just replace a single protocol analyzer: it replaces a number of them. It is very common for engineers to attempt to *mentally* decode waveforms because they don’t have the right protocol analyzer at hand. This new capability eliminates this difficult process forever.

### Key Functionality

In order to debug a serial bus, it is essential to have both decode as well as triggering. Depending upon the protocol, the meaning of *decode* can mean a variety of things. For example, in a high-speed serial protocol such as PCI-Express, decode can mean anything from symbols embedded in the waveform to a packet listing. In lower speed protocols like SPI, decode is simply a listing of the payload values.

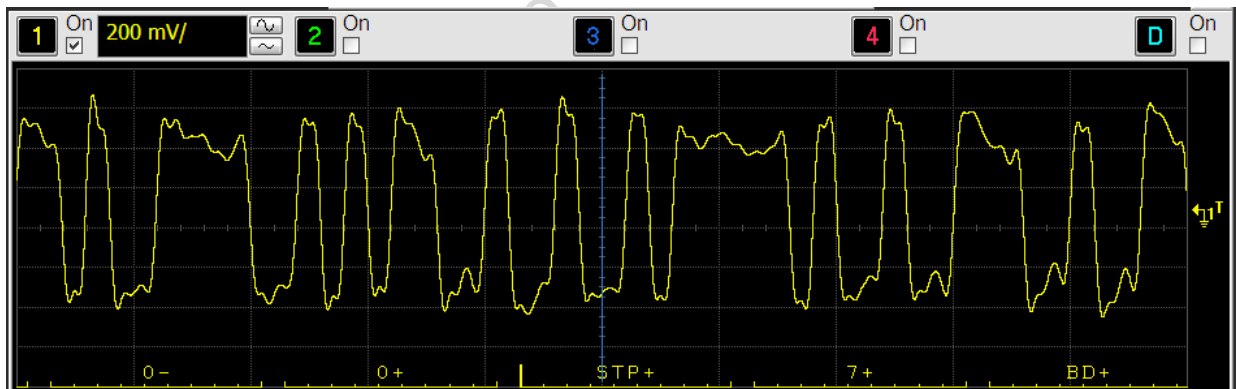
## Embedded Decode

Embedded Decode uses the oscilloscope's waveform display to overlay the decode information in the graticule. This provides very tight correlation between a waveform and the corresponding decode. This is functionality that protocol analyzers do not provide. It also eliminates the common problem of having to mentally decode the waveform. (It may sound strange, but many engineers have told me that they do this on an oscilloscope). An example of an embedded decode is shown in Fig. 1.



**Fig. 1: Example Of Embedded Decode Of PCI-Express Fields**

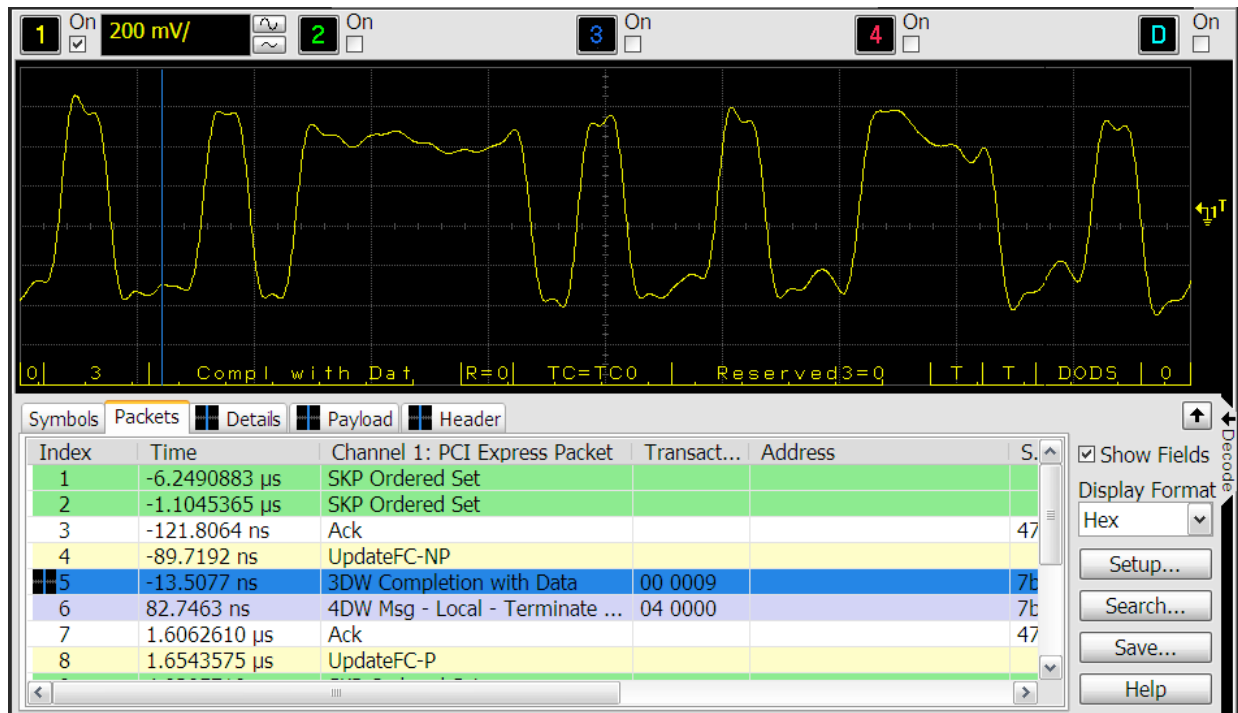
Note that for high-speed protocols, the embedded decode can either be in symbols (such as 8B/10B symbols) or packets. This is two different levels of abstraction. An example of embedded symbols is shown in Fig. 2.



**Fig. 2: Example Of Embedded Decode Of 8B/10B Symbols**

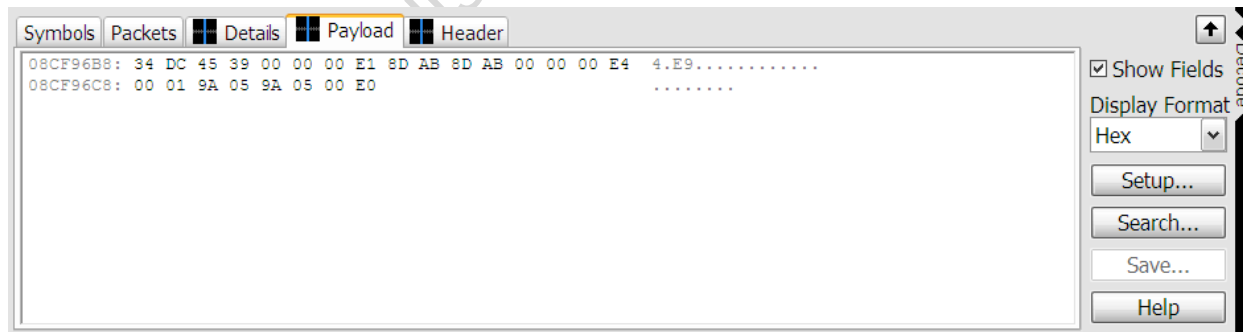
## Decode Listing

A listing is a very efficient method for displaying data because it shows a complete packet on a single line (see Fig. 3). This is the display that is used on protocol analyzers. However, on an oscilloscope, there is correlation between the waveforms and the selected packet. Notice there is a selected packet (highlighted in blue) and there is also a blue line in the waveform display showing the corresponding waveform.



**Fig. 3: Decode Listing Of Packets Correlated With Waveforms**

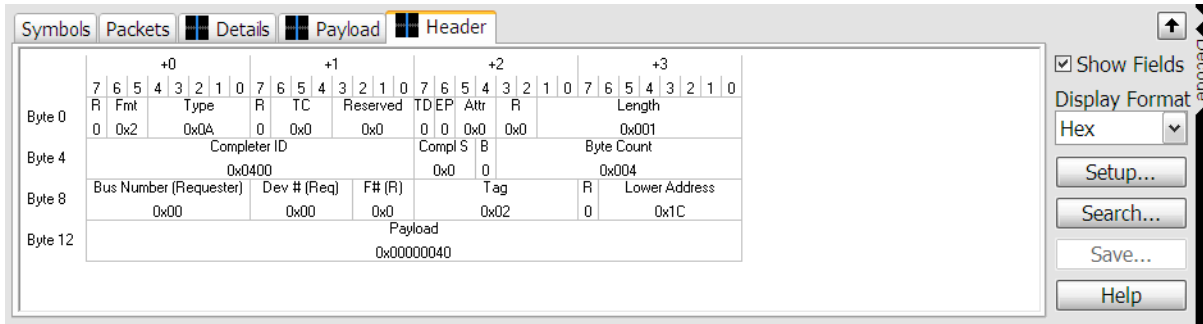
Some information can easily be shown in a column of a table but some cannot. The most obvious example is a payload, which can consist only of a couple of bytes or dozens of bytes. In order to be able to view a payload, use the Payload tab (see Fig. 4). This shows all of the bytes in this payload (which happens to be high-speed USB) along with an ASCII display. Having a separate Payload tab is essential because it allows entire payloads to be displayed regardless of length. Users can also change the base (hex/binary/decimal) to whatever format is preferred for them.



**Fig. 4: Payload Tab Shows Far More Information Than Can Fit In Column Of Table**

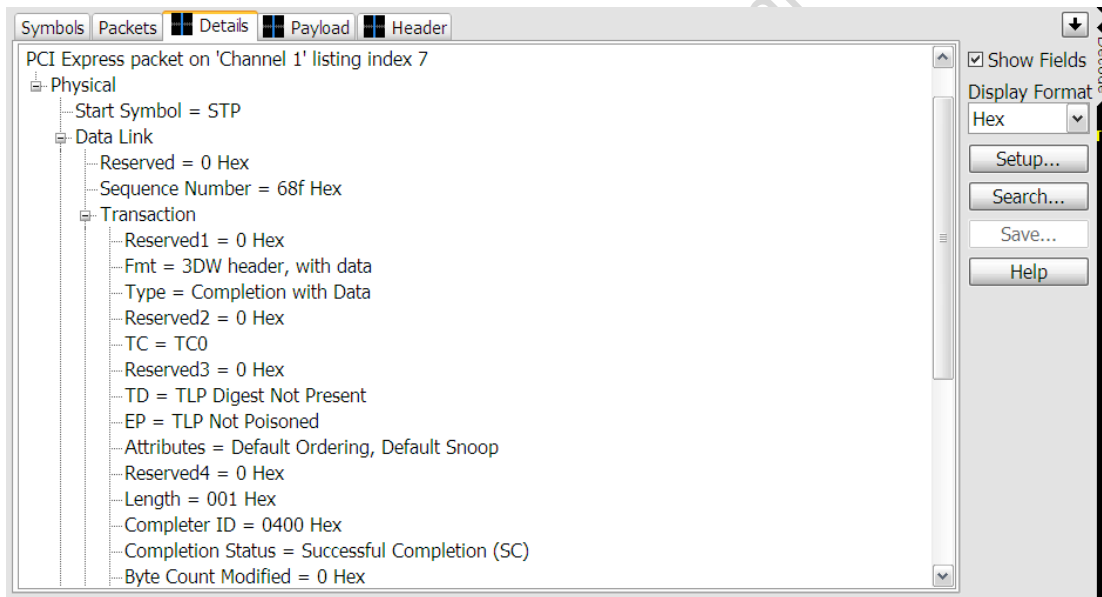
When we did research on how engineers like to see serial data, we found that they could not agree on a single format to be used all of the time. Showing them a variety of formats they requested all of them. The point they made is that a single view of the data does not cover all situations and a variety of formats would give them flexibility. Some relatively simple protocols like I<sup>2</sup>C, RS232, or SPI require nothing more than a listing of packets and a Payload tab. But high-speed protocols like PCI-Express, SATA, SAS, and MIPI require more views of the data.

One of the formats that users requested is the Header tab (see Fig. 5). The header is broken down by fields in format similar to the one used in protocol specifications. Having a view of the data that matches the way the engineers were taught the protocol is very important.



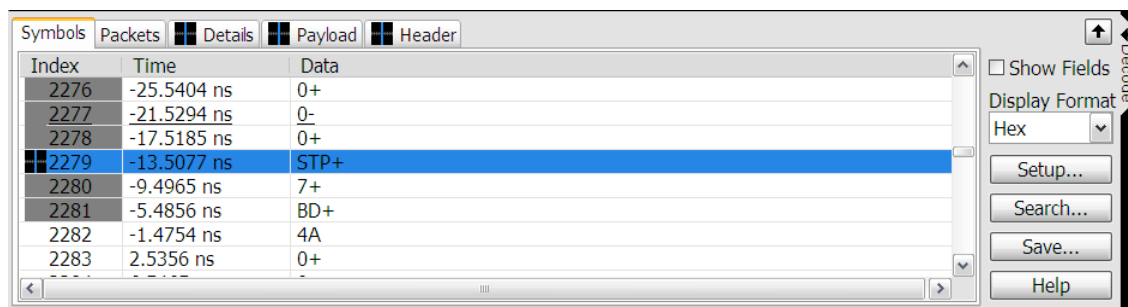
**Fig. 5: Header View Of Packet Matches Way They Are Shown Protocol**

Another format that the engineers requested is a hierarchical view called the Details tab. This recognizes that there can be a hierarch in protocols (such as fields at different layers of the protocol). An example (Fig. 6) shows where the Physical, Data Link and Transaction Layers are.



**Fig. 6: Details Tab Shows Physical/Data Link/Transaction Layers Of High-Speed Protocol**

The last view that the Engineers requested is a Symbol view (see Fig. 7).

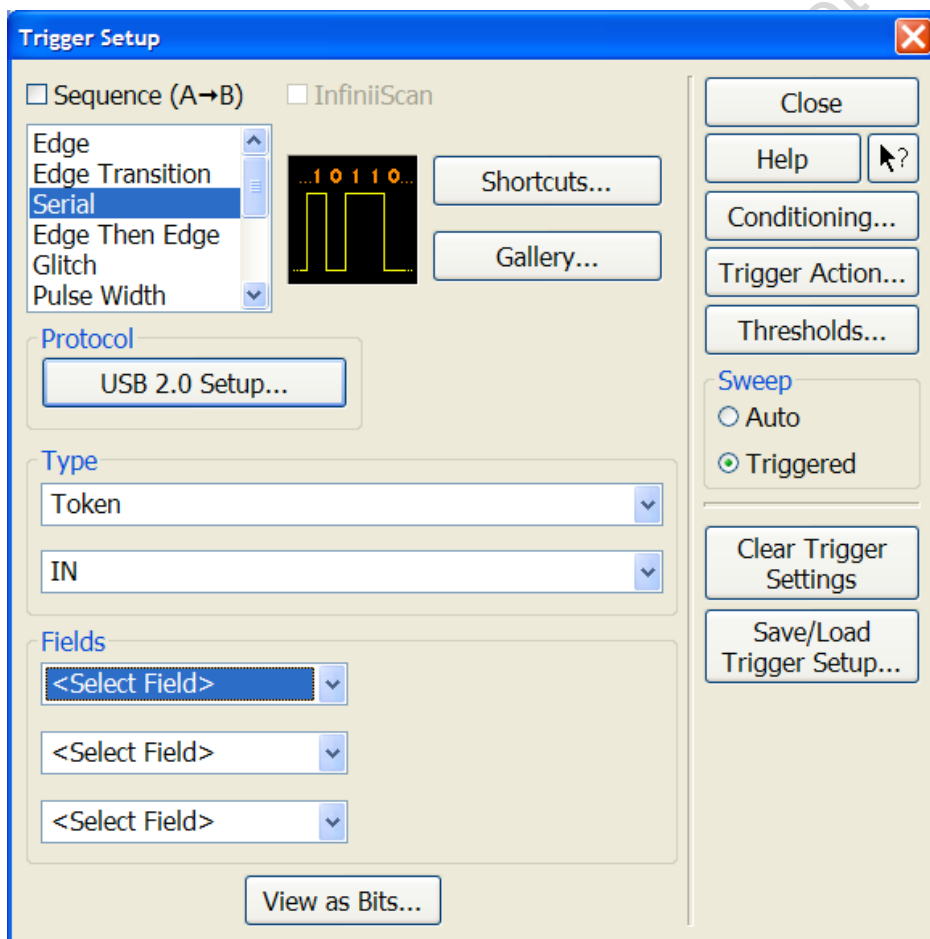


**Fig. 7: Example Of Symbol Listing**

## Triggering

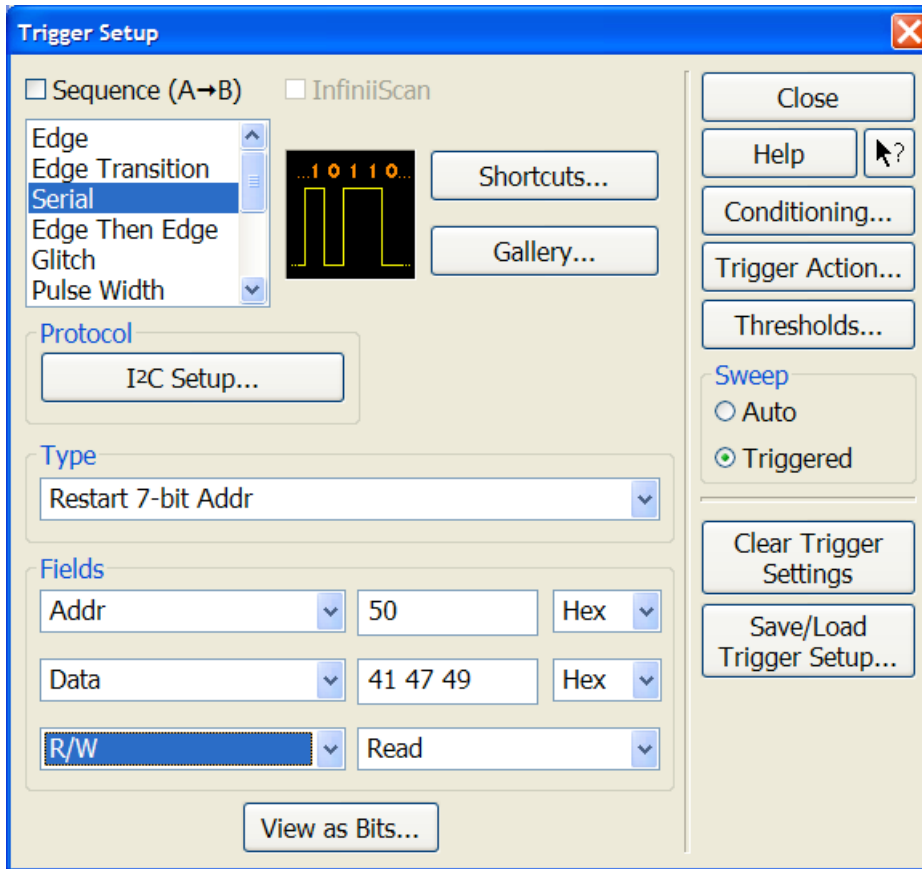
While doing research on oscilloscopes and serial protocols, we asked engineers if serial decode, or triggering, was more important. Most engineers simply could not answer the question because both are essential to debugging a serial bus. After all, what good is it to see the bus if you can't find what you want? That is why oscilloscopes provide both serial decode and triggering.

Every oscilloscope user is familiar with common triggers such as edges and pulse widths, but an entirely new suite of triggers is required for serial. It is not enough to simply trigger on some aspect of the waveform: the oscilloscope needs to trigger on aspects of the protocol itself. For example, a common trigger that users desire for USB would be on IN tokens. Or, for I<sup>2</sup>C, an example would be a Read that occurs as result of a restart. These are known as packet types because they allow users to trigger only on a specific type of packet (see Fig. 8)



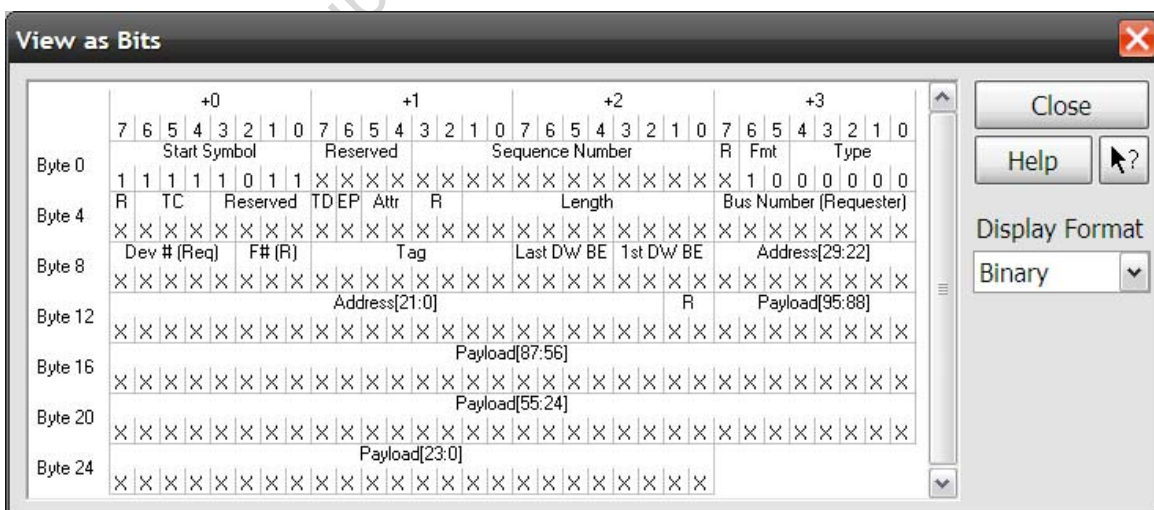
**Fig. 8: A USB Trigger On All Packets Where Token Is IN**

However, there are many cases where simply triggering on a packet type simply isn't enough: It is essential to be able to trigger on specific fields within a packet just like protocol analyzers do. An example is to trigger on a specific data and address value in I<sup>2</sup>C (see Fig. 9).



**Fig. 9: I<sup>2</sup>C trigger On Read Of Address 50 With Data 41 47 49**

While we were doing our research, we discovered that even though engineers wanted the ability to specify packets and fields as triggers, they also wanted the ability to see exactly what the definition of the trigger is. That is the purpose of the “View as Bits” dialog shown in Fig. 10.



**Fig. 10: View as Bits Dialog Shows Bit-By-Bit Definition Of Serial Trigger**

A final type of triggering is called the Symbol Sequence. Again, this is for high-speed serial buses where engineers want a list of specific symbols to trigger on. An example (Fig. 11) begins with a K code (SDP), ends with a K code (END) but uses data values for the rest of the sequence.

	0	1	2	3
0	SDP	BC	XX	BC
4	86	45	89	END

**Fig. 11: Example Of Symbol Sequence**

## Setting Up A Serial Bus

Setting up a serial bus on an oscilloscope is not necessarily simple. There are a variety of factors that must be specified correctly in order for both triggering and serial decode to work. These factors include sample rate, memory depth, trigger levels, measurement thresholds and clock recovery (high-speed buses only). For simplicity, the Agilent 9000 series oscilloscopes include an Auto Setup button that will automatically setup all of these parameters for the user.

## Sample Rate

In general there is an optimal sample rate defined by the speed of each serial bus. Naturally, if the sample rate goes below this level, the serial decode cannot be performed. Traditionally, however, oscilloscopes have optimized their sample rate based upon the current setting of the time per division. This optimizes the view of the current waveform (which is best for finding signal integrity problems). This is a problem when viewing serial decode because zooming in on a waveform will cause the serial decode to disappear. Therefore, until users want to investigate a signal integrity issue, it is recommended that the sample rate be set to a fixed point.

## **Memory Depth**

Memory Depth is another key issue. If the depth is very large, this can slow down the update rate of the oscilloscope. However, making the depth too small can result in the trace consisting of a single fragment of a packet (which cannot be decoded). Ironically, the serial triggering hardware will still work fine because it sees the complete packet even though the depth is not great enough to save it. In many cases, the users will need to experiment to find their optimal memory depth. While triggering helps them to find a packet of interest, it is also essential to have a fair amount of context around the packet to see what happened before and after.

## **Trigger Levels**

Trigger Levels are familiar to oscilloscope users because they are used in virtually all oscilloscope triggers. Serial triggering is relatively simple because, if the trigger is set to the middle of the waveform, it will generally work fine. Trigger Levels are probably the easiest issue to deal with in serial debugging.

## **Measurement Thresholds**

Measurement Thresholds are very similar to Trigger Levels even if they are somewhat less familiar. The biggest confusion with Measurement Thresholds is how they are used. Trigger Levels are used by the oscilloscope hardware to trigger. Measurement Thresholds are used by the oscilloscope software to figure out the decoding. The two functions are independent: an oscilloscope that is triggering just fine on a serial bus may fail to decode. Compounding the problem is that Trigger Levels are prominently displayed in the main window of every oscilloscope, while Measurement Thresholds are buried. To alleviate this problem, Agilent automatically sets up both the Trigger Levels and Measurement Thresholds whenever the Auto Set button is pressed. However, if the user manually changes a Trigger Level (such as by turning the front panel knob), serial triggering may no longer work. In a similar manner, if a user changes any of his Measurement Thresholds, serial decode may no longer work. The best advice for most engineers is if either triggering or decode doesn't work, press the Auto Setup button again. In most cases, that will take care of the problem.

## **Clock Recovery**

Most high-speed serial buses use clock recovery instead of a separate clock signal. The Autoset function will attempt to choose a method for a specific protocol, but it may be up to the user to specify the correct method.

## Limitations

While oscilloscopes can perform many of the functions of a protocol analyzer, there are some limitations. The biggest is memory depth. Oscilloscopes acquire samples to be able to recreate waveforms while protocol analyzers only need to recreate packets. Therefore, the number of packets that can be stored by protocol analyzers is vastly greater.

A second limitation of oscilloscopes is performance. Because protocol analyzers have dedicated hardware to decode packets, while oscilloscopes use software decode, for large memory depths the oscilloscopes will be sluggish. (A notable exception to this rule is the Agilent 7000 series which have dedicated hardware for serial decode.)

Protocol Analyzers often have specialized performance analysis capabilities that go well beyond just triggering and decoding. For these capabilities, a protocol analyzer is required.

## Conclusion

While oscilloscopes are essential tools for electrical engineers to view analog waveforms, they are also a great serial debugging tool. Key advantages of oscilloscopes include correlation of waveforms to packets, support for a wide variety of protocols, and multiple instruments on one box. This makes the oscilloscope even more versatile than the past.

## About The Author

Doug Beck is a Senior Usability Engineer with Agilent Technologies in Colorado Springs. He holds a PhD in Industrial Engineering from The University of Michigan. He has 12 patents in communications and electronics. His career has focused on user interface design and customer research for engineering applications.

