JLAB-ACC-97-17

DESIGN AND IMPLEMENTATION OF A SLOW ORBIT CONTROL PACKAGE AT THOMAS JEFFERSON NATIONAL ACCELERATOR FACILITY

J. van Zeijts, S. Witherspoon, and W. A. Watson

Thomas Jefferson National Accelerator Facility

Abstract

We describe the design and implementation of a C++ client/server based slow orbit and energy control package based on the CDEV[1] software control bus. Several client applications are described and operational experience is given.

1 INTRODUCTION

The slow orbit control system at Jefferson Lab. was initially implemented as several Tcl[2] scripting applications[3, 4]. This was a good interim solution when most resources needed to be spent on commissioning the low level control system. To develop more robust high level applications an effort was started to convert the most demanding applications to C++. The first project completed was the conversion of the DIMAD[5] based on-line optics database from Tcl to a C++ client/server framework[6]. That effort included the first application of the CDEV client/server framework in operations. Next we started the conversion of the slow orbit/energy control and auto-steering applications.

2 REQUIREMENTS

The general orbit control package requirements are summarized in a Software Requirements Document[7]. The requirements call for locking beam positions and energy at a maximum rate of 1Hz. The focus is on extensive exception handling, and particular attention is given to the determination of misbehaving responders.

3 DESIGN & IMPLEMENTATION

Here we describe the various parts needed to build an operational lock. The CDEV library has available a robust and flexible generic client/server framework[8]. We use this to design a multiple client and multiple server implementation. External control of the each server is handled by sending CDEV messages. We describe the design of each part and its implementation.

3.1 Lock Server

The Lock Server contains a set of related locks which are handled by a 'Lock Manager' described below. Each lock can potentially be using a different algorithm. Multiple named instances of Lock Servers may exist on the network.

• Lock Manager

The Lock Manager handles the scheduling and interaction of the related locks. The simplest instance is a periodic manager which periodically triggers each lock in series. For steering applications we provide an 'On Demand' manager. Lock to Lock interaction is handled in more involved managers. We plan to support SLAC type adaptive handlers[9].

• Type/Algorithms

Multiple lock types are supported. They are mostly distinguished by the use of different algorithms for locking and steering the beam. The basic algorithm is Singular Value Decomposition for calculating the state of the system. Standard feedback control state space algorithms are supported. For demanding applications in orbit steering we use a special purpose algorithm[10]. The locks can be configured to any of the applicable types.

3.2 Configuration Server

The configuration of each lock is kept in a Database server. There is once instance of this server. This database keeps a set of configurations, where each configuration has a list of named locks and a lock manager type. Each lock has a list of responders and actuators and selection information, and has an algorithm type.

3.3 Optics Information & Model Server

The response matrices needed for the operation of the locks are retrieved from a 'Model' server. These models can represent the design optics, or contain measured data. In the simulated control system environment these models can produce simulated beam position readings.

3.4 'Gold' Orbits Server

The Gold Orbit Server is the centralized repository for the 'gold' orbits. This is a set of orbits to which the machine should be steered. Access to this server is needed by safe/restore clients and by all the steering and feedback applications. In particular, each lock can be notified when the gold orbit changes. The server provides 'xGold' and 'yGold' attributes for all beam position monitors. The use of this server has significantly improved the management of the operational gold orbit values.

3.5 Error Logging

A centralized client/server facility will be used to report errors[11]. This facility accepts error messages from clients distributed throughout the control system. Query messages and GUI query tools are used to monitor the state of any one of the systems.

3.6 Alarm Reporting

Alarm reporting will be done using the EPICS alarm handler program running over CDEV. Operators will hear an audible alarm when actuators go out of range or a feedback loop stops running because of a control system or hardware alarm.

3.7 Data Logging

The multiple lock servers present data to the outside world in a form such that standard CDEV data logging tools can be used. We use the generic StripTool application and will use the upcoming CDEV archiving standard.

3.8 User Interfaces

Even though the locks can in principle be operated by sending CDEV commands from the command line, this is clearly not sufficient for an operational interface. We use the Tcl/Tk tool box to provide lightweight graphics interfaces. The Tcl interpreter is dynamically extended with commands to access arbitrary CDEV devices and send/receive arbitrary cdevData packets. Asynchronous monitoring fits well in the Tcl event model.

4 OPERATIONAL IMPLEMENTATIONS

Here we describe the lock servers which are in use in operations at JLAB.

4.1 OpsLock Server

This server contains the default set of operational slow feedback loops. We have 3 slow energy locks and multiple slow orbit feedback loops, including orbit feedback locks in front of the experimental hall targets.

4.2 ArcDiagnostics Server

This server runs in calculate only mode and is used to diagnose energy shift problems throughout the machine. A momentum deviation is calculated for each of the 9 arcs and displayed on a strip chart tool.

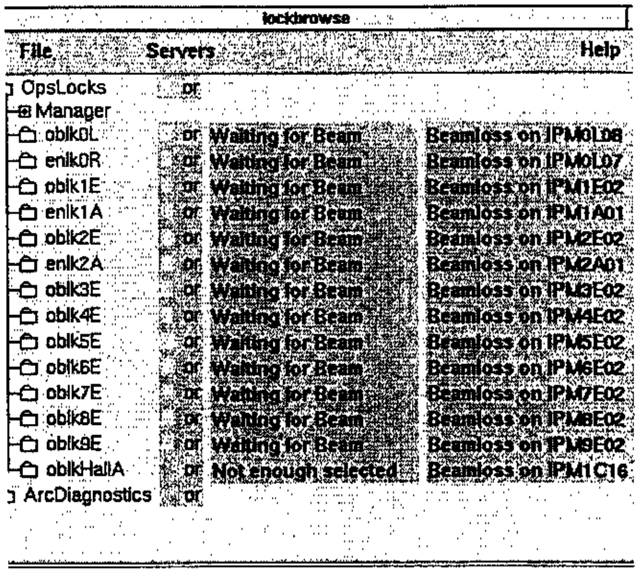
4.3 OpsSteering Server

This server is configured with beam steering algorithms for the linacs and the arcs.

5 CLIENT IMPLEMENTATIONS

5.1 Operator Control GUI Client

The main operational interface is shown below. Operators can start/stop a feedback loop and get a status readback from both for the lock status and the beam status. All operational servers can be accessed from this main panel. An expert panel allows selection/deselection of elements and changing lock parameters. Multiple instances of these browsers can be up at the same time and changes are coordinated by the CDEV server monitoring capability.



```
OpsLocks        or
 Manager
 oblk0L         or  Waiting for Beam       Beamloss on IPM0L08
 enlk0R         or  Waiting for Beam       Beamloss on IPM0L07
 oblk1E         or  Waiting for Beam       Beamloss on IPM1E02
 enlk1A         or  Waiting for Beam       Beamloss on IPM1A01
 oblk2E         or  Waiting for Beam       Beamloss on IPM2E02
 enlk2A         or  Waiting for Beam       Beamloss on IPM2A01
 oblk3E         or  Waiting for Beam       Beamloss on IPM3E02
 oblk4E         or  Waiting for Beam       Beamloss on IPM4E02
 oblk5E         or  Waiting for Beam       Beamloss on IPM5E02
 oblk6E         or  Waiting for Beam       Beamloss on IPM6E02
 oblk7E         or  Waiting for Beam       Beamloss on IPM7E02
 oblk8E         or  Waiting for Beam       Beamloss on IPM8E02
 oblk9E         or  Waiting for Beam       Beamloss on IPM9E02
 oblkHallA      or  Not enough selected    Beamloss on IPM1C16
ArcDiagnostics  or
```

Figure 1: Main Operations Lock Overview Panel

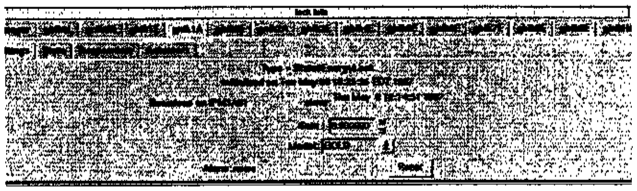


Figure 2: Expert State of Lock Display

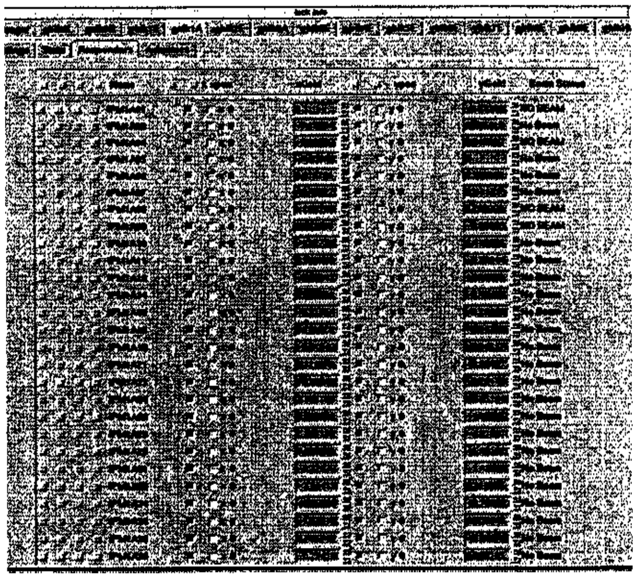
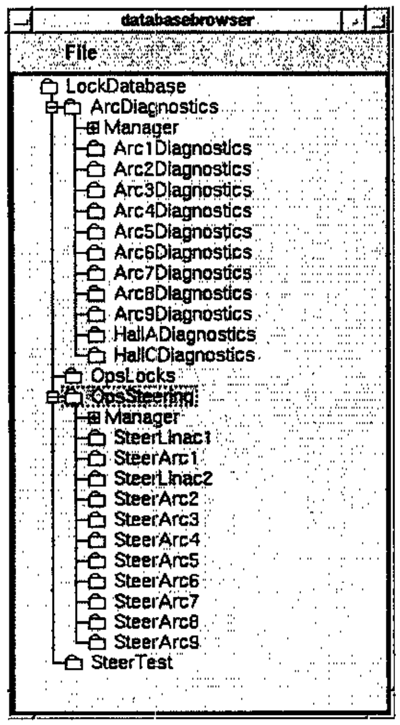


Figure 3: Responder Configuration Panel

5.2 Lock Database Configuration Tool

A browser GUI was developed to query and set the lock database. Persistent changes to the configurations can be made from this panel.



```
databasebrowser
File
LockDatabase
  ArcDiagnostics
    Manager
    Arc1Diagnostics
    Arc2Diagnostics
    Arc3Diagnostics
    Arc4Diagnostics
    Arc5Diagnostics
    Arc6Diagnostics
    Arc7Diagnostics
    Arc8Diagnostics
    Arc9Diagnostics
    HallADiagnostics
    HallCDiagnostics
  OpsLocks
  OpsSteering
    Manager
    SteerLinac1
    SteerArc1
    SteerLinac2
    SteerArc2
    SteerArc3
    SteerArc4
    SteerArc5
    SteerArc6
    SteerArc7
    SteerArc8
    SteerArc9
  SteerTest
```

Figure 4: Lock Database Configuration GUI

5.3 Strip Chart Client

For operational diagnostics purposes we can use a standard CDEV tool like the StripTool to display a live strip chart of parameters calculated by any of the Lock servers. Here we show the dp/p momentum deviation for 2 arcs in the CEBAF accelerator.

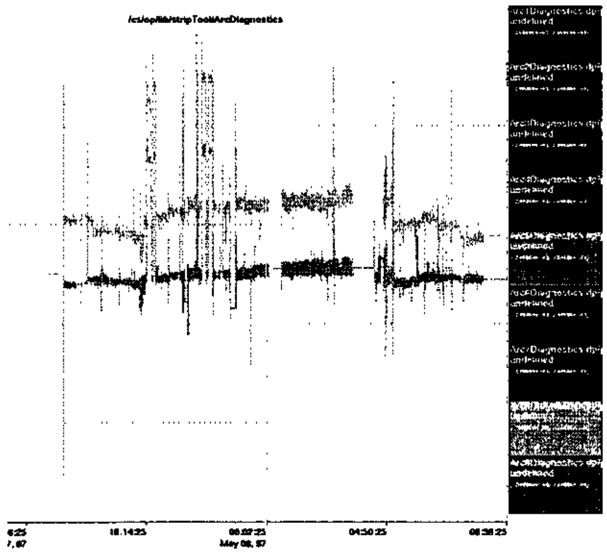


Figure 5: ArcDiagnostics Strip Chart

6 COMMISSIONING

Before releasing the code in operations it needs to be tested exhaustively. We take advantage of the CDEV device/attribute paradigm which gives us the ability to re-route control system devices to different servers. We build a 'fake' control system server and point the control system devices to this by a simple change in the routing information database. The application code is not changed. All the exception handling can be exercised by manipulating the fake control system to trigger arbitrary exceptions.

7 UPGRADES

The Database server will be backed up by an object-oriented persistent database. The code for any of the Database clients will need no change when we switch the database server to this. Likewise we can move from the current gold orbit server, to a persistent database server with no change in client code.

8 ACKNOWLEDGMENTS

We would like to acknowledge Walt Akers for providing the code and documentation for the CDEV Generic Server engine. This work is supported by the U.S. Department of Energy, contract DE-AC05-84ER40150.

9 REFERENCES

[1] W. A. Watson, 'A Portable Accelerator Control Toolkit', PAC 1997 Proceedings.

[2] J. K. Ousterhout, 'Tcl and the Tk Toolkit', Addison-Wesley Professional Computing Series (1994).

[3] J. van Zeijts, 'Rapid Application Development Using the Tcl/Tk Language', PAC 1995 Proceedings, Vol 4 p 2241.

[4] J. Karn et al., 'Development of Digital Feedback Systems for Beam Position and Energy at the Thomas Jefferson National Accelerator Facility'.

[5] R. Servranckx et. al., DIMAD Manual.

[6] B. Bowling et. al., 'Integrated On-Line Modeling at CEBAF', PAC 1995 Proceedings.

[7] J. van Zeijts, 'Software Requirements for Slow Locks', Jefferson Lab., internal document.

[8] W. Akers, The CDEV Generic Server, Jefferson Lab. CDEV Documentation. www.jlab.org/cdev/doc_1.5/cdevGenericServer.html

[9] T. Himel, 'Requirements and Design Overview for Adaptive Cascaded Feedback'. SLAC Preprint, October 1991.

[10] Y. Chao, 'Prosac Algorithm', CAP 96, 319.

[11] J. Chen, CMLOG documentation, Jefferson Lab., internal document.