Fermi National Accelerator Laboratory

FERMILAB-Conf-93/288

The Epicure Control System

E. Dambik, D. Kline and R. West

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

September 1993

Presented at the *International Conference on Accelerator and Large Experimental Physics Control Systems*, Berlin, Germany, October 18-22, 1993

Disclaimer

The Epicure Control System

E.Dambik, D.Kline, R.West
Fermilab*, P.O. Box 500, Batavia. IL 60510, USA

Abstract

The Epicure Control System supports the Fermilab fixed target physics program. The system is distributed across a network of many different types of components. The use of multiple layers of interfaces for communication between logical tasks fits the client-server model. Physical devices are read and controlled using symbolic references entered into a database with an editor utility. The database system consists of a central portion containing all device information and optimized portions distributed among many nodes. Updates to the database are available throughout the system within minutes after being requested..

1 Introduction

The Epicure control system assists in the delivery of different types of particle beams to the fixed target experimental areas of Fermilab. The control system is distributed across a network of VAX[1] computer nodes, VME modules, and CAMAC modules. The VMS[1] operating system executes on each VAX node, the nodes are connected via Ethernet, and network communication is via DECnet[1]. A user at a VAX work station interacts with the system in a windowing environment supported by VWS[1], DECwindows[1], SMG[1], and MOTIF[2] software interfaces. VT100-compatible terminals connected to the system are supported using SMG.

The fixed target experimental areas consist of three principal sections or beamlines. CAMAC crates containing many different types of controllers and data acquisition modules have been installed in the beamline areas for various purposes. All the crates in one area are connected to a single serial CAMAC link. The three CAMAC links are accessed by the Epicure control system through the CAMAC serial link multiplexer and arbiter. One VAX node directly interfaces with each beamline and is called a data acquisition front-end. If one of the three beamline front-end nodes fails, the work of that node is distributed between the remaining two nodes to maintain control system operations on that beamline.

A *Q-bus to VME Interface* (QVI) module connects a data acquisition front-end VAX node to a VME crate containing several special purpose modules. The software executing in the VME modules controls the timing and issuing of CAMAC commands. The QVI maps VMEbus memory into the address space of the VAX. Communication with the software executing on the VAX is via queues residing in the VME common memory.

Epicure front-end nodes are also situated in several cryogenic areas. In these locations, the front-end configuration includes a Multibus interface and must be able to run as an isolated system whenever network access is unavailable. In contrast to the beamline areas, no capability exists to maintain control functions in a cryogenic area if its front-end fails.

The software implementing the standard Epicure interfaces for menu interaction and data acquisition may be installed on any VAX node with the same version of the VMS operating system used on the Epicure nodes. As a result, any user's VAX can interact with the Epicure control system if DECnet network links can be established with the Epicure nodes.

2 Device Specification

Individual Epicure devices are specified by a database entry corresponding to a unique device name. Device entries are based on generic templates that define parameters pertaining to a particular type of hardware. Each database entry contains hardware addressing and other parameters for the set of all reading and setting properties available for that device. These properties apply to various aspects of the physical beamline device and controller and include analog reading, analog setting, digital status reading, digital control setting, and alarms. In the database, each device property includes numerous parameters defining specific attributes of the property. These attributes include hardware addressing, scaling constants and equations, text and bit masks for interpreting digital status and control data, and assigned device protection bit flags. Composite devices may be defined that synthesize a reading using a selected expression and the readings of a set of other devices.

*Work supported by the U.S. Department of Energy under contract No. DE-AC02-76CH03000.

[1] Trademark of Digital Equipment Corporation.

[2] Trademark of Open Software Foundation, Inc.

3 Distributed Database Subsystem

The Epicure database subsystem incorporates both centralized and distributed philosophies to provide simple and consistent interfaces to manipulate and access devices. The centralized portion implements the device modification and reporting functions, whereas the distributed portion serves to divide database accesses among the clients performing data acquisition. The subsystem is composed of several VAX-based processes working in conjunction to synchronize database transactions within the control system.

3.1 Database Overview

The centralized and distributed database philosophies are implemented using two record formats. The central database serves as the master instance for every device within the control system. It contains every property and attribute associated with a device. At the initial design phase, the central database was to be implemented using a commercial relational database product for reasons of consistency and modifiability. However, the products evaluated were found to be expensive, sluggish, and incompatible with the control system configuration and data types. Therefore, due to information obtained from investigation and benchmark testing, the central database was implemented using the VAX/VMS Record Management Services (RMS) with a variable length record format. The Optimized Access (OA) databases are derived from the central database and are implemented as a set of binary files. The OA databases represent a subset of device properties and attributes using a series of fixed and variable length data structures that are primarily relevant for data acquisition.

The subsystem centralizes its database transactions by focusing them through a central database server process. The server is responsible for managing requests for device modifications and synchronizing the derivation of the OA databases. Device requests for data acquisition are serviced by the OA database server process. The request load is distributed by placing copies of the OA databases and server processes on destinations that are standalone or clustered VAX nodes.

The distribution and management of the OA databases are performed by the distributor server process. It is responsible for delivering OA databases to remote destinations and managing their consistency throughout the control system.

The reporting functions are performed by the report server process. It first caches a subset of device information into virtual memory. The server then processes the requests for device information by scanning the cache and comparing each device with the search criteria. Devices matching the criteria are returned to the client along with the requested properties and attributes.

Several applications have been written by system programmers to enhance the interface between users and the database subsystem. By using these applications, users can modify devices, control OA database derivation and distribution, generate device reports, control remote server processes, and monitor system performance and integrity.

3.2 Database Architecture

The central database is implemented using the RMS system services and is an indexed sequential file with variable length records. Each record is partitioned with fixed and variable length data structures combined to represent a device's properties and attributes. Offsets from the beginning of a record are used to access individual data. The database is divided into two RMS files. One file contains all devices in the control system and the other file provides templates used to derive a device.

The optimized databases are derived from the central database and contain a subset of device properties and attributes. For efficiency, two types of optimized databases exist: the *OA database* contains every device in the control system and the *Look Aside List (LAL) database* contains only devices modified since the last OA database derivation. Both databases are similar in format and contain device records composed of fixed and variable length data structures.

The OA and LAL databases are separated into three binary section files: hash, device, and shared. All files contain a serial number used to determine consistency.

The hash section file provides general information about the file set and the initial means to locate a device record. This file is organized into both root and hash partitions. The root partition provides general information about the database. The hash partition contains hash tables and nodes organized as a linked list. Each hash node describes the location of a device in the device section.

The device section file contains the majority of device properties and attributes. This file consists of device records organized as fixed and variable length data structures. These structures contain offsets to the individual properties and attributes within a device record in either the device or shared section.

The shared section file contains the remainder of the properties and attributes common among devices. This file provides a means of efficiently sharing data among devices. The file consists of variable length records that are not in any particular order but are located by offsets. Properties and attributes are intermixed and are referenced using the offsets from the device section file.

3.3 Database Server Processes

The database subsystem consists of VAX-based server processes that manage and execute database transactions on behalf of clients. Processes are accessed from a set of services provided by the control system. Some services are accessible by clients and others are implemented exclusively for system programmers. Each service layer describes a well defined protocol for requesting device modifications, report generation, and management.

The *central database server* is responsible for processing transactions that require device modification, searching for device information, and synchronizing OA derivation and distribution. The server manages requests to modify devices and logs them to a record-oriented file for tracking. Requests are also placed into a Device Modifications Queue (DMQ) that is used to derive the LAL database. The OA database is automatically derived and distributed on a periodic basis, superseding the LAL database. After the OA derivation, the DMQ is reset, indicating that previous versions of the LAL have been integrated into the OA. The server implements a write-through cache that provides efficient access for locating and reading devices. As device modifications occur, cache updates are synchronized with the central database for consistency. When clients need device modifications to be propagated, the server spawns the Source to OA (SOA) subprocess to derive the LAL database and then synchronizes the distribution of the LAL with the distributor server process.

The *distributor server* is responsible for managing OA and LAL database distributions and monitoring database consistency throughout the control system. The server opens both OA and LAL databases, maps them into its virtual address space, and compares the serial numbers. Periodically, the distributor server connects to OA database server processes and checks the OA and LAL database serial numbers for consistency. If a discrepancy exists, the server restarts the remote server in error. If the error condition exists for a prolonged number of connects, the distributor server automatically distributes the appropriate OA databases to the destination. Primarily, OA database distributions are requested by the SOA subprocess. However, they can be forced by authorized clients.

The *OA database server* is responsible for servicing client requests for device information. The server initializes by mapping the OA databases as virtual memory and comparing serial numbers between section files. The offsets in the device section are used to locate device properties and attributes, accessing them as consecutive memory locations. Clients request properties and attributes by using interface services. The requests are processed upon receipt and return the desired information.

The report-generating functions are implemented by the *report server*. The server uses a memory cache to store a subset of device information that can be retrieved by a client. The report system services are used to build a complex set of search criteria and specify which properties and attributes to return. The server linearly scans the cache, compares each device with the criteria, and returns the specified data to the client.

All these server processes collect and maintain statistical and usage data that can be read by authorized clients. The data can be analyzed to evaluate usage patterns and performance and to identify bottlenecks.

3.4 Database Applications

Several applications have been written by system programmers to provide clients with simple interfaces to the database subsystem. The applications were written using the standard Epicure interfaces to communicate database requests to the appropriate server processes. With these applications, clients can modify devices, generate reports, derive OA and LAL databases, and manage the subsystem.

The *DBedit* application is used by clients to view device properties and attributes and to perform database transactions such as device adds, modifies, and deletes. The application uses DECforms[3] to display the device properties and attributes. The central database services are used to retrieve and modify device information and request LAL derivations. The derivation process can be monitored by a status screen that will indicate the completion of the distribution.

The *DBhist* application reads the device-modification tracking database and displays a detailed description of which devices, properties, and attributes have been modified, as well as when and by whom each modification was requested. Clients can tailor the search by specifying different time windows, properties, and attributes. Device modification history can be displayed on the screen and/or written to a file.

The *DBbuild* application displays detailed information about OA database derivations using information recorded by the central database server. Clients can specify time frames, OA database types, or users for a given search. The information collected can be displayed on the screen and/or written to a file.

The *DBrpt* application provides device-reporting functions to clients. Various pull-down and pop-up menus are provided to specify the search and sort criteria and the properties and attributes to return. The return data is written to a file and displayed as an editing session. Clients can save in files selected menu options and recall them in subsequent sessions. In addition, the application can be executed in a batch environment, reading selected option files and generat-

[3]Trademark of Digital Equipment Corporation.

Operation	Time
Central database device add	350 ms
Central database device modify	150 ms
Central database device delete	300 ms
Central database device report	50 ms
OA compression & distribution	20 min
LAL compression & distribution	1.5 min
OA/LAL distribution failure rate	0%
Average OA derivations	1/day
Average LAL derivations	4/day
Average time to retrieve all devices from the report server	325 ms

Table 1: Database Timings

ing reports.

The *Epicure Distributed Database System Status* (EDDS) application provides functions to manage the subsystem. Pull-down and pop-up menus are used to monitor and maintain all server processes within the subsystem. Various display screens provide statistical information about individual servers and overall system utilization.

3.5 Database Statistical Information

Statistical information is maintained in each of the server processes and can be retrieved and viewed from the EDDS application. Table 1 contains average timings for various database operations monitored over a six-month period.

4 Data Acquisition

On the Epicure control system, applications call data acquisition and setting services, which provide a generic means of reading and setting beamline devices. The application specifies a reading or setting property and a sample time or rate. The data size and the offset into the data array may be specified but are optional. Depending on the requested device and property, the returned data may be a single reading, a state bit mask, an array of readings, or a structure of information.

The sample time of a data acquisition request is specified by a *Frequency Time Descriptor* (FTD). An FTD contains three fields: a type, a clock event number, and a number of milliseconds. The type field indicates a periodic rate, a clock event plus a delay, or a delay starting now. A reading may be executed once or repeatedly. A setting is executed only once. Most data collection is done at intervals of two seconds or greater. The highest data rate the system supports is 30 Hz (33 ms).

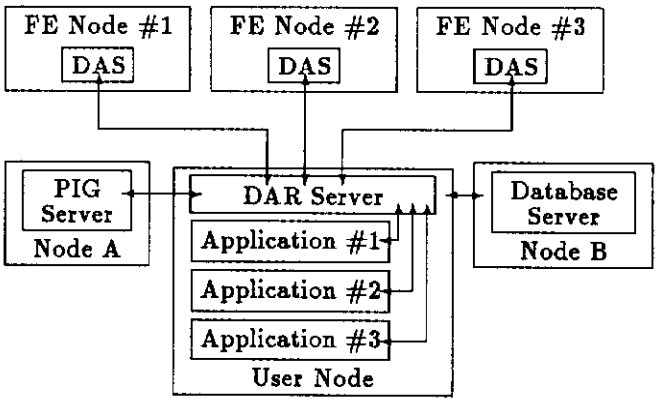

Figure 1: DAR and Connected Subsystems

The data acquisition functions of the Epicure control system are handled by server processes distributed across numerous VAX systems. These servers communicate via DECnet, perform database lookups of hardware addressing information, make application requests for device readings or settings, verify user access to protected devices, and communicate with low-level data acquisition processes. Logical definitions are used to determine the node or list of nodes to check for a server. Load sharing is achieved by changing the order of nodes in a list defined by these logical names from VAX to VAX. When a server connection is broken, these logical names are also searched in order to attempt a fail-over connection to another server.

Device reading and setting requests and data readbacks are routed through a *Data Acquisition Requester* (DAR) server. The main purpose of DAR is to provide an interface for applications to collect data and to control beamline, security system, and cryogenic devices. This interface consists of various *User Task Interface* (UTI) library routines callable from C or FORTRAN. The UTI provides routines to read or write data for devices defined in the Epicure database.

DAR is connected to various servers, each performing a different function. These subsystems may be distributed across several nodes or reside on one node. Figure 1 shows the connections between DAR and these other subsystems.

Each subsystem in Figure 1 has a particular function. The application is the end user of the data acquisition system. A DAR server must be running on the same node for an application to use DAR UTI routines. All applications on a node use the DAR server executing on that node. The maximum number of applications simultaneously interacting with the DAR is limited only by system resources. An application calls UTI routines to make data reading or setting requests to the DAR server.

The *Protection Interface and Grants* (PIG) server supplies information for DAR to determine whether the particular user of the application has the privileges to read or write the requested device and property. The database server supplies the DAR server with the hardware-related information necessary to make data acquisition or setting requests. The database also supplies DAR with information to interpret or scale the data DAR forwards to the application.

The DAR server combines and forwards data requests on behalf of the applications to the *Data Acquisition Server* (DAS). A DAS executes on each front-end (FE) VAX node and serves as the intermediary between a DAR and the processes executing in the VME modules. One VME module is a timer that performs the necessary synchronization. The acquisition VME module executes the CAMAC commands to perform the device read and write operations. Communication between DAS and the VME modules, and also between the VME modules, occurs via queues residing in the VME shared memory. Data is returned from the acquisition module to DAS using another queue in shared memory. DAS returns the data to the requesting DAR via DECnet. The DAR server notifies the application when data has arrived via the UTI routines. If readings are to be repeatedly done, the data acquisition module returns the list to the timer module for rescheduling. A repetitive request list continues to cycle until it is cancelled by the DAR server.

5 Epicure Applications

A number of applications and utilities are available to the Epicure user. Applications can be run on VT100-compatible terminals or workstations and are menu-based using the SMG run-time library procedures.

A user executes an Epicure application by selecting the corresponding entry from a displayed menu. The user may page forward or backward within a menu to see all the available options. A system-wide Epicure application menu is defined, but local menus may also be created and selected. Selecting a menu entry runs an executable image or initiates a command procedure or script. A menu entry may also be the name of another menu; thus, the user can have multiple menus and easily switch between them.

The *Epicure Screen Management* (ESM) software allows the user interface of different applications to have a similar appearance and operation. ESM provides menu bars, pop-up menus, dialogue boxes, and field maps. User Task Interface (UTI) routines provide control system services for database accesses, data acquisition, timing, and interprocess communication. By calling the ESM and UTI routines, users of the control system can develop software to collect and display Epicure data to satisfy their specific requirements.

The *Parameter PAGE* application allows the user to display analog readings and digital status readbacks, change settings, and set control bit flags for devices. The PAGE screen consists of a menu bar usable via a mouse or keyboard and 22 rows of information. Each row contains device name, setting value, analog readback, units of the setting and readback values, text strings for digital status readings, and an overlayable field. The overlay may display either the analog reading rate or the device's text description from the database. To begin reading a device, the user enters a device name in the name field. The setting of a device may be modified by typing a new value into the setting field. Control bit flags may be sent to the device by selecting the desired status bit text. For example, the user may select in the status column the TRP text to reset a device or the NEG or POS text to change the polarity. The menu bar allows the user to (1) write the displayed set of devices to a disk file, (2) view a list of such page files and select one for display, (3) change the reading display units between raw data, intermediate, and engineering units, (4) print the screen, and (5) switch the overlay field between descriptive text and reading-rate parameters. Most menu functions are also available through specific keys.

The *Fast Time Plot* application provides for the plotting of one to four devices on the same axes. The data value is plotted on the Y-axis, for which the user must specify the expected minimum and maximum values. The X-axis is a time interval within a single accelerator cycle. Both the start time and the stop time are specified in seconds relative to the start of the cycle or to some beam or accelerator clock event within the cycle. The data collection rate is specified in seconds between data points with a minimum value of 0.033333 seconds (30 Hz). The data display is erased at the start of an accelerator cycle, with the number of cycles between erasures specified by the user. All the parameters relative to a single plot may be written to a file for later recall.

Acknowledgements

The Epicure control system was developed by the Controls Group of the Electrical/Electronics Department in the Research Division of Fermilab. Individuals who directed the efforts of this group during successive periods were Al Thomas, Terry Lahey, and Therese Watts. Other people involved in the specification, design, implementation, and testing of the hardware and software components of the system were: D. Baddorf, C. Chopp, K. Dabous, E. Dambik, R. Demaat, J. DeVoy, T. Fink, D. Kline, W. Knopf, M. Kozlovsky, B. Kramper, M. Larwill, L. Lee, M. Moy, F. Nagy, J. Schmidt, and R. West.