07/08/20

Lumistar Dynamic Simulator (TDSIM)

Simulate The Cards

The TdSim application allows control of the Lumistar 70V1 and the 70V2 cards, with room for expansion for other vendors as needed, all at the same time, for a total of up to 8 streams. If you're like a lot of others out there, there aren't enough bucks to go around for everyone to have these cards for developing test routines.

The TdSim application allows you to simulate the existence of the cards, so you are simulating a simulator (or up to 8 of them). For the most part, the simulation of the data is about the only thing really meaningful. The simulation of the transmitter portion of the card is only useful if you want to save the setup file with the transmitter setup information ahead of time, before you do your test with actual hardware. Just go into the TdSim System-HwConfig and tell the system which vendor and card model to simulate the card and how many streams you want to simulate.

There is also a Virtual card selection, which is just a simulated simulator, without having to select a particular vendor card to simulate. The only real reason to simulate a vendor card is for the transmitter section to simulate.

The simulation of the card does allow you to do everything with the data that you could do with a real card. It just doesn't go anywhere (unless you are using the FPI Shared Memory output or the FPI UDP output). The only caveat is with timing. Without a card, there is no access to a timer that is granular enough to get the bit rate set just right and the interrupt rates to be consistent. It has to use the Windows timer, and we all know how accurate that is. The Windows timer will allow you to get close with your bit rate, and you can do whatever it is you want to do with the data. It even does video and audio quite well with the Windows timer. It works great with TMoIP gateways, sending them simulated data over the network, where they turn the network packets into clock and data lines.

Once you've set up your test scenario, you can just save off the setup file(s), and then copy those files to the appropriate directories on the PC that has the actual hardware, and you're ready to go. This can save you a lot of time, and allow you to use your laptop to set up your test scenarios.

LDPS Interface

Simulating the data with a simulated simulator is a little cryptic for most folks, if you have no way to see what the final results are when it comes through a decom, because all you get to see is a frame dump of the data, which is just the raw PCM data words.

LDPS has the ability to take data in from just about anything if you can get it into the PC. The simulated simulator input is no exception.

There are 2 methods to get data out of the TdSim application to feed some other application (like LDPS) with the PCM data. Both methods send the PCM stream in buffers of FPI length. The receiving application just parses the data and feeds it to the processing section of that application.

1. Shared Memory – This is the fastest method, but only works if the receiving application is on the same PC.
2. UDP – This allows you to have your receiving application on a different PC.

LDPS allows the data to come in from other than a decom via TMoIP device interface. (See the LDPS documentation). This means you can do 99% of your testing and setup, without having either a simulator card nor a decom card. Pretty slick, eh?

Depending on your method chosen above, you would either select the TdSimShMem.exe for the shared memory interface, or the TdSimNet.exe for the UDP interface or the TMoIP for multiple UDP packet definitions, such as RCC-218.

This LDPS interface works with real cards, as well as simulated cards. At the point where data is shoveled out of the TdSim application, it doesn't know if it is simulating a card or not, and LDPS doesn't care.

More on timing.

Windows is a non-deterministic operating system. You cannot get very accurate under about 10 milliseconds, but close, to within 1 millisecond. That is only if not a lot of other services are interrupting the system. Lots of factors here.

This gets worse with faster bit rates. The granularity of the Windows Sleep() is 1 millisecond. On faster rates, that 1 millisecond affects the bit rate much more.