ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV011

INTERFACING EPICS AND LabVIEW USING OPC UA FOR SLOW CONTROL SYSTEMS

Jalal Mostafa, Armen Beglarian, Suren A. Chilingaryan, Andreas Kopmann
Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany

Abstract

The ability of EPICS-based control systems to adapt to heterogeneous architectures made EPICS the defacto control system for scientific experiments. Several approaches have been made to adapt EPICS to LabVIEW-based cRIO hardware, but these approaches, including NI EPICS ServerI/O Server: (1) require a lot of effort to maintain and run, especially if the controllers and the process variables are numerous; (2) only provide a limited set of metadata; or (3) provide a limited set of EPICS features and capabilities. In this paper, we survey different solutions to interface EPICSwith LabVIEW-based hardware then propose EPICS OPCUA device support as an out of the box interface between LabVIEW-based hardware and EPICS to preserve most of EPICS features and provide reasonable performance for slow control systems.

INTRODUCTION

The KArlsruhe TRItium Neutrino (KATRIN) experiment is a large-scale scientific experiment to determine neutrino mass using Tritium beta decay [1]. Large-scale scientific experiments like KATRIN employ diverse hardware from different vendors to support the control system infrastructure, e.g., NI cRIO, Siemens S7, custom hardware chips, etc. For instance, KATRIN employs around 10,000 process variables that are distributed in bunches of 100 to 300 process variables on different NI cRIO, NI cFieldPoint, and Siemens S7 devices. The heterogeneity of such systems imposes a new challenge of providing a unified layer of control and interoperability between these heterogeneous systems and other services like alerts, data archiving, and analysis. The Experimental Physics and Industrial Control System (EPICS) solves this problem using an intermediate C++ software layer.

EPICS is a set of tools and libraries to develop distributed server-client control systems for large-scale scientific experiments e.g. particle accelerators. It can solve the challenge of heterogeneity by acting as an abstract software layer for all devices through implementing a C++ abstraction interface called Device Support. Operator's setpoints and sensor measurements are then channeled through EPICS server or Input/ Output Controller (IOC) using one of EPICS network protocols: Channel Access (CA) and its successor Process Variable Access (PVA). Figure 1 shows a simplified illustration of the EPICS server architecture of the EPICS server. An EPICS server keeps all process variables in in-memory structures called Database. When a client asks for a process variable, the EPICS server access these structures using a module called Database Access. The EPICS server can read data from the hardware using the device support layer where the data is mapped to the corresponding process variable through database access and then published on the network using CA or PVA. Operator's setpoints work similarly, but in the opposite direction: an EPICS client (the operator) sets a process variable on an EPICS server using CA or PVA, which is implemented on the hardware using Device Support.

This architecture imposes new challenges on the interoperability between EPICS and NI LabVIEW-based cRIOs. The LabVIEW programming environment offers very tight integration with EPICS. Several approaches have been made to provide an integration for LabVIEW with EPICS, but they usually require an effort to maintain and run, provide a limited set of metadata, or provide a limited set of EPICS features and capabilities. In this paper, we propose an EPICS-LabVIEW integration based on Open Platform Communications Unified Architecture (OPC UA) protocol.

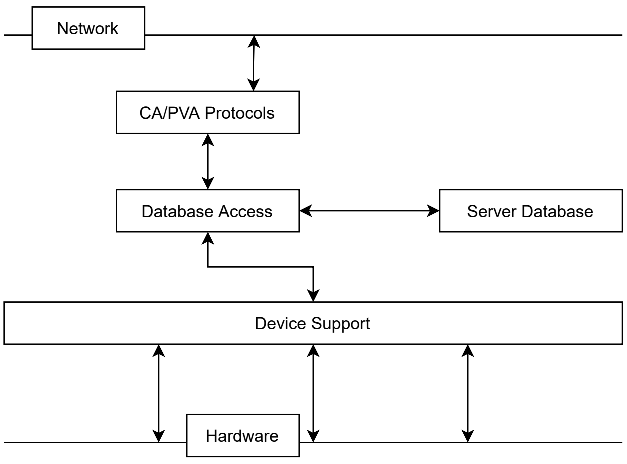

Figure 1: EPICS Server Simplified Internal Architecture.

RELATED WORKS

Several works have been done before to interface LabVIEW with EPICS: most notable is *EPICS Server I/O Server* [2]. *EPICS Server I/O Server* is a complete implementation of EPICS CA protocol for LabVIEW from NI. It allows the developer to create LabVIEW Shared Variables to publish EPICS process variables on the network, a complicated process that involves many clicks to accomplish a simple task, especially when the process variables are numerous. The metadata that EPICS Server I/O Server provides is limited to alarms only leaving behind important metadata like description and engineering units.

Another LabVIEW/EPICS interface that depends on Shared Variables is *NetShrVar* [3]. NetShrVar uses the proprietary NI-PSP protocol to bind a process variable in an EPICS C++ server to a LabVIEW shared variables using a set of complex configurations written in XML.

lvPortDriver [4] allows users to develop EPICS device support in LabVIEW code and to start an EPICS C++ server through the same LabVIEW program. This is possible through an *asynDriver* [5] C++ layer and a LabVIEW VI library. Although *lvPortDriver* inherits platform independency from EPICS but compilation and deployment are still needed for each specific NI target CPU architecture.

[6–8] employs shared memory between LabVIEW program and EPICS C++ server to interface EPICS with LabVIEW. Other than being in need to compile for each NI target architecture, these solutions are platform-specific and they are hard to port from one operating system to another.

IRIO [9], *Nheengatu* [10] allow developers to run EPICS on cRIO and other NI devices based on a middleware library that allows direct access to NI hardware, but it depends on the target architecture.

EPICS-LABVIEW INTERFACE USING OPC UA

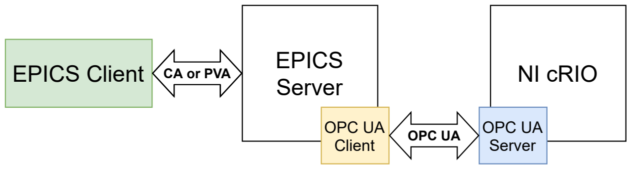

Figure 2: EPICS/OPC UA Architecture.

OPC UA provides seamless integration with EPICS. It is an open service-oriented protocol that can run on any hardware and operating system and thus provides a good fit for interoperability between EPICS and LabVIEW-based hardware.

Figure 2 shows how EPICS can interface with LabVIEW. The LabVIEW developer writes standard OPC UA code on the target device. An standard EPICS server can communicate to the OPC UA server running on the LabVIEW target using a device support module for OPC UA [11].

The OPC UA Device Support for EPICS provides the easy representation of hierarchical data structures, access control, and data monitoring. All standard data types for both input and output are supported. It adds an extra record type *opcuaItem* which allows representing OPC UA objects in EPICS by destructing the OPC UA object to standard data types records.

The OPC UA server/client architecture abstracts the communication of EPICS with the cRIO and thus allows deploying a one-time compilation EPICS server on any commodity server or virtual machines or containers to add modularity, isolation, and scalability to the architecture.

PERFORMANCE EVALUATION

To evaluate this architecture for slow control systems like KATRIN, we launch an OPC UA server on NI cRIO 9047 and then run an EPICS server running on a workstation with Intel Xeon CPU E5-1630 v3 @ 3.70 GHz and 1 Gbps Ethernet connection, and linked to the cRIO through the EPICS OPC UA Device Support module. The OPC UA objects in our experiments are represented using a flat data architecture: each process variable is represented by an OPC UA variable inside a distinct OPC UA object. We operate experiments by simulating sensors (read from cRIO to EPICS) and setpoints (write to cRIO through EPICS). Table 1 shows the experimental setup of 300 process variables based on the requirements of the KATRIN control system.

Table 1: Update Rate and Expected № of Elements for Each of the 300 Values

Update Rate	Exp. № of Values in 1 Hr.
10 Hz	36000
2 Hz	7200

Sensor Mode

We launch two experiments for the sensor mode evaluation where, in both experiments, the OPC UA server on the cRIO is generating incremental values for each of the 300 process variables for a duration of an hour: (1) 2 Hz experiment, one value for each of the process variables every 500 ms; and (2) 10 Hz experiment, one value every 100 ms. The EPICS server is monitoring all the 300 variables in both experiments over one OPC UA subscription, then we use *PyEPICS*[12] to read the values from the EPICS variables through CA monitoring. The EPICS OPC UA Device Support is configured to use the timestamp from the OPC UA protocol.

In an hour interval, we noticed that the expected number of values for each record is attained. No loss of data in both 10 Hz and 2 Hz experiments.

Setpoints Mode

We simulate setting values to actuators through EPICS and that are controlled by the OPC UA server. We then launch an EPICS client developed in C with real-time priority threads (one thread per process variable), as illustrated in Fig. 3. We want to check two metrics in this mode: lost values rate and response time (time delay between setting the value in EPICS and implementing the setpoint in the OPC UA server). This is possible through having two records for each EPICS process variable in the EPICS database: (1) the *ao* output record to output the variable to the OPC UA server using EPICS timestamp, and (2) the *ai* readback record to read the process variable values back using an OPC UA subscription and OPC UA timestamp. We then monitor both records; the loss rate is then computed by comparing values from both monitors (output and readback).

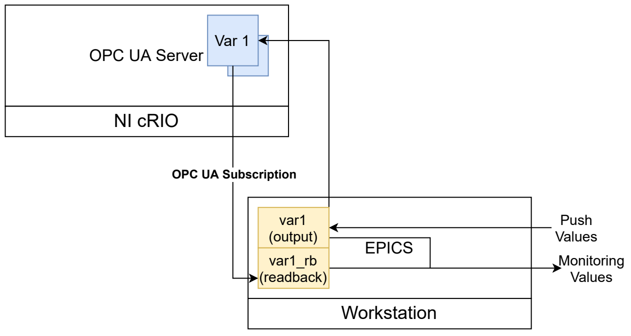

Figure 3: Setpoints Mode Experimentation.

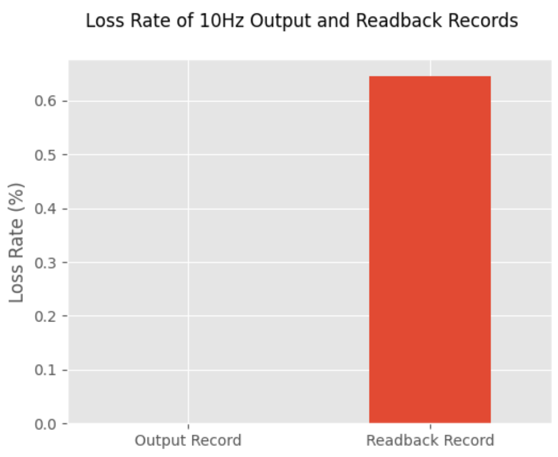

Figure 4: Comparison of Data Loss between the Output Record and the Readback Record in 10 Hz Experiment for the 300 Process Variables.

While the loss at 2 Hz is zero in the output and the readback records, the loss rate at 10 Hz is zero for the output record but around 0.6% for the readback in an hour experiment interval which means EPICS has processed the setpoints very well, but the OPC UA server did not implement them. Figure 4 shows the loss rate of the output record using the output monitor and that of the readback record using the readback monitor.

CONCLUSION

Using OPC UA protocol can be a good connection interface to integrate EPICS C++ code and NI LabVIEW-based hardware without a strong impact on performance for slow control systems. Comparing both 2 Hz and 10 Hz experiments, we notice the performance is limited by the OPC UA server which depends on the performance of the cRIO or the system hosting it. Therefore, we plan to use this architecture for the KATRIN experiment where the maximum update rate is 2 Hz and can guarantee no data loss.

REFERENCES

[1] J. Wolf, "The KATRIN neutrino mass experiment," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 623, no. 1, pp. 442–444, 2010, 1st International Conference on Technology and Instrumentation in Particle Physics, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2010.03.030. https://www.sciencedirect.com/science/article/pii/S0168900210005942

[2] A. Veeramani, T. Debelle, W. Blokland, R. Dickson, A. Zhukov, and O. RAD, "Options for interfacing EPICS to COTS hardware through LabVIEW," 2009, THD004.

[3] Network shared variable epics support module, (2013), http://epics.isis.stfc.ac.uk/doxygen/NetShrVar/

[4] lvPortDriver, (2020), https://github.com/lanl/lvPortDriver

[5] Epics asyn driver, (2021), https://epics.anl.gov/modules/soft/asyn/

[6] D. Thompson and W. Blokland, "A shared memory interface between LabVIEW and EPICS," in *Proceedings of ICALEPCS2003*, Korea, 2003, pp. 275–277.

[7] A. Liyu, W. Blokland, and D. Thompson, "Labview library to epics channel access," in *Proceedings of the 2005 Particle Accelerator Conference*, 2005, pp. 3233–3234. DOI: 10.1109/PAC.2005.1591423.

[8] G. Li and J. Zhao, "Application of LabVIEW-EPICS in measuring and monitoring system of BEPCII," *Nuclear Electronics and Detection Technology*, vol. 26, no. 2, pp. 222–225, 2006.

[9] M. Ruiz *et al.*, "IRIO technology: Developing applications for advanced DAQ systems using FPGAs," in *2016 IEEE-NPSS Real Time Conference (RT)*, 2016, pp. 1–5. DOI: 10.1109/RTC.2016.7543090.

[10] D. Alnajjar, G. Fedel, and J. Piton, "Project Nheengatu: EPICS support for CompactRIO FPGA and LabVIEW-RT," in *17th International Conference on Accelerator and Large Experimental Physics Control Systems*, Oct. 2019. DOI: 10.18429/JACoW-ICALEPCS2019-WEMPL002.

[11] R. Lange *et al.*, "Integrating OPC UA Devices in EPICS," in *18th International Conference on Accelerator and Large Experimental Physics Control Systems*, Oct. 2021, MOPV026.

[12] Pyepics, (2014), https://pyepics.github.io/pyepics/