USING EPICS ENABLED INDUSTRIAL HARDWARE FOR UPGRADING CONTROL SYSTEMS*

E. Björklund, Los Alamos National Laboratory, Los Alamos, NM 87545, U.S.A.
A. Veeramani, T. Debelle, National Instruments, Austin, TX 78759, U.S.A.

Abstract

Los Alamos National Laboratory has been working with National Instruments (NI) and Cosylab to implement EPICS Input Output Controller (IOC) software that runs directly on NI CompactRIO Real Time Controller (RTC) and communicates with NI LabVIEW through a shared memory interface. In this presentation, we will discuss our current progress in upgrading the control system at the Los Alamos Neutron Science Centre (LANSCE) and what we have learned about integrating CompactRIO into large experimental physics facilities. We will also discuss the implications of using Channel Access Server for LabVIEW which will enable more commercial hardware platforms to be used in upgrading existing facilities or in commissioning new ones.

WHAT IS CompactRIO?

CompactRIO is a “Programmable Automation Controller” (PAC) manufactured by National Instruments. I/O modules plug into a 4 or 8 slot bus where they are directly connected to the input pins of an FPGA. A PCI bus connects the FPGA to a Real-Time Controller (RTC), which connects to the Ethernet. Both the RTC and the FPGA are programmed with LabVIEW.

LabVIEW is supported on a variety of platforms including Microsoft Windows (2K, XP, Vista), Linux, and Macintosh. LabVIEW Real-Time, the embedded systems solution for LabVIEW, is supported on Pharlap and VxWorks targets on hardware platforms such as PXI and CompactRIO respectively.

The CompactRIO product fits nicely into the niche traditionally occupied by Programmable Logic Controllers (PLCs). It was an attractive choice for us because it is designed for harsh environments, is faster than traditional PLCs, and easily configurable through the FPGA to match the behaviour of the home-built equipment we were replacing.

WHY PUT EPICS ON CompactRIO?

Typically, Ethernet-attached devices such as PLCs require two network trips to reach an EPICS application such as an archiving engine or an operator display. The first Ethernet trip takes the data from the device to an EPICS record, usually via a vendor-specific proprietary protocol. The second Ethernet trip takes the data from the EPICS record to the application. Obviously, we could avoid the extra Ethernet hop if EPICS ran directly on the Real Time Controller.

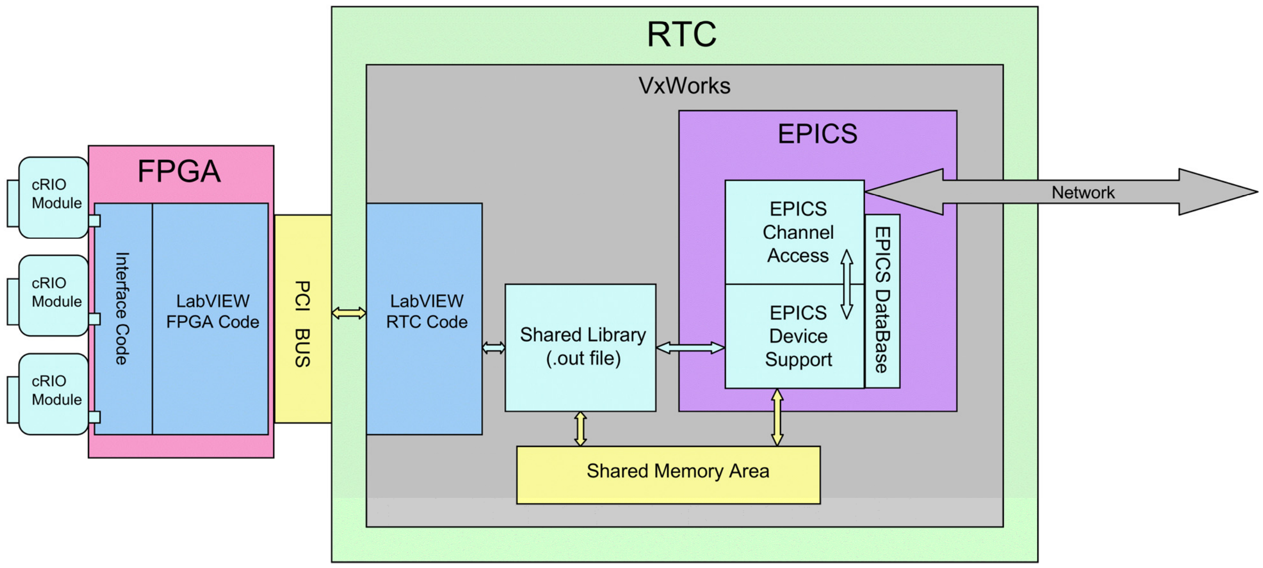


Figure 1: System architecture for running EPICS on the NI CompactRIO platform.

*Work supported by the U.S. Department of Energy under contract DE-AC52-06NA25369

Running EPICS directly on the RTC also gives us access to all the EPICS tools such as the sequencer, "bumpless reboot", and our locally developed diagnostic and introspection utilities. This, in turn, provides a lot of flexibility in how the application is partitioned.

RUNNING EPICS ON CompactRIO

The CompactRIO RTC is a Power PC processor running VxWorks – an architecture already supported by EPICS. Cosylab modified the CompactRIO Board Support Package (BSP) to include NFS and Telnet, two utilities we needed to support our EPICS environment. They also provided a prototype shared memory library to interface between EPICS and LabVIEW. The priorities of the EPICS-enabled VxWorks kernel are adjusted so that EPICS runs at lower priorities than the LabVIEW code. This prevents EPICS from interfering with LabVIEW's time-critical functions.

The following files are needed to run EPICS on the CompactRIO RTC:

- **An EPICS-enabled VxWorks kernel**. This file will reside in the RTC's non-volatile memory file system. It is best to give this file a different name (e.g. VxWorks_epics) than the pre-loaded VxWorks kernel.
- **A startup script.** This script will load the EPICS IOC code, database, and applications. This file also resides on the RTC's non-volatile memory file system.
- **An EPICS/LabVIEW shared memory library**. This library comes in two parts and is described in the next section.

Once you have loaded the EPICS-enabled kernel and the startup script into the RTC's non-volatile file system, you will need to modify the VxWorks boot parameters to boot the EPICS-enabled kernel and to invoke the startup script.

The architecture of the running system is shown above in Figure 1.

THE SHARED MEMORY LIBRARY

The shared memory library is similar in concept to the LabVIEW/EPICS shared memory library implemented at the Spallation Neutron Source [1]. It comes in two parts; a LabVIEW interface file, and an EPICS interface file.

The LabVIEW Interface Library

The LabVIEW portion of the shared library is responsible for allocating and controlling access to the shared memory resources. It may only call VxWorks routines. The RTC LabVIEW code interfaces to this library through "Call Library Function Node" VI's. This part of the library also resides on the RTC's volatile memory file system. It is automatically loaded when the RTC VI that references it is loaded.

There are actually two versions of this file. The first version is a VxWorks shared object library (an EPICS ".munch" file will work too). This is the version that lives on the RTC's non-volatile file system and it must have a ".out" suffix. The second version is a Windows DLL file. It resides on the PC where you develop the LabVIEW code and provides the interface information to the "Call Library Function Node" VI's. Since its only use is as an interface specification, it may contain just "stub" routines. It does not need to contain any EPICS or VxWorks references.

The EPICS Interface Library

The EPICS portion of the shared library is basically the EPICS "Device Support" layer. It interfaces to the LabVIEW portion of the library for read/write access to the shared memory resources. This part of the library lives on the EPICS development host computer as a standard EPICS shared application.

It is worth pointing out that there need not be only one LabVIEW/EPICS shared library. Different libraries with different functionality can happily exist on the same system and not interfere with each other.

EXPERIENCE

In the spring of 2009, we replaced the binary input and output controls for one accelerating module with an embedded-EPICS CompactRIO system. Once installed, the new system ran flawlessly for the duration of the run cycle. The bulk of the application was implemented in the FPGA. The RTC LabVIEW code's only function was to serve the data between EPICS and the FPGA. The EPICS code's only function was to serve the data between the operator interface and the CompactRIO. As a result, the RTC itself was very lightly loaded, and the relative priorities between EPICS and LabVIEW were not a problem. Because we implemented the bulk of the logic in the FPGA, we did experience some difficulty getting the logic to fit on the 3 million gate Virtex 2 FPGA. Our next prototype will use the newer CompactRIO systems with Virtex 5 FPGAs.

We also found that we had to create separate CONFIG files in the EPICS "base/config/os" directory because the CompactRIO used a different VxWorks version than our other IOCs used. We did this by copying the ".vxWorks-ppc603_long" files and giving them a ".vxWorks-cRIO" suffix.

ENABLING COTS HARDWARE WITH EPICS

While CompactRIO was used for this specific application, there will be need for other hardware such as PXI-based instruments and even non-NI hardware. To make these commercial-off-the-shelf (COTS) hardware natively communicate over a Channel Access network either an EPICS IOC running on the hardware is required or at the minimum a Channel Access server is needed to publish the values as process variables (PV). To run an IOC embedded on a hardware is non-trivial task and would need extensive development time. With LabVIEW 2009, you can take advantage of the built-in EPICS Server to make your hardware EPICS "compatible".

The EPICS server in LabVIEW is really a Channel Access server that publishes data points from within LabVIEW as process variables over a channel access network. Instead of using the EPICS IOC infrastructure to write your function blocks and schedule them, you can use LabVIEW to write all your control and data acquisition functions using the thousands of built-in I/O libraries, control and analysis function blocks in LabVIEW.

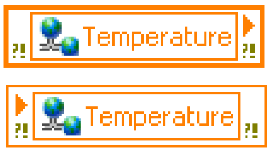


Figure 2: LabVIEW shared variable nodes.

To publish the PVs over channel access, LabVIEW uses the shared variable engine (SVE) that runs alongside LabVIEW and LabVIEW Real-Time applications [2]. The SVE provides a generalized mechanism for LabVIEW applications to interact in a distributed manner with other LabVIEW and third-party applications. Developers use 'shared variables' to read and write values to and from the SVE. Shared variables classified as "memory tags" are bound to networked data sources such as other shared variables, OPC tags, or EPICS process variables. The SVE uses a plug-in architecture which makes it possible to interface with a wide range of data protocols.

LabVIEW already includes Channel Access client support which is implemented using this plug-in.

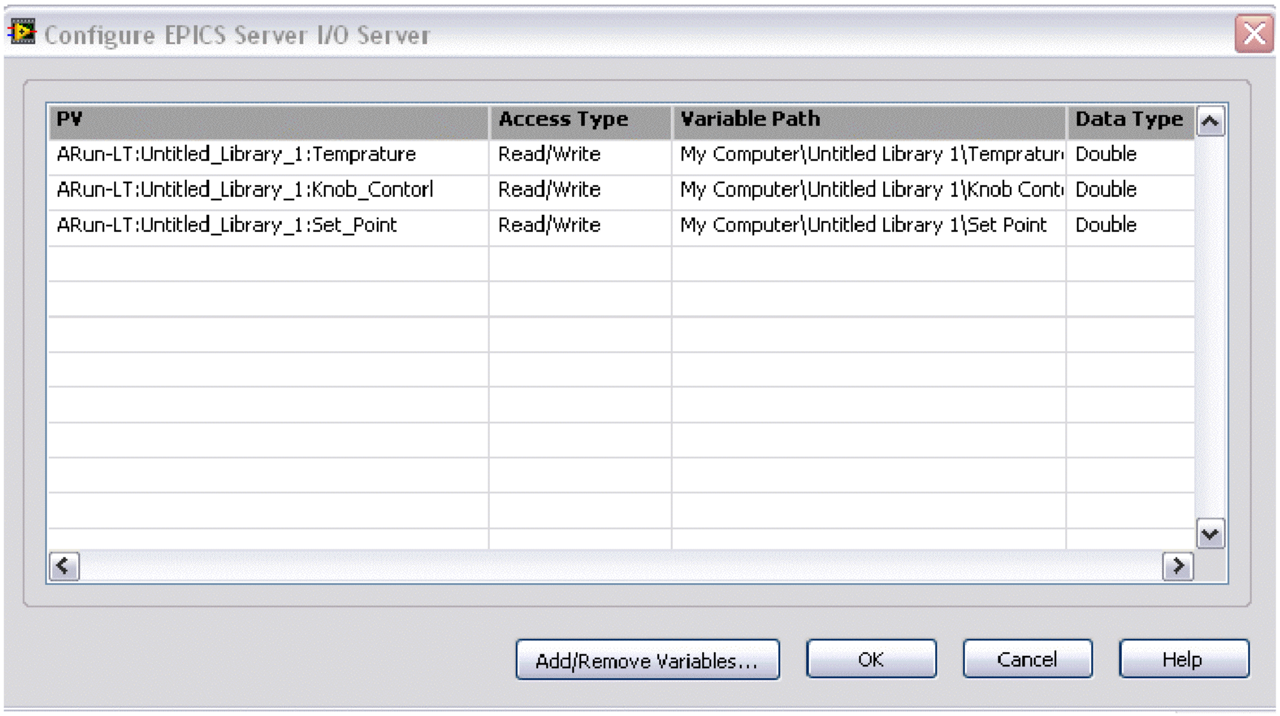
ACKNOWLEDGEMENTS

The authors would like to thank David Bonal, and Ryan King of National Instruments for their invaluable support and assistance with the LabVIEW portion of this project. We would also like to thank Rok Šabjan of Cosylab for his work on the VxWorks and EPICS portions, as well as for the initial shared library implementation that integrated all the parts together.

REFERENCES

[1] A. Liyu, W. Blokland, D. Thompson "LabVIEW Library to EPICS Channel Access", PAC'05, Knoxville, TN. May 2005, FPAT053, p 3233.

[2] A. Veeramani, K.E. Tetmeyer,R. Sabjan, A. Zagar "Interfacing EPICS IOC and LabVIEW for FPGA Enabled COTS Hardware", PCaPAC'08, Ljubljana, Slovenia. October 2008, TUX01.



PV	Access Type	Variable Path	Data Type
ARun-LT:Untitled_Library_1:Temprature	Read/Write	My Computer\Untitled Library 1\Tempratur	Double
ARun-LT:Untitled_Library_1:Knob_Contorl	Read/Write	My Computer\Untitled Library 1\Knob Cont	Double
ARun-LT:Untitled_Library_1:Set_Point	Read/Write	My Computer\Untitled Library 1\Set Point	Double

Figure 3: Configuring variables in LabVIEW to be published over Channel Access.