ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA146

INTERFACE BETWEEN EPICS AND ADO*

A. Sukhanov†, J. P. Jamilkowski, A. Marusic, BNL, Upton, NY 11973, USA

Abstract

EPICS is widely used software infrastructure to control Particle Accelerators, its Channel Access (CA) network protocol for communication with Input/Output Controllers (IOCs) is easy to implement in hardware. Many vendors provide CA support for their devices. The control systems of the Collider-Accelerator Department (C-AD) at Brookhaven National Laboratory (BNL) is a complex system consisting of approximately 1.5 million [1] control points. The control of all devices is unified using an Accelerator Device Objects (ADO) software abstraction layer. In this paper we present software solutions for cross-communication between two different platforms. They were implemented for the integration of a NSLS II Power Supply Controller hardware into the RHIC Controls System.

INTRODUCTION

EPICS Software Infrastructure

EPICS is used in more than 100 of independent projects, number of controlled process variables (PVs) ranges from 1K to 300K.

The key features of the EPICS infrastructure:

- Client-server model: device is controlled by an Input/Output Control program (IOC), which provide support for 'records'. Records provide access and control to process variables.
- Client-server protocol: EPICS Channel Access protocol (CA).
- Transport layer: TCPIP.
- Name server: No central name server.
- Number of lines in the Base Code: 200K of C code.
- Source availability: Open source.

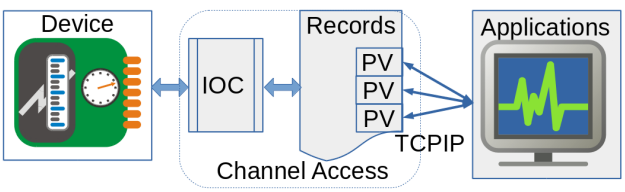

Figure 1: EPICS client-server model.

RHIC Controls Infrastructure

The RHIC control system provides the operational interface to the collider and injection beam lines. The equipment under control includes more than 350 VME crates, and hundreds of network-capable devices. The control of all devices is unified using Accelerator Device Objects (ADO) software abstraction layer

The key features of the RHIC Controls infrastructure:

- Client-server model: device is controlled by an ADO Manager program, which hosts the process variables.
- Client-server protocol: RPC.
- Transport layer: TCPIP.
- Name server: CNS, an RPC-based server program, connected to central Sybase database.
- Number of lines in the Base Code: pure Python implementation: 5K, (original C++ implementation: ~200K).
- Source availability: proprietary.

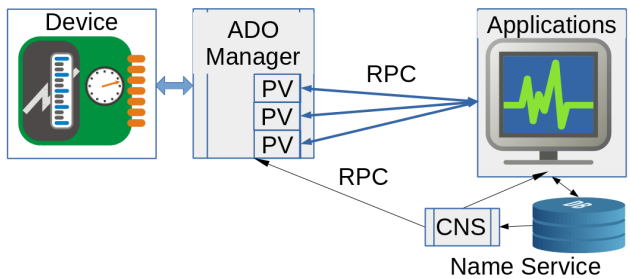

Figure 2: RHIC Controls client-server model.

CONTROL OF THE EPICS-MANAGED DEVICE IN RHIC ENVIRONMENT

The server for an EPICS-managed device should monitor and react on two sources of changes:

- monitor changes of the EPICS PVs (e.g using EPICS API ca_pend_event() loop) and change the corresponding ADO PV.
- monitor changes of the ADO PVs (e.g. using ADO API HandleNextEvent() loop) and change the corresponding EPICS PV.

The logically simplest approach is to build two separate applications: epics2ado (based on camonitor from EPICS API) and ado2epics (based on adoIf from ADO API) as illustrated on Fig. 3.

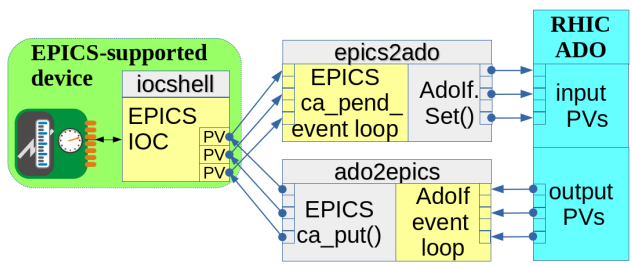

Figure 3: Two-way translation between EPICS and ADO.

* Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA146

This approach have been implemented using standard C/C++ tool chains, provided with the each of the API. It exposed following disadvantages:

- Very complicated compilation and linking. The both applications have to be linked to full sets of EPICS and ADO libraries.
- Difficult transition from 32bit to 64bit architecture.
- A separate ADO manager is required to host the ADO PVs.

Python ADO Manager With PyEpics Module

The Python interfaces for both systems had been developed recently: PyEpics [2] for EPICS and PyADO [3] for ADO. These interfaces provide an easy way to develop an ADO manager which directly communicates with the EPICS IOC. The PyADO package is pure Python, the PyEpics requires libca.so shared library.

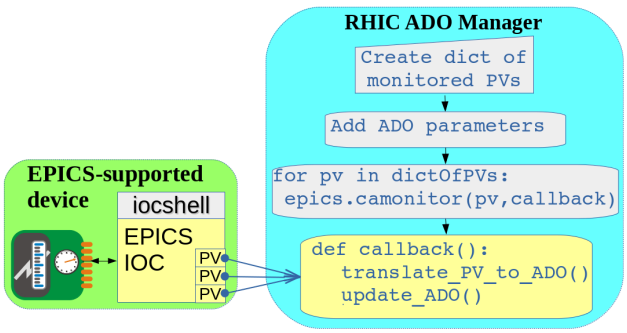

Figure 4: Python ADO Manager for the EPICS IOC.

During initialization, the ADO manager creates a dictionary between the EPICS PV names and the ADO PVs and methods. During the PV creations, the parameters, which should be translated to EPICS have the set() function defined to use epics.caput() for setting the parameter. The manager calls the epics.camonitor() with the same callback method for each PV. The callback method will be called when any of the EPICS PV have been changed, in that case the update_ADO() is called to set the corresponding ADO PV.

Advantages:

- Link to EPICS API is simple, the only requirement is that the PYEPICS_LIBCA environment variable should point to a valid shared library libca.so.
- 32bit/64bit architecture independence.
- The program is much more concise, for example, the number of source lines is ~300 as opposed to ~2200 for the two-way translation approach.
- The performance is not affected, the PV access time is ~1 ms, it is mainly defined by the handling inside of the RPC layer.

Optionally, instead of using epics.camonitor() the monitoring of the EPICS PVs can be done by launching a system call to execute a camonitor program. This is particularly useful for ssh access to EPICS IOC. The PV access time in that case is ~ 0.4 s.

CONTROL OF THE ADO-MANAGED DEVICE IN EPICS ENVIRONMENT.

For the reversed problem, when a device, provided with the ADO support, needs to be used in the EPICS environment, the solution is shown on Fig. 5.

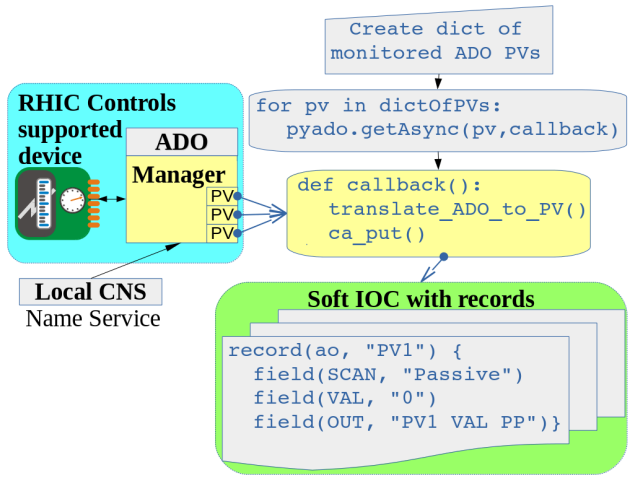

Figure 5: Control of the ADO-managed device in EPICS environment.

The soft IOC program, serving the device, calls the pyado.getAsync(), which is the ADO equivalent of the epics.camonitor(), to assign the same callback function to every EPICS PV. The callback() translates incoming ADO PV to outgoing EPICS PV and calls ca_put() to set it. The soft IOC is hosting the database of 'ai' and 'ao' records.

The CNS program should be configured to resolve the ADO names using a local file.

CONCLUSION

The described solutions have been applied to control a 180 degree bending magnet for Low Energy RHIC Electron Cooling Project [4], controlled by an EPICS-based IOC [5]. The number of PVs provided by the device IOC is close to one hundred. It was successfully implemented into the RHIC Control System and supplied with the GUI, similar to the native Control System Studio. The most effective approach is to use the Python ADO Manager with PyEpics module, it provides the same performance as C/C++ solution and eliminates the need for complicated compilation and linking.

The solution for reversed task, when the device is delivered with the RHIC Control support needs to be used in the EPICS environment, is presented on Fig. 5.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA146

REFERENCES

[1] K. A. Brown, "C-AD Controls Systems various notes on history, architecture, and modern systems", in Collider-Accelerator Department Documentation - Notes, 2012.

[2] PyEpics: Epics Channel Access for Python, http://cars9.uchicago.edu/software/python/pyepics3

[3] K.A. Brown , T. D'Ottavio, W. Fu, A. Marusic, J. Morris, S. Nemesure, A. Sukhanov, October 2017, paper TUPHA153, this conference.

[4] J. Kewish, A.V. Fedotov, D. Kayran, S. Seletskiy, October 2016, paper WEA4CO05, Proc. of NAPAC2016.

[5] Y. Tian, W. Louie, J.Ricciardelli, L.R. Dalesio, G. Ganetis, October 2009, paper WEB005, ICALEPCS2009.