REAL-TIME TELEMETRY ON A PC

Dan Smith, Chief Software Engineer
Doug Steele, Principal Engineer
L-3 Communications – Telemetry-West

ABSTRACT

Near real-time telemetry acquisition, processing and analysis on a desktop PC have always been difficult. Many factors complicate working with real-time data, including operating system latencies, design inefficiencies and hardware limitations. These problems are further compounded when data from multiple sources had to be integrated, increasing design complexity. Current design solutions for analyzing data in near real-time now utilize the latest hardware implementations and software designs, taking advantage of new hardware and language features. This paper will discuss several issues found with PC-based telemetry systems and how new designs are addressing these issues.

KEYWORDS

(1) Telemetry
(2) Real-Time
(3) Operating System

INTRODUCTION

It is common knowledge that the personal computer (PC) is a low cost alternative to high-end workstations and specialized embedded hardware. Not only is the hardware less expensive, but the majority of skilled users will already have the necessary knowledge to use applications running on a desktop operating system. This is not a profound insight, but rather a realization that many developers and users of telemetry systems alike have come to understand. With this realization, numerous approaches have been taken to deliver telemetry solutions based on the PC architecture.

Many problems surround the implementation of a telemetry system on a desktop PC whose purpose is then to acquire, process and analyze data in real time[1]. The operating systems of desktop computers are not designed to process data in real time. They were designed for an application environment where delays lasting many milliseconds are not considered critical.

---

[1] The term "real time" suggests that data is processed with a very small latency between sampling and analysis, archiving, or display. The reality for PC-based telemetry systems is that this latency is the result of many different factors and is difficult to characterize on a generic scale. For the purposes of this paper, the term "real time" should be interpreted to mean near real-time, or as close as possible given the operating environment.

This requires that details such as general operating system and interrupt latency (due to mouse and video interrupts, et al) have to be addressed.

This paper discusses how the latest advances in PC hardware provide support for near-real time data analysis on any given platform by offloading much of the processing to dedicated hardware.

FIRST GENERATION PC-BASED TELEMETRY SYSTEMS

Solutions for obtaining deterministic data processing used to require significant investment in and training on specialized technology. Over time, technology has gotten smaller and faster, and advancements in user interfaces have decreased the amount of training required to operate that technology. These changes have culminated in the development of many low-cost telemetry acquisition and data analysis solutions. These solutions have not been without problems.

First generation PC-based telemetry systems relied heavily on hardware for acquiring a telemetry stream while the vast majority of the data processing was handled in software running on the host processor. There were many reasons for this, but the most prominent reason was that technology had not yet advanced to a point where low-cost configurable hardware components could be designed that provided a greater level of processing power to an application. Hardware engineers would design modules to transfer received data to the software in a format that could be easily used while relying on the software to perform most data analysis and display functions.

These designs were not necessarily bad in many circumstances. Data recording is an example of an application where the format of the data may not be of concern to the user, as long as the data is available for later playback and analysis. The ability to support multiple or user data formats may be desired, but the issue can often be worked around by application software.

DEVELOPER CONSIDERATIONS

From the start, PC based telemetry solutions have primarily been limited by the host computer processor, memory, peripherals, operating system and applications. A manufacturer could supply a PC with the system and control these factors to some extent, but these items still defined the basic system limitations. Over time, processors, memory and peripherals have gotten smaller, faster and less expensive; however, operating systems and installed applications grew, requiring larger amounts of memory and more CPU cycles to operate. As users became more comfortable with the PC environment, expectations concerning telemetry system feature sets also grew. The result was that even though more PC power was available, the resource problem never really went away.

The installed operating system presented problems to the telemetry system developer. Desktop operating systems were not designed for high-performance, real-time telemetry analysis. They were designed to allow the user to run desktop applications, such as word processors, email clients or web browsers. Optimization of the operating system was oriented towards these other types of applications. This is not to suggest that these operating systems were designed inefficiently, only that they were never designed with the express purpose of meeting the needs of a real-time application. The challenge to the telemetry system developer was then to create an

application that could perform high-speed data acquisition and analysis on a system that was not expressly designed for this purpose.

Another problem for the developer required that he or she have an understanding of how the user would actually use the system. The advantage and disadvantage to the desktop operating system is that the majority of users understand how to use it and there are tools available from many different vendors that can be run inside of this environment. While these are good things, they can lead to problems when users attempt to modify a system configuration by adding in their own personal or commercial software, hardware, or configuration (resolutions, themes, or performance enhancing software or tweaks), thus changing the operating environment. What this means is that the telemetry system developer cannot rely on the system always being maintained in the exact condition it was delivered or that all system resources will always be available for telemetry application use.

These problems were further compounded when data from multiple sources had to be integrated, increasing design complexity. First generation hardware modules designed for a PC tended to operate independent of other modules. One PCM stream might be decommutated on one PC card, while another PCM stream would require another card for decommutation (see Figure 1). The problem with this type of design has always been time correlation between these separate streams. Each data sample should be processed in the time synchronous order it was received regardless of stream origin. Failing to do so in the hardware makes real time processing of the data difficult and CPU intensive. This problem, like many others, only gets worse as new data processing requirements are added to the system. This is typically not a problem on small-scale telemetry systems. Small-scale systems are designed for only one or two data streams and multi-stream problems may only be seen as system requirements increase.

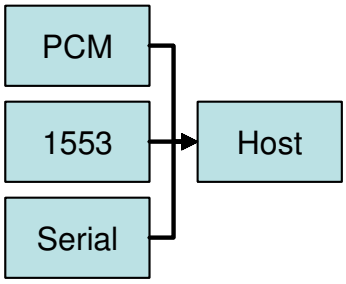

Figure 1 – Multiple Stream Integration

All of these problems can be addressed with both hardware and software solutions. Hardware designs are required in order to support the processing of data from multiple sources in real time and software architectures are required to provide support for analyzing this data. It took a combination of these solutions to provide the next stage in the evolution of telemetry systems.

As technology advanced, manufacturers began to combine multiple hardware modules onto a single card. Multifunction Telemetry cards are an example. Cards of this type typically contain multiple bit syncs, PCM decommutators, PCM simulators and IRIG generator/decoders. There may be configuration limitations, but these types of cards allow multiple data streams to be input

into a single card. While these types of architectures assist with the multiple-source data stream problem, they do not resolve it.

At this point, it should be understood that while PC based systems may not have contained the processing power required by many advanced applications, there are embedded systems that do meet this need. Larger asymmetric processing architectures based on high-speed back planes or switched fabric serial busses in VME or cPCI form factors provide advanced data acquisition and analysis for complex systems, but require far more investment in hardware and training. Systems of this nature by design allow multiple processors to act simultaneously on the same data sets and eliminate the latencies associated by such things as processor contention and non-critical application interrupt processing. In situations where this high level of power is not required, telemetry users have often made do with systems that offered lower performance but met the basic needs of the application. This approach has often been called the "good enough" approach to system selection. This approach also allows systems to be obtained within ever decreasing budget constraints.

BOXED PRODUCT LIMITATIONS

One solution to the resource and configuration problems identified earlier is to isolate the system from the user. This is done by embedding all the system functionality into a single module and isolating critical functionality from the user; preventing them from modifying the configuration with third party hardware or software. This can be done with proprietary system designs or specialized operating system implementations. The problem with these designs is lack of portability and potential difficulty with integration. While these designs are often easily incorporated into existing telemetry networks, they can easily reach limitations if more advanced functionality is later needed. The greatest flexibility is best achieved by providing some level of support for third party and customer designed hardware and software.

REAL-TIME OPERATING SYSTEMS

Several real-time operating systems on the market offer deterministic interrupt latency and pre-emptive scheduling. Operating systems such as VxWorks® and Integrity® are popular in the embedded environment and offer some distinct advantages over traditional desktop operating systems when dealing with interrupt response times. These operating systems have expensive per-seat development costs and do not offer a desktop environment useful for an end user. In some cases, these operating systems can be used to provide real-time performance, such as archiving, but they are still faced with the limitations of getting real-time data to a desktop interface for display and analysis. One solution to this is to provide a two-processor solution, one processor running the desktop operating system and a second processor to run the real-time applications. This presents more of a configuration and maintenance challenge for the user, although it is an attractive option for system designers.

Some vendors have created real-time extensions to popular desktop operating systems such as Windows® and Linux®. These extensions can offer real-time performance while still providing the ability to run common user applications and tools. On Windows NT/2000/XP/2003 systems, replacing or extending the Hardware Abstraction Layer (HAL) can add real-time capability. The

HAL source code can be licensed from Microsoft, thus allowing vendors to create a replacement HAL or extend certain real-time features to it. Adding the ability to pre-empt any Windows process, defer certain interrupts and add higher resolution clocks and timers is essential for real-time performance. As an example, a hardware interrupt from a telemetry board could be processed before a mouse click interrupt, regardless of which occurred first. One of the disadvantages of changing the HAL is that when Microsoft® releases new versions or service packs of the operating system, they can break or adversely affect the real-time extension. This puts the developer at the mercy of the extension developer to come out with new releases of their library in a timely manner. The extensions themselves are often expensive and may require run-time royalties. Venturecom's RTX, Nematron's Hyperkernel and TenAsys's INtime are examples of real-time extensions for Windows.

In the Linux open-source based community, there are two main players in the real-time extension arena. These are RTLinux and RTAI. They both offer the following features:

- A small real-time core
- One-shot and periodic timer support
- A real-time scheduler
- Real-time threads
- Real-time FIFO's and shared memory
- A real-time interrupt handler

They are also available at no cost under a general public license. In both these cases, the real-time kernel inserts a layer of code between the Linux kernel and the interrupt control layer, thus allowing it to intercept any interrupt prior to it being handled by the kernel layer. This essentially makes the actual Linux OS run at a lower priority from the real-time portion. See Figure 2. Unfortunately, these tools require certain versions of the Linux kernel to work properly. It may also require the developer to be knowledgeable enough to rebuild Linux kernels using mostly command line tools. These solutions do offer a lower cost alternative than a Windows solution and Linux itself is gaining in popularity.

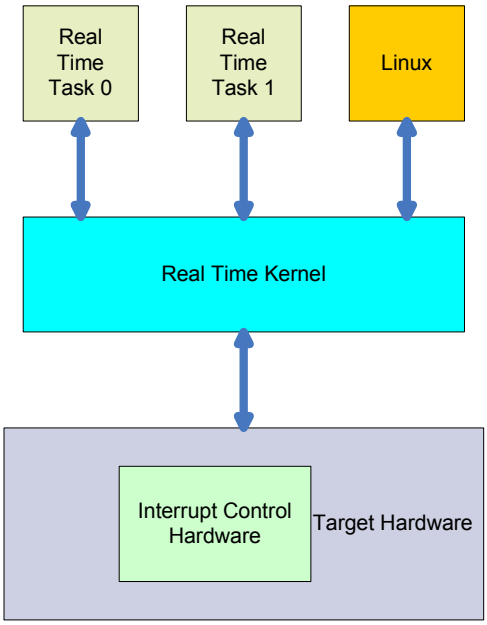

Figure 2 – Example Kernel Layering

NEXT GENERATION PC-BASED TELEMETRY SYSTEMS

If the developer accepts that complete control over the host environment cannot be achieved, then he or she can begin looking at ways to circumvent these potential pitfalls and provide the user with a stable, high-speed, feature rich system. Next generation telemetry systems are characterized by embracing the limitations that are inherent in PC architectures while taking advantage of all the features available.

Next generation PC-based telemetry systems are simply the next evolution in telemetry systems. They are extensible and designed with the understanding that a given application will most likely grow over time. These systems are designed to be flexible in meeting that growth. These new systems perform multiple tasks, including advanced data processing functions, high rate archiving, and 3D visualization with a high level of software integration. This is important because historically manufacturers have seemed to focus more on function than integration. While individual products offered exceptional performance for individual applications, they did not work well together. This held true even for products developed by a single manufacturer. These next generation systems that we are discussing attempt to resolve these and previously discussed difficulties with new hardware and software designs.

After finishing the design and market release of its 4th generation PC telemetry processing system about 6 years ago, L-3 Telemetry West came to the realization that new technologies and architectures were not easily being integrated into systems that existed internally and in the general marketplace. The result of this realization was a substantial internal research and development investment by L-3 of an entirely new software product and hardware platform that would provide the foundation for supporting current customer requirements while being flexible enough to integrate into new telemetry architectures and the latest technologies. The resultant

solution for this tightly integrated, high performance, multi-stream telemetry processing requirement was the Vista Enterprise Software Suite and the Vista 350P.

Vista is a Java based, host and platform agnostic, enterprise-level application environment that provides a feature rich processing, analysis, archiving, and display solution. Based on a network centric architecture, it provides the software framework for multiple hardware designs while allowing customers to migrate towards a more generic platform and assists with integration issues.

The L-3 Telemetry West 350P is a PC based telemetry system that is built around two primary hardware components: the Multifunction Telemetry Card and the PCI PMC Carrier. The Multifunction Telemetry Card is a well-known product that provides many functions that used to exist on disparate cards into a single device, and can be used in almost any PC while providing the stepping stone necessary to go to the next level.

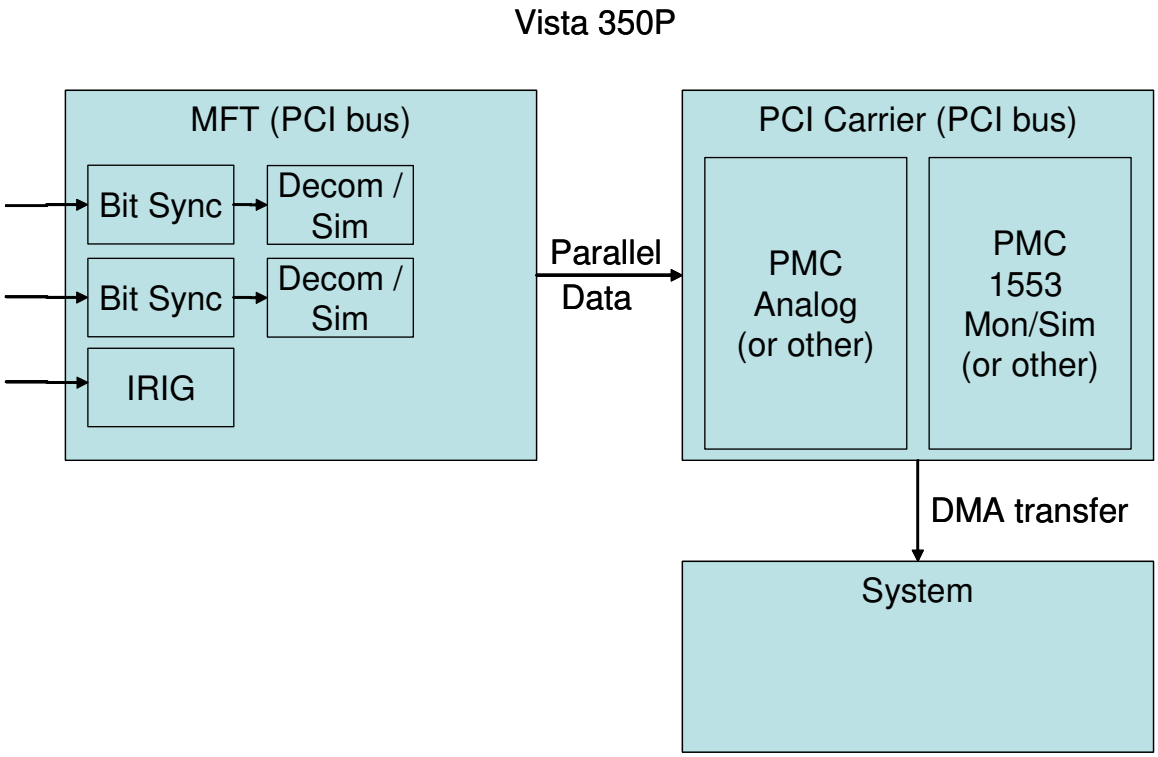

Figure 3 – Vista 350P Architecture

Like many technologies, new products are often smaller and higher performance than their predecessors. An example of this progression is the movement from PCI to PMC (PCI Mezzanine Cards) mezzanine modules that provide high performance functions on a small form-factor device. All mezzanine modules require a carrier module on which they attach. For the 350P, this carrier is called the PCI PMC Carrier. This carrier is unique in that it provides functions beyond simple bus extension including high-speed interconnection, multiple interfaces and support for third party devices. These advanced functions propel the 350P to Next Generation status. See Figure 3.

Embracing inherent limitations in the architecture does not mean that performance will always be a problem. One solution to this problem is to add additional independent, special purpose processor modules to the PC. This is called Asymmetric Multi-Processing (AMP). AMP allows these processor modules to run independent of the host processor and permits the host processor to offload some CPU intensive tasks. The obvious advantage is a significant gain in performance. By integrating these processor modules within the existing system design, additional performance gains can be realized by reducing host processor interaction of many of the data handling tasks.

LOOKING FORWARD

Looking into the future, there continues be a significant need for telemetry systems that are easily integrated into network architectures such as the Test and Enabling Network Architecture (TENA) and integrated Network Enhanced Telemetry (iNET) architecture. Telemetry systems may require integration with other well-defined interfaces such as those provided by the JTRS (Joint Tactical Radio Systems). While an implementation integrating these designs can be done primarily in software, it is best to consider them from a system design perspective and let the requirements generated by these system designs flow down into lower level module designs. This means that new designs cannot be solely based on proprietary implementations but must be designed so that "open" standards can be easily integrated into the system.

Initiatives such as lifecycle testing require that telemetry systems be adaptable to changing requirements and flexible enough to expand as the needs of programs change. From conception to deployment, any new system that generates telemetry data will have changing needs as the project progresses. The requirements for a sub-system lab environment will be different from those for a systems integration facility, which will still yet be different from a flight line telemetry check out system. New telemetry systems must be designed so that investments made in one portion of the telemetry environment will be realized throughout the project lifecycle. This relates to the old axiom "Don't reinvent the wheel."

CONCLUSION

Telemetry system developers can no longer afford to focus implementations on strictly proprietary designs. New requirements will mandate the use of commercial standards to assist with the integration of equipment from multiple vendors into a combined, easily administered and fully capable telemetry network. Focusing on this integration in new designs will make this effort less painful for telemetry users and telemetry vendors.

REFERENCES

Hyperkernel. Nematron. June 5, 2005. <www.nematron.com>

Integrity Real Time Operating System. Green Hills Software, Inc. June 5, 2005. <www.ghs.com/products/rtos/integrity.html>

INtime. TenAsys. June 5, 2005. <www.tenasys.com/intime.html>

Ivchenko, Alex "Real-time Linux", *Embedded Systems Programming* May 1, 2001. <http://www.embedded.com/showArticle.jhtml?articleID=9900126>

Java Technology. Sun Microsystems, Inc. June 5, 2005. <www.sun.com/java>

Microsoft Developers Network . Microsoft Corporation. April 10, 2005. <msdn.microsoft.com>

RTX. Ardence (formerly VenturCom). June 5, 2005. <www.vci.com>

Vista Software. L-3 Communications, Telemetry-West. June 5, 2005. <http://www.tw.l-3com.com/telemetry_west/products/vista.htm>

VxWorks Operating System. Wind River Corporation. June 5, 2005. <www.windriver.com>