07/08/20

Telemetry Dynamic Simulator (TDynaSim)

Theory

The Dynamic Simulator application is a very modular application to allow the user to output PCM data (either RF and/or cable). Similar to LDPS, the user has hooks to implement his own code at the INPUT, PROCESS, and OUTPUT sections (albeit the OUTPUT section has little to do). Ties to LDPS archive files as well as any other archive format are available for playback purposes. Likewise, ties to LDPS setup files (e.g. decom setup file, data dictionary files) are also provided.

As of this writing, this application is designed to control three different models of dynamic simulators from

- Ls70V1 (Lumistar)
- Ls70V2 (Lumistar)
- Virtual

Normally, you will only have one type of dynamic simulator model, but you can control all at the same time if desired. As far as the user is concerned, the only differences in operation are the capabilities of the RF section. The data section is all handled the same for each model (as the user sees it).

The hardware can be simulated (the Virtual is just a simulated simulator), and therefore if you don't need actual clock/data outputs from the hardware and just want the network output, you just select one of the models of hardware, tell the system to simulate the existence of the hardware, and how many streams you want it to have.

Since the dynamic simulator is limited on hardware space, the complexity of PCM simulation is quite limited if the simulation algorithms were left to the cards' capability for simulation. Therefore, all simulation is performed in software, leaving the dynamic simulator card as a clock and data output source. The dynamic simulator operation mode will be as if it was in "Archive Playback Mode" (as defined in the card manuals), where a buffer of data is sent to the dynamic simulator via DMA transfer through the respective model hardware interface DLL and kernel drivers.

By tasking the application to simulate the data instead of the card, the application can be the item that changes with user requirements, and not having to redesign the card. This method will allow complex streams to be generated (not just played back), i.e.

- Embedded asynchronous streams
- Embedded synchronous streams
- Chapter 8 streams
- Error generation of data on a bit by bit basis, including frame sync loss.
- Embedded time (various formats)
- Any or all words with fixed/canned/user defined value algorithm

- PCM word values can span any number of frames
- Archive playback (any format)
- Burst data
- Fill data (of any value, at any time)
- Easier insertion of embedded video and audio, with error generation
- Format switching / mode code handling
- Number encoding (2's comp, 1's comp, 1760, IEEE formats, TI formats, DEC formats, etc) for any or all values on the fly.
- PRN patterns, with error generation

This application controls up to 8 PCM streams.

Data Flow

There are three major modes of operation, for each stream, in this application. Depending on the stream operational mode, the data flow changes slightly, but the concept is the same for all modes. These four modes are as follows:

- User Defined PCM
  - User defines a PCM frame, then fills in the data values as desired.
- Playback
  - User plays back a previously archived file
- PRN
  - Selected PRN patterns are sent out.
  - BERT checks on data received (optional on some model cards only)

The over simplified version of the data flow is that the Card Manager collects a major frame of data from the Major Frame Data Manager, and pushes the data out the respective model card. From the user's standpoint, this system works with a major frame at a time.

Below is a brief description of the two main managers within this application. Detailed descriptions have their own chapters.

Card Manager

The Card Manager is a thread responsible for keeping the card buffer full of data when the card requires new data. All three models of cards have different memory capabilities, so the Card Manager divides the major frames of data up as required to output the data to the card, as the card can handle the data. The card itself works with a set of ping-pong buffers. As it is outputting data from one buffer, it is calling this application to fill in the next buffer for output. This ping-pong switching takes place based on the bit rate selected, and the size of the ping-pong buffers. It is the

responsibility of the Card Manager to keep the ping-pong buffer full, and all in sync with the request from the card for new data.

You cannot turn the cards' ping-pong buffers off. If the PC is on, then the cards' ping-pong buffers start outputting data and clock. If the application is turned on, and you have Data Output enabled, then the cards' ping-pong buffers get filled with new data. Otherwise, the data is what was last in the two buffers on the card. If you card has an RF transmitter on it, the default is RF Off, and you must use the application to control the RF Out state.

PC's cannot operate at just any rate (particularly with a non-real time operating system like Windows). If the request for new data is too fast, then Windows will not be able to keep up with the processing, and you will lose data. The cards' ping-pong buffer will wait for no one. If the request for new data is too slow, then your latency from data manipulation to when it finally goes out the cards buffer can be unacceptable for certain tests. Therefore, there is an adjustment for this, the Frames Per Interrupt (FPI). This adjustment value, coupled with the bit rate, determines how often the PC is interrupted for a new data buffer. With the FPI, the frame is a minor frame, not a major frame, of data.

In addition to sending the data buffers to the respective card, the Card Manager also sends the data buffers to optional output devices.

- TMoIP – The data buffers are sent out the network via UDP packets
- Shared Memory – The data buffers are sent out to shared memory

This allows you to have other applications, whether on the same PC or different PCs, to receive the manipulated data. If you don't need RF, and you don't need the Clock/Data output of the card, then you really don't need a card to get data to your other application.

There is one caveat, however, if you do not have a card. Remember, this is the Windows operating system. Deterministic timing with Windows is at best, poor. The data will be pushed out one or both (as enabled) based on a Windows timer, not based on the hardware timer from the card. The faster the interrupt rate, the worse the timer gets, so bear that in mind. Minor frame rates less than 400 Hz are generally pretty good. If you're feeding the data to a gateway device for reconstruction to clock and data, they are usually pretty good at dealing with the Windows timing issues and generate the correct clock.

Major Frame Manager

The Major Frame Manager is the guy that manipulates the data as you desire. There are six main sections to this manager. Every main section works on the entire major frame before passing the major frame off to the next main section. Some operational modes do not perform any operation in all main sections.

The six main sections are as follows, in the order they are called within the Major Frame Manager.

1. ***Word Initial Values***. – Fill the major frame with initialized values. The result of the data is also called Raw Data. The source of the initial data can come from various sources, depending on the operational mode.
    a. Ls50 Simulator Style – Each word value in the first minor frame is set to a specific value, and then for each minor frame in the major frame, the same value is used.
    b. Playback – A major frame of data is retrieved from the playback DLL.
    c. User defined Shared Memory – The user has an application that sets the initial value of every word in the major frame.
    d. PRN Data – If the operational mode is BER, then the major frame is filled with the correct PRN data for the pattern selected.

2. ***Word Algorithm Selector*** – Once the initial word values for the major frame have been filled in, you can go into the major frame, on a word by word basis, and tell the manager to adjust the initial values based on one of the following choices (for User Defined and Playback operational modes only)
    a. Use Raw Data – This is the default. Don't change the value that was assigned in step 1 above.
    b. Use Manual Controls – There are a few manual controls (sliders/joystick/etc) you can adjust while the data is being processed to set the value(s).
    c. Use Canned Wave Form – There are a number of canned wave forms provided, i.e. Sin Wave, Cos Wave, etc. you can have the program set the values to based on the commutation for that word. The frequency of the wave form is based on your bit rate, commutation for the word, and the number of words to complete the cycle.
    d. User Defined Function – This is the all powerful method. The user can write a DLL that will allow him to adjust the data however he sees fit, including reading data from a file to insert into the desired word location(s).

3. ***User Defined Major Frame Shared Memory Function*** – (Not available in BER operational modes.) Once the major frame has completed with step 2, the user has the option to pass the entire major frame off to an application he wrote, which he can then change the entire major frame from the values assigned in step 2 above.

4. ***FSP/SFID Insertion*** - (Not available in BER or Playback operational modes.) This is an option. If enabled, it will insert the correct FSP and SFID in the correct locations based on the PCM setup information provided. Most of the time, this will be on, but sometimes, you don't want anything to occur to the data after step 3.

5. ***User Defined CRC Data Insertion*** - (Not available in BER operational mode. Really shouldn't be used in Playback mode, but that is your call.) This is an option. After step 4 is complete, you can have the entire major frame passed off to your user written DLL, to have the CRC data filled in as you desire.

6. ***Data Error Injection*** – Once the major frame is filled out with all the steps above, you can now inject errors into the major frame if you wish. This allows you to simulate the reception of wrong data due to the many factors that can cause the data to be received incorrectly at the receiving end, i.e. multi-path, encoder errors, etc.

RF SECTION

If your card has an RF transmitter on it, then there are more things to play with. The RF section operations are not operational mode dependent. They work the same in all operational modes.

Depending on the transmitter model, and card model, some are really simple functions, while others can get quite complex (like Doppler Profiling). There are separate chapters on the RF section, because they are so diverse, depending on many factors. The Data Section is the same for all models.