How to Simplify Debugging and Documenting Wireless Applications

By (Troy Gentry, Applications Engineer Principal, Cypress Semiconductor Corp.)

Executive Summary

Wireless application debugging sessions are prone to analyzer capture buffer overruns and mountainous serial-based protocol to unravel. Often this leaves engineers with the troublesome time-consuming task of manually stitching together and decoding elaborate back-to-back captures to analyze the inner workings of a single event. This article details the ease of debugging and documenting your wireless application utilizing PC-based tools.

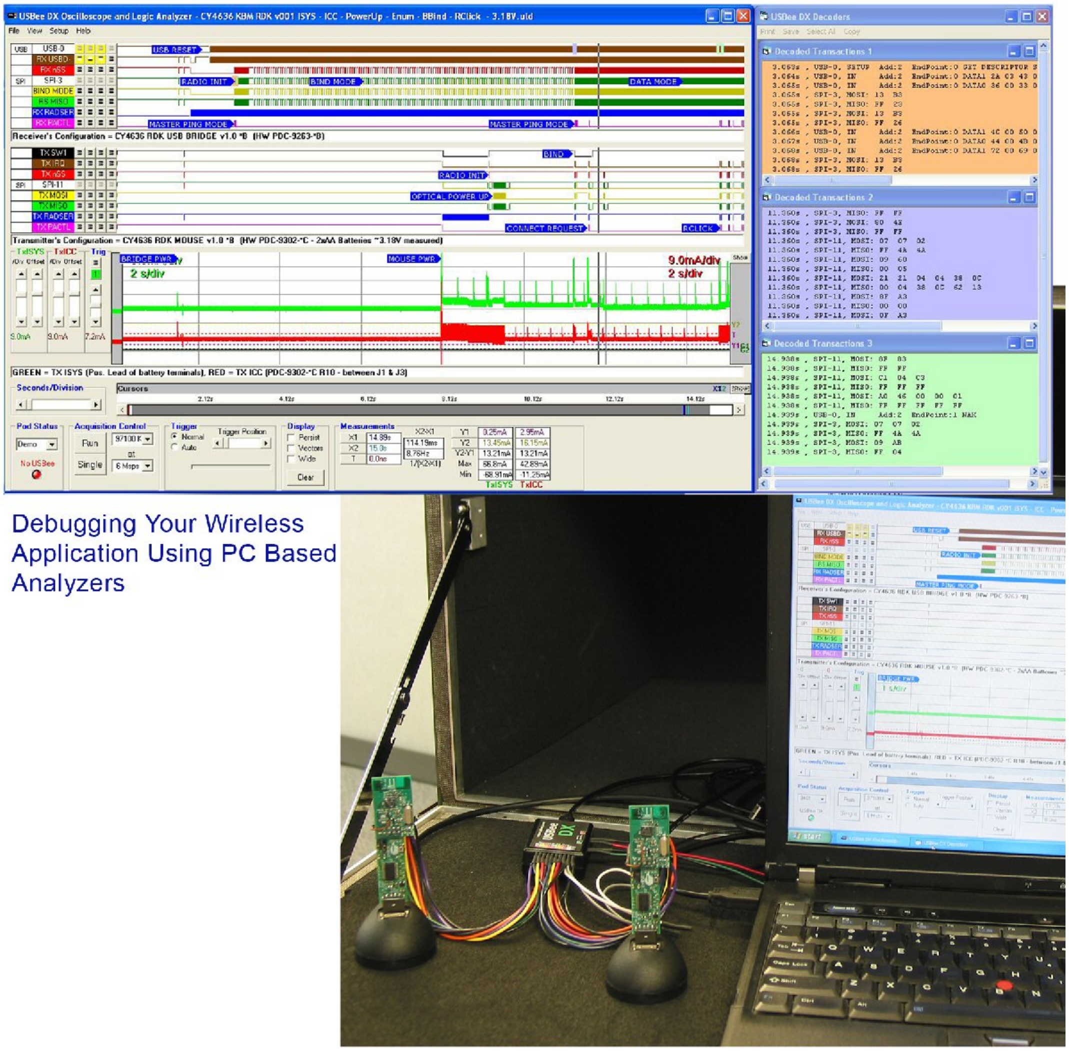
Wireless system-level interactions-based debugging captures often exceed the fixed length capture buffer capacities of traditional test equipment even for a seemingly simple application such as a wireless desktop mouse design. Debugging even the most rudimentary functions – such as pairing two wireless devices initiated by a person pressing a button on the first device followed by pressing a button on the second device – can render traditional analyzers useless.

Inadequate sizing of an analyzer's capture buffer forces wireless engineers to take on the tedious task of manually stitching together back-to-back debugging captures. PC-based analyzers designed with the capability to continuously stream capture buffer data into a PC's memory not only simplify the mundane task of capturing data for lengthy debugging sessions, they also greatly reduce the time required for post-processing and analyzing the data by eliminating undesirable stitching. First-time viewers of lengthy streamed capture buffer sessions are often surprised to discover previously unobserved system-level interactions-based events that they had not been privy to with traditional test equipment.

Even when an entire debugging session can be captured in a continuous buffer, developers are still left with the tedious task of decoding mountains of serial-based protocol data. Decoding bit-for-bit and byte-for-byte, left as a human exercise, is error-prone and excessively time-consuming. Additionally, given the high speed of wireless interactions, the desired debugging events are typically buried somewhere in a long debugging data capture with huge voids of unrelated data between them.

Traditional wireless application debugging strategies such as selectively sprinkling printf debug statements throughout application code can ease some of these pains. However, they achieve this at the expense of incurring unwanted side-effects such as code bloat, code execution inconsistencies, and code obfuscation, as well as requiring additional hardware resources. PC-based analyzers armed with flexible bus-level decoding facilities can greatly reduce cycle-to-cycle debugging time by effortlessly unraveling and accurately displaying large chunks of capture buffer data at the click of a mouse button. Figure 1 shows a typical PC debugging screen.

Figure 1: PCs have flexible bus-level decoding capabilities that can greatly reduce cycle-to-cycle debugging time.


USBee DX Oscilloscope and Logic Analyzer - CY4636 KBM RDK v001 ISYS - ICC - PowerUp - Enum - BBind - RClick - 3.18V.uld
File View Setup Help
USB RESET
RADIO INIT
BIND MODE
DATA MODE
MASTER PING MODE
Receiver's Configuration = CY4636 RDK USB BRIDGE v1.0 *0 (HW PDC-9263-*0)
BIND
RADIO INIT
OPTICAL POWER UP
CONNECT REQUEST
RCLICK
Transmitter's Configuration = CY4636 RDK MOUSE v1.0 *8 (HW PDC-9302-*C - 2xAA Batteries ~3.18V measured)
BRIDGE PWR
MOUSE PWR
2 s/div
9.0mA/div
2 s/div
GREEN = TX ISYS (Pos. Lead of battery terminals), RED = TX ICC (PDC-9302-*C R10 - between J1 & J3)
Seconds/Division
Cursors
Pod Status
Acquisition Control
Trigger
Display
Measurements
Run
Single
Clear
USBee DX Decoders
Print Save Select All Copy

Decoded Transactions 1

```
3.063s , USB-0, SETUP  Add:2  EndPoint:0 GET DESCRIPTOR S
3.064s , USB-0, IN     Add:2  EndPoint:0 DATA1 2A 03 43 0
3.065s , USB-0, IN     Add:2  EndPoint:0 DATA0 36 00 33 0
3.065s , SPI-3, MOSI: 13  B3
3.065s , SPI-3, MISO: FF  23
3.065s , SPI-3, MOSI: 13  B3
3.065s , SPI-3, MISO: FF  26
3.066s , USB-0, IN     Add:2  EndPoint:0 DATA1 4C 00 50 0
3.067s , USB-0, IN     Add:2  EndPoint:0 DATA0 44 00 4B 0
3.068s , USB-0, IN     Add:2  EndPoint:0 DATA1 72 00 69 0
3.068s , SPI-3, MOSI: 13  B3
3.068s , SPI-3, MISO: FF  26
```

Decoded Transactions 2

```
11.360s , SPI-3, MISO: FF  FF
11.360s , SPI-3, MOSI: 80  4E
11.360s , SPI-3, MISO: FF  FF
11.360s , SPI-11, MOSI: 07  07  02
11.360s , SPI-11, MISO: FF  4A  4A
11.360s , SPI-11, MOSI: 09  60
11.360s , SPI-11, MISO: 00  05
11.360s , SPI-11, MOSI: 21  21  04  04  38  0C
11.360s , SPI-11, MISO: 00  04  38  0C  62  13
11.360s , SPI-11, MOSI: 8F  A3
11.360s , SPI-11, MISO: 00  00
11.360s , SPI-11, MOSI: 0F  A3
```

Decoded Transactions 3

```
14.938s , SPI-11, MOSI: 8F  83
14.938s , SPI-11, MISO: FF  FF
14.938s , SPI-11, MOSI: C1  04  C3
14.938s , SPI-11, MISO: FF  FF  FF
14.938s , SPI-11, MOSI: A6  46  00  00  01
14.938s , SPI-11, MISO: FF  FF  FF  FF  FF
14.939s , USB-0, IN     Add:2  EndPoint:1 NAK
14.939s , SPI-3, MOSI: 07  07  02
14.939s , SPI-3, MISO: FF  4A  4A
14.939s , SPI-3, MOSI: 09  AB
14.939s , SPI-3, MISO: FF  04
```

Debugging Your Wireless Application Using PC Based Analyzers

Automating bus-level decoding is fast, efficient, and speeds resolution of complex system-level interactions-based wireless application bugs by allowing engineers to step away from what would otherwise be the boring task of bit-level decoding into event and results-driven debugging sessions. And, since all the details are readily available, debugging results can be explained in detail and clearly documented.

Traditional test equipment offers engineers a scanty supply of Save As facilities for post-processing and analyzing captured data. Routinely, these analyzers save their capture buffer data in proprietary formats, often resulting in the lost of vital debug information. Habitually, engineers have been forced to spend countless hours converting their saved capture buffer data into ASCII formats and imported into PC-based software for final post-processing and analysis.

PC-based analyzers natively save capture buffer data directly onto a computer's hard drive, easing the importation of information into spreadsheets such as Microsoft Excel for post-processing and data manipulation while maintaining the ability for anyone to access captured data in its original format using PC-based virtual instrumentation software. This ease of transferring and transform data into different formats is quite useful since wireless application subject matter experts are often split across varying engineering disciplines – and location sites – and need to evaluate data in different ways.

Reducing analyzer-based captured data post-processing time also has the benefit of increasing the amount of time engineers can spend viewing and analyzing captured data. Having the capability for anyone to play back known good system-level interactions-based capture buffer data and comparing its behavior to that of the debugging system’s capture buffer data is an invaluable wireless application debugging tool. Play back can also be used as a simple training tool for better understanding and documenting various wireless application interactions and dependencies that typically only occur in the field.

Debugging wireless applications in the field challenges even the brightest of engineers. Lugging bulky test equipment through the world's airports and shuttles burns life's most miserable set of memories onto one's brain. Additionally, the ritual of having to reorganize one's test bench to accommodate the latest "debugging effort of the week" often exceeds the time it takes to actually capture the data. PC-based analyzers designed to take advantage of a multitude of a PC’s available resources are often small enough to use on an airplane’s tray table, making it so that when debugging data offline, the sky really isn’t the limit.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.