ANL/ASD/RP--87267

EPICS Overview

EPICS OVERVIEW

EPICS consists of a set of software components and tools with which Application Developers can create a control system. The basic components are:

OPI

Operator Interface. This is a UNIX based workstation which can run various EPICS tools.

IOC

Input Output Controller. This is VME/VXI based chassis containing a Motorolla 68xxx processor, various I/O modules, and VME modules that provide access to other I/O buses such as GPIB.

LAN

Local area network. This is the communication network which allows the IOCs and OPIs to communicate. EPICS provides a software component, Channel Access, which provides network transparent communication between a Channel Access client and an arbitrary number of Channel Access servers.

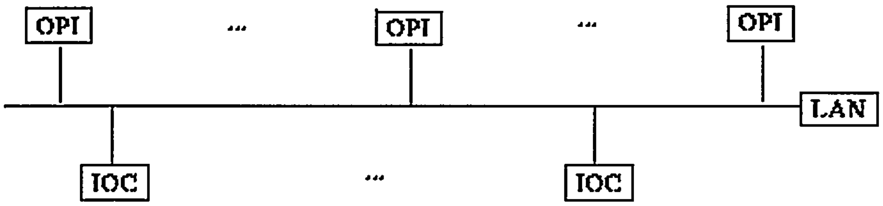

Figure 1 : EPICS based Control System

Figure 1 shows the basic physical structure of a control system implemented via EPICS. The rest of this chapter gives a brief description of EPICS:

Basic Goals

A few basic design goals on which EPICS was developed.

Platforms

The Hardware and Software platforms EPICS supports.

IOC

Software EPICS supplied IOC software components.

Channel Access

EPICS software that supports network independent access to IOC databases.

OPI Tools

EPICS supplied OPI based tools.

EPICS Core

A list of the EPICS core software, i.e. the software components without which EPICS will not work.

Basic Attributes of EPICS

Tool Based:

EPICS provides a number of tools for creating a control system. This minimizes the need for custom coding and helps ensure uniform operator interfaces.

Distributed:

An arbitrary number of IOCs and OPIs can be supported. As long as the network is not saturated, no single bottle neck is present. A distributed system scales nicely. If a single IOC becomes saturated, it's functions can be spread over several IOCs. Rather than running all applications on a single host, the applications can be spread over many OPIs.

Event Driven:

The EPICS software components are all designed to be event driven to the maximum extent possible. For example rather than having to poll IOCs for changes, a channel access client can request that it be notified only when changes occur. This design leads to efficient use of resources as well as to quick response times.

High Performance:

A SPARC based workstation can handle several thousand screen updates a second with each update resulting from a channel access event. A 68040 IOC can process more than 6,000 records per second including generation of any channel access events.

Hardware – Software Platforms

OPI

Hardware

Unix based Workstation. Currently Sun4s. Hope to support HP RISC workstation in near future.

Software

UNIX
X Windows
Motif Toolkit

LAN

Hardware

Ethernet now. FDDI in the future.

Software

TCP/IP protocols via sockets.

IOC

Hardware

VME/VXI bus and crates.
Motorola 68020 and 68040
Various VME modules (ADCs, DAC, Binary I/O, etc.)
Allen Bradley Scanner (Most AB I/O modules)
GPIB devices
BITBUS devices

Software

vxWorks operating system.

Real time kernel
Extensive "Unix like" libraries

IOC Software Components

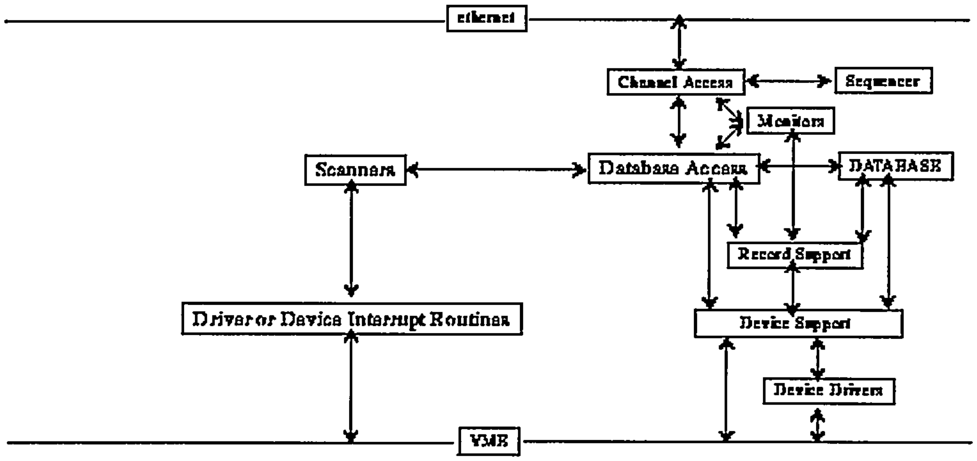

Figure 2, above, contains an overview of the IOC software components and their interactions.

- DATABASE:
  The memory resident database plus associated data structures.
- Database Access:
  Database access routines. With the exception of record and device support, all access to the database is via the database access routines.
- Scanners:
  The mechanism for deciding when records should be processed.
- Record Support:

Each record type has an associated set of record support routines.

- Device Support:
  Each record type has one or more sets of device support routines.
- Device Drivers:
  Device drivers access external devices. A driver may have an associated driver interrupt routine.
- Channel Access:
  The interface between the external world and the IOC. It provides a network independent interface to database access.
- Monitors:
  Database monitors are invoked when database field values change.
- Sequencer
  A finite state machine.

Lets briefly describe the major components of the IOC and how they interact.

DATABASE

The heart of an IOC is a memory resident database together with various memory resident structures describing the contents of the database. EPICS supports a large and extensible set of record types, e.g. ai (Analog Input), ao (Analog Output), etc.

Each record type has a fixed set of fields. Some fields are common to all record types and others are specific to particular record types. Every record has a record name and every field has a field name. The first field of every database record holds the record name, which must be unique across all IOCs attached to the same TCP/IP subnet.

A number of data structures are provided so that the database can be accessed efficiently. Most software components, because they access the database via database access routines, do not need to be aware of these structures.

Database Access

With the exception of record and device support, all access to the database is via the channel or database access routines. See Chapter for details.

Database Scanning

Database scanning is the mechanism for deciding when to process a record. Four types of scanning are possible: Periodic, Event, I/O Event, and Passive.

- Periodic:
  A request can be made to process a record periodically. A number of time intervals are supported.
- Event:
  Event scanning is based on the posting of an event by any IOC software component. The actual subroutine call is: post_event(event_num)
- I/O Event:
  The I/O event scanning system processes records based on external interrupts. An IOC device driver interrupt routine must be available to accept the external interrupts.
- Passive:
  Passive records are processed as a result of linked records being processed or as a result of external changes such as channel access puts.

Record Support, Device Support, and Device Drivers

In order to remove record specific knowledge from database access, each record type has an associated record support module. Similarly, in order to remove device specific knowledge from record support, each record type can have a set of device support modules. If the method of accessing hardware is complicated, a device driver can be provided to shield the device support modules. Many record types, in particular all types not associated with hardware, do not have device support or drivers.

The IOC software is designed so that the database access layer knows nothing about the record support layer other than how to call it. The record support layer in turn knows nothing about it's device support layer other than how to call it. Similarly the only thing a device support layer knows about it's associated driver is how to call it. This design allows a particular installation and even a particular IOC within an installation to choose the set of record types, device types, and drivers it wishes to use. The remainder of the IOC system software is unaffected.

Because an Application Developer can develop his own record support, device support, and device drivers. Every record support module must provide a record processing routine. It is this routine that is called by the database scanners. Record processing consists of some combination of the following functions (particular records types may not need all functions):

- Input
  Read inputs. Inputs can be obtained, via device support routines, from hardware, from other database records via database links, or from other IOCs via channel access links.
- Conversion
  Conversion of raw input to engineering units or of engineering units to raw output values.
- Output
  Write outputs. Output can be directed, via device support routines, to hardware, to other database records via database links, or to other IOCs via channel access links.
- Raise Alarms
  Check for and raise alarms.
- Monitor
  Trigger monitors related to channel access callbacks.
- Link
  Trigger processing of linked records.

Channel Access

Channel access is discussed in the next section.

Database Monitors

The routines described in this section provide a callback mechanism for database value changes. This allows the caller to be notified when database values change without constantly polling the database. A mask can be set to specify value changes, alarm state changes, and/or archive changes.

At the present time only channel access uses database monitors. No other software should use the database monitors. Because they are of interest only to channel access, the monitor routines will not be described.

Channel Access

Channel access provides network transparent access to IOC databases. It is based on a client server model. Each IOC provides a channel access server which is willing to establish communication with an arbitrary number of clients. Channel access client services are available on both OPIs and IOCs. A client can communicate with an arbitrary number of servers.

Client Services

The basic channel access client services are:

- Search
  Locate the IOCs containing selected process variables and establish communication with each one.
- Get
  Get value plus additional optional information for a selected set of process variables.
- Put
  Change the values of selected process variables.
- Add Event
  Add change of state callback. This is a request to have the server send information only when the associated process variable changes state. Any combination of the following state changes can be requested: change of value, change of alarm status and/or severity, and change of archive value. Many record types provide hysteresis factors for value changes. In addition to process variable values, get and add event requests can also request any combination of the following additional information:
- Status
  Alarm status and severity.
- Units
  Engineering units for this process variable.
- Precision
  Precision with which to display floating point numbers.
- Time
  Time when the record was last processed.
- Enumerated
  A set of ascii strings defining the meaning of enumerated values.
- Graphics
  High and low limits for producing graphs.
- Control
  High and low control limits.
- Alarm
  The alarm HIHI, HIGH, LOW, and LOLO values for the process variable.

It should be noted that channel access does not provide access to database records as records. This is a deliberate design decision. This allows new record types to be added without impacting any software that accesses the database via channel access. A channel access client can communicate with multiple IOCs having differing sets of record types.

Search Server

Channel access provides a server which waits for channel access search messages. These are generated when a channel access client (for example when an Operator Interface task starts) searches for the IOCs containing process variables the client uses. This server accepts all search messages, checks if any of the process variables are located in this IOC, and, if any are found, replies to the sender.

Connection Request Server

For each IOC containing process variables it uses, the channel access client issues connection requests. The connection request server accepts the request and establishes a connection to the client. Each such connection is managed by a separate task (actually two tasks). Ca_get and ca_put requests map to dbGetField and dbPutField database access requests. Ca_add_event requests result in database monitors being established. Database access and/or record support routines trigger the monitors via a call to db_post_event.

Connection Management

Channel access provides a connection management service. When a channel access server fails (e.g. it's IOC crashes) the client is notified. and when a client fails (e.g. it's task crashes) the server is notified. When a client fails, the server breaks the connection. When a server crashes, the client automatically re–establishes communication when the server restarts.

OPI Tools

EPICS provides a number of OPI based tools. These can be divided into two groups based on whether or not they use channel access. Channel access tools are real time tools, i.e. they are used to monitor and control IOCs.

Channel Access Tools

- MEDM
  Motif version of combined display manager and display editor.
- DM
  Display Manager. This tool reads one or more display list files created by EDD, establishes communication with all necessary IOCs, establishes monitors on process variables, accepts operator control requests, and updates the display to reflect all changes.
- ALH
  Alarm Handler. This is a general purpose alarm handler driven by an alarm configuration file.
- AR
  Archiver. This is a general purpose tool to acquire and save data from IOCs.
- Sequencer
  A tool which runs in an IOC and emulates a finite state machine.

Other OPI Tools

- DCT
  Database Configuration Tool. This tool is used to create a run time database for an IOC
- EDD
  Display Editor. This tool is used to create a display list file for the Display Manager. A display list file contains a list of static, monitor, and control elements. Each monitor and control element has an associated process variable.
- SNC
  State Notation Compiler. It generates a C program that represents the states for the IOC Sequencer tool.
- Build Tools
  Tools are available to create the various database components from ASCII definition files.
- Source/Release
  EPICS provides a Source/Release mechanism for managing EPICS.

EPICS Core Software

EPICS consists of a set of core software and a set of optional components. The core software, i.e. the components of EPICS without which EPICS would not function, are:

- Channel Access – Client and Server software
- DATABASE
- Scanners
- Monitors
- DCT
- Build Tools
- Source/Release

All other software components are optional. Of course any application developer would be crazy to ignore tools such as MEDM (or EDD/DM). Likewise an application developer would not start from scratch developing record and device support. Most OPI tools do not, however, have to be used. Likewise any given record support module, device support module, or driver could be deleted from a particular IOC and EPICS will still function.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.