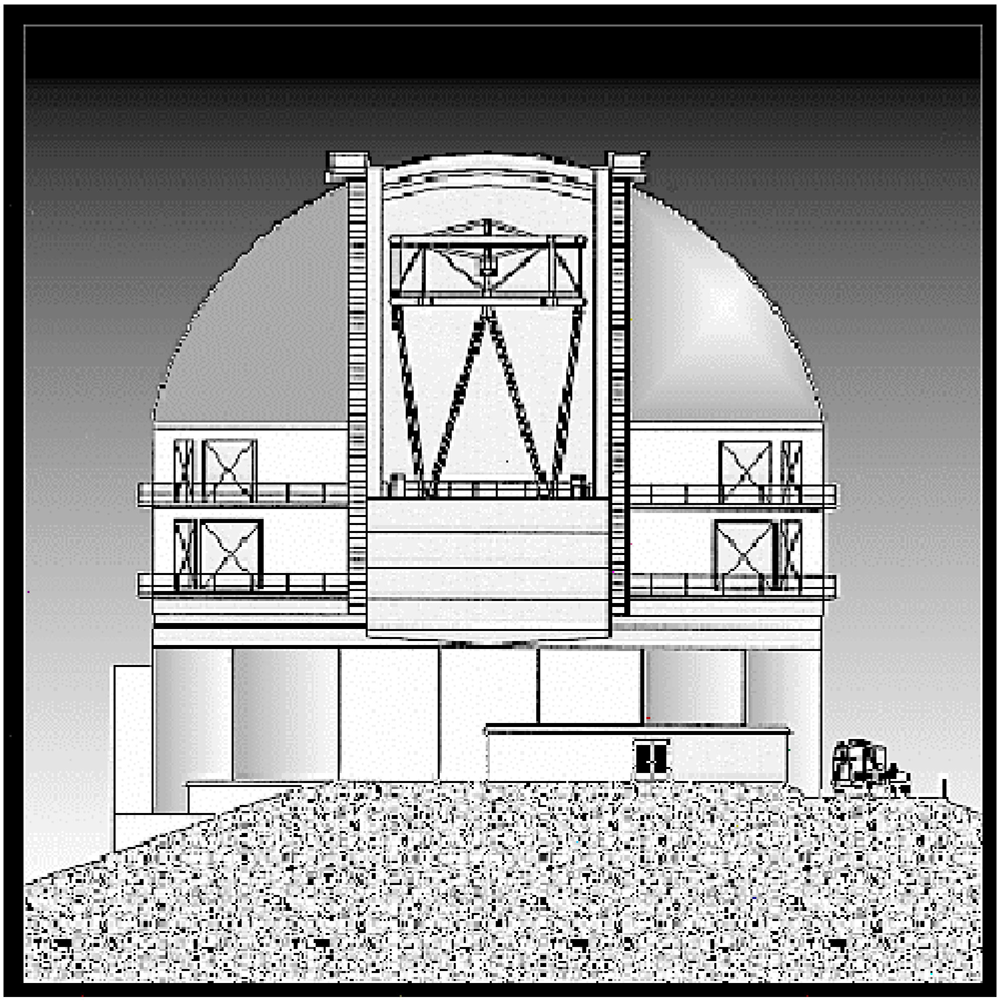
GEMINI
8-M Telescopes
Project

# RPT-C-G0012

# Evaluation of Control System Development Tools



**P.M. McGehee**
Controls Group

August 26, 1993

GEMINI PROJECT OFFICE 950 N. Cherry Ave. Tucson, Arizona 85719
Phone: (520) 318-8545 Fax: (520) 318-8590

1. Introduction 1
I .1. Purpose 1
1.2. Overview of Products 1
1.3. Method 1
1.3.1. Installation and Overview 1
1.3.2. Running of Demo Applications 2
1.3.3. Creation of Simple Application 2
1.4. Products Tested 2
2. DATA VIEWS 3
2.1. Installation And Overview 3
2.1.1. Data Types 3
2.1.2. Routine Types 3
2.2. Running of Demo Applications 4
2.3. Building A Simple Application 4
2.4. Dataviews: What Next? 4
3. RTAP 5
3.1. Installation And Overview 5
3.2. Running of Demo Applications 5
3.3. Building A Simple Application 5
3.3.1. Schematic: CCD_schem 5
3.3.2. Display Panel: CCD-main 5
3.3.3. Display Panel: CCD-accept 6
3.3.4. CCD Status Report 6
3.3.5. Database: 6
3.4. RTAP: Files 7
3.4.1. General RTAP Files 7
3.4.2. Application Environment Files .7
3.4.3. Application ASCII Database Files 7
3.4.4. Application Snapshot Database Files 8
3.4.5. Application Description Files 8
3.5. RTAP: What Next? 8
4. EPlCS 9
4.1. What Is EPICS? 9
4.2. Product Status And Contacts 11
4.2.1. Original Source 11
4.2.2. Industrial Partners 11
4.2.3. Discussion with partners 12
4.2.4. Enhancements 13
4.3. Documentation 14
4.4. Tools 15
4.4.1. Host (UNIX) 15
4.4.2. Target (VxWorks) 15
4.5. Record Types 17
4.6. EPICS Structure 18
4.7. EPICS Drivers 19

4.8. CCD Controller 20
4.8.1. Configuration 20
4.8.2. Required EPICS work 20
4.8.3. Applications Setup 20
4.8.4. Database 20
4.8.5. OPI Screens 20
4.8.6. SNL Program 21
4.9. Encoder Test Station ...................... 21
4.9.1. Environment 21
5. Conclusions 22
5.1. EPICS 22
5.2. Dataviews 22
5.3. RTAP 22
5.4. Recommendations 22
6. Appendix A: Ccd-Sim Database 23
7. Appendix B: Ccd-Sim Snl Program 24

# Disclaimer

This report may compare products from different vendors and may contain a specific decision to purchase which may be based on this comparison. The evaluation contained in this report is related only to the specific needs and requirements of Gemini, and should not be construed to apply to the needs and requirements related to any other application.

The contents of this report should in no way be viewed either as an endorsement by Gemini for a particular vendor's product or as Gemini's opinion that other vendor's products are unsuitable for any application.

Gemini makes this report available in the spirit of keeping its user community informed of the background behind its decisions. It urges this community to make decisions based on its own requirements.

# 1. INTRODUCTION

## 1.1. Purpose

The Controls Group of the Gemini project is evaluating three different products that have applications for the development of control systems and their associated user interfaces, databases, and drivers.

The products under consideration are:

- V.I. Corporation's Data Views.
- HP-Calgary's RTAP (Real Time Applications Platform).
- EPICS (Experimental Physics and Industrial Control System) originally codeveloped at the Argonne and Los Alamos National Laboratories, now marketed by a joint venture of Kinetic Systems and Titan.

## 1.2. Overview of Products

| **Product:** | **DataViews** | **RTAP** | **EPICS** |
|---|---|---|---|
| **Interface(Library):** | Yes | Yes | Yes |
| **Interface (Menu):** | Yes (Prototyping Only) | Yes | Yes |
| **Database:** | No | Yes | Yes |
| **VxWorks interface:** | No | No | Yes |
| **Recommended By:** | CFHT | ESO | ESO |
| **Used By:** | JPL: Voyager 11 VNESSA<br>Lockheed: HST Telemetry | ESO: VLT | ANL: APS<br>LANL: GTA<br>SSC |
| **Cost:** | $12,400 for first license<br>$2,100 for each additional | $31,000 to<br>$47,000 depending on database size. | $8,500 for first license<br>see 4.2.3. |

## 1.3. Method

The method used for product evaluation consisted of three distinct phases: installation & overview, running of demo applications, and creating a simple application.

The principal considerations associated with each stage are outlined in the following sections.

### 1.3.1. Installation And Overview

Ease of installation, general sense as to what the product can be used for.

### 1.3.2. Running of Demo Applications

Use of available features, conformance to documentation.

### 1.3.3. Creation of Simple Application

Ease of product use for own applications.

## 1.4. Products Tested

- DV-Draw and DV-Tools running on a SUN SPARC 2 under SunOS 4.1.2 and OpenWindows Version 3.
- RTAP/PLUS Version A.05.01 Demo System running on a HP 9000/710 under HP-UX. [S/N 72323D51000101]
- EPICS running on a SUN SPARC 2 under SunOS TBD, OpenWindows Version 3, with VxWorks release 5.0.2b running on the workstation and a MVME147 VME crate controller.

# 2. DATAVIEWS

## 2.1. Installation And Overview

Installation of the DataViews products (DV-Draw, DV-Tools, tutorials, and sample programs) was accomplished in the standard manner by exporting the icarus:/scr directory to gemini and using the 1/4-inch tape on that machine.

The DV-Tools library of user interface design routines is quite complex and is structured according to the vendor's own model of how the elements of a GUI interact.

There are a set of routines that deal with the concept of a data source variable which is a variable bound to a file or to the output of a process. There is no formal concept of a database or of DV-Tools sponsored sequencing.

The DV-Draw package gives the user the ability to create prototype GUls without programming. But the prototype can not be bound to real applications data - the full-blown environment must be described by the developer's C code.

### 2.1.1. Data Types

Private Structures:

| | | |
|---|---|---|
| axis descriptor | display formatter | data group |
| data source list | drawport | data source |
| data source variable | event request | hash table node |
| hash table | interaction handler | prototype environment |
| symbol tablenode | symbol table | text array |
| variable descriptor | view | |

Graphical Objects:

| | | |
|---|---|---|
| arc | circle | data group |
| ellipse | icon | image |
| input | line | polygon |
| rectangle | subdrawing | text |
| vector text | | |

Non-Graphical Objects:

| | | |
|---|---|---|
| color | deque | drawing |
| dynamic control | edge | input technique |
| location | node | pixmap |
| point | rule | screen |
| slotkey | thresholdtable | variable descriptor |
| transform | | |

2.1.2. Routine Types

| **Routine Prefix** | **Function** |
|---|---|
| T routines | Screens, Drawports, and Views |
| VO routines | Objects |
| VU routines | Event Handlers |
| VN routines | Interaction Handlers |
| VD routines | Display Formatters |
| VG routines | Graph Formatters |
| VP routines | Plot Formatters |
| VT routines | Hash Table and Symbols |
| GR routines | Graphics |

## 2.2. Running of Demo Applications

For the most part the demo applications and tutorials ran smoothly. A few comments are in order:

- Objects and actions are activated by a single mouse button click. This is contrary what many other GUIs use which is single mouse button click to select, double mouse button click to activate.
- Many of the demos ran slowly. The afflicted demos either used large bitmapped graphics or had simulation processes running. For both types the SUN's CPU performance monitor maxed out at 100.
- The DV-Draw menu driven GUI builder had a fairly steep learning curve and would have been difficult to use without full documentation

## 2.3. Building A Simple Application

The tutorials and demo applications were very comprehensive and included analysis of sample user software thus giving an adequate description of the product's capabilities. To realistically model a typical Gemini application (like a CCD controller) would have taken significant programming and was not done for this test. Several sample screens were constructed using the prototyping features of DV-Draw.

## 2.4. Dataviews: What Next?

The DV-Tools and DV-Draw packages are very useful for the creation of more complete tools like RTAP, EPICS, or CFHT's PEGASUS. We would favor a system that based itself on the concept of a run-time distributed database. Although DataViews has implemented the concept of a data source variable it has not expanded it to the extent of RTAP or EPICS.

# 3. RTAP

## 3.1. Installation And Overview

Installation of the RTAP system comprised several stages due to the software and hardware coming from separate locations.

The software arrived from Calgary in the form of a 60mm DAT tape along with documentation describing the installation and operation of the demo applications. Formal documentation was not
supplied.

After several lengthy discussions with the Phoenix sales office a demo HP 710 was delivered to the Gemini office although without the required DAT tape drive. A loaner tape drive was located and shipped within a week.

Initial product installation was straightforward although the demo did not function correctly even after tuning of operating system kernel parameters. After consultation with the Calgary staff it was decided to reload the entire HP-UX system.

During the system rebuild from CD-ROM it was noticed that the ethernet port was not terminated. After it was properly terminated the RTAP demo system functioned correctly.

## 3.2. Running of Demo Applications

The demos performed as described with the following exceptions to the supplied documentation:

- A few menu names had been changed.
- A control panel testing mode did not exist - it had been phased out in the current demo release.

## 3.3. Building A Simple Application

For our test it was decided to build a CCD controller interface. This consisted of a top-level schematic, two dialog areas, a report generator, and corresponding database entries.

### 3.3.1. Schematic: CCD Schem

A box labeled Cassegrain Focus:Instrumentation which contains a smaller box labeled CCD that triggers the display panel CCD-main.

### 3.3.2. Display Panel: CCD-Main

A group of radio buttons collectively labeled Exposure and individually labeled bias, dark, flat, object, comp, and calibration.

A text edit object labeled Time (sec):
A text edit object labeled Iterations:
A text edit object labeled Sequence will be:
A text edit object labeled Filename:.
A group of radio button collectively labeled Median data option: and individually labeled bone, and normal.

A text edit object labeled Object:.
A text edit object labeled Observer:.

A page edit object labeled Comments:.

An action button labeled Accept that triggers the display panel CCD-accept.
An action button labeled Save Values that triggers the CCD report.
An action button labeled Defaults that does nothing.
An action button labeled Cancel that closes the display panel CCD-main.
An action button labeled Help that does nothing.

3.3.3. Display Panel: CCD-Accept

A text edit object labeled Filename:.

A read-only text edit object labeled Sequence Number:.

A group of read-only radio buttons collectively labeled CCD status: and individually labeled Expose and Readout.

A read-only text edit object labeled Remaining Time:.

An action button labeled Stop that closes the display panel CCD-accept.
An action button labeled Abort that does nothing.
An action button labeled Hold that does nothing.
An action button labeled Break that does nothing.

3.3.4. CCD Status Report

The report types out the current state of all the CCD database parameters into the terminal window. See the next section for a description of the database entries.

3.3.5. Database:

The main branch of the database is labeled CCD Setup and describes three objects which are labeled exposure, observer, and source. Each object is described by a list of attributes with each object.attribute associated with a unique data structure.

**Object:** exposure

| **Attributes:** | **Name** | **Type** | **Meaning** |
|---|---|---|---|
| | .time | rtINT16 | Total exposure time in seconds. |
| | .remaining | rtlNTl6 | Exposure time left in seconds. |
| | .median_control | rtLOGICAL | Median control (NONE/NORMAL). |
| | .type | rtINT16 | Exposure type (bias,dark,...) |
| | .sequence | rtBYTES20 | Sequence ID string. |
| | .iterations | rtINT 16 | Number of iterations. |
| | .filename | rtBYTES32 | Disk filename string. |
| | .hardware_status | rtLOGICAL | CCD status (EXPOSE/READOUT). |
| | .hold | rtLOGICAL | Exposure on hold (YES/NO). |

**Object**: observer

| **Attributes:** | **Name** | **Type** | **Meaning** |
|---|---|---|---|
| | .name | rtBYTES128 | Observer name string. |

**Object:** source

| **Attributes:** | **Name** | **Type** | **Meaning** |
|---|---|---|---|
| | .name | rtBYTES128 | Sourcename string. |
| | .comment | rtBYTES256 | Comment string. |

### 3.4. RTAP: Files

The RTAP environment for each application is defined by a number of files. These files are described below by their general and specific functions as well as their locations.

#### 3.4.1. General RTAP Files

| **Location** | **Function** |
|---|---|
| rtap/etc/RtapEnvList | Defines known demo environments. |
| rtap/bin/start_CCD | Defines logical names, starts RTAP. |

#### 3.4.2. Application Environment Files

| **Location** | **Function** |
|---|---|
| rtap/CCD/RtapEnvTable | Defines RTAP environment - which processes run and how they are invoked. |

#### 3.4.3. Application ASCII Database Files

| **Location** | **Function** |
|---|---|
| rtap/CCD/config/CCD_Setup/exposure/exposure | Defines exposure database object attributes. |
| rtap/CCD/config/CCD_Setup/observer/observer | |

| | |
|---|---|
| | Defines observer database object attributes. |
| rtap/CCD/config/CCD_Setup/source/source | |
| | Defines source database object attributes. |
| rtap/CCD/config/support_data/... | Object definition files for general RTAP system unloaded from water environment. |

3.4.4. Application Snapshot Database Files

| **Location** | **Function** |
|---|---|
| rtap/CCD/RtapDiskDB | Copy of disk database. |
| rtap/CCD/RtapHdrSnapO | Unknown. |
| rtap/CCD/RtapRamSnapO | Copies of RAM database. |
| rtap/CCD/RtapRamCtrl | Unknown. |
| rtap/CCD/rtap_log | Log file. |

3.4.5. Application Description Files

| **Location** | **Function** |
|---|---|
| rtap/CCD/panels/CCD-accept | Accept and start exposure dialog. |
| rtap/CCD/panels/CCD-main | Main CCD control dialog. |
| rtap/CCD/reports/CCD | CCD status report format. |
| rtap/CCD/schematics/CCD_schem | Top-level schematic. |

### 3.5. RTAP: What Next?

For a complete appraisal of the usefulness of the RTAP system the following features would need
to be explored:

- Use of the Simulator and Sequencer to mimic telescope and instrument behavior.
- Creation of User Written Procedures.
- Implementation of Alarms.
- Use of Trends to plot system behavior.
- Use of Distributed database features for remote updates.

Unfortunately, given the limited time that the HP hardware was available and the state of the supplied documentation we were not able to fully exercise the RTAP system. We can, however, state that RTAP may be the system of choice for the host system. Information that could change this are:

- Costing (if we are required to equip all development sites), and
- The results of the upcoming EPICS evaluation.
- Fully supported port to SunOS (Solaris).

# 4. EPICS

## 4.1. What Is EPICS?

EPICS stands for Experimental Physics and Industrial Control System. The original development was done by the following U.S. National Laboratories from ~1985 to present for use in particle accelerator control systems:

The Controls and Automation Group (AT-8)
Ground Test Accelerator
Accelerator Technology Division
Los Alamos National Laboratory

and

The Controls and Computing Group
Accelerator Systems Division
Advanced Photon Source
Argonne National Laboratory.

As of March 1993 EPICS is currently in use at Los Alamos, Argonne, the Super Conducting Super Collider, Boeing, U.S. Department of Transportation, Duke University, and Lockheed.

Two industrial partners have been set up to transform EPICS into a commercial product. The first is Tate Integrated Systems in Baltimore, MD which specializes in process control systems, the second is a collaboration between Titan in Albuquerque, NM and Kinetic Systems in Denver, CO who target high-speed CAMAC and VXI applications. Final business plans should be drawn up by
April 1993.

EPICS is built on top of Wind River Systems' VxWorks real-time operating system and includes both host and target modules. The heart of EPICS is a real-time distributed database system that resides on the system VME crate controllers that can be accessed via a TCP/IP-based channel access library.

| | |
|---|---|
| Supported hosts: | SunOS |
| Supported targets: | VxWorks<br>(processor support is currently just for MC68xxx but the IOC core can be recompiled for any VxWorks target). |
| Supported channel access hosts: | SunOS, VxWorks, VAX/VMS |

Each field in the database has an associated record type that can be purely software, related to a specific hardware channel, or linked via input, output, or forward processing sequence to another database record. The current list of record types is quite lengthy but includes:

- analog input/output
- binary input/output

- calculation of arbitrary algebraic and logical expressions
- event generators
- fanout processing of other records
- histogram calculation
- multi-bit binary input/output
- client-server synchronization
- PID (Proportional, Integral, and Derivative) control algorithms
- selection of inputs based on index, having an extreme value or returning median value
- stepper motor control
- string input/output
- interface to non-EPICS subroutine package
- timer module control
- waveform input

The target or IOC (Input-Output Controller) runs the bulk of the EPICS system. Besides the database and its associated access routines the following tasks are present:

- event driven record processor
- clock driven record processors (10Hz, 5Hz, 2Hz, lHz, 0.5Hz, 0.2Hz, 0.1Hz)
- VME Input/Output event driven record processor
- device drivers
- device support routines that interface drivers to database record types
- sequencer code

The host or OPI (OPerator Interface platform) is the actual development area. Cross development is done with the GNU toolkit. Besides the standard VxWorks development environment the following EPICS tools are available on SunOS hosts:

- database configuration tool
- display editor
- display manager
- alarm handler
- data archiver
- data archive reviewer
- state notation language (sequencer code) compiler
- channel access clients for WINGZ, Mathematica, and PV-WAVE

### 4.2. Product Status And Contacts

#### 4.2.1. Original Source

Address: LANL
Controls and Automation Group (AT-8)
Accelerator Technology Division
MS H811
Los Alamos, NM 87545
Phone: (505) 667-3414
Contact: Bob Dalesio(dalesio@luke.atdiv.lanl.gov)

#### 4.2.2. Industrial Partners

There are two industrial partners which have been offered a "co-exclusive" contract meaning that although they are in competition with each other no other company can license the system.

The partners have different markets in mind and will enhance the original system independently. They will also produce upgraded documentation.

There will be sharing of upgrades between the industrial partners and the EPICS consortium which is comprised of Los Alamos, Argonne, and the Super Collider.

We have yet to choose the source from which we will obtain the EPICS system.

4.2.2.1 First Industrial Partner

One commercial version of EPICS will be produced by Tate Integrated Systems.

Address: Tate Integrated Systems
Baltimore, MD
Phone: (410) 581-0422
Contact: Day Stephens
Notes: Background: Process control, MVME147 based systems.
Planned EPICS improvements: redundancy, remote I/O, low cost platforms

4.2.2.2 Second Industrial Partner

The second version of EPICS will be jointly supported by Titan and Kinetic Systems.

Address: Titan
Albuquerque, NM
Phone: (505) 764-5337
Contact: Dave Fugelso
Notes: Background: Accelerators, data acquisition

Address: Kinetic Systems
Denver, CO
Phone: (303) 220-7455
Contact: Arun Sheth
Notes: Background: CAMAC
Planned EPICS improvements: very high speed data archival

4.2.3. Discussion With Partners

Given that the proposed Gemini control system may more closely resemble an experimental physics environment than an industrial process control system, communication was first established with the second EPICS industrial partner.

A conversation with Dave Fugelso of Titan on 2 March 1993 is summarized below.

1) Business plan will be defined within 2-3 weeks.
2) Product will most likely be called EPIX since Los Alamos is attached to original EPICS name.
3) Driver support for all of Kinetic System's line of CAMAC and VXI products along with some VME support.
4) DataViews version of upgrade for EDD/DM (EDitor for Displays/Display Manager).
S) The partners plan on targeting high speed applications - both in terms of data collection rates (transient digitizers, etc) and for fast control loops. They are informed about our servo rate requirements.
6) General product support will be defined by June (end of 2nd quarter 1993).
   Courses:
   - Introduction to EPIX.
   - Programming in EPIX (adjunct to VxWorks training)
   - Advanced User Topics in EPIX
   - EPIX Hardware Support
7) A standard maintenance contract will be defined.
8) The manuals will be rewritten especially the sections on installation and release control.
9) Infrastructure notes:
   Kinetic Systems will operate as sales representative and distributor, Titan will provide support. Market division and joint development schemes with the other industrial partner (Tate) is TBD.
10) Local Kinetic System representative is:
    Dave Norgord
    Phoenix
    (602) 820-4015
11) As of yet, Kinetic Systems sales reps have not been trained to sell EPIX. This will be changed by April along with release of initial product brochures.

12) Pricing structure: The prices listed here are for first copy only, additional copies will be 20% of listed cost. Separate pricing (if any) for government or university sites is under negotiation and will be known by April 1993. Pricing is being driven by LANL.

| | | |
|---|---|---|
| IOC core | Target software | $1000 |
| DCT | Database Configuration Tool | $500 |
| SNC | Sequencer | $500 |
| EDD/DM | Display Editor/Manager | $ 1000 |
| | Currently both LANL-X version and ANL-Motif version are shipped. | |
| ALH | Alarm Handler | $1000 |
| AR | Data Archives | $1000 |
| ARR | DataArchiver Review | $500 |
| KNOBS | GUI for Motor Control | $250 |
| PROBE | Motif based single channel debugger | $100 |
| WIN(ZZ client | | $ 100 |
| Mathematica client | | $ 100 |
| PV-Wave client | | $ 100 |
| Core System - everything for a complete site (1 host + 1 target) | | $8500 |

### 4.2.4. Enhancements

The following features are currently being added to EPICS:

- New graphical version of DCT (Database Configuration Tool) available by the end of
- February. It operates like a schematic capture program using the EDITH format which is a
- netlist standard.
- Motif version of EDD/DM (EDitor for Displays/Display Manager). This comes from the
- Argonne co-development site. Los Alamos comments are:
    - runs at half speed of Los Alamos version.
    - limited color map (standard set of ~40 colors).
    - not as flexible as Los Alamos version.
    - Los Alamos version has ability to create 3-D look & feel using available tool.

### 4.3. Documentation

This is a list of documentation available from the Los Alamos codeveloper. Most of it is in a draft
form. Enhanced versions of all system documentation will be available from the industrial partners.

| | | |
|---|---|---|
| 1. | Installation Notes<br>Source/Release Control<br>Bob Zieman and Marty Kramer<br>share/site/README. INSTALL | <br><br>1 Jan 1993 |
| 2. | Working in the EPICS Environment Tutorial Slides<br>Roger Cole | <br>3 Feb 1992 |
| 3. | Input Output Controller (IOC) Application Developer's Guide<br>Martin R. Kraimer | <br>14 Jun e 1991 |
| 4. | Channel Access Reference Manual<br>Jeff Hill | <br>16 Sept.1991 |
| 5. | Input Output Controller (IOC) Record Reference Manual<br>Janet B. Anderson / Martin R. Kraimer | <br>14 June 1991 |
| 6. | The Operator Interface for EPICS - Overview | 7 Jan 1992 |
| 7. | AR and ARR User Information Manual<br>Roger Cole | <br>10 Jan 1992 |
| 8. | State Notation Language and Run-time Sequencer Users Guide<br>Version 1.8<br>Andy Kozubal | <br><br>1 May 1992 |
| 9. | Alarm Configuration Tool System Requirements Definition<br>Janet Anderson | <br>2 June 1992 |

### 4.4. Tools

4.4.1. Host (Unix)

EPICS calls the UNIX Host an OPI (OPerator Interface machine). The user accessible programs include:

| | |
|---|---|
| ALH | Alarm Handler |
| AR | Data Archives |
| ARR | Data Archive and Real-Time Display |
| CAU | Channel Access Utility |
| DCT | Database Configuration Tools |
| DM | Display Manager |
| EDD | Display Editor |
| SNC | State Notation Compiler |

CAU provides for interactive access to the database via channel access.

DCT enables you to define everything about each record. A partial list is channel name, alarm limits, alarm monitors, links to other records, physical location within IOC system (in terms the device support routines can understand), deadbands, conversion between raw and engineering units Provision is made for non-hardware related records.

EDD is a graphical display editor which allows you to create the screens used by DM. EDD allows for interaction with the distributed database via channel access and includes changes in the
animation of objects based on channel values or alarm status.

SNC is a compiler for the EPICS state notational language. Full channel access is included along with the ability to define events, signals, and states. The inclusion of C language code is possible. SNC was implemented using the UNIX utilities lex and yaw and generates code for the VxWorks target.

4.4.2. Target (VxWorks)

EPICS calls the VxWorks target processor an IOC (Input-Output Controller or Crate). The user accessible tools include:

| | |
|---|---|
| CAU | Channel Access Utility |
| DBGF | Get value from a channel |
| DBIOR | Print IO_report for all drivers. |
| DBL | List database record names. |
| DBPF | Put value to a channel. |
| DBPR | Print a database record. |
| DBTGF | Test getting a value from a channel. |

| | |
|---|---|
| DBTPF | Test putting a value to a channel. |
| DBTR | Put a value to a channel, then process and print record. |
| LOPI | Local Operator Interface. |
| SEQ | Execute a Sequence Program. |

CAU is the target version of the host's CAU.
DB* routines are for single request accesses to the database.
SEQ runs a program created by SNC.

## 4.5. Record Types

The term 'world,' as used below, can mean one of the following sources of information:

- raw values from hardware,
- engineering values converted from raw hardware values,
- database link,
- channel access link, or
- purely software defined.

| Record Type | Name | Description |
|---|---|---|
| Ai | Analog Input | Read analog value from world. |
| Ao | Analog Output | Write analog value into world. |
| Bi | Binary Input | Read binary value from world. |
| Bo | Binary Output | Write binary value into world. |
| Calculation | Calculation | Calculate the value of an algebraic, elational, or logical expression based on up to 12 inputs and write into world. |
| Compression | Circular buffer | Useful for sending high-speed data to a slow host. |
| Fanout | Fanout | Trigger processing of up to 6 other records. |
| Histogram | Histogram. | Compute a histogram from world values and write back into the world. |
| Mbbi | Multi-bit Bi | 2**N states. |
| Mbbo | Multi-bit Bo | 2**N states. |
| Permissive | Client-Server | Synchronize requests. |
| PID | PID | Implements PID control algorithm. |
| Select | Input selection | Selection based on index, having min value having max value, or returning median value. |
| State | Store a state. | As an ASCII string. |
| Stepper controllers. | Stepper Motor | Support for various stepper motor |
| Stringin | String Input | Read an ASCII string from world. |
| Stringout | String Output | Write an ASCII string into world. |
| Subroutine | Call a subroutine | Easy way to incorporate non-EPICS target applications. |
| Timer | Timer modules | Support for various timer modules. |
| Waveform | Read an array | Read world data into an array. |

### 4.6. EPICS Structure

| | |
|---|---|
| CA: | Channel Access is for database requests that make no assumption as to whether or not the record is stored locally. |
| DA: | Database Access is for access of local database. |
| DB: | Database indicates direct access of memory mapped local database without use of Database Access routines. |
| OTHER: | Connections to other utilities or hardware. |

| **Program** | **CA** | **DA** | **DB** | **OTHER** |
|---|---|---|---|---|
| DM | YES | NO | NO | Host Display |
| ARR | YES | NO | NO | Host Display, Archive Files |
| Monitors | YES | YES | NO | NONE |
| eventScanTask | NO | YES | NO | NONE |
| periodicScanTask | NO | YES | NO | NONE |
| ioEventScanTask | NO | YES | NO | Driver Interrupt Routines |
| RecordSupport | YES | YES | YES | Device Support |
| Device Support | NO | NO | YES | Drivers,Record Support |
| Driver Interrupt | NO | NO | NO | VME |
| Device Drivers | NO | NO | NO | VME |

Channel Access Search Server accepts all search messages and if any process variables are located
at it's IOC replies to the sender. "I have xxx"

Channel Access Connection Request Server establishes connections to requesting clients who are using variables owned by it's IOC.

There are a few background tasks on the host that handle time, host channel access, and logging of IOC messages: nserver, CArepeaterz and iocLogServer.

## 4.7. EPICS Drivers

| | |
|---|---|
| Ab | Allen-Bradley Remote Serial IO |
| At5Vxi | LANL AT-5 designed VXI Modules |
| BB232 | BitBus -> RS-232 |
| BBMsg | BitBus driver for message based I/O |
| Bb902 | Burr-Brown 902 Binary Output |
| Bb910 | Burr-Brown 910 Binary Input |
| BitBus | XYCOM XVME-402 (a.k.a. IST/DATEM DMC804) BitBus Controller |
| Comet | Comet 4 Channel 5 MHz Transient Digitizer |
| CompuSm | Compumotor 1830 Stepper Moter Controller |
| Dvx | Analogic 2502 A/D Scanners |
| ExampleVxi | Skeleton VXI driver |
| Fp | FP 10S ’fast protect’ |
| Fpm | FP 1 OM ’fast protect master’ |
| Gpib | National Instruments Nl-1014/NI-1014D GPIB Controller |
| Hpl404a | HP E1404A VXI Bus Slot Zero Translator |
| Hpel368a | HP E1368A and HP E1369A VXI Microwave Switoh |
| Jgvtrl | Joerger 25Mhz Waveform Digitizer |
| KscV215 | Kinetic Systems V215 VXI 16-bit 32-channcl ADC |
| Msg | Generic Messaged Based I/O Driver |
| Mz8310 | Mizar 8310 Counter/Timer |
| Oms | Oregon Micro Systems Six Axis Stepper Motor Driver |
| Rs232 | RS-232 driver for VxWorkts tty driver |
| Stc | AMD STC (Am9513) |
| Time | Support for Timing Drivers (Mz8310,AT5_TIME,DG535) |
| Vmi4100 | XYCOM VMI4100 Analog Output |
| Xy010 | XYCOM SRM010 Bus Controller |
| Xy210 | XYCOM 210 Binary Input |
| Xy220 | XYCOM 220 Binary Output |
| Xy240 | XYCOM 240 Digital I/O |
| Xy566 | XYCOM 566 Analog Input |

FYI: The Keck chopper controller uses the following VME modules

| **Module** | **Function** | **EPICS driver** |
|---|---|---|
| VMI4116 | Analog Output | Vmi4100 |
| XY203 | Counter/Timer | None: substitute Mz8310 or equiv OR modify Time driver |
| XY240 | Digital I/O | Xy240 |

## 4.8. CCD Controller

The initial test of the EPICS system was to create a demo CCD controller. Since we did not have the supporting hardware all database records were SOFT or RAW SOFT and the hardware performance was mimicked by SNL code running on the IOC. Appendices A and B list the database and sequencer code required for this test.

### 4.8.1. Configuration

The host consisted of a SUN SPARCstation 2 running SunOS 4.1.2. For the CCD demo this was the existing icarus machine although a new host for the encoder lab is on order.

The target consisted of a 68030 based MVME147 running VxWorks 5.0.x (soon to be 5.1).

There was also a Delta Tau PMAC-VME card on the VMEbus which was not be used for this test.

### 4.8.2. Required EPICS Work

The EPICS version that will be loaded into the target was that originally developed for the 68020 based Heurikon HKV2F. The other target supported by the current distribution is the 68040 based MVME167. The EPICS installation manual does include instructions for porting to other VxWorks supported target architectures.

The database was created using the DCT configuration tool.

The EDD display editor was used to create the operator interface screens.

Sequencer code was written and compiled via SNC.

### 4.8.3. Applications Setup

The applications directory, /scr/epics/apple/ccd-sim, was set up according to the guideline in the following document:

> Experimental Physics and Industrial Control System (EPICS) Source/Release Control
> Chapter 5 Applications Developer - Basic.

### 4.8.4. Database

The database for the CCD simulator is listed in Appendix A at the end of this document.

### 4.8.5. bOPI Screens

The OPI screen for the CCD simulator was crcated by the display editor EDD.

#### 4.8.6. SNL Program

The listing of the SNL program, timer.st, can be found in Appendix B at the end of this document.

### 4.9. Encoder Test Station

The encoder test rig will be used as an EPICS evaluation system. This is a full blown implementation of an EPICS application consisting of the following parts:

- Device Driver for the Delta Tau PMAC board.
- Device Support routines to connect record aocess to the above device driver.
- EDD/DM based user interface.
- Full EPICS database.

#### 4.9.1. Environment

| | |
|---|---|
| Host: | SUN SPARCstation 2. |
| Target: | Motorola MVME147 68030-based VMEbus controller. |
| Modules: | Delta Tau PMAC board. |

# 5. CONCLUSIONS

## 5.1. EPICS

At this point EPICS is the leading contender for the development tool on which to base the proposed Gemini control system. This status could be changed pending the results of the completed encoder test station and further interactions with the industrial partners (Titan/Kinetic Systems). As Kinetic Systems has been a leader in the field of nuclear instrumentation for over the past decade we are not seriously concerned about the stability of this arrangement.

Another consideration is that the number of EPICS-like systems are increasing - major examples in progress are the ESO LCU software system and the Sac Peak rcal-time system. The proposed interface between the Gemini Virtual Telescope Controller running on the host and any real-time system can be made general enough to accommodate any distributed run-time database system implemented in the TCP/IP and VxWorks environment.

The existence of a (relatively) architecture-independent database aocess library (so called "channel
aocess") is a very powerful feature of this system.

## 5.2. Dataviews

This product is more suited to building a system like RTAP or EPICS. It is no surprise that there is a planned upgrade of the EPICS EDD/DM system to a DataViews version.

To use this system we would have to provide run-time database management and the rcal-time software required to communicate with the database in the target environment.

## 5.3. RTAP

Although RTAP is a strong system which makes ample use of distributed databases, GUIs, and sequencers/simulators it is hampered by the lack of a predefined interface to any true rcal-time system like VxWorks. It is also strongly tied to one particular hardware vendor (HP) and the high-end pricing structure is an additional detriment.

To use this system we would have to create our own interface between the RTAP environment and the VxWorks real-time system.

## 5.4. Recommendations

The following step is necessary to complete the recommendation of the EPICS system as a proposed basis for the Gemini control system

- Demonstration of Encoder Test Station with ANL/Motif EDD/DM version. The specifications for this are bcing drawn up. The fine tuning of the demo may be contracted out.

## 6. APPENDIX A: CCD-SIM DATABASE

The following table is taken from the report file generated by DCT's One-line listing command:

| **Type** | **Name** | **DESC** | **DTYP** |
|---|---|---|---|
| ai | ccd:time_remaining | Timeremainingin seconds | Soft Channel |
| ao | ccd:delay | Timer delay (for tests) | Soft Channel |
| ao | ccd:time | Exposure time in seconds | Soft Channel |
| bi | ccd:expose_status | Exposing status | Soft Channel |
| bi | ccd:readout_status | Reading out status | Soft Channel |
| bi | ccd:shutter_status | Shutter status | Soft Channel |
| bo | ccd:break_exposure | Break exposure command | Soft Channel |
| bo | ccd:hold_exposure | Hold exposure command | Soft Channel |
| bo | ccd:reset_ccd | Reset CCD command. | Soft Channcl |
| bo | ccd:start_exposure | Start exposure command | Soft Channel |

## 7. APPENDIX B.: CCD-SIM SNL PROGRAM

```
program timer

/*
*	A few definitions of binary states:
*/

#define OFF 0
#define ON l

#define CLOSED 0
#define OPEN l

#define NO 0
#define YES l

/*
*	Macro definitions for sequencer variable definition.
*/

#define II(vtype, vname, cname) \
	vtype vname; \
	assign vname to cname;

#define OO(vtype, vname, cname) \
	II(vtype, vname, cname) \
	monitor vname;

#define AO(vname, cname)	OO(int, vname, cname)
#define AI(vname, cname)	II(int, vname, cname)

#define BO(vname, cname)	OO(char, vname, cname)
#define BI(vname, cname)	II(char, vname, cname)

/*
*	PV Prefix: ccd:
*
*	User Controls:
*	start_exposure
*	break_exposure
*	hold_exposure
*	reset_ccd
*/
```

```
BO(start_exposure, "ccd:start_exposure")
BO(break_exposure, "ccd:break_exposure")
BO(hold_exposure, "ccd:hold_exposure")
BO(reset_ccd, "ccd:reset_ccd")

/*
*       Status Lines:
*
*       shutter_status
*       expose_status
*       readout_status
*/

BI(shutter_status, "ccd:shutter_status")
BI(expose_status, "ccd:expose_status")
BI(readout_status, "ccd:readout_status")

/*
*       CCD setup:
*       time
*       delay
*/

AO(time, "ccd:timeE")
AO(delay_time, "ccd:delay")

/*
*       CCD feedback:
*
*       time_remaining
*/

AI(time_remaining, "ccd:time_remaining")

/*
*       Event Flags:
*/

evflag timer_finished; /* Set by ss timer */
evflag reset_timer;
evflag start_timer;
evflag hold_timer;
evflag release_timer_hold;

/*
```

```
*       Internal variables:
*/

int elapsed_ticks;

ss controller
{
        state init
        {
                when ()
                        {
                                printf("CONTROLLER: Init.\n");

                                efClear(timer_finished);
                                efClear(reset_timer);
                                efClear(start_timer);
                                efClear(hold_timer);
                                efClear(release_timer_hold);

                                expose_status = OFF;
                                pvPut(expose_status);
                                readout_status = OFF;
                                pvPut(readout_status);
                                shutter_status = CLOSED;
                                pvPut(shutter_status);
                                efSet(reset_timer);
                        } state ready
        }
}
state ready
{
        when (start_exposure)
                {
                        printf("CONTROLLER: Exposure started.\n");
                        expose_status = ON;
                        pvPut(expose_status);
                        shutter_status = OPEN;
                        pvPut(shutter_status);
                        efSet(start_timer);
                } state exposing

                when (reset_ccd)
                {
                        printf("CONTROLLER: CCD reset.\n");
                        delay(0.5);
```

```
                reset_ccd = NO;
                pvPut(reset_ccd);
            } state init
}

state exposing
{
    when (start_exposure)
    {
        start_exposure = NO;
        pvPut(start_exposure);
    } state exposing

    when (reset_ccd)
    {
        delay(0.5);
        reset_ccd = NO;
        pvPut(reset_ccd);
        printf("CONTROLLER: CCD reset during exposure.\n");
    } state init

    when (hold_exposure)
    {
        printf("CONTROLLER: Exposure on hold.\nw
        expose_status = OFF; pvPut(expose_status);
        shutter_status = CLOSED; pvPut(shutter_status);

        efSet(hold_timer);
    } state onhold

    when (break_exposure)
    {
        delay(0.5);
        printf("CONTROLLER: Sequence broken.\n");
        break_exposure = NO;
        pvPut(break_exposure);
    } state init

    when (efTest(timer_finished))
    {
        printf("CONTROLLER: Timer finished.\n");
        expose_status = OFF;
        pvPut(expose_status);
        shutter_status = CLOSED;
        pvPut(shutter-status);
```

```
                readout_status = ON;
                pvPut(readout-status);

                efClear(timer_finished);
        } state readout
}

state onhold
{
        when (!hold_exposure)
        {
                printf("CONTROLLER: Exposure released from hold.\n");
                hold_exposure = NO;
                pvPut(hold_exposure);
                expose_status = ON;
                pvPut(expose_status);
                shutter_status = OPEN;
                pvPut(shutter_status);

                efSet(release_timer_hold);
        } state exposing
}

state readout
{
        when (delay(2.0))
        {
                printf("CONTROLLER: Readout completed.\n");
                readout_status = OFF;
                pvPut(readout_status);
        } state ready
}

ss timer
{
        state init
        {
                when ()
                {
                        printf("TIMER: Init.\n");
                        time = 0;
                        pvPut(time);
                        time_remaining = 0;
                        pvPut(time_remaining);
```

```
			efClear(reset_timer);
		} state ready
	}

	state ready
	{
		when (efTest(start_timer))
		{
			printf("TIMER: Started.\n");
			time_remaining = time;
			pvPut(time_remaining);
			elapsed_ticks = tickGet();

			efClear(start_timer);
			efClear(reset_timer);
		} state ticking
	}

	state ticking
	{
		when (time_remaining <= 0)
		{
			elapsed_ticks = tickGet() - elapsed_ticks;
			printf("TIMER: Took %ld ticks (%.3f secs) to complete.\n",
				elapsed_ticks, elapsed_ticks/60.0);
			efSet(timer_finished)
		} state ready

		when (efTest(hold_timer))
		{
			printf("TIMER: on hold.\n");
			efClear(hold_timer);
		} state onhold

		when (efTest(reset_timer))
		{
			printf("TIMER: Reset during ticking.\n");
			efClear(reset_timer)
		} state init

		when (delay(delay_time/60.0))
		{
			time_remaining -= l;
			pvPut(time_remaining);
		} state ticking
```

```
        }

        state onhold
        {
                when (efTest(reset_timer))
                {
                        printf("TIMER: Reset from hold.\n");
                        efClear(reset_timer);
                } state init

                when (efTest(release_timer_hold))
                {
                        printf("TIMER: Hold released.\n");
                        efClear(release timer hold);
                } state ticking
        }
}
```