CHANNEL BASED SAMPLING IN A NETWORK BASED DATA ACQUISITION SYSTEM

Joseph Sulewski
Lead Software Engineer
L-3 Communications Telemetry East

Chris Dehmelt
Director of Systems Engineering
L-3 Communications Telemetry East

ABSTRACT

Over the last few years, PCM based data acquisition systems have become known as "Traditional PCM" systems. This terminology modification is a sign of the evolution of the next generation of telemetry/data acquisition systems based on network topologies. This has come about due to users clamoring for functionality that has not been available in the traditional systems, such as supporting increased data rates, providing access to onboard archived data, supporting on-the-fly reconfiguration, and simplifying data distribution and delivery.

The iNET standard is using standard network technology to improve device interoperability and data acquisition. To minimize impact on existing data acquisition system devices, the initial effort of this approach has included the transmission of “Traditional” fixed PCM frames within a network message based structure. This approach, however, squanders network bandwidth, as a PCM frame includes all samples of all channels, and requires significant processing power for even simple tasks.

Delivering on the promise of a more flexible transmission method requires a change in how data is acquired in the data acquisition devices. The iNET standard defines such a packet based transport system, which supports channel based packet formats besides “Traditional PCM” to efficiently deliver data products. This paper will provide background on the benefits of these methods and an overview of methods by which these formats can be implemented.

KEYWORDS

iNET, NETWORK, DATA ACQUISITION

INTRODUCTION

Today's PCM telemetry systems have served the community well over the last number of decades. Although PCM generally refers to the encoding of the telemetry stream, solutions for data storage, analysis, decommutation, and other requirements have been developed and used reliably for years. However, the systems required for some tasks such as decommutation and analysis have generally required highly specialized and powerful devices.

Over the last number of years there has been a push to transition from PCM transmission methods in favor of modern network based protocols. This push has been so great that PCM based data acquisition systems have become known as "Traditional PCM" systems. This terminology modification is a sign of the evolution of the next generation of telemetry/data acquisition systems based on network topologies. The immediate question that should come to mind is why, as the existing technologies have served the community well. One must be careful that network based telemetry is not a solution for which there is no problem.

TRADITIONAL PCM ARCHITECTURES

In traditional PCM based systems, there is often a mix between high rate analog data and slower rate bus data. PCM relies on a constant rate bit stream that is derived from sampling many different types of sensor and bus inputs. Meeting this requirement via the different sampling rates requires the commutator to sample some measurement types regardless of whether new data is available – this is specifically true of bus monitor measurements that occur asynchronously to the operation of the commutator. This physical constraint is a core part of the architecture and a number of interesting techniques have surfaced helping to create flexible and capable PCM acquisition systems.

This flexibility however, comes at a cost. In more complicated setups where multiple Data Acquisition Units (DAU) are required to collect the data, master remote setups must be employed. Due to the nature of PCM this creates its own complexities because the remotes must sample at a rate that is an even multiple of the master. Therefore, the master has control over the sampling rates of the remotes and instrumentation engineers must work within these bounds often leading to large amounts of oversampling. Therefore, the careful work required to be efficient with the usage of bits is wasted due to the large amount of oversampling.

NETWORK ARCHITECTURES

The sampling constraints described above are due to the physical limitations implicit to the PCM architecture. Network based telemetry could turn the physical constraint of PCM into an artificial constraint fundamentally altering the approach of telemetry systems. PCM relies on a single constant rate output stream; while network based systems have many different simultaneous output streams where each stream is free to define its own output rate.

Another advantage of the network architecture is that multiple DAU's may be placed on the same network with each broadcasting many streams of data at varying rates. However, one must be concerned with understanding the issues created by the relationship between data, time

synchronization between devices, and management of these systems. The strong demand for network based systems and the desire to overcome these issues has created the demand for standards such as the iNET standard. The goal of the iNET standard is to set design guidelines for how to handle timing, configuration, decommutation, and the general management of these systems.

Understanding how iNET defines the time synchronization and management of devices is beyond the scope of this paper. Rather, we will explore the implication of the features network based DAU's can provide. Considering that the IEEE-1588 Precision Time Protocol (PTP) standard is able to accurately time synchronize independent DAU's amongst each other, we can now reevaluate whether remote and master configurations are necessary. Since multiple DAU's can be placed on the network and act without regard for one another, the timing constraints imposed by master and remotes are no longer an issue for the user. Therefore, each DAU can configure itself without concern for others and achieve a more efficient stream.

NETWORK DATA ORGANIZATION

Another benefit the network provides is the ability to construct DAU's according to the types of data they acquire. The instrumentation engineer could, for example, construct a DAU specifically for low sample rate data to produce a slow stream of information reducing oversampling and making more efficient use of bandwidth. One consequence of this design is the decreased need for highly specialized systems to decommutate the data - not necessarily because there is less useful data to process, but because there is less unused data to process, and discriminating which data to handle becomes markedly simpler.

With a traditional PCM stream, all data must be decommutated for the user to access the desired measurements. In a network system, only some of the information needs to be decommutated because a network packet will contain specific measurement sets rather than the standard complete set of all measurements. Therefore, small devices such as cockpit displays can consume slow streams containing only the measurements it needs, and ignore the faster higher rate data that is contained within separate packets. This lessens the burden on small devices to process information making it much easier for the devices to extract the samples and display the data.

Initial approaches have focused on a single stream of data and attempting to fit a PCM approach into a network based system. If a single DAU could produce multiple streams of data then it would be possible to group measurements according to a number of factors, for example, sample rates, type, safety of flight concerns, etc. This offers us another fundamental shift in how data acquisition is approached. Rather than fitting the rates to a PCM style matrix, measurements can be treated as unique entities that can be broadcasted into any number of messages onto the network. Just as networking broke down the barrier of timing constraints imposed by the proprietary master-remote relationship, changes to the sampling methodology of network systems can break down the sampling barrier measurements imposed on one another due to the PCM frame.

This approach is generally referred to as channel based sampling and there are several ways to implement its design with each method having associated pros and cons.

PCM DATA ACQUISITION ARCHITECTURES

A traditional PCM data acquisition system is comprised of two primary components:

- The encoder/controller. This component provides all system interfaces for configuration and control, as well as the mandatory PCM output interfaces (whether filtered PCM or clock and data). This component is responsible for executing a fixed commutation schedule that is used to control access to the various input channels at specific times to collect data for its output ports.
- A backplane. Depending upon the system, this could be analog and/or digital, and is usually proprietary to support the unique characteristics of the controller's method for accessing acquired data. Modern data buses have typically not been usable due to the design of the data acquisition system and its need to support dynamic installations and the mandatory requirement to support deployment in ruggedized environments.

This combination results in a sampling/data access architecture that is typically known as command-response (or in other terms, a schedule based pull mechanism) in which the controller collects data from a specific channel at specific intervals to create the user defined PCM output stream.

CHANNEL BASED SAMPLING

Current Network Based Systems

Most of the existing network based systems have been "upgraded" to migrate away from traditional PCM by adding a network gateway device to an existing system. This gateway provides basic capabilities that result in no impacts to the scheduling or acquisition aspects of the DAS:

- Eliminates the traditional serial configuration interface
- Incorporates a PCM framer that takes the data collected by the same data acquisition schedule use in the "traditional" PCM system and creates a network packet that consists of one or more PCM frames with an application specific header.

These functions are typically implemented in software on an embedded Linux type module. This solution allows for easy transition from one type of network specific application to another format (for example the NetDAS NET-810 network can be configured by the user for Chapter 10 packet formatting, iNET TmNS formatting, or custom L-3TE formatting via configuration information). However with these solutions we still find ourselves applying PCM disciplines to a network based system thus limiting the potential of the system. There are several steps that can be taken to move away from PCM disciplines toward the channel based sampling approach.

Requirement Overview

Network based packetized PCM frames however, still have the same burden as its predecessor in that a receiving device still must receive and decommutate all data from a source, whether it needs it or not. As previously noted we are still applying PCM disciplines to a network system

and as a result, many devices are not equipped to handle the amount and rate of data being generated by newer data acquisition systems, and methods must be developed to create data sets that are tailored to the requirements of the user devices.

Background

In one of its previous lives, L-3 TE (Aydin Monitor Systems) had been part of a system team that designed and built a number of ground decommutation and processing systems (PAX River RPTS III being one example). In these ground processing systems the data acquisition hardware was physically segregated from the host computer with different interfaces employed to transfer real-time data and status. These interfaces included various parallel computer specific interfaces, commercial-off-the-shelf fiber-optic shared memory interfaces, and standard Ethernet. To offload processing, these systems were required to segregate channels into multiple buckets using the unique capabilities provided by the specific interface.

- Networking: different packets with the packet header containing a unique identifier (ID).
- Shared Memory and computer specific interfaces: specific memory blocks used for different data sets. A form of Direct Memory Access in which the data acquisition hardware is provided with the ability to directly write into unique memory blocks, based on data contents. When combined with external interrupt generation capability, a powerful method to minimize the need for the host computer to actively ingest data from the data source.

Channel Based Data Collection Problem

This ground application is quite similar to that of the airborne network based data acquisition, in that there is an acquisition system that is cognizant of the data that is being sampled and hence, has the ability to immediately create the unique data sets that a receiving device requires. The reason that this is true is that the configuration tools have all of the pertinent measurement sampling and distribution requirements that allows for device specific configuration information to be generated. The problem that now needs to be addressed is how to integrate this into the acquisition system's controller device, which has been designed to not "know" anything about the data that it is collecting and to simply output data immediately upon sampling and not create specialized packets of data.

Software-Only Solution

The simplest method to provide the functionality of channel-based sampling is to implement a solution in software at the controller level. In this implementation, the controller would continue to run the existing schedule with frame buffers built in memory, but instead of being immediately output, would be routed to a pre-output processing function that would segregate the frames into a variety of output buffers based on the measurements that are needed by a receiving device. The output buffer types could include all samples of one specific measurement, all samples from those measurements that have a relationship, or all messages from a specific bus.

This method would be preferred, as it has no impact whatsoever on the underlying hardware. However, this method could be limited to maximum bit rate, the number of unique output

packets that could be provided, or the affect on latency because the solution is based solely on software and is dependent upon the performance of the underlying processor capabilities.

Hardware Solutions

To support higher throughput rates, better latency control, and/or more packets, improvements can be made in the underlying hardware to assist the software (there is also the obvious possibility of using higher performance processors with the likely negative accompaniment of higher power consumption and more heat to dissipate).

However, as most controllers include both a processor and a supporting FPGA, the question at hand is whether that secondary device can provide any initial sorting functions ala similar mechanisms that are implemented by the aforementioned ground systems, and if so, where is the software-hardware split and what information needs to be supplied to the hardware layer?

Scheduling Implementation

To truly make use of the hardware layer, the existing scheduling mechanism needs to be overhauled. The current data acquisition schedule is a general one – there is nothing in the schedule to identify how to map the data that is being collected to the output products that are needed by various users. As part of the configuration process, the measurements required by each unique receiver needs to be defined such that an aggregate "schedule" can be generated. The facets that need to be considered as part of this process include:

- What can the data acquisition system backplane support? For example, the NetDAS TEBus is a digital backplane that supports a sustained data rate of 133 Mbps (or 8 million bus transactions). The bus architecture however is implemented via traditional command-response (pull) mechanism, with the signal conditioning module's bus interface designed with that in mind. As most PCM applications require data rates no greater than 20 Mbps, how can the additional unused 85% bandwidth be used by a new scheduling mechanism?
- Analog channels still need to be sampled at fixed intervals in order to provide useful information to functions such as frequency analysis. This implies that a baseline schedule consisting of a fixed rate data acquisition is still required.
- How to handle asynchronous sources, in which data will sometimes be available, sometimes will not be?

The above considerations drive the need for a smart or dynamic scheduler that knows something about the data that is being collected, and hence can run a fixed schedule that will support the access of the synchronous channels at the specific timing intervals, mixed with a secondary dynamic schedule that allows for access of the other non-synchronous inputs to determine when data is available.

This secondary schedule is akin to a polling mechanism, as the asynchronous input device, such as a MIL-STD-1553 Bus Monitor, provides status information that allows for the controller to recognize when new data is available and needs to be read. Again, using NetDAS as an example, the unused bus cycles, which are No-Operations transactions, would consist of one status poll read followed by 1-34 data reads (32 data words plus the message time tag for a 1553 message) to be read between the fixed synchronous samples.

To allow data to be sorted as it is acquired, the hardware schedule must be provided with the necessary information to either route a sample to a specific buffer, and hence an output stream, or provide it with a unique tag. Whether it is routed to a buffer or given a tag is dependent upon whether the controller's hardware architecture provides sufficient capabilities, such as DMA, to create the buffers automatically, or whether a single data interface mechanism, such as a First-In-First-Out (FIFO) buffer is employed.

Hardware Buffering

The ground system output module is responsible for building Low Latency (LOLA) buffers in external host memory while simultaneously creating separate buffers for localized archiving. Unlike the archiving function, which allows data to be recorded in large blocks over a period of seconds and does not have latency requirements, the LOLA function needed to provide continuous streaming data to external users that allow them to monitor a test in progress. Buffers are then processed when one of two events occurred, which triggered an interrupt into the host:

1) A buffer was filled.
2) A timeout occurred. A predefined timeout period that is established keep data flowing smoothly to processing and distribution tasks within the external host workstation.

In either case, the same interrupt was fired as the information needed was supplied by the two read-only parameters, which allows the host to process all received data from start to the end of the current buffer without having to perform any unique processing functions. A couple of major differences between what is required for an airborne system and what has been previously implemented in the ground system include:

1. It is well known what is being sampled in the airborne system (sampling rates for each entity, channel or bus, number of channels, etc.).
2. The number of managed buffers is significantly larger and ground based computer systems had significantly more memory for buffering.

Buffering Processing

Figure 1 shows the basic flow of data from the TEBus to the individual buffers. The FPGA has as an input, the scheduling information, in which each instruction not only includes the channel to be sampled, but where the data should be written once received.

- The controller FPGA executes the desired schedule which is made up of new instructions that include the channel address and the target buffer, so that immediately after data has been read, it can be written to the correct buffer. To support measurements that are to be included in multiple packets, secondary sorting by the processor will be required.
- The schedule would be configured to read ("standard") channels at the appropriate sampling rate, similar to what is done now for the PCM application.
- Messages: With the inclusion of bus cards, there is a possibility for thousands of unique measurements to be defined by the user (PCM tends to discourage identifying all parameters due to impact on format size, bandwidth, etc.). There has to be some limit on the number of unique measurements that can be individually packetized (based on memory, processor, latency requirements, etc), so to address bus data, an initial approach

would be to buffer data on a message-by-message basis (thus allowing for packetization in a Chapter 10 type method).To support this, the schedule could be setup to read an entire message in one burst at a periodic rate (note that it is assumed that message rates are known so the number of times to read any one specific message/second can be applied to the schedule).

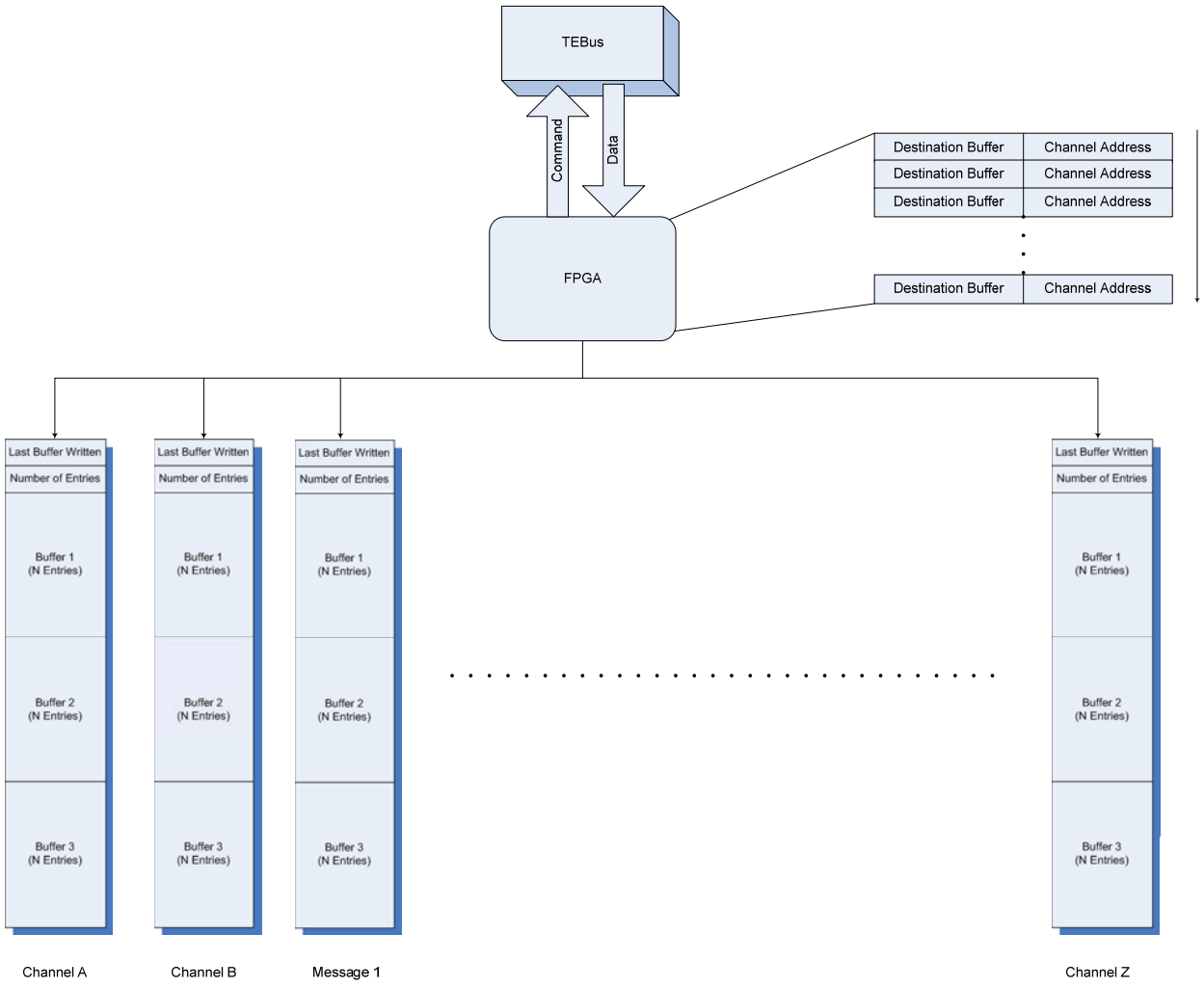

Figure 1 - Schedule and Data Routing

After the buffer is filled, the processor will cycle through each message, checking the status to determine if the following data is repeat or new. If new, then the message contents (only) would be copied to a packet assembly buffer (note that time of message receipt could also be included in each message entry). This method is an acceptable alternative to having the status poll in the data acquisition process itself because it removes the specialized decision making from the hardware layer.

Output Processing

Network based systems have to manage their output based upon various criteria, including latency that is induced by the required buffering processing, and separately, minimizing network overhead, which is induced when packets of non-optimal size are generated. This latter item can occur if the system is generating many small packets at high rates.

Latency is a critical parameter to address because it is impacted by interrupts that can be generated when a buffer is completed or a timeout has occurred. Because the timeout (for any specific buffer) would be based on that buffer's fill rate, these two events should be theoretically identical. So the question becomes, is it necessary to actually trigger on a buffer full, or does the FPGA generate interrupts at specific intervals that allow the processor to handle those buffers that should have been completed "automatically"?

In the simple example shown in Figure 2, channels A-C require output after 500 samples are collected or timeout period "t" has expired. Channels D-F also require output after 500 samples a second, but the sample rate is 2X faster, therefore, the timeout period is instead t/2.

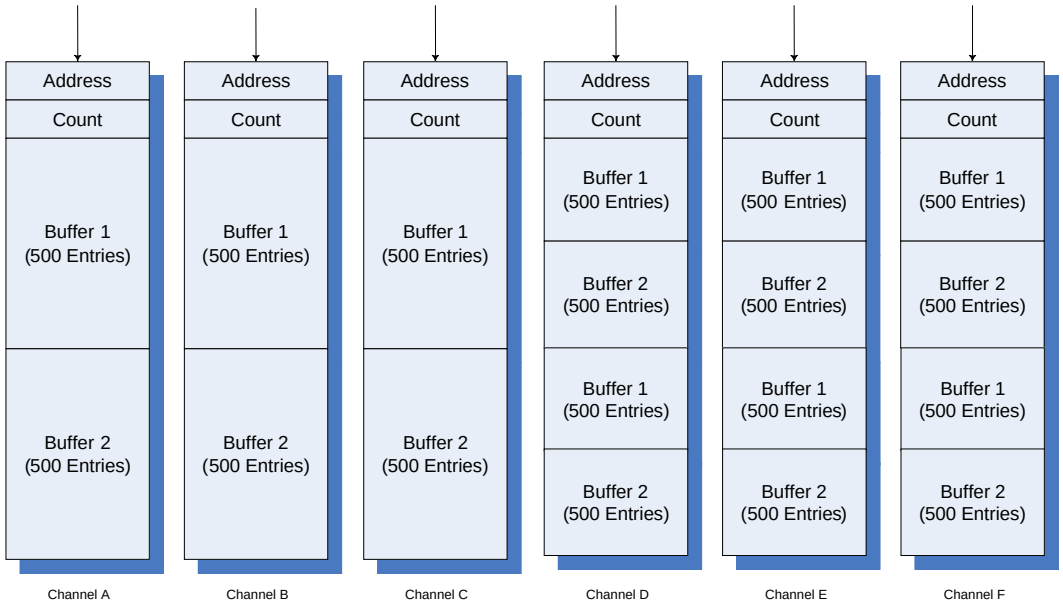

Figure 2 – Buffer Organization and Interrupt Triggering

Triggering on every completed buffer would result in interrupts that are fired very quickly as each buffer "simultaneously" completes. The time between interrupts from the different buffers would be based on the actual data collection rate and could be as quick as 240 nanoseconds (50 mbps format with 12 bits/word). The end result would be a processor that is overloaded with service interrupts inhibiting it from performing any actual processing functions.

However, knowing that there are relationships between channels and that there will likely be similar sampling requirements, the configuration software could likely build a sampling schedule and interrupt schedule such that the receipt of the last measurement for a specific buffer triggers a group of buffers to be handled. For example, in the case above, Channel F would trigger an

interrupt every time it fills; every odd interrupt resulting in Channel D-F buffers being output, every even interrupt resulting in all channels being output.

CONCLUSION

If one were to design a system from scratch today, one would build in the necessary facilities to easily support the acquisition and collection of data in specific packet formats. This effort would include a detailed analysis of the software and hardware tradeoffs to optimize the architecture for this specific application. However, this paper has shown that there are a variety of paths forward that allow the user to take a phased approach with existing hardware and software components to support this capability today without incurring the costs of developing a new system, and would additionally provide a solid foundation to support the evolution of the technology as resources allow to meet the expected continued increase in system data rates.