May 2007

Tools Target Real-Time Data Acquisition Systems

Software modules in the real-time and non-real-time domains of a proposed model for real-time hardware recording systems help maintain performance while easing software development tasks.
By Rodger Hosking,Pentek, Inc.

With each new opportunity, designers of real-time data acquisition and analysis systems confront a unique set of development challenges specific to the requirements at hand. Although methodology and lessons learned in previous projects are invaluable, over time each new system introduces increasingly more complex hardware and software. Guessing correctly about how long development will take, choosing the right approach and selecting the appropriate tools are crucial for delivering the project on time and coming out ahead on the bottom line.

The Real-Time Data Acquisition Hardware Environment

A typical real-time data acquisition system includes high-speed A/D and D/A converters plus the associated circuitry for clocking, gating, triggering and synchronizing multiple channels (Figure 1). Digital up-converter and down-converter ASICs provide frequency translation for communication and radar applications. FPGAs handle tough real-time signal processing tasks such as FFTs, decoding and encoding, decryption and encryption, modulation and demodulation, and beamforming. A fast memory helps boost efficiency of data transfers by buffering blocks of real-time signal data. A control processor manages system resources and often performs additional signal processing and data formatting tasks. Often, a fast hard disk allows the system to handle real-time storage and playback of signal data.

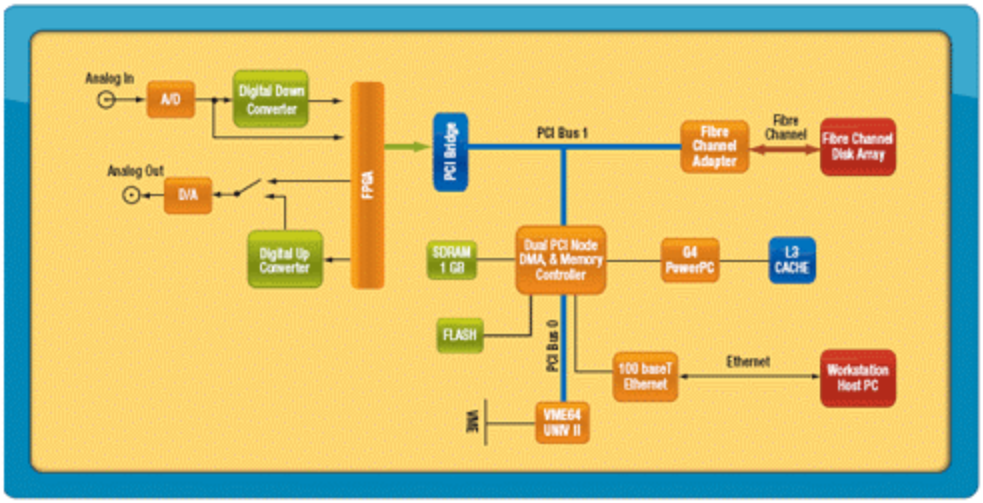

Figure 1 Real-Time Record/Playback System Hardware Components.

Connected through an Ethernet link, a host PC workstation running Windows or Linux provides essential, non-real-time hardware resources for the operator interface, including monitor, keyboard and mouse. The PC also provides high-capacity disk storage for archiving data files and network connections to the office or facility. All of the necessary hardware components are present, but which software components will ensure that the system performs the way it should?

Software Considerations for Data Acquisition and Analysis

A good choice for the real-time control processor is usually a DSP or RISC processor. It should be freed from tasks not directly related to essential data movement, formatting or processing. During real-time operation, there should be minimum interaction with the PC. However, before and after real-time operations, the Ethernet link can be extremely useful for initializing the real-time hardware, configuring modes of operation and moving data between the real-time disk and the PC disk file system.

A proposed software model for the real-time hardware recording system is partitioned into real-time and non-real-time domains and joined by Ethernet (Figure 2). While this partitioning may seem complex, each software module serves a well-defined, essential function to maintain performance while easing software development tasks.

Inside the Real-Time System

In the real-time software module, the real-time data acquisition system uses an RTOS with low latencies and deterministic behavior to guarantee that no data will be lost while managing the critical tasks it needs to perform. The board support libraries feature well-defined subroutine calls to implement low-level control for all of the real-time front-end hardware resources. This includes configuring the A/D

and D/A converters, setting parameters for the digital up- and down-converters and defining modes in the timing section for triggering, gating and synchronization.

The SCSI file system uses RTOS calls to manage real-time transfer to and from the local Fibre Channel hard disk. A Fibre Channel protocol layer provides a complete, high-speed file system interface suitable for real-time recording and playback. Nearly every RTOS also provides a native stack for Ethernet, TCP/IP drivers and support for Sockets, a very popular application-layer protocol for networks.

Each of these three resource groups—the board support libraries, the Fibre Channel interface and a network socket interface—are integrated by the record/play server API into a set of intuitive high-level functions to simplify development of custom server applications.

Acting as the executive in charge, the record/play server application has complete access to sending and receiving Ethernet commands and status, directing real-time data streams on and off of the Fibre Channel hard disk, and controlling all operating parameters and modes of the front-end data acquisition hardware.

With these software components, the real-time hardware has now assumed the role of a complete, stand-alone, functional server subsystem. It is capable of responding to Ethernet commands that are interpreted by the server application and then dispatched efficiently by the server API. When the function is complete, the server application can issue an Ethernet message along with any relevant parameters about the operation.

By extending the code in the server application so that it understands and implements different or more complex commands, the real-time subsystem can acquire new features. These new commands simply implement a new set of calls to the existing collection of record/play server API functions. For example, a new Ethernet command might fetch data from the Fibre Channel disk and deliver back through the Ethernet port.

Inside the Non-Real-Time Workstation Host PC

The non-real-time workstation host PC assumes the role of the client, by sending and receiving Ethernet messages to and from the real-time server subsystem. Virtually all workstation operating systems include native support for Ethernet, TCP/IP and Sockets to support message transfers.

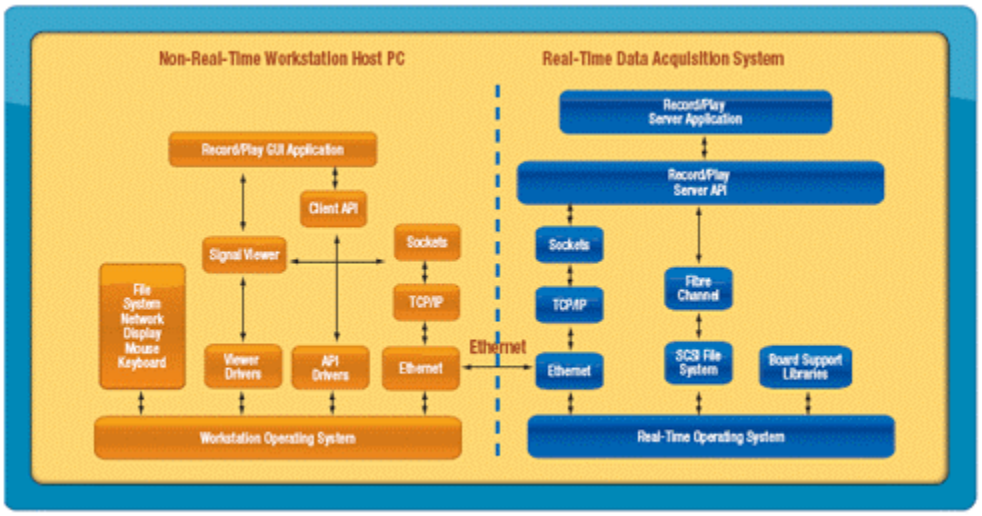

Figure 2 Real-Time Record/Playback System Software Components.

The client API manages the Socket message traffic by appropriately forming outgoing Ethernet commands so they are understood by the real-time server, and interpreting status messages returned by the server. Like the record/play server API in the real-time domain, it presents a set of easy-to-use, high-level commands suitable for client user applications.

Additional socket connections deliver Ethernet data from the server into the signal viewer. This application delivers a graphical representation of signals on the PC screen, providing the operator with an oscilloscope display for viewing signal data. This can be useful for checking snapshots of live data from the A/D converter before recording, for verifying the live output of a digital down-converter and for viewing the data recorded on the Fibre Channel hard drive after recording.

	CLIENT Non-Real-Time Workstation PC	SERVER Real-Time Play/Record Sub-System
Physical System	Desktop, laptop	VME, cPCI, PC/104
Processor	Pentium, PowerPC	PowerPC, DSP
Operating System Sockets, TCP/IP Ethernet, File System	Windows Linux Solaris	eCos VxWorks LynxOS
Application	JAVA, Visual C Visual Basic, C, C++	C, C++
API (client/server)	JAVA, C, C++	C, C++
Signal Viewer	LabView	-
Board Support Libraries	-	C, C++
FPGA Development	Foundation ISE Gate Flow	-

Figure 3 Candidates for System Hardware and Software Components.

Any such operations required by the client application must first be implemented in the client API and then supported as formal commands by the server application. Of course, the necessary functions for executing those commands must be available in the record/play server API. If they are not, additional API functions can always be created as required.

One typical client application is a virtual instrument panel GUI displayed on the monitor. With buttons, knobs, sliders, switches, indicators, status windows and parameter entry windows, the operator simply uses the mouse and keyboard to control operations. Such an application could be written in Visual C, Visual Basic or Java to make a visually attractive and functional layout. The GUI would make the appropriate calls to the client API according to which buttons are pushed or which parameters are entered.

A larger client application might need a record/playback subsystem as an I/O resource. In this case, the larger application might be written in C, C++ or any other language supported by the operating system. Like the GUI, it would also make calls to the client API to set up the hardware, start and stop the recording and then fetch data back into the application. By adding this type of functional subsystem that is easy to use and fully characterized, system designers and integrators can slash development time and reduce risks.

The rationale for each of the many software blocks in this software partitioning scheme should now be appreciated. Instead of a single monolithic program, the modular architecture of this system helps custom application developers take advantage of the standardized, well-defined interfaces between the modules to add new features, commands and functions. The existing commands and subroutine structures offer excellent examples for building new ones that are fully compliant with the rest of the system. Operating system revisions, maintenance upgrades and ports to different operating systems all benefit from this modularity.

When it comes to appropriate candidates for the various modules discussed, there are many choices available (Figure 3). Client workstation platforms for these systems range from hand-held devices, blade servers, embedded PCs, laptops and desktop PCs to networked clusters of high-end multiprocessing systems. In each case, the processors must support a diverse set of infrastructure functions best handled by operating systems such as Windows, Linux, Unix or Solaris.

Client applications and the client API can be written in C or C++, and GUI components can use visual versions of these tools. A popular trend of using Java for both the client applications and API helps with portability across platforms and operating systems. The signal viewer application is a good candidate for LabVIEW because the software offers tools specifically oriented to signal processing and display and is now available for many workstation environments.

By its nature, the real-time server system is less heterogeneous and runs under operating systems such as VxWorks, eCos or LynxOS. Most of the components—including the board support libraries, server application and API—and the network and disk drivers are all written in C or C++, while some lower-level functions are coded in assembly language.

Putting It All Together

SystemFlow is one implementation of this software framework. It was developed to address hardware platforms like the real-time recording/playback system discussed above, which is similar to the Pentek RTS2504 Real-Time Recorder system. SystemFlow includes all of the software modules described above, with client workstation software modules written in Java and LabVIEW and server real-time modules written in C. By following this proposed software architecture, SystemFlow successfully fulfills two different product objectives for the same hardware: a ready-to-use record/playback instrument and a real-time signal processing development platform.

To meet the needs of the record/playback instrument, special enhancements were made including a complete virtual instrument GUI, a real-time file manager and a full-featured signal viewer. The workstation GUI was written in Java and runs under both Windows and Linux (Figure 4). Intuitive buttons, indicators, status windows and parameter entry windows are geared for novice users who simply want to capture signals and transfer files to their workstation.

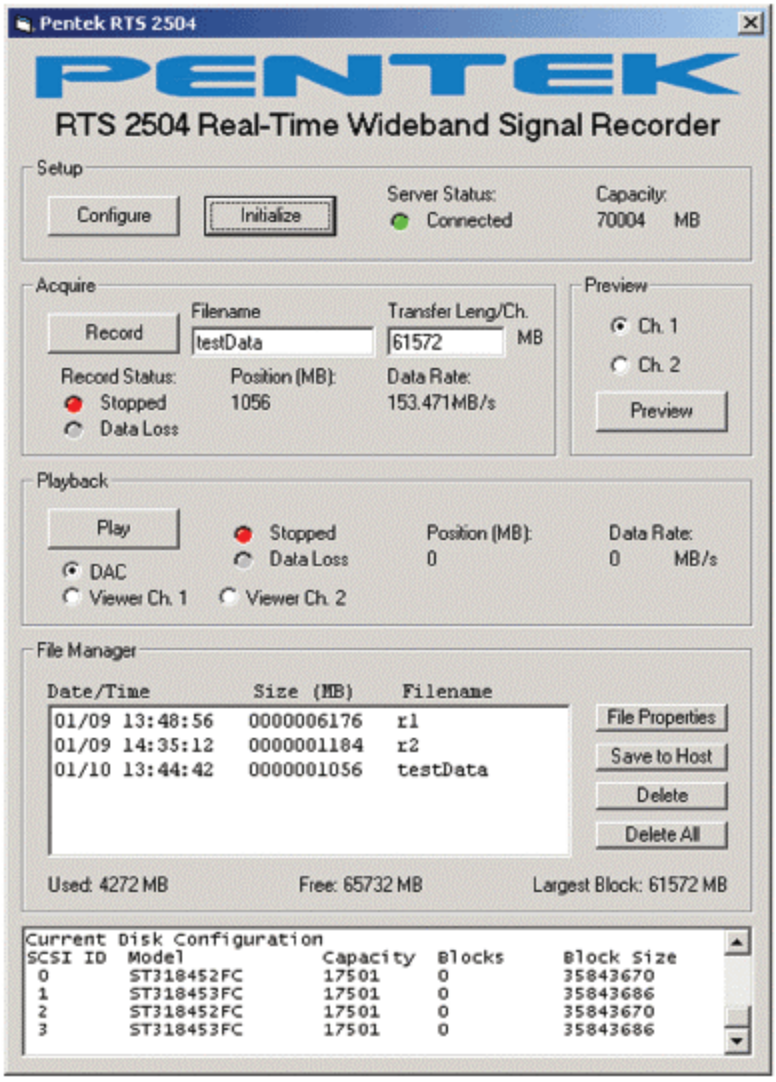

Figure 4 Pentek RTS2504 Virtual Instrument GUI.

A sophisticated file manager is implemented through extensions to both the client API, also written in Java, and the server API, written in C. It supports user-named files and headers that automatically store important system parameters in each recording. The client signal viewer, written in LabVIEW, includes display windows for time and frequency domains, dual annotated cursors and automatic calculation of critical signal parameters such as harmonic distortion and signal/noise ratios.

To meet the alternate needs of a real-time signal processing development platform, SystemFlow includes source code for all software modules created for the record/play instrument. System developers can start with this fully functioning instrument and incrementally extend, replace or modify each module as required to meet their custom requirements.

For customizing workstation modules, Java source code is provided for the client GUI application and client API, and the LabVIEW script is provided for the signal viewer.

For customizing server modules, a complete eCos development environment running under Windows offers license-free, open-source tools including the GNU compiler, Insight for the GDB debugger, CYGWIN make utilities, an eCos kernel configuration utility and a TFTP server. C source code is supplied for the record/play server application and server API, the file manager and the socket interface. ReadyFlow board support libraries include C source code for the data movement, mode and parameter initialization, timing and control of all hardware resources on the boards.

For FPGA code development, Xilinx's ISE Foundation Tool Suite is installed on the Windows workstation. Pentek's GateFlow FPGA Design Kit contains all ISE project files and VHDL source code for the specific hardware boards. In this way, FPGA developers can build upon the standard interfaces and structures already instantiated. An FPGA code loader utility transfers newly created FPGA bitstreams through the Ethernet link into the FPGA.

All software development tasks for the workstation, the real-time server and the FPGAs are performed on the Windows workstation. All real-time server development tasks are supported across the Ethernet link through drivers and utilities. Except for the optional Xilinx ISE tools and GateFlow Design kit, all of the above resources are bundled into the SystemFlow package.

The unique requirements of each real-time embedded system will drive choices in the hardware, the nature and function of the software modules, the operating systems and software languages. However, the philosophy of the software architecture outlined here should prove valuable in helping to make those decisions when starting a new design.

Pentek
Upper Saddle River, NJ.
(201) 818-5900.
[www.pentek.com].