A MODULAR INTERFACE BETWEEN CUSTOM PCI INSTRUMENTATION AND COMMERCIAL SOFTWARE

L. Day, J. Power, M. Stettler, Los Alamos National Laboratory, Los Alamos, NM, USA

Abstract

Control of custom instrumentation as accomplished through platforms consisting of unique hardware, drivers, process and control programs, and user interfaces can alternatively be implemented with custom PCI hardware, custom driver software, and commercial process and control applications including the user interface. Process control of SNS custom diagnostics PCI instrumentation is accomplished through applications developed from standard commercial software. The system's design maximizes modularization of the hardware, software, and operator interface, which simplifies the complexity of development by requiring minimal overlap of expertise between the hardware designers, software engineers, and operators. Cost of development is also kept minimal by integrating commercial software packages at the lowest level possible. SNS diagnostics hardware designers independently create LabVIEW applications to verify the functionality of their instruments. The software driver modules are implemented as DLLs containing exported functions callable by any number of client processes. A library of sub-VIs is pre-configured with driver function calls establishing the communications connection for instrument control and data acquisition.

INTRODUCTION

The diagnostic systems designed for the Spallation Neutron Source (SNS) beam-line were designed to be developed modularly using commercial products, simplifying the design and development of intelligent instrumentation. The standard PC was chosen as the platform with the idea that commercially available application development environments (ADE) and driver development tools could be used for the diagnostic application and the interface to the custom hardware respectively. This would enable a stand-alone system to be built simplifying development and testing. Figure 1 shows the stand-alone software structure design.

A mechanism for integrating with the SNS global controls would be provided and could be installed when the system was commissioned.

The use of both commodity and real-time OSs were considered. Microsoft Windows was chosen due to the large variety of commercial tools available that supported both application and system software development. Windows does not support real-time, deterministic data acquisition and requires support at the hardware and/or driver level. However, real-time extensions [1] are available for Windows if driver level implementation becomes cumbersome. Other advantages of commodity systems include a self hosting development environment and the ability to develop on common desktop units. Finally, embedded versions are available to reduce the OS to the minimal footprint for deployment.

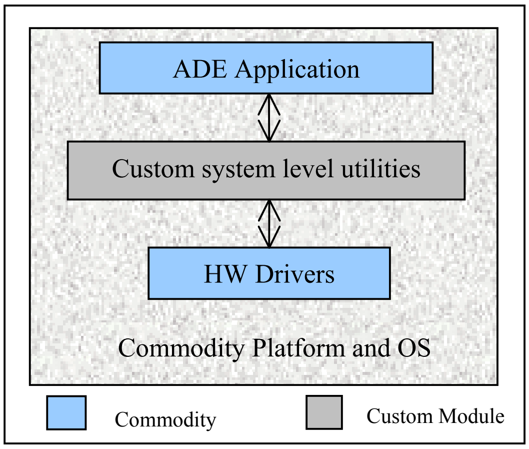

Figure 1: Basic Software Structure.

For the low-level driver development, a kit available by Jungo Software Technology [2], WinDriver, was selected. This generic driver product provided an API for immediate access to hardware at the user level system support modules. WinDriver also handles OS interface details like installation and hardware detection (plug and play).

For the ADE, National Instruments' LabVIEW [3] was chosen. With its powerful tools for analyzing data, signal processing applications were created easily. It also includes the ability to link to external software modules for extended functionality. This facilitates this system's API development based on Jungo driver. LabVIEW's use in custom instrumentation has already been demonstrated on other diagnostic projects such as Low Energy Demonstration Accelerator [4].

DLLs were developed to provide stable APIs between commercially developed modules. One DLL was created for communications between LabVIEW and WinDriver. Another was developed for efficient high-level system communications, allowing other applications to access the system's data without interfering with the LabVIEW control application. While not an integral part of the instrument functionality, this DLL was developed to interface with the SNS controls, EPICS [5].

MODULARIZATION

The use of commercial products contributed to the natural modularization of the system resulting in segregation of development tasks and providing design flexibility. The most notable advantage of this design was the reduction of the amount of specialized software required due to the ability to use a commercial ADE as the user interface. The hardware designer was free to develop and debug his hardware with the ADE of his choice. Specialized low-level data processing and analysis routines were no longer needed.

The savings blanketed over areas including time-and-effort, system cost, and development coordination resulting in the ability for a small team to deliver an easily maintainable system of economical commercial products in a timely manner.

HARDWARE

Further modularization of the system was done at the hardware level. The hardware was designed to consist of a general PCI motherboard that supports various custom acquisition DFEs [6]. The hardware designer's insight of the advantages of this design is note worthy.

The system's specialized software requirements were again reduced with the use of one generic motherboard. One device driver was needed. In addition, the development of the driver was done during the design phase of the acquisition hardware. As soon as a motherboard prototype was available with the basic register map, the driver was developed to ~90% completion. Details of the register map and driver were worked out as the acquisition modules became available. The remaining testing was left to the hardware designer using his choice of ADEs.

DRIVER

The device driver was created using Jungo's commercial driver development tool, WinDriver. WinDriver is an API providing a library of standard driver functions for configuring and accessing a variety of supported hardware designs. This API enables a driver to be developed and debugged in user mode, within a system DLL or low-level application.

SNS diagnostic systems use WinDriver to detect and register its custom PCI card(s). It maps the card's physical address space into the local process's virtual address memory according to user level configuration specifications. This includes contiguous, non-paged system pool buffers for DMA data transfers. WinDriver also comes with a default interrupt service routine for basic interrupt handling and contains a mechanism to spawn an interrupt handling thread for customized interrupt processing.

The availability of these API features to access the card's user and system resources through memory addressing provide a significant benefit to the diagnostic system's development cycle by eliminating the necessity of a specialized, kernel mode driver. WinDriver's API is optimized for minimal performance overhead and is more than adequate for this system's support. However, another Jungo development tool, KernelDriver, is available for running performance critical modules at the OS kernel level.

WinDriver also supports automated simulation capabilities. If WinDriver scans the bus and does not find a diagnostics PCI card, a user level DLL proceeds to configure a simulated card's address space. This highly advantageous capability allows ADE processing algorithms to be developed before actual hardware is available or attached. Thorough testing can be completed in a controlled environment, eliminating any questions of hardware affects on the process control.

The complete driver control configuration is encompassed within a user level DLL. The DLL provides a complete and easily interfaced API for the ADE layer and includes both driver support as well as SCADA control. This is shown in Figure 2 and discussed in the following section.

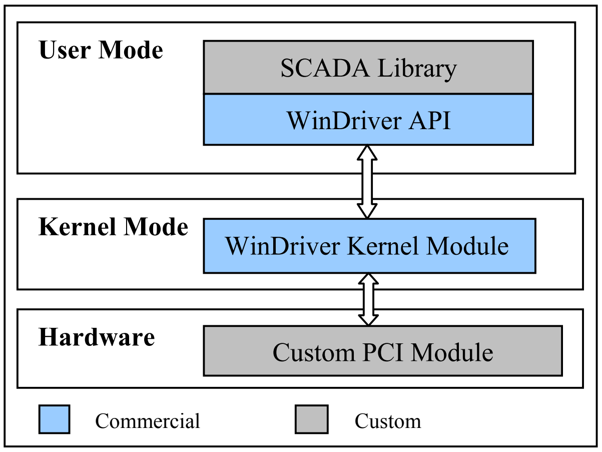

Figure 2: Hardware Control

API

The API to the hardware is a system DLL providing the complete driver interface and SCADA control library for the ADE layer. Functions are exported from the DLL to be imported into the ADE application. These functions include both high-level process routines and low-level card access as well as hardware simulation support.

As mentioned previously, the hardware is designed as separate modules including a general PCI motherboard supporting various DFE daughter cards. The DLL's high-level routines provide applications with generic functions for SCADA control. These function's interface with the motherboard and provide routines that complete operations such as initialization, status inquiries, and DMA requests. Within these functions are calls to the DFE routines to complete the processing by supplying details such as address specifications and reading/writing appropriate registers.

The low-level DFE calls are also exported and are useful when debugging the hardware. Individual commands can be initiated to test hardware functionality such as register read/writes, interrupt handling, and memory transfers.

The simulation capabilities are extremely valuable during the application development phase. Thorough testing of the application's processing algorithms can be completed off-line and without any confusion as to the affects of the hardware.

This API is to be used for single process control. A separate, higher-level API is available enabling inter-process communications and data distribution.

ADE

Although there were many choices of ADEs for the controls application, LabVIEW was chosen due to its familiarity among the development team and its strength as an engineering programming language. The hardware designer was able to develop sophisticated signal processing algorithms quickly using LabVIEW. An example of a LabVIEW data acquisition and processing application is the SNS BPM system [6]. This system acquires eight channels of I and Q data from RF probes. The LabVIEW application uploads and processes this data to determine the beam phase and beam position. Phase is determined relative to the reference and beam magnitude signals and position is determined by comparing the signal amplitudes between the four probes.

Because LabVIEW includes a mechanism for dynamically linking to system DLLs, control of the hardware and uploading of the I and Q data was easily accomplished using the driver/SCADA API. This along with the full processing capabilities resulted in a system that was completely built and tested on a stand-alone system with minimal team effort.

DISTRIBUTED CONTROL

For SNS commissioning, the diagnostic controls data was distributed to the SNS global controls system, EPICS. A high-level DLL was developed to communicate with the controls protocol, EPICS Channel Access. This protocol distributes data in small, logically related blocks. Therefore, the DLL was designed to create a database like structure in system memory, where the diagnostics data would be held. The data would be available in small blocks of related data, which could easily be transferred between itself and the SNS network.

However, control of the instrumentation is limited to a single process and was to remain within the LabVIEW application. Both LabVIEW and the SNS controls require the ability to update and retrieve data from the DLL's system memory. Therefore, the DLL exports functions that both processes call to maintain current knowledge of the system's status. LabVIEW retrieves control requests originating from the SNS system and updates the data sets to be distributed. This is shown in Figure 3.

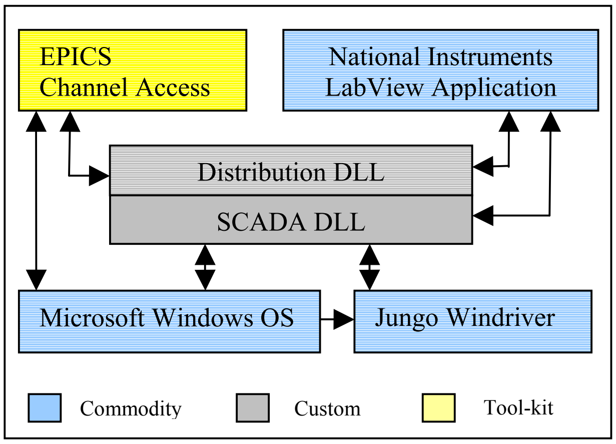

Figure 3: System Distribution

This distribution mechanism provides valuable benefits. While the system meets the SNS integration requirements providing distributed data for remote processes, it is easily converted back to a stand-alone system providing the ability to maintain and upgrade off-line. An additional benefit is that processes other than SNS controls can access the data, for example, an IDL application could complete secondary processing on the data in real-time.

CONCLUSION

The use of a commodity system and commercial software simplified and reduced the cost of deploying the SNS diagnostics custom instrumentation controls. The modular design proved to have significant advantages in the areas of development, maintenance, cost, and team support. Most notably, the use of commercially available products reduced the amount of specialized software required and simplified the development process.

REFERENCES

[1] Venturcom, http://www.vci.com
[2] Jungo Inc, http://www.jungo.com
[3] National Instruments Inc, http://www.ni.com
[4] "Automated Control and Real-Time Data Processing of Wire Scanner/Halo Scraper Measurements," L. Day et al., particle Accelerator Conference, 2001
[5] Experimental Physics and Industrial Control System
[6] "Beam Position Monitor Systems for the SNS LINAC", J. Power et al., Particle Accelerator Conference, 2003