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 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.

EPICS DEVICE in RHIC ENVIRONMENT

Option 1: Two-way translation

Prerequisites:
- Device is provided with EPICS support.
- The target ADO already exists at RHIC Controls.
- The host computer has all EPICS Base libraries installed.

Two processes on the host are compiled and linked with ADO and EPICS libraries:
1. `epic2ado`: EPICS to ADO translator, runs `ca_pend_event_loop` to monitor changes in all EPICS PVs, the ADO parameters are updated using `addif.Set()`.
2. `ado2epics`: ADO to EPICS translator, runs `adoif.event_loop` to monitor changes in ADO parameters, the EPICS PVs updated using `caput()`.

Both programs are using the same translation table of PV names.

Disadvantages:
- Complicated linking
- Number of code lines: 1200 in (1) + 1000 in (2)

Option 2: Python-based ADO manager & pyepics

Prerequisites:
- Device is provided with EPICS support.
- Python package `PyEpics` is installed on the host.

Methods:
- The soft IOC is hosted on the host.
  - Using `caput()` to update ADO parameters
  - Using `epics.camonitor()` to monitor changes in ADO PVs
- The `ai` records are handled in the callback function of the `pyado.getAsync()` and they are set using `caput()`. The `ai` records are handled in the callback of the `epics.camonitor()` and they are passed to `pyepics.set()`.

Option 3: Python-based ADO manager ssh access to EPICS PVs

Prerequisites:
- Same as option 2 but using subprocess.Popen() to execute camonitor and caget program on the server.

Advantages:
- Device can be behind the firewall.
- No EPICS components installed on the host.
- Number of source code lines: 300

Disadvantages:
- PV access time: ~ 400 ms

RHIC Controls

Fig 1 EPICS Client – Server Model

Fig 2. RHIC Controls Client-Server Model

Fig 3. Two-way software bridge

Option 2

Device

Option 3

Device

Fig 4. Python-based ADO manager & pyepics

Fig 5. Control of an ADO-managed device in EPICS environment

EPAD-managed Device

in EPICS Environment

Prerequisites:
- Device is provided with the python-based ADO manager.
- Python package `pyado` is installed on the host.

Methods:
- The soft IOC program is using `pyado.getAsync()`, which is the ADO equivalent of `epics.camonitor()`.

The Soft IOC is hosting database of records. The 'ai' records are handled in the callback function of the `pyado.getAsync()` and they are set using `caput()`. The 'ai' records are handled in the callback of the `epics.camonitor()` and they are passed to `pyepics.set()`.

SUMMARY

If a device, provided with the EPICS support needs to be used in the RHIC Control environment then the most optimal solution is Option 2: Python-based ADO manager, using camonitor method from pyepics Python library.

This approach was implemented to control a 180 degree bend magnet for Low Energy RHIC electron Cooling Project1, controlled by an EPICS-based NSLS-II power supply control system2.

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

REFERENCES

1) http://slidedeplayer.com/slides/6537483/
2) http://www.c-adi.bnl.gov/esfd/LE_RHICeCooling_Project/LERec.htm#Reviews

ACKNOWLEDGMENTS

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