

## **Enterprise Data Recorder**

*Using Telemetry Front-end Equipment and Network Attached Storage  
Connected to Form a Real-time Data Recording and Playback System*

---

Tim Gatton  
Product Line Manager, Telemetry and Data Systems  
Wyle Laboratories, Inc.



**Telemetry and Data Systems**

## **ABSTRACT**

The use of traditional telemetry decommutation equipment can be easily expanded to create a real-time pulse code modulation (PCM) telemetry data recorder. However, there are two areas that create unique demands where architectural investment is required: the PCM output stage and the storage stage. This paper details the efforts to define the requirements and limits of a traditional telemetry system when used as a real-time, multistream PCM data recorder with time tagging.

## **Enterprise Data Recorder**

### **INTRODUCTION**

Wyle Laboratories, Incorporated, Telemetry and Data Systems (TDS) was invited to propose a solution to a high availability (A<sub>0</sub>) PCM recording requirement supporting 24/7 space operations. The requirements included:

- Microsecond time tagging
- Support for up to ten simultaneous PCM streams
- No single point of failure (for example, if a PCM input malfunctions, subsequent troubleshooting and repair of that channel's electronics can have no impact on any other channel's operation or electronics)
- System availability of .9999, which means that for one year or 525,600 minutes of operation, the total down time may not exceed ~52 minutes
- Ability to simultaneously record and reproduce all data as well as the ability to be able to record data while playing data
- System level support of 300 to 600 Mbps aggregate

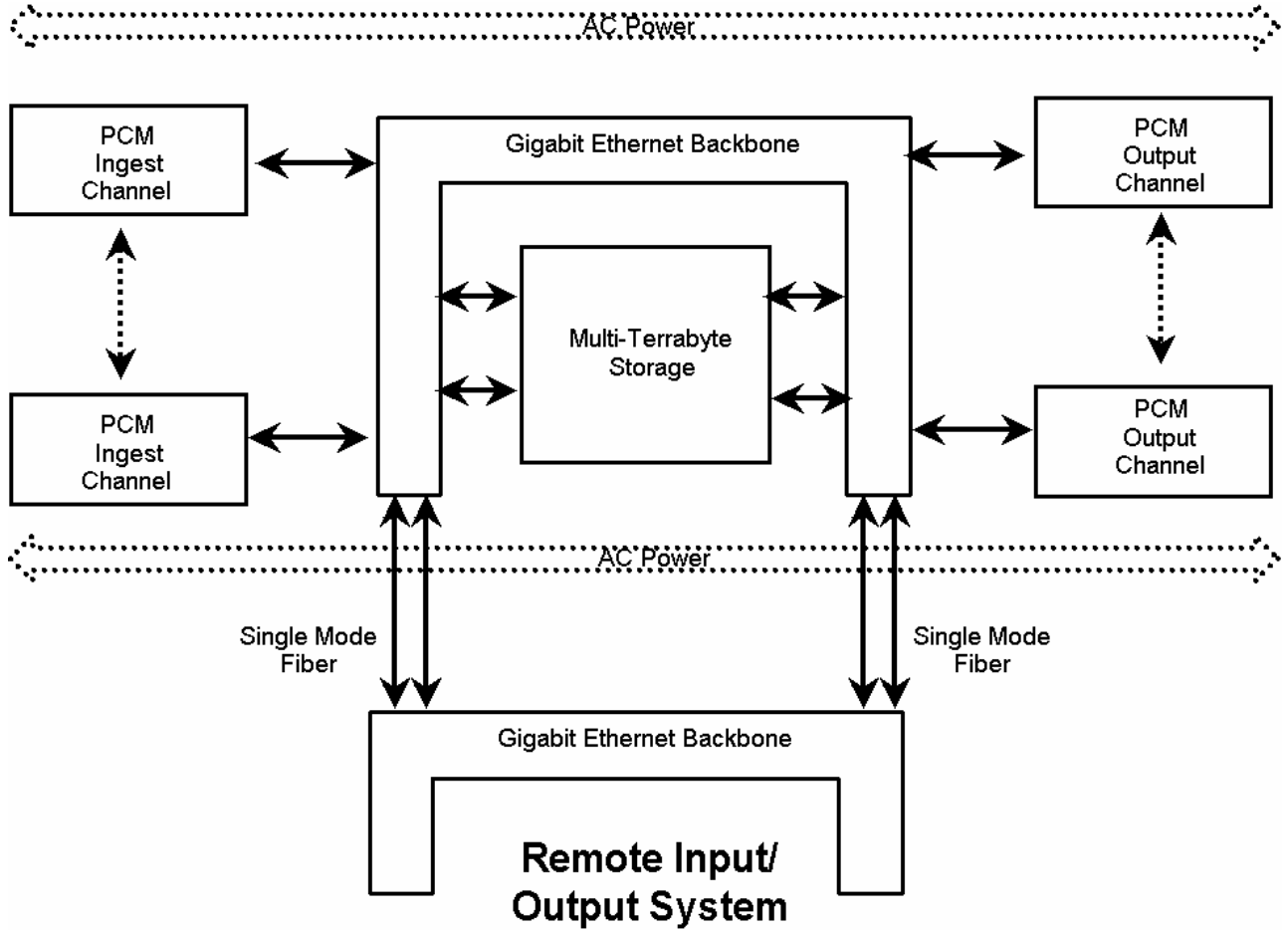
Traditional data and telemetry recorders are certainly robust and provide many hours of continuous operation; however, typical recorders are not architected to operate on a 24/7 basis. If a core element such as the media or a backplane becomes defective, troubleshooting brings the entire system down for repair. Therefore, traditional recorders are not an acceptable approach for solving this challenge.

### **BLOCK DIAGRAM**

The previous approach was to mechanically, logically, and electrically isolate each major system element. This included placement of input channels, output channels, media, and backplane—each in its own subsystem chassis. Functionally, this is broken out as shown in Figure 1.

- Each PCM channel resides in its own chassis. The backplane connecting them is a gigabit Ethernet backbone. The media is a network attached storage (NAS) device.
- Two power rails with independent power buffering are installed in the racks, and each component is outfitted with auto-switchover, dual-redundant power supplies.
- With each PCM ingest and output path as its own Ethernet node, any input or output channel can malfunction. Troubleshooting has no impact on any other system components.

*Using Telemetry Front-end Equipment and Network Attached Storage Connected to Form a Real-time Data Recording and Playback System*



**Figure 1. -High Level Block Diagram**

The network implemented was a 1000-Base-TX and a 1000-Base-F network using an HP Pro-Serve managed switch as the backbone. Copper (TX) connections were used in all local connectivity points while single-mode fiber was used for the remote input and output paths located several miles away with six input/output nodes at each location.

## SOFTWARE ARCHITECTURE

With the stability that Microsoft has established in its XP operating system (OS), XP was seen as a viable OS for a recorder application. People tend to think of Windows as an environment that is not very stable; however, empirical data shows that lack of stability is a result of users not testing applications and drivers properly, as well as not maintaining a compliant universe (see side bar). If an application is properly tested and properly maintained, today's XP environment can be every bit as stable as any other commercial operating system.

The newest language for software development under the Microsoft ".Net" (pronounced dot-net) framework is C# (pronounced C-sharp). A perceived disadvantage of C# is that it appears to remove some of the flexibility of C++ in creating code. In reality, this perceived disadvantage is actually one of C#'s advantages.

Programmers initially resist the change in programming flexibility, but the improvement in stable applications is well worth the effort. In addition to a Java-like garbage collector, C# also has strong memory management rules. For instance, in C++, it is relatively easy to misallocate memory causing leaks and blue screens. In C# under the .Net framework, most of those pitfalls are inherently prevented from ever occurring by the development environment itself.

### The Compliant Universe

Do you write memos on your oscilloscopes? Of course not! So why do people think it's acceptable to treat personal computers as desktop toys instead of the precision test equipment they are? If a personal computer is used for high reliability tasking (such as recording or real-time displays), it is imperative that it be maintained in a "compliant universe" — know the versions and service releases of every application on the computer. Once known, ensure that your system is tested before being released into its mission support role. Never allow anyone to amend or alter any aspect of the personal computer. Ideally, have one disk as the mission disk, which is protected and used only for missions, and a second disk for noncritical testing and support.

## DATA INPUT AND OUTPUT

Bringing 50 Mbps PCM data into and out of each node is done using traditional PCM decoms and PCM simulators (see Figure 2). Unique aspects of the decomp and simulators exploited for this recording solution were:

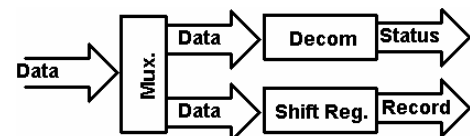


Figure 2. - Data I/O Diagram

- The decomp operates in two modes simultaneously:
  - Mode 1** – Shift register mode where data is buffered into registers and shifted out with no lost bits and no processing to look for data validity.
  - Mode 2** – Decom mode where data is buffered and searched via a frame sync correlator to establish a data quality value.
- The simulator operates in three modes:
  - Mode 1** – Memory is loaded with values, and a serial stream is created (this is the traditional PCM simulator mode from most vendors today).
  - Mode 2** – Memory is mapped to an application, and bits are driven from software.
  - Mode 3** – Memory is mapped to a data file, and data playback moves data files into a serial stream.

Having decoms with two parallel modes enabled one path to capture and store every bit that came in without regard to frame sync quality (think “accident investigation!”), while the second path allowed parallel data-quality inspection and a report path so the user could monitor data quality while recording. For the output, the PCM simulator was well suited for the task because it used the two modes that allowed either a) an application to stream data to a PCM output port or b) an application to stream data from an archive file to a PCM output port.

Both the decom and the simulator have IRIG A, B, and G time input/output ports so that data can be time tagged at input with microsecond time and reproduced at output with the same microsecond fidelity.

## **STORAGE COMPONENT**

The heart of the system is the storage subsystem. During their investigations, engineers discovered that the performance of the system was determined by several factors: storage type, NAS-head operating system, and disk type. This section speaks to the data gathered during these investigations.

### **STORAGE TYPE**

In most high-demand environments, a fiber-based storage area network (SAN) is typically advised; however, the costs of SAN systems are typically much higher than NAS. Between the high cost of fiber components and the high cost of network management software, SANs were seen as the ideal technical choice but less than ideal for cost containment. Therefore, a NAS was considered to be more attractive. Features that contributed to the NAS selection were: the self-aware nature on the network, its requirement for very little special software to manage, its standard gigabit Ethernet connectivity, and its allowance for the 30-60 percent usable bandwidth required to meet system needs. NAS was seen as a better solution to this challenge. The hunt was still not over.

### **NAS AND THE OPERATING SYSTEM**

In testing and analyzing available NAS systems, a wide variety of units were examined:

- Microsoft-based server software
- Linux-based server software
- iSCSI-based architectures
- Proprietary server software

The representative samples for a Microsoft-based product seemed to have reasonable ability on the input side to accept real-time data sets, but on the reproduce side the demand often outstripped supply. Discussions with the original equipment manufacturers (OEMs) yielded inconsistent theories as to why. The most reasonable explanation is that the output buffers are usually smaller than the input buffers, and thus outputting data files is more demanding in real-time applications (but this is only one vendor’s theory). Most of the Microsoft-based products are assembled from open-market piece parts and the internal knowledge of the RAID operations is typically, but not always, low.

Linux-based systems seemed to run faster than the Microsoft-based systems and they should. If the OEM has the ability to strip away all of the drivers for miscellaneous media that are not of interest and can adjust the queue characteristics of a system for only one purpose, this should yield better performance. But again, while the performance was faster, the output demands of real-time applications pushed (but did not break) the system limits. Note that this is not a vote for or against Microsoft and Linux—that discussion is well beyond the scope of this paper.

A new iSCSI-based system was tested and showed promise with benchmarks that were from 50 to a 100 percent faster than other systems. However, the beauty of this architecture quickly waned when it was discovered that iSCSI architectures did not allow access to the media by more than a single IP address at a time. Note: The vendor we tested professed that third-party software existed to plug this performance void; however, it never provided verification of this fact.

Finally, proprietary operating systems where the vendor has the flexibility to adjust the code to support one and only one environment and purpose should have yielded the best performance and it did. The proprietary environments tested produced superior performance right out of the box.

For this application, all versions (except iSCSI) would allow operation, but customization and tweaking will be required, with Microsoft versions requiring the most work and the proprietary software requiring the least amount of effort.

## **DISK TYPE**

The next component to select was the disk family: SCSI/FC or ATA/IDE. In reviewing these technologies, it was difficult to discern clear differences until the basics of SCSI versus ATA were studied.

First, remember that all disks are, within themselves, an “operating system” and as such are responsible for some level of housekeeping and error/flow control. For example, disks monitor their temperatures and recalibrate each and every time a temperature delta of ‘x’ occurs (the physical position of a magnetic domain is always known; heat changes will expand the media and move a block location to another physical position). This environment typically exists asynchronously to the computer environment with little to no feedback on what is happening—other than the “why is this taking an extra two seconds to open” head scratching.

Furthermore, it is commonly known that SCSI-based disks typically come with a more robust firmware environment while ATA-based disks typically come with a simpler environment. A simple disk environment requires more cycles of the host operating system. Indeed, in discussions with vendors, no one will cite specific examples when comparing an ATA disk that is recovering from a media failure versus a SCSI disk recovering from the same type of failure—but numbers of one to two minutes for ATA versus one to two seconds for SCSI were commonly used.

In examining the differences in speed and robustness between the SCSI and ATA designs, the difference in the reaction times when a disk goes off-line (bad) is noticeable—again, seconds versus minutes. Finally, when a disk does fail “hard” and a system needs to remap and rebuild the data/parity sets, an ATA disk-based system can take from 20 to 480 minutes while a SCSI/FC-based disk is typically quoted as 10 to 240 minutes (naturally, these times vary widely based on system activity and processing power in the NAS head).

In considering all of these factors, as well as the cost of ATA vs. SCSI/FC-based RAIDs, the desire is for an ATA-based RAID if the “nonavailability” of an ATA system can be accepted. The availability of ATA devices is impacted by two primary factors: a) the medium takes longer to correct itself, and b) often ATA disks cannot have a “hot swap” controller; thus, a single point of failure exists. This can be overcome by using a virtual file manager program that shares volume information among all nodes on the backbone and shares storage space among them. In the event of a NAS failure, this would allow the individual node disks to be used temporarily until the NAS is back on line—but virtual file manager software has its own complexities and limitation that impact overall system performance (such as reconsolidating data from all nodes when the NAS becomes available again). The primary impact of this approach is that if the NAS is off-line for repair, previously recorded data is not available. Therefore, dual-disk controller architecture with this combination can achieve the required .9999 availability and still be sensitive to the price sensitivities that are a fiscal reality. If .99999 is required, a SCSI or FC-based NAS would be the better solution.

## **CONCLUSION**

The design of a high-availability solution must take into account the acceptable level of serial versus parallel design elements, the reliability of each element, and the level of system robustness that is required (.9999 for ~52 minutes of downtime per year or .99999 for ~five minutes of downtime per year). While the discussion of parallel “k of n” binomial mathematics and the reliability block diagram components is fascinating, it is well beyond the venue of this paper to provide the analysis. Suffice it to say that with all other infrastructure being the same, a SCSI- or FC-based NAS can provide a .99999  $A_0$ , while an IDE/ATA based NAS will typically be limited to .9999.

For an excellent primer on reliability information, a good web site is:

[www.weibull.com/systemrelwebcontents.html](http://www.weibull.com/systemrelwebcontents.html)

Note: This does not constitute any endorsement of the website nor its content, accuracy, or availability.

## **NOTICE**

The life of information about storage architectures and available features is extremely short. For example, iSCSI solutions with multinode simultaneous access are very common. Therefore, use this paper as an example of the items to be sensitive to when designing your system solution.

## **NOMENCLATURE**

1000-Base-T	gigabit Ethernet
ATA	advanced technology attachment (official name of IDE drives)
FC	fiber channel
IRIG	Inter-range Instrumentation Group
IDE	Integrated Drive Electronics
iSCSI	Internet SCSI; SCSI protocol over IP
NAS	network attached storage
PCM	pulse code modulation
RAID	redundant array of inexpensive disks
real-time data recording	Process by which a continuous and contiguous data stream is captured and reproduced as continuous and PCM.
SAN	storage area network
SCSI	small computer system interface
single mode fiber	Optical network connections that use spectrally pure wavelengths of optical energy.
telemetry front end	Components that receive and simulate one or more telemetry streams.

### **For More Information:**

Wyle Laboratories, Inc.  
Telemetry and Data Systems  
44417 Pecan Court  
California, MD 20619

301-737-1555  
telemetry.info@wylelabs.com  
www.wyletds.com