INTERFACING EPICS IOC AND LABVIEW FOR FPGA ENABLED COTS HARDWARE

A. Veeramani, K.E. Tetmeyer, National Instruments, Austin, TX, U.S.A.
R. Sabjan, A.Zagar, Cosylab, Ljubljana, Slovenia

Abstract

Several attempts have been made to integrate EPICS functionality with National Instruments LabVIEW. With existing EPICS code, labs want to reuse the code while still being able to use LabVIEW to interface with FPGA enabled embedded controllers and other COTS hardware. In this paper, we will show how we can run EPICS IOC simultaneously with LabVIEW on VxWorks based hardware. We will go into the implementation details and the benchmarks that will be obtained from the LANSCE-R project at Los Alamos National Labs. We will also examine ways to implement a Channel Access(CA) server natively in LabVIEW. This will open up the opportunity to use a variety of IO and different operating systems that LabVIEW can interface with. The native LabVIEW CA server will implement all Channel Access functionality exposed by a standard EPICS IOC such as synchronous and asynchronous publishing of data, alarm processing, and response to connection requests by CA clients. We will finally cover the programming of FPGA allowing for custom solutions.

Several attempts have been made to integrate EPICS functionality with National Instruments LabVIEW. Notably, the Spallation Neutron Source implemented a shared memory based interface that allows LabVIEW to interface directly with an EPICS IOC [1]. GSI implemented LabVIEW and EPICS DIM interfaces to allow the two execution systems to interact [2] [3]. These solutions provide various levels of interaction between LabVIEW and EPICS entities, but many users have expressed interest in a more integrated solution that allows LabVIEW VIs to interact directly with EPICS networks. To this end, we propose the inclusion of a Channel Access Server (CAS) for LabVIEW and LabVIEW Real-Time.

LABVIEW TECHNOLOGY BACKGROUND

National Instruments LabVIEW is a cross-platform, high-level, and general purpose programming language. It is used in a wide range of test and measurement, control system, and embedded applications. 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 CompactRIO and PXI.

LabVIEW includes a shared variable engine (SVE) execution system that runs along-side LabVIEW and LabVIEW Real-Time applications. The SVE provides a generalized mechanism for LabVIEW applications to interact in a distributed manner with other LabVIEW and 3rd party applications. LabVIEW application developers use a simple read/write API to access shared variables on a LabVIEW block diagram. Shared Variables classified as "memory tags" are bound to networked data sources such as other shared variables, OPC tags, or EPICS process variables.

Figure 1: LabVIEW Shared Variable Nodes

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 architecture.

BRIDGING LABVIEW AND EPICS

There are several ways to integrate LabVIEW and EPICS to take advantage of the numerous commercially available hardware that are available through LabVIEW. This paper discusses in detail two ways:

- Cohabitation of EPICS and LabVIEW Real-Time on the FPGA-based platform – NI CompactRIO
- Channel Access (CA) server in LabVIEW enabling any LabVIEW enabled hardware to appear as an EPICS node

COHABITATION OF EPICS AND LABVIEW REAL-TIME

The approach discussed here involves running a full EPICS IOC on cRIO's vxWorks operating system alongside with LabView Real-Time. A prototype of this solution was developed and is in use in the Los Alamos National Laboratory.

The main requirements for the system were:

- System must run full EPICS IOC on the VxWorks platform
- The FPGA must be configurable using LV FPGA
- The two main processes (LV RT and EPICS) must be able to efficiently exchange data using different data types and arrays
- System must be configurable without a need for re-compilation of the EPICS source code (use of text configuration files)

Architecture and Design

It was decided to base our work on the Windows-based shared memory approach implemented by Spallation Neutron Source [1]. An important decision was to set priorities of all EPICS related tasks lower to those of LV RT.

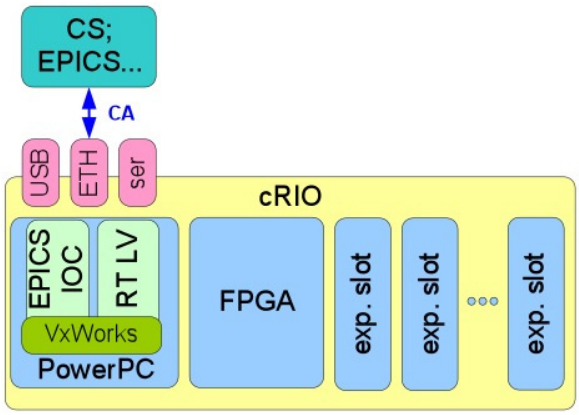

Figure 2: Basic co-habitation architecture

Due to simplicity of the prototype, it was decided that the shared memory would only provide separate memory blocks for each data-type. Additionally, no event-based communication was implemented, thus relying on the polling only.

Implementation

Several issues required attention during implementation. First, we needed to change the CompactRIO's default VxWorks image to include NFS (network file system) and NTP (network time protocol) support. The NTP was even required to have a smooth EPICS compilation for the platform without source modifications. NFS was mainly used for convenience to speed up development time as it eliminated file copying for every change.

We tested several EPICS modules that are very popular and common. We did not experience any challenges with general modules such as the asyn driver and stream device. However, we experienced challenges with the sequencer (on the Windows side of the compile) and the VxStats package. We left the sequencer out of the package and only a small sub-set of VxStats functions was used.

A set of special LabVIEW VIs was implemented taking advantage of the shared library call node. On the EPICS side a device support module was developed for each supported data type.

Benchmarks

We measured the performance for round-trips of data pairs. We used two variables, where LV was increasing one of them if they were equal and EPICS was trying to level them by increasing the second one. Thus we obtained a measurable transfer rate.

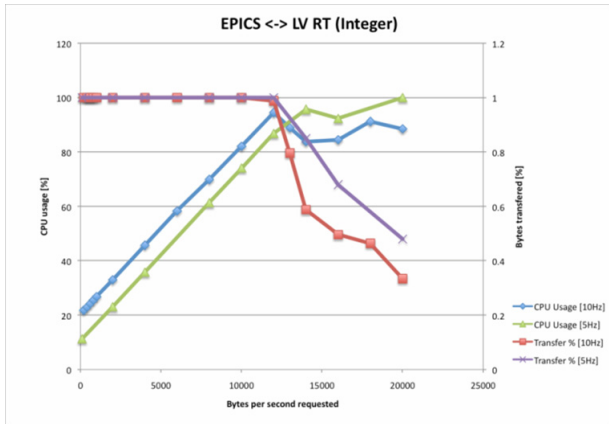

Figure 3: Data transfer rate and CPU usage for integer pair exchange.

The Figure 3 shows four data series. Two of them (blue and green) depict CPU usage that increases linearly with the requested data rate until it reaches the vicinity of 100%. The remaining series depict the percentage of successful data trips. EPICS tasks were set to lower priorities and were starved of CPU time thus failing to finish the full circle of the data roundtrips on some occasions.

The maximum data exchange rate was obtained using arrays – we reached 2-3 MB transfer rate, which seems to be sufficient for the proposed application.

Future Scope

Prototype demonstrates a simple interaction between LabVIEW and EPICS both running on the same cRIO machine and VxWorks operating system. It lacks interrupt/event driven mechanism to provide more synchronous and responsive messaging between both systems. Instead communication is based on setting and polling values of predefined shared variables contained by an independant shared library acting as a middleman.
Such concept was taken from the SNS solution of LabVIEW Shared Memory Interface to EPICS IOC, which serves a similar purpose only for different operating system, i.e. Microsoft Windows. In the SNS solution, shared memory is based on memory-mapped file implemented in Windows API. Our prototype is in fact even simpler and is using statically allocated memory that can due to VxWorks' primitive memory management be accessed from any process running on the same system. Protection against concurrent access has not yet been used nor tested although it will be required in the future.

NATIVE CHANNEL ACCESS SERVER IN LABVIEW

A new Channel Access (CA) server in LabVIEW will allow LabVIEW or LabVIEW Real-Time developer to write a LabVIEW-based application that integrated seamlessly with existing EPICS networks.

In order to allow LabVIEW-based control applications to participate in EPICS-based control systems National Instruments will develop a Channel Access Server plug-in for the Shared Variable Engine that exposes Shared Variables as EPICS Process Variables.

This proposal will allow users to implement all the control functions using the function blocks in LabVIEW and use the Chanel Access (CA) server plug-in for the Shared Variable Engine that would be developed to expose the Shared Variables as EPICS Process Variables (PVs).

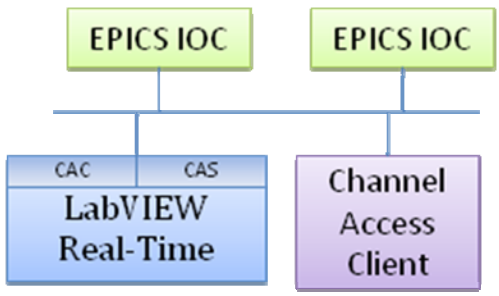

Figure 4: LabVIEW RT on a Channel Access Network.

Process Variable Naming

Maintaining unique PV names is necessary when implementing a distributed EPICS control system. In contrast to the flat PV namespace used in the CA protocol, LabVIEW utilizes the DNS name of the computing target to define a hierarchal namespace for shared variables. It is necessary to provide a mapping between these two name representations.

The network path for a shared variable in a LabVIEW application would have the general form:

\\TargetName\Library Name\Variable Name

Internal to LabVIEW, users can access the variable using this path. Since Channel Access uses a flat namespace for process variables, the LabVIEW Channel Access Server could publish a process variable in the general form:

VariableName.FieldName

where the field name could correspond to those fields defined by LabVIEW shared variables, such as description and alarm values and limits. The LabVIEW developer would be responsible for configuring which shared variables to publish via Channel Access and to provide a mapping to a PV name. One option is to handle this mapping automatically with-in LabVIEW by maintaining some form of the hierarchy structure. For example, PVs could have the general form:

TargetName:LibraryName:VariableName

An alternative is to provide the user with a mechanism to pre-pend an arbitrary string to the variable name, such as:

UserString:VariableName

Under this type of naming option, the application developer would be responsible for de-fining unique PV names within and between targets since LabVIEW's native namespace mechanism would no longer detect name collisions.

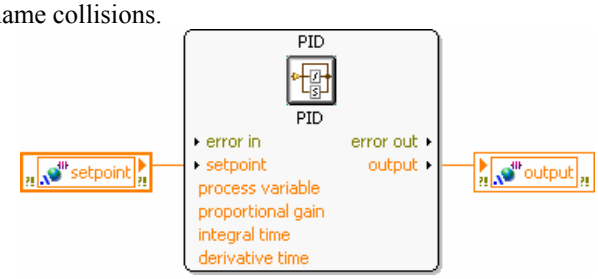

Figure 5: Use of LabVIEW Shared Variables.

This proposal outlines a method for integrating additional EPICS support into LabVIEW utilizing communications via the Channel Access protocol. It is intended to allow LabVIEW developers to create applications that interact with existing EPICS systems while allowing the developer to continue to use the full power of LabVIEW.

Another appealing alternative to these above two solutions would be a hardware implementation for the control system connectivity. This would be accomplished by developing a cRIO extension module with a separate CPU that would run an open operating system (e.g. Linux) and EPICS and would communicate with the main CPU through the cRIO bus and FPGA.

REFERENCES

[1] D. Thompson, W. Blokland, "A Shared Memory Interface Between LabVIEW and EPICS," http://epaper.kek.jp/ica03/PAPERS/TU514.PDF

[2] D. Beck, H. Brand, "Control System Design Using LabVIEW Object Oriented Programming," http://ics-web4.sns.ornl.gov/icalepcs07/TPPA01/TPPA01.PDF

[3] "LabVIEW DIM Interface," http://wiki.gsi.de/cgi-bin/view/CSframework/LVDimInterface#Native_interface_LVDimInterface