Logic Analyzers Interpret Protocols

Jon Titus, Senior Technical Editor

Small pocket-size logic analyzers can capture and display signals from parallel buses, but many engineers now use these capable instruments mainly to examine I2C, SPI, UART, CAN, and other protocols. A PC, connected via a USB cable, displays these serial signals and helps engineers interpret them.

The logic analyzers also can "watch" several different serial buses simultaneously. "Say a PC sends a serial command to a controller," said Jerry Merrill, CEO at Tech Tools. "The controller sends back a reply and then forwards data to an I2C device and adjusts a PWM output. A logic analyzer lets you see all the communications and what happens as a result. One customer bought our DV3100 just to monitor 18 UART debug channels on one board."

"No matter how much data you acquire, it's important to center attention on something of interest and triggering helps you do that," said Merrill. "We dedicate 70 percent of our logic resources to implement flexible trigger conditions. But we hide the complexity behind a graphical user interface [GUI], although users can see our trigger code, too. The GUI lets them graphically 'connect' inputs to level, edge and range detectors, logic gates, and counter/sequencers to create complex trigger conditions. If you had to define trigger conditions with an equation or C code, things would get overwhelming quickly."

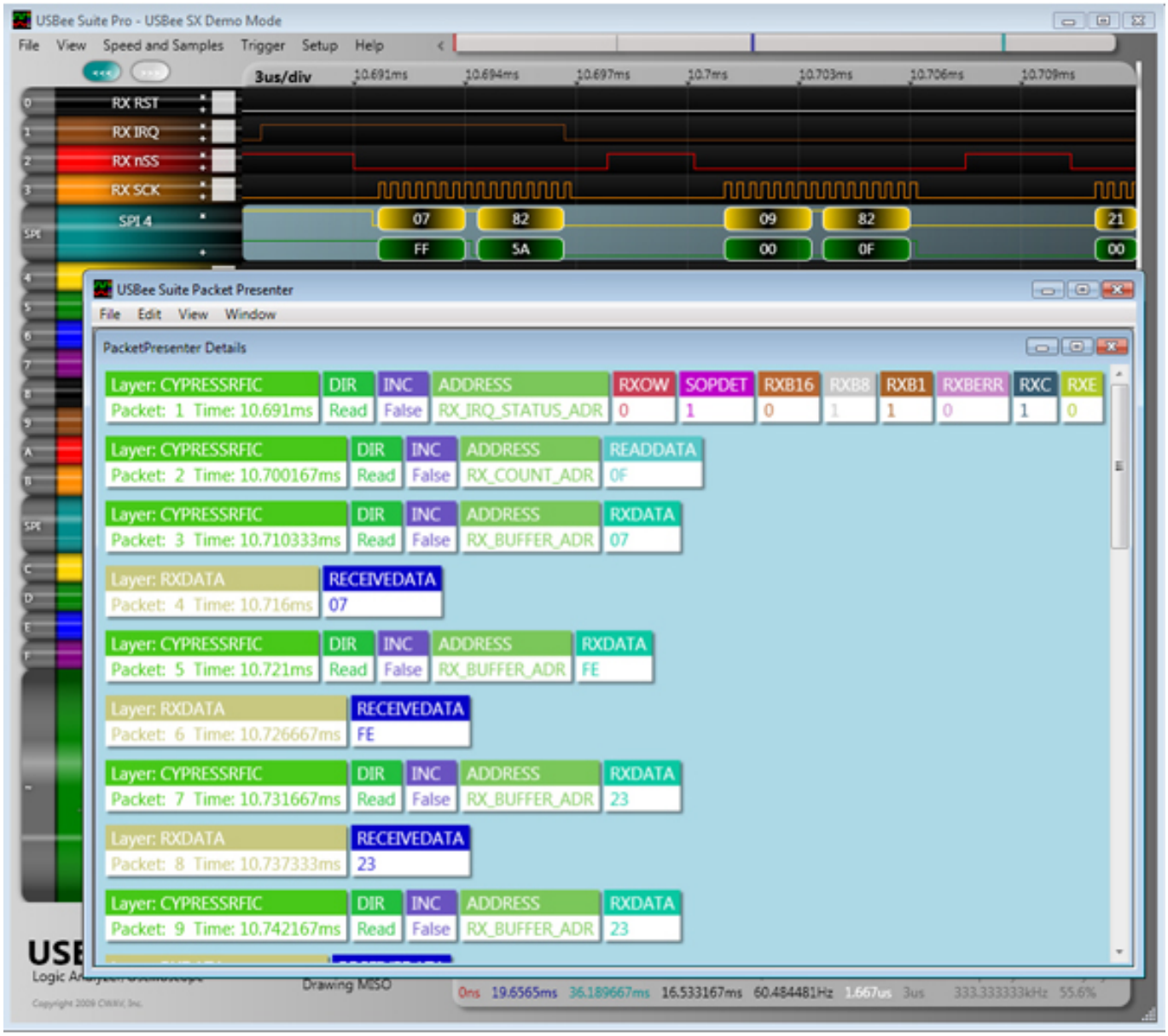

Figure 1. This USBee Packet Presenter screen shows the raw logic signals (background) and the interpretation based on a specific protocol that shows packet addresses, control bits, and register commands.

According to Tim Harvey, president of CWAV, which manufactures the USBee logic analyzers, many times people set their trigger conditions and hope to capture the fault they looked for, but they often must change trigger conditions many times to home in on a problem. "Instead, we let people capture millions of samples. Then they use a 'smart search' to find all occurrences of the fault, not just the one that would have triggered a logic analyzer."

"A customer had problems with I2C communications so he captured thousands of I2C transactions and found 13 or 14 examples of the problem in one trace," said Harvey. "Then he compared the problems to determine their cause. Usually a trigger shows where a fault occurred once, but you need to see when the fault happens again and again because problems are not always consistent."

"Complicated triggers aren’t a high priority for us," said Joe Garrison, the owner of

Saleae. "When you can capture multiple seconds of data and then examine it separate from the acquisition step, triggering becomes less important. So far, level and edge triggers satisfy most of customers' needs. We do have an arbitrary state-machine based trigger written, but haven't implemented the GUI for it yet."

After engineers capture data, they can look at individual bits to start. "Typically, engineers first work to get an MCU's internal I/O devices to operate properly," added Merrill of Tech Tools. "They have to tweak timer settings and control bits. Then when a I/O device works properly, they want to ensure their firmware sends the proper command over an I2C bus, for example, so they need analysis tools that decode bus information and make it easy to understand, usually by displaying hexadecimal or other values instead of bits."

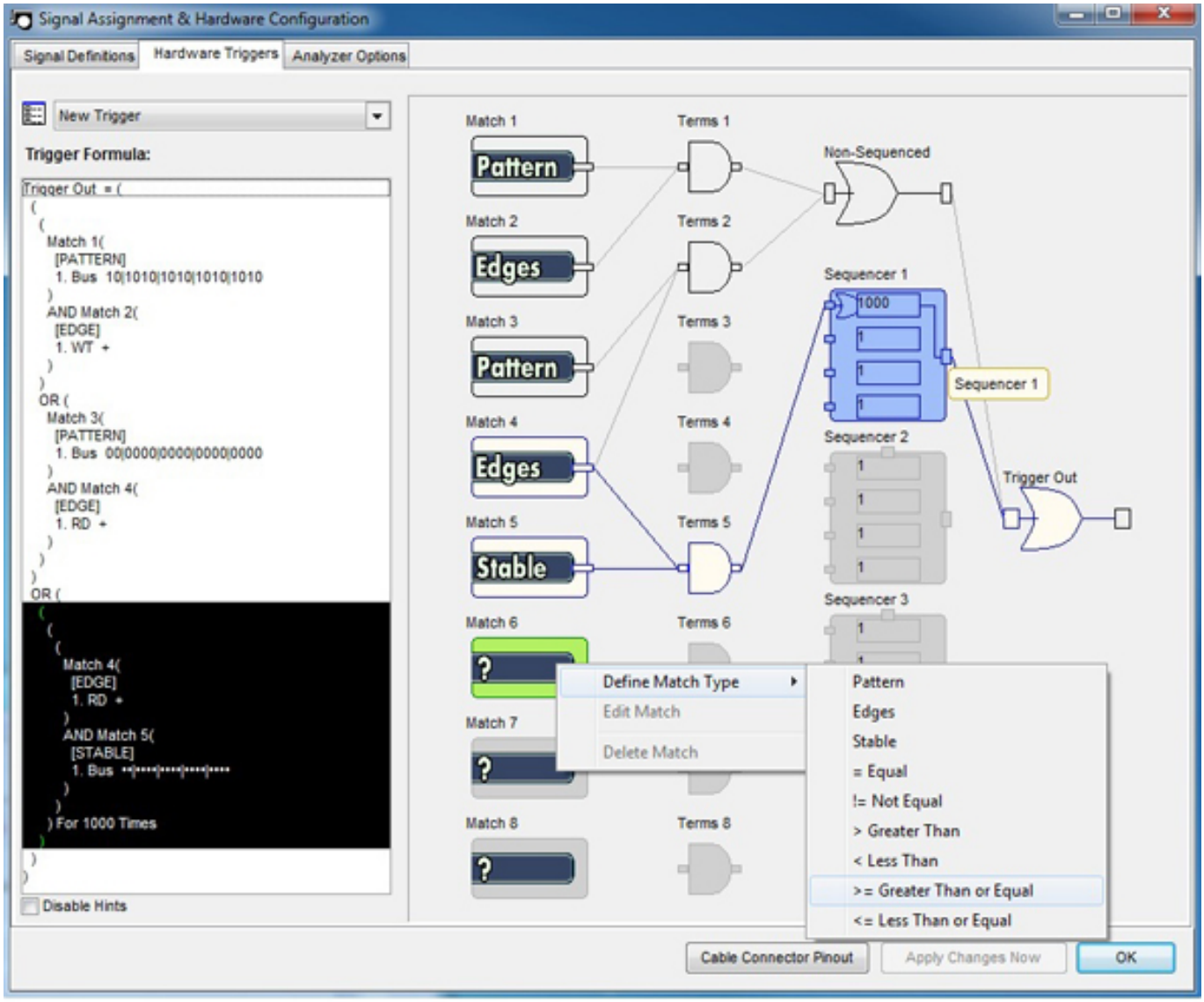

Figure 2. The Signal Assignment & Hardware Configuration display for the Tech Tools logic analyzers shows the types of sophisticated trigger actions engineers can configure.

"Decoding the packets is a big step, but it gets you only part of the way, said CWAV's Harvey. "After your communications work properly you want to look at the protocol so you can see commands and responses in a packet format. Rather than

see 47h, for example, you want to see 'Read XYZ,' or you might want to see an ADC output as a voltage value. The PacketPresenter software performs this function."

"Engineers can adapt that programmable interpreter to their data and command exchanges on a bus." added Harvey. "When you see a packet the same way a data sheet describes it, a problem cannot hide for long. You can debug at the level--from bits to packets--appropriate for what you want to investigate."

But suppose you have a custom protocol. "You create a packet-definition text file that specifies what a packet means, how you start a packet, how you end it, and the packet's fields, parameters, and so on," said Harvey. "Large companies can set up their proprietary packet-definition files and share them with design teams."

"We expect to have a high-level SDK [software-development kit] available in a few weeks," said Garrison of Saleae. "Then people can create their own protocol analyzers. Basically, you build a DLL [dynamic link library] and it comes up on the list of protocol analyzers you have for the software. The DLL takes captured data from the main application, crunches it, and produces results you can overlay on the display. We use the same SDK to create our own protocol analyzers."

"When you aim to decode a custom protocol, you need something predictable and controllable to test it against, so we also provide a way to generate simulated data," added Garrison. "The SDK lets you generate plausible data that matches your protocol and any settings it might use. We use this tool ourselves so we can test all sorts of protocol variations without the need to create special hardware."

Acknowledgement

Thanks go to Harrison Young at PC Test Instruments, who provided background information for this article.

Source URL (retrieved on *05/23/2013 - 5:18am*):
http://www.ecnmag.com/articles/2011/01/logic-analyzers-interpret-protocols?qt-video_of_the_day=0