!a_waste_of_time

Larry Creel, Miguel Torres
Systems Engineering Branch, White Sands Missile Range, NM

ABSTRACT

Time has always been the elusive fourth dimension – until now. Using a common programming language and a network-to-PCM interface, the power to generate time codes is facilitated using a non-traditional approach. This novel approach to time simulation addresses the common conundrums concerning time code simulation and testing. Practical applications will be discussed along with an intriguing technical demonstration.

BACKGROUND

Motivation

The Telemetry Data Center (TDC) at White Sands Missile Range (WSMR) happened on time simulation merely out of curiosity. Software decommutation had already been implemented to support a project with a complex telemeter. This "soft decom" had proven that data processing could be done entirely in software without any special hardware. This off-the-shelf computer was fast enough to process real-time data.

This lead to logging binary files in real-time for development purposes. The next step was to be able to playback the data from the computer. With a few modifications, the soft decom could process a logged binary file. This allowed development testing quite a bit of latitude. Development tests could be run from the computer without tying up a complete telemetry system. However, this method could not check out the mission configuration of the existing hardware systems.

In order to carry out such a test, there would need to be a data bridge that could easily convert network data to PCM such that it would be clocked out at the necessary bit rate. The NetAcquire F20-SIO unit has that capability. This unit can be given a network data stream and can clock out data at a specified bit rate. This is now a reliable data-to-PCM bridge. The logged data in the computer can be fed through the existing hardware systems for testing purposes. There is one small snag: time.

This setup does not have the capacity to use playback timing. In fact it cannot even pass coordinated universal time (UTC). This poses a serious problem: how do you test a telemetry processing system without the useful reference point of playback timing? Moreover, how do you process telemetry data real-time without time?

This is the motivation to use the tools of soft decom to generate IRIG B time. First, a description of the equipment used is necessary.

NetAcquire F20-SIO

The NetAcquire F20-SIO server consists of four serial input/output (SIO) channels, two network interface cards (NIC), an IRIG B time card, and a Pentium 4 3.2 GHz processor running NetAcquire's version of the RTX operating system. See Figure 1.1.

Figure 1.1 – A NetAcquire F20-SIO

The F20 is a server and therefore can be accessed from a remote computer using a web browser. NetAcquire recommends using Netscape. The web pages that control the F20 reside on the server. The SIO (Figure 1.2a) and TCP/IP Gateway (Figure 1.2b) pages were used to configure the F20 for time simulation.

!a_waste_of_time

Figure 1.2a – The SIO Configuration page on the F20

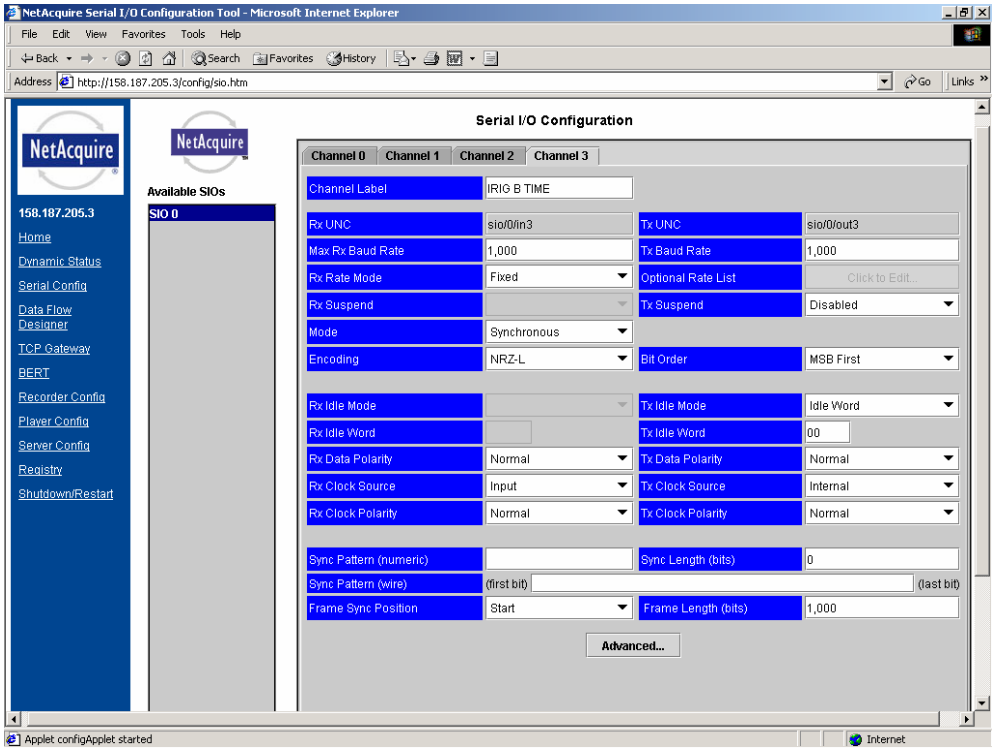
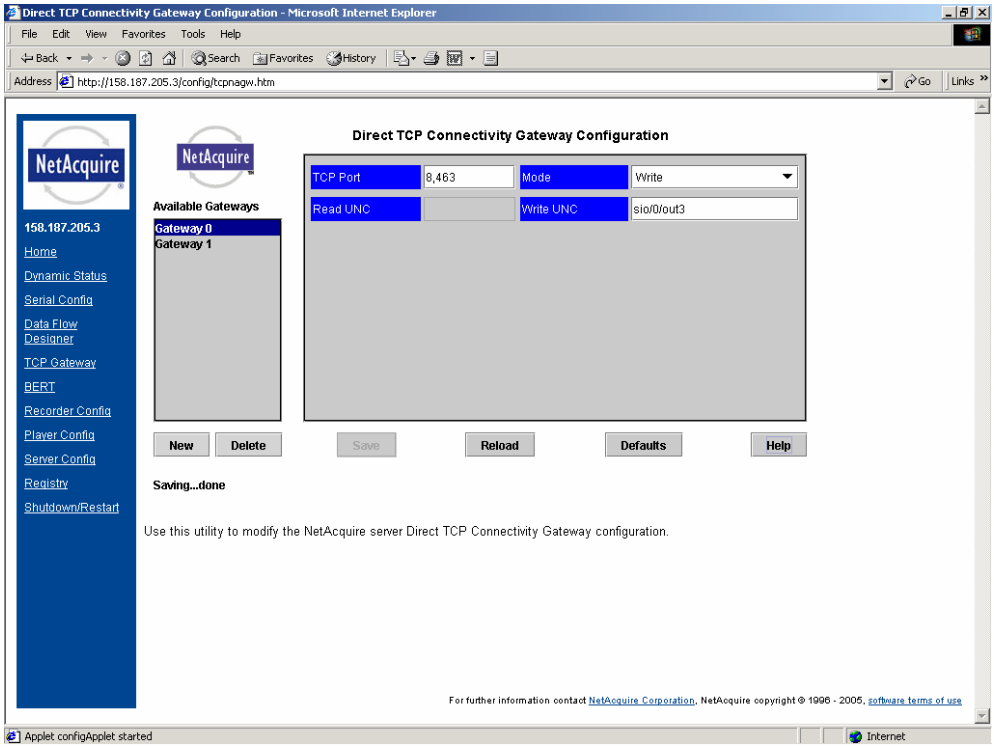


Figure 1.2b – The TCP/IP Configuration page on the F20



Apple Dual G5 and GNU Software package

The computer used for time simulation is an Apple computer with Dual 2.5 GHz PowerPC G5 processors, 4 GB SDRAM, no internal hard drives, and two NICs. The Apple computer is run by an external 60 GB LaCie hard drive with Mac OS X version 10.3.9. See Figure 1.3.
All of the time simulation files are coded in C. The GNU software package, including Emacs, is used to code, compile, and debug the program. Emacs provides a command line-friendly environment and the ease of an accessible compiler, gcc, and debugger, gdb.

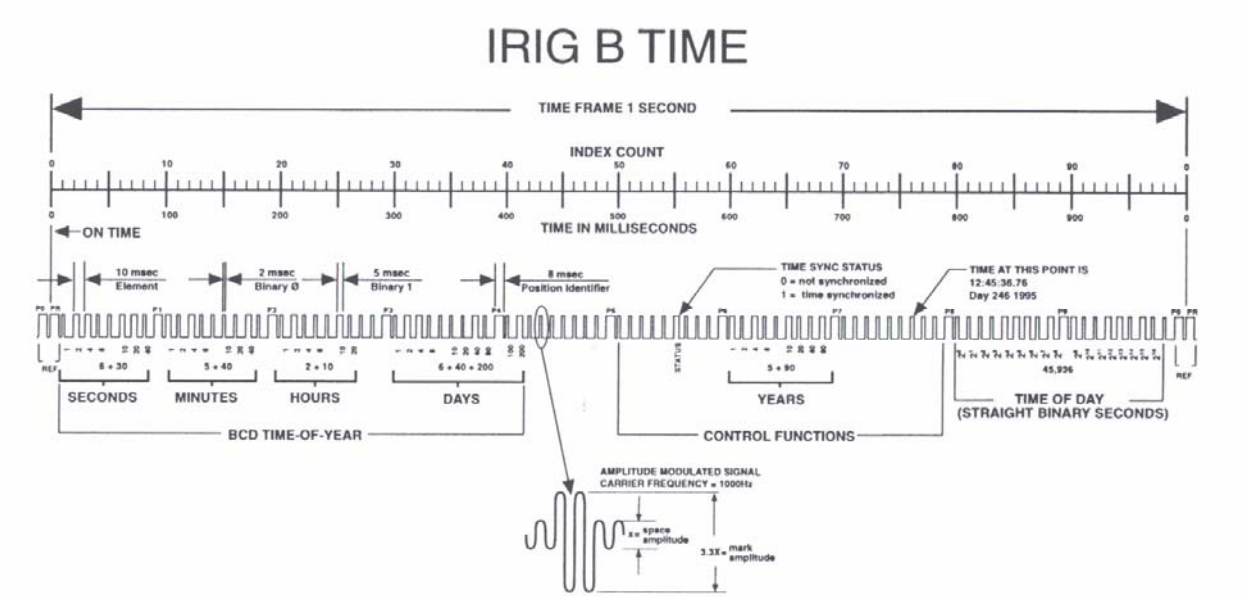
Figure 1.3 – An Apple G5 with external hard drive

IRIG B Timing

In order for time simulation to occur, one question had to be answered: what does IRIG B Timing look like? The simplest way to understand this analog signal is to represent it in a digital PCM-like form. See Figure 1.4.

_____

Figure 1.4 – IRIG B Timing in DC Code



This is referred to as DC Code. The time frame is represented as one bit for each millisecond, a rate of 1 KHz. Each element is ten bits, or ten ms, and each section has ten elements, or 100 ms, and each frame has ten sections, or 1 second. An element can have one of three values: post, mark, or space. A post is defined as eight "ones" and two "zeros". A mark is defined as five "ones" and five "zeros". A space is defined as two "ones" and eight "zeros". These definitions allow for a simple binary representation of a one-second IRIG B Timing frame.

A time frame consists of seconds, minutes, hours, days, and years in a binary coded decimal (BCD) format and time of day in straight binary seconds. The first section begins and ends with a post. All other sections end with a post. There are three sections that are reserved for control functions that may be populated in the future. Following is a table showing where the time components are located in the frame. See Table 1.1.

Table 1.1 – Time component locations

Section #	Time Component
1	BCD Seconds
2	BCD Minutes
3	BCD Hours
4	BCD Days
5	BCD Days
6	Time Sync Status
7	BCD Years
8	Control Functions
9	Time of Day (Straight Binary Seconds)
10	Time of Day (Straight Binary Seconds)

Trak Systems Modular Time Code Processor

The unit used to test the simulated time is the Trak Systems Modular Time Code Processor model 9000B. This unit has four basic operating modes: Standalone, Synchronized Generator, Reader/Generator, and Time Code Reader. In order to use the simulated time the timing unit must be set in Reader/Generator mode. This mode allows the unit to read and output IRIG B timing. The unit was also set to read DC Code for IRIG B.

IMPLEMENTATION

Layout

Now that all of the equipment has been identified, the next step is to clarify the physical connections. The time simulator code resides in the G5. The G5 is connected to a NetAcquire F20-SIO server via an Ethernet network. The F20-SIO server connects to the Trak Systems Modular Time Code Processor via coaxial cable. The Time Code Processor is connected to existing Telemetry Data Handling Systems (TDHS) that require timing through coaxial cable. See Figure 2.1.

Figure 2.1 – Layout Diagram

The Code

The time simulator code consists of two main parts: Initialization and Steady State. As a minimum Initialization establishes a TCP/IP connection to the NetAcquire F20-SIO at a well-known port and asks for a start time in "hours:minutes:seconds" format. If this connection fails, so will the program. During this one-time-only part of execution, other housekeeping tasks are performed as needed. The time simulator now enters the Steady State.

The Steady State is really a continuous loop that calls three functions: *build_and_update_2d_lookup_table(), build_frame_for_time_stream(&f20sio_time_frame[0]),* and *send_data_to_f20sio(&f20sio_time_frame[0]).*

In *build_and_update_2d_lookup_table()* a 4- x 100-element two-dimensional array is initialized. Each element is in a 32-bit container. This array represents four IRIG B time frames, or 4 seconds of time. It is initialized during the first call to the function and updated in every call. The initialization places the posts in their respective locations as mentioned above and spaces elsewhere. This function gets called every four seconds.

In the IRIG B DC Code representation, one time frame is 1000 bits long. Interestingly enough this representation is byte-divisible. However, the elements are only 10 bits long and so each element must be extracted from the two-dimensional array and "streamed" into a one-

dimensional array. The *build_frame_for_time_stream(&f20sio_time_frame[0])* function does exactly that. The *f20sio_time_frame[0]* array is one-dimensional with 32-bit containers. Even though one time frame is byte-divisible, it is not int-divisible, or 4-byte-divisible. This is the purpose for using four time frames as a buffer. This streamed array, four time frames neatly packed in 32-bit containers, is what is sent to the F20-SIO server.

The *send_data_to_f20sio($f20sio_time_frame[0])* function takes the streamed array and sends it to the F20-SIO server via the TCP/IP connection. Since this array represents four seconds of time, it is going to take the F20-SIO server four seconds to clock out the data at the IRIG B rate of 1 KHz. The reason for using the F20-SIO server is that the server will throttle the data from the G5. In other words, the G5 has to wait until the F20-SIO server has taken all of the data before it can send the next batch of time frames. The F20-SIO server preserves the bit rate of the timing data and is fed into the Time Code Processor. The timing unit does not notice a difference and so it locks up on the DC Code representation in a matter of seconds.

This is how time is simulated at the TDC at WSMR.

APPLICATIONS

Future Digital Data Recording/Playback on a Computer

One useful application of this technology is digital data recording and playback using a commercial-off-the-shelf computer, i.e. an Apple G5 computer. Using the F20-SIO it is feasible to record data and timing onto a hard drive. With this system data could be reconstructed into PCM effortlessly, however playback timing was lost. Only the data could be reconstructed. This method of time generation allows playback timing to be reconstructed as well as the data. This introduces a new obstacle: data-driven timing vs. time-driven data.

The flexibility of this system can allow for both. Only one solution is practical: time-driven data. Data-driven timing means that the data will not have any dropouts. The actual dropouts are reflected in shifts in time. This is not practical due to the nature of timing units and the IRIG B time frame. It takes a matter of seconds for a timing unit to lock on to the timing signal. It would be hard to identify a time reference for any drop in data since the timing unit would be trying to resolve the jump in time.

Time-driven data, on the other hand, would incorporate skips in the data by simply zeroing the frame synchronization pattern of the frame. The data can now be accurately represented with all of its data gaps. Time would not fluctuate.

Diagnostic Tools with Simulated Timing

Another useful application of time generation is Latency Measurements. By specifying the start time, the generated timing signal could be piped through all timing channels and then measured at the origination point for total latency. This would provide ranges with accurate measurements of latencies between stations.

CONCLUSION

With the capability of producing a telemetry PCM stream from a binary file residing on disk, other applications abound: simulation building, data correction, time correction, data merging, and frame building. For all of these to be viable, time generation is essential.

ACKNOWLEDGEMENTS

Armando Juarez, Jesus Benitez, Juan Guadiana — Telemetry Data Center
Preston Hauck, Steve Proudlock — NetAcquire Corporation

RELATED TECHNICAL PAPER

Soft Decommutation and Integration	Jesus Benitez, Juan Guadiana	ITC 2006