The CEBAF Control System[1]

William A. Watson III, Continuous Electron Beam Accelerator Facility,
MS 12H, 12000 Jefferson Av., Newport News, Virginia, 23606, USA

Abstract

CEBAF has recently upgraded its accelerator control system to use EPICS, a control system toolkit being developed by a collaboration among laboratories in the US and Europe. The migration to EPICS has taken place during a year of intense commissioning activity, with new and old control systems operating concurrently. Existing CAMAC hardware was preserved by adding a CAMAC serial highway link to VME; newer hardware developments are now primarily in VME. Software is distributed among three tiers of computers: first, workstations and X terminals for operator interfaces and high level applications; second, VME single board computers for distributed access to hardware and for local control processing (complex sequences, limit checking, some process control); third, embedded processors where needed for faster closed loop operation. In some cases, multiple VME processors transparently access a single serial highway for improved performance. This system has demonstrated the ability to scale EPICS to controlling thousands of devices, including hundreds of embedded processors, with control distributed among dozens of VME processors executing more than 125,000 EPICS database records. To deal with the large size of the control system, CEBAF has integrated an object oriented database, providing data management capabilities for both low level I/O (calibration, alarm limits, etc.) and high level machine modelling (optics properties, etc.). A new callable interface which is control system independent permits access to live EPICS data, data in other Unix processes, and data contained in the object oriented database (extensible to other sources).

INTRODUCTION

The Continuous Electron Beam Accelerator Facility

CEBAF is a 4 GEV electron accelerator in the process of commissioning in Newport News, Virginia, with the first experiments expected to run this summer. The unique features of this facility are its continuous beam and high luminosity -- ideal for experiments requiring large samples of events with minimal accidental coincidence rates.

The accelerator consists of two 0.4 GeV superconducting RF linacs connected by two 180^{o} arcs. Each linac consists of 20 cryomodules, each containing 8 accelerating cavities. Beam is recirculated through the machine for up to 5 passes yielding an energy of 4 GeV. After any pass, the beam may be split and sent to any of the 3 halls, allowing simultaneous operation at the same or different (modulo 20%) energies. Two injectors (one thermionic, one polarized) and a 3 slit chopper will allow different halls to receive different beam intensities and polarization.

Two of the three halls house conventional small solid angle spectrometers, and will use the full beam intensity (200 uA). Hall B will house the CEBAF Large Acceptance Spectrometer (CLAS), which will require beam currents 3 or 4 orders of magnitude lower. In Halls A and C, parity violation experiments will require measurements accurate to a part in 10^{7}. This flexibility in beam delivery and constraints upon beam stability (both current and polarization) place complex demands upon the control system.

The control systems for both the accelerator and the experimental facilities are based upon EPICS -- Experimental Physics and Industrial Control System. [1] EPICS was selected as a replacement for the original control software when problems with scaling to the full machine were encountered nearly two years ago. The following discussion will describe the controls hardware at CEBAF, the use of EPICS in this system, and the higher level software being added above EPICS.

CONTROL SYSTEM ARCHITECTURE

Standard Model

The control system follows what has been referred to as the “standard model”: a client-server system consisting of a collection of Unix workstations and X-terminals connected by a network to multiple servers running device control software. At CEBAF the network is a switched ethernet, which allows simple scaling to high bandwidths as needed. The server machines are VME single board computers running the EPICS real-time database. The client machines are HP workstations configured as two clusters for redundancy (Figure 1).

Conversion to EPICS

EPICS was selected as a replacement for the original CEBAF control system (TACL) when problems were encountered scaling it to over 25,000 control points. [2] The switch to EPICS was accomplished incrementally during machine commissioning, starting with the linacs and arcs a little over a year ago, and ending with the injector (except the gun) this past winter. The gun will be converted following an upgrade to its control hardware this summer.

The two systems were operated concurrently for much of the year, with information being exchanged between the systems. This co-existence was made easier by the fact that both EPICS and TACL are name based control systems -- applications address parameters in the machine by the name of the parameter, and not its hardware address.

[1] Work supported by the Department of Energy, contract DE-AC05-84ER40150. Work performed by the various groups at CEBAF and within the EPICS community.

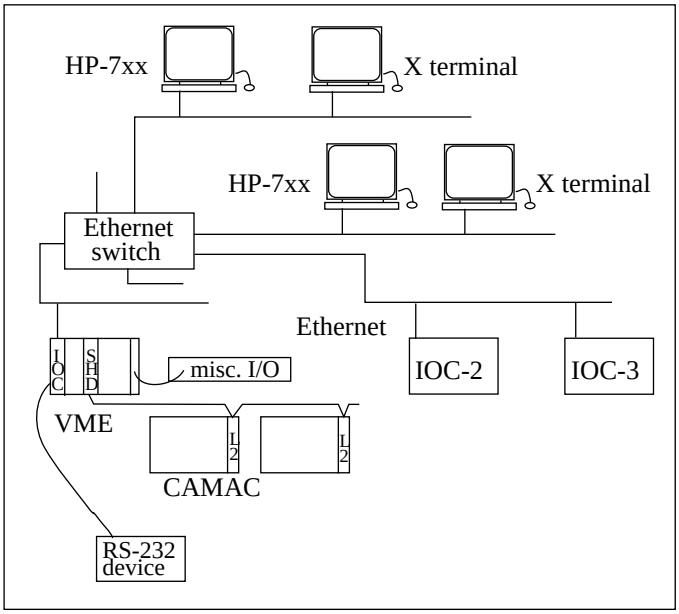

Figure 1: EPICS Architecture showing several displays and IOC's. CAMAC and other hardware is shown for one IOC.

The success of this conversion is evident in the success of machine commissioning. First beam on target (one pass) was delivered within 2 weeks of a date specified 7 years earlier! CEBAF has so far operated beam around 7 of the 9 arcs, and has delivered both pulsed and cw beam to Hall C for detector commissioning activities.

Hardware

In the previous control system, almost all of the machine hardware was interfaced to CAMAC. This investment in hardware has been preserved by replacing the GPIB interfaces to CAMAC with type L2 controllers, which are connected to VME via a serial highway -- with typically 2-8 crates per highway. Newer devices are interfaced directly to VME, or via standards such as GPIB, RS-232, and Arcnet. (Many other buses and interfaces are supported by EPICS at other sites).

Controlled Devices

The bulk of the control system deals with 3 types of devices: RF cavities (~350), magnets (~2000), and beam position monitors (~500). In addition, there are harps, beam viewers, beam loss monitors, gun and injector devices, and ancillary controls and monitoring.

Each RF cavity has a dedicated 8 MHz '186 embedded processor continuously (>20 Hz) adjusting setpoints of an analog feedback system. In addition, these processors monitor for certain critical faults, taking appropriate steps to protect the hardware. A CAMAC card provides an addressable 32 word buffer between the microprocessor and the VME processor, with interrupt support for messages to the '186. During the switch from TACL to EPICS, the embedded software was changed only slightly to support a message based communication protocol.

Embedded processors are also used for each magnet, but with much more limited functionality -- set the current and read the current. Communication with each magnet controller is via RS-485 serial lines, 32 controllers per line. At present, ramp functions are handled by the VME processor, but will be migrated to the embedded processors to improve response time for ramping large numbers of magnets.

Most of the remaining control system hardware is interfaced with commercial or custom CAMAC modules, with the exception of the newer beam position monitor electronics. These devices use a mixture of commercial and custom VME modules to support high speed acquisition of position information (easily tens of kilohertz). Personnel safety is handled by a completely separate PLC based system, and machine protection has both hardware and software components, with the software generally monitoring the state of the hardware.

Additional information about the migration of the low level device control from TACL to EPICS is given in a companion paper. [3]

EPICS

EPICS has been described in previous papers, [1,4] and is well documented on the World Wide Web, [5] so only a brief overview will be given here. EPICS provides a client-server architecture in which the server, called an IOC (I/O controller), executes a real-time database in which each record describes an input, an output, or a calculation.

Database Records

Each record contains a large amount of functionality; for example, an analog input record monitors its input, converts from raw counts to engineering units, compares the value to 2 upper and lower limits (more and less severe alarms), and may cause other records to process. In addition, the record detects significant changes (changes exceeding either of two pre-determined thresholds to support 2 classes of clients), and causes the clients to be notified. Alarms may additionally have hysteresis so that they don't oscillate when near the alarm limit. Simulation mode allows fetching a value from a location other than hardware.

Records may be processed periodically or in response to an external event -- either operator induced, hardware triggered, or software triggered. Records scanning at higher rates preempt slower records, improving real-time behavior.

Channel Access

Most fields within the record are accessible over the network (some read-only). An accessed field is referred to as a "channel", and the network protocol is correspondingly "channel access". [6] At present, channel access servers run only on the IOC, a VxWorks based system. A portable version of the channel access server is in development at LANL to allow any network process to be a server. [7]

The channel access application programming interface (API) is optimized for high performance applications, including buffering all requests and responses and containing support

for unlimited asynchronous replies (in the case of monitoring a value or an alarm status). Clients connect to servers by broadcasting the name of the desired channel (record.field), and broken connections are automatically re-established when a server becomes available again. Several hundred connections per second may be made to a single server, and monitoring several thousand changes each second produces a negligible load on a workstation (6% on an HP 715/50 for 2000 values/sec).

Algorithms

Control algorithms may be implemented via a number of techniques within EPICS. Database records, including subroutine and calc records, may be linked together to form an algorithm, with data transferred from one record to another. Most low level applications use this technique. More complex algorithms may in addition use a state machine sequencer, using a special language and compiler to facilitate this approach. A sequence runs as a channel access client, and may access both local and remote databases as well as any other resources on the VME system. High performance algorithms are implemented as tasks on the IOC, controlled and monitored through database records. This is the approach used for CEBAF's beam position monitors and fast feedback systems. [8] Finally, Unix applications (typically in C or C++) may interact with the EPICS database through channel access.

Utilities

EPICS includes several main general purpose client programs. (1) a save/restore utility, which includes basic check before restore and save-only capabilities; (2) general purpose operator interfaces (one X based, the other Motif); (3) an alarm manager to present alarm status organized into trees of arbitrary depth; (4) an archiver utility supporting 3 styles of data acquisition: (i) periodic sampling, (ii) record on significant change, and (iii) event driven sampling. In the third mode, a change in one channel can initiate recording of values for a set of other channels.

There are a wide variety of other general purpose clients including diagnostic utilities, a knob manager, a parameter page display -- with more being written each year.

EPICS also includes graphical and text based database creation tools, and scripts and other tools to facilitate building and managing the databases.

Integration with other software

EPICS has been integrated with a large number of other packages, including tcl/tk, PV-Wave, IDL, Mathematica, WingZ and others. In each of these packages, EPICS variables are accessible by name through channel access, so that channel access has functioned as a limited form of software bus.

CONTROL DEVICE API

One difficulty encountered with EPICS for high level applications is the fact that the implementation details of a low level algorithm are in many cases too visible: the high level application knows the names of the various records and fields. A change in the low level algorithm which adds or deletes records (moving the needed information to other records) will invalidate the high level application.

A new layer (cdev, for control device) has been added above channel access to provide implementation hiding as well as several additional features. In this new API (defined by a team from all major EPICS sites), all I/O in the system is in the form of messages to devices such as *on* or *off* or *get current*. (Note: cdev builds upon ideas in earlier work done at ANL/APS [9]). In cdev, a device is a virtual entity potentially spanning multiple servers, and even multiple underlying services such as EPICS channel access, an archive data server, a host-based database, or a legacy control system.

The cdev layer routes messages to the appropriate service (such as channel access) based on the device name and the message. In this way, one can obtain the length of a magnet (from a static database) as easily as the current (from the real-time database). The application program is unaware of the source of the data: if the low level application is changed, it is only necessary to fix the mapping, and all high level applications using that information are correct. An architectural diagram of cdev is shown below. All services at the lowermost layer are dynamically loaded, and new services (interfaces to other systems) may be added without recompiling any other cdev sources (in fact without stopping the running application).

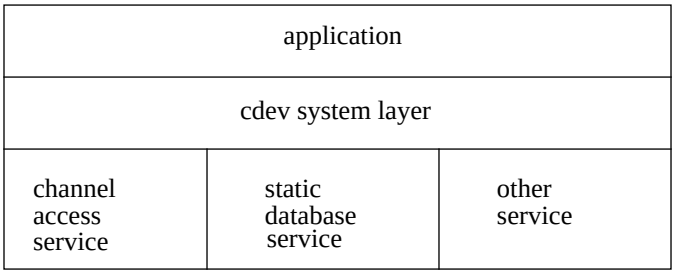

Figure 2: Block diagram of a cdev application with 3 services currently loaded.

An implementation of cdev in C++ has demonstrated that this additional layering introduces on the order of 10% additional I/O processing for both establishing connections and for receiving asynchronous replies. Advantages gained include a device abstraction, implementation hiding, access to a wider variety of data (including a centralized database, described below), wildcard query capability (not present in EPICS), and the ability to treat a collection of devices as a single device.

cdev Applications

A useful cdev demonstration application using tcl/tk for a windowing interface has been written at CEBAF. It allows selecting devices by regular expression, and can read/write/monitor device attributes.

A more sophisticated cdev application now in development will allow measuring correlations among parameters in the control system. This correlation package takes its inspiration from the SLAC Correlation Plot package, which was designed to "measure anything as a function of anything else".[10] Any number of parameters may be systematically

stepped, and at each step point, any number of other parameters may be measured. All I/O will be performed through cdev, making the package independent of EPICS or any other control system. In addition to device I/O, the package will support pop-up windows for operator input or prompting, and the ability to execute shell scripts (including tcl) at any point in the process.

In the long run, most of the general purpose EPICS applications will be re-written to run over cdev. This will decouple them from the EPICS database (giving record implementation independent request files) and allow them to be used in a wider variety of (non-EPICS) environments. For example, the EPICS display program DM (due to be converted this summer) could then be used to control or monitor the control systems of other non-EPICS laboratories -- the only integration expense will be in writing the cdev service layer. A template service will be available to facilitate this integration. Hopefully this ability to exchange utility programs will provide another opportunity to share software development among laboratories.

INTEGRATING AN OO DATABASE

Because of the large size of the control system (over 125,000 records), CEBAF decided a year ago to integrate the ObjectStore object oriented (OO) database with EPICS. Prior to this effort, EPICS contained no centralized database for tracking configuration parameters (hardware addresses, etc.), and operational parameters (alarms and limits, calibrations, etc.). The only tools available were the tools used to construct the databases, and the save/restore utility (smaller systems could also use a spreadsheet for save/restore). Database building tools are not easily used by operators and are generally only convenient for changing all instances of a parameter, for example the skew rate in a magnet. Using save/restore to manage these infrequently changing numbers is possible, but is cumbersome and increases the time to save the machine state.

With a centralized database, instance specific data is more easily managed. Information in the “static” (non-real-time) database is used to construct the real-time database at boot time. Access to this data at run time is through the higher performance IOCs. Persistent changes to these parameters may be made by writing to both the IOC and the oo database. In the future, support may be added for automatically migrating changes from the real-time systems to the static database.

An object oriented database was chosen because it avoids the costly step of table joins to link together dissimilar data, as in displaying relationships among records. Furthermore, the oodb is more naturally accessed by a C++ program in that objects in the database are handled the same way objects in memory are handled. A more detailed discussion of this project is contained in a companion paper. [11]

MODEL DRIVEN APPLICATIONS

High level accelerator control applications generally model the machine as an optical system, with the model parameters stored in a file, a database, or in memory; various applications access the shared parameters. At CEBAF these model parameters will be maintained in memory by a dedicated server process. Access to the parameters of the model (twiss parameters, transfer matrices, etc.) will be by network calls using the cdev interface. [12]

A consequence of this level of integration of the model information and the rest of the control system is that parameters of the model may be displayed and manipulated by the general purpose display programs. Diagnostic information about the operation of the server process may also be monitored in this fashion.

Initially, high level accelerator control applications were developed using the scripting language tcl [13] and its associated packages. A tcl interface to channel access was written at CEBAF [14] which allows tcl to read, write, and monitor EPICS database fields. During the commissioning process, tcl based programs were produced to perform the following functions:

1. linac energy management (adjusts acceleration elements and quads based upon demand energy)
2. energy lock (slow feedback, about 1/2 Hz, to correct the linac energy based upon beam position in high dispersion region)
3. orbit lock (slow feedback from BPM’s to magnets)
4. automated orbit centering in quads
5. automated arc steering
6. automated linac steering
7. optics (general diagnostic package)

Each of these applications used DIMAD [15] to calculate the relevant optics parameters; model information was obtained from a server process via RPC. As the new model server becomes available, the tcl based applications will be modified to use the new model server; programs requiring high performance will be written (or re-written) in C and C++.

SUMMARY

The migration of the control system at CEBAF from TACL to EPICS during a year of commissioning has been largely successful. EPICS is now successfully operating a system an order of magnitude bigger than any previous installation (APS has simultaneously scaled up to a comparable size), and shows every indication of being able to scale up another order of magnitude (with appropriate upgrades in network bandwidth). One can safely say that architecture is no longer an issue -- the “standard” model works.

New features are being added to facilitate the management of large control systems, including an object oriented database for device instance parameters.

CEBAF and others are now in the process of adding high level accelerator control applications above EPICS, with sufficient applications already in place to support commissioning. These new applications are being added in a way which preserves the open toolkit approach adopted by EPICS.

ACKNOWLEDGEMENTS

The conversion of the CEBAF control system was accomplished by deputy head Karen White and other members of the CEBAF controls department, with considerable debugging assistance from the CEBAF operations crew and technical support by visitors from LANL. Work on the cdev specification was done in collaboration with Claude Saunders and other members of the EPICS collaboration at APS, LBL, LANL, DESY, and Keck; the cdev implementation was done by Jie Chen in the Physics Data Acquisition Group, and by Walt Akers and Danjin Wu in the Controls Department. Development of high level applications was done by Hamid Shoaee (on sabbatical from SLAC), and by Johannes Van Zeijts and other members of the accelerator physics and accelerator performance groups.

REFERENCES

[1] Leo R. Dalesio, et. al., "The Experimental Physics and Industrial Control System Architecture: Past, Present, and Future", *International Conference on Accelerator and Large Experimental Physics Control Systems*, Oct. 1993.

[2] William A. Watson III, et. al., "The CEBAF Accelerator Control System: Migrating from a TACL to an EPICS Based System", op. cit.

[3] Sally Schaffner, et. al., "Device Control at CEBAF", this conference.

[4] Leo R. Dalesio, et. al., "The EPICS Architecture", ICALEPCS, 1991.

[5] The web site for EPICS is http://epics.aps.anl.gov.

[6] Jeff Hill, "Channel Access: A Software Bus for the LAACS," ICALEPCS, 1989.

[7] Jeff Hill, private communication.

[8] Mahesh Chowdhary et. al., "A Prototype Fast Feedback System for Energy Lock at CEBAF", this conference.

[9] Claude Saunders, private communication.

[10] L. Hendrickson et. al. "Correlation Plot Facility in the SLC Control System", ICALEPCS, 1991.

[11] Matthew Bickley et. al. "Managing Control Algorithms with an Object-Oriented Database", this conference.

[12] Bruce Bowling et. al., "Integrated On-Line Accelerator Modeling at CEBAF", this conference.

[13] John Ousterhout, "Tcl and the Tk Toolkit" Addison-Wesley, 1994.

[14] Johannes Van Zeijts, private communication.

[15] R. V. Servranckx, User's Guide to the Program DIMAD, SLAC Report 285 UC-28 (A), May 1985.