EPICS: OPERATING-SYSTEM-INDEPENDENT DEVICE/DRIVER SUPPORT*

Martin R. Kraimer#, Argonne National Laboratory, 9700 South Cass Ave, Argonne IL 60439, USA

Abstract

Originally EPICS input/output controllers (IOCs) were only supported on VME-based systems running the vxWorks operating system. Now IOCs are supported on many systems: vxWorks, RTEMS, Solaris, HPUX, Linux, WIN32, and Darwin. A challenge is to provide operating-system-independent device and driver support. This paper presents some techniques for providing such support.

OVERVIEW

EPICS [1] (Experimental Physics and Industrial Control System) is a set of software tools, libraries, and applications developed collaboratively and used worldwide to create distributed, real-time control systems for scientific instruments such as particle accelerators, telescopes, and other large scientific experiments. An important component of all EPICS-based control systems is a collection of input/output controllers (IOCs).

An IOC has three primary components: 1) a real-time database; 2) channel access, which provides network access to the database; and 3) device/driver support for interfacing to equipment. This paper describes some projects related to providing device/driver support on non-vxWorks systems.

In order to support IOCs on platforms other than vxWorks, operating-system-independent (OSI) application program interfaces (APIs) were defined for threads, semaphores, timers, etc. [2]. Providing support for a new platform consists of providing an operating-system-dependent implementation of the OSI APIs.

RTEMS

RTEMS is an open-source, real-time operating system [3]. So far at least three sites are using RTEMS IOCs.

CLS – Canadian Light Source

The CLS uses RTEMS and Linux IOCs for their control system [4]. The hardware platforms consist of EROC embedded controllers and standard rack-mounted PCs. An EROC is a low-cost embedded controller developed for the CLS. It is used to control most of the power supplies. It is also used for other control applications such as stepper motor control.

SSRL – Stanford Synchrotron Radiation Laboratory

As part of a major upgrade to the SSRL, the control system is also being upgraded. The upgrade is using VME-based IOCs with RTEMS as the operating system [5]. The IOCs use a number of VME I/O modules that were previously supported only on vxWorks IOCs. EPICS already provided an API (devLib) that was almost an OSI API. devLib provides methods for connecting to interrupts, allocates VME memory, etc. Till Straumann modified devLib so that it works on RTEMS. For the SSRL VME I/O modules, he then modified the existing EPICS drivers to use devLib rather than vxWorks-specific APIs. His changes have been incorporated into EPICS base so that devLib is now a supported OSI API. Thus any driver that uses it can work on both vxWorks and RTEMS.

NSLS – National Synchrotron Light Source

Many beamlines at NSLS have recently been upgraded to use EPICS. NSLS is also investigating the use of RTEMS instead of vxWorks [6]. They are using some of the driver support implemented at SSRL and are using the devLib API for other VME modules.

LABVIEWS

The Spallation Neutron Source (SNS) uses LabViews-based systems for diagnostics. Two methods have been developed for communication between LabViews and EPICS: activeX controls and a shared memory DLL [7]. The shared memory DLL is the preferred method because it provides high performance. Data flows in both directions through the DLL. With this interface a LabViews system appears to the rest of the control system as an ordinary IOC.

PC104

The Jefferson National Accelerator Laboratory (JLAB) has a portable NMR instrument. Previously the system consisted of both the NMR chassis and a VME crate. The VME chassis has been replaced by a PC104 attached to the NMR chassis. The EPICS IOC is a PC104 using Linux for its operating system [8]. The only interface between the IOC and the NMR is a serial port. Thus it was easy to write special device support that talks to the serial port.

GPIBCORE

EPICS, before the OSI extensions, provided support for GPIB. EPICS base provided support for the National Instruments NI1014 VME GPIB controller. Benjamin Franksen converted the code to support additional GPIB controllers [9]. His approach was to define a controller-independent API, implemented by drvGpib, that device support uses, and an additional API that drvGpib uses to access drivers for specific controllers. Benjamin

* Work supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

mrk@aps.anl.gov

implemented a driver for the Hewlett Packard LAN GPIB server. His implementation works on both a workstation and on a vxWorks IOC.

Eric Norum and Marty Kraimer adapted Benjamin's GPIB support to use the EPICS OSI APIs. Thus the support now works on all EPICS-supported platforms. In addition, low-level serial drivers were implemented . As long as the device support doesn't use GPIB-specific methods, like SRQs, it works for serial as well as GPIB instruments. gpibCore support has been used on vxWorks, Solaris, Linux, and Darwin IOCs.

NETWORK-BASED DEVICES

Program Logic Controllers (PLCs) and other intelligent devices come with an Ethernet interface. At KEK in Japan work is underway to provide a common framework to support such devices [10]. The effort uses the EPICS OSI interfaces to provide platform independence.

ASYNCHRONOUS DRIVER SUPPORT

Asynchronous Driver Support (ADS) will be the successor to gpibCore [11]. The goal is to provide a general-purpose facility for interfacing device support code to device instances via a communication driver.

At the present time EPICS sites use several different methods for communicating with serial devices. At each site, however, normally only one method is used. Each method has its own device support interface and also provides low-level drivers for communicating with serial interfaces. On vxWorks/VME systems a widely used serial interface is the SBS octalUART, which is an Industry Pack (IP) module that has eight serial ports. EPICS support for this is available only for vxWorks. In fact, several different versions of the octalUART support are being used. Once a site has chosen a method for communicating with serial devices, it is very hard to change to another method that provides a different device support interface.

With ADS, each existing serial support method can appear the same at the device support layer, but all can use common driver support. ADS is not limited to serial devices but is intended to support any message-based communication interface including serial, GPIB, and Ethernet.

ADS provides the following features:

- A thread for each communication interface.
- A queue of requests to access the communication interface. A queue request is a synchronous non-blocking call.
- When a request is taken from the queue, a callback specified with the queue request is called. This callback can make calls to the communication interface, which sends/receives messages from the actual device.

At present ADS defines the following APIs:

- asynQueueManager – This defines methods for communicating with a device over a communication interface. The implementation provides a thread for each interface. Methods are provided to connect to an interface, queue requests, and lock a set of requests. A method (registerDevice) is called by lower-level drivers to register each communication interface.
- asynDriver – This provides methods to connect and disconnect from a communication interface and to report status about the interface.
- octetDriver – This provides basic methods for communicating with any message-based device, i.e., methods to read and write messages.
- gpibDriver – An API for device support to access GPIB-specific functions, e.g., SRQ handling, addressed commands, universal commands, etc.
- gpibDevice – An API implemented by low-level GPIB drivers and accessed by gpibDriver.

A low-level serial driver must implement asynDriver and octetDriver. A low-level GPIB driver implements only gpibDevice, which includes support for asynDriver, octetDriver, and GPIB-specific methods.

Most device support needs only asynQueueManager, asynDriver, and octetDriver. If device support uses only these interfaces then it will work with ANY low-level communication interface.

The following is a typical sequence of events for establishing a connection to a device:

- asynQueueManager:connectDevice is called. This establishes a connection to the communication interface attached to the device.
- asynQueueManager:findDriver is called to find octetDriver.

The following is a typical sequence of events for issuing a command and getting a response from an instrument:

- queue request
  - asynQueueManager:queueRequest is called. Associated with a request is a callback. When the request is removed from the queue the callback is called.
  - It is possible to prevent other users from accessing the communication interface between queue request by calling asynQueueManager:lock.
- call driver from the callback
  - poctetDriver->write(…)
  - poctetDriver-<read(…)

The queue request is a synchronous nonblocking operation, i.e., it can be called by EPICS scan threads. The queue request has an associated callback routine. The callback calls write to send a message to the device and read to get the response. The write and read requests are handled by a low-level communication driver that interfaces to the actual device.

REFERENCES

[1] EPICS home page
http://www.aps.anl.gov/epics

[2] M.R. Kraimer et al., "EPICS: A Retrospective on Porting iocCore to Multiple Operating Systems," ICALEPCS 2001, San Jose, CA USA, Nov 2001, 238-240, 2002.

[3] W.E. Norum, "EPICS on the RTEMS Real-Time Executive," Proceedings of SRI2001, Madison, Wisconsin, August, 2001, American Institute of Physics, Review of Scientific Instrumentation, January 2002.

[4] G. Wright, "RTEMS & EPICS at CLS," EPICS collaboration meeting, Jefferson Lab, Newport News, Virginia , Fall 2003.
http://www.jlab.org/intralab/calender/archive02/epics

[5] S. Allison, "RTEMS at SSRL," European EPICS Meeting, Abingdon, England, Spring 2003.
http://www.diamond.ac.uk/Activity/EPICS

[6] K. Feng, "EPICS RTEMS OSI @ NSLS," European EPICS Meeting, Abingdon England, Spring 2003.
http://www.diamond.ac.uk/Activity/EPICS

[7] D.H. Thompson and W. Blokland, "A Shared Memory Interface between LabView and EPICS," these proceedings.

[8] A. Freyberger, "PC-104 as an IOC," EPICS collaboration meeting, Jefferson Lab, Newport News, Virginia , Fall 2003.
http://www.jlab.org/intralab/calender/archive02/epics

[9] B. Franksen, "GPIB Update," European EPICS Meeting, Berlin-Adlershof, Germany, Fall 1998.
http://www-csr.bessy.de/control/Epics98

[10] J.O Odagire et al., "EPICS Device/Driver Support Modules for Network-based Intelligent Controllers," these proceedings.

[11] M. Kraimer et al., "Asynchronous Driver Support," European EPICS Meeting, Abingdon England, Spring 2003.
http://www.diamond.ac.uk/Activity/EPICS