THE EVOLUTION OF THE GENERIC LOCK SYSTEM AT JEFFERSON LAB*

B. Bevins† and Y. Roblin
Jefferson Lab, Newport News, VA 23606, USA

Abstract

The Generic Lock system is a software framework that allows highly flexible feedback control of large distributed systems. It allows system operators to implement new feedback loops between arbitrary process variables quickly and with no disturbance to the underlying control system. Several different types of feedback loops are provided and more are being added. This paper describes the further evolution of the system since it was first presented at ICALEPCS 2001 and reports on two years of successful use in accelerator operations. The framework has been enhanced in several key ways. Multiple-input, multiple-output (MIMO) lock types have been added for accelerator orbit and energy stabilization. The general purpose Proportional-Integral-Derivative (PID) locks can now be tuned automatically. The generic lock server now makes use of the Proxy IOC (PIOC) developed at Jefferson Lab to allow the locks to be monitored from any EPICS Channel Access aware client. (Previously clients had to be Cdev aware.) The dependency on the Qt XML parser has been replaced with the freely available Xerces DOM parser from the Apache project.

INTRODUCTION

The low-level accelerator control systems at Jefferson Lab are built with EPICS running on IOC front-end computers. The "Slow Locks" are a group of programs that implement feedback loops to stabilize various machine parameters against disturbances at < 1Hz [1,2,3]. They come in various flavors and each lock type uses a distinct server, GUI, and configuration format. Adding new lock types has been difficult and time consuming. Even adding new lock instances requires time from a software developer.

A "Generic" Lock System has been developed at Jefferson Lab [4] to integrate the Slow Locks into a common framework with better performance that is more easily maintainable and extensible. Because it is configuration-driven rather than code-driven, it is more flexible and can be reconfigured by non-experts while running. This enables on the fly creation and configuration of feedback loops using any available control system I/O signals. Thus feedback loops can easily be created as prototypes for new control ideas or to satisfy temporary operational requirements with absolutely no disruption of the existing control system and no new programming effort required.

The system has been in use and under continuing development for over two years with very positive results. The Operations staff is very happy with the ability to create new locks on demand. Many of these have achieved permanent status as required parts of the control system. The PID Locks have been enhanced and several new lock types have been added to the system. The effort to fully integrate the older systems is continuing.

SYSTEM DESIGN

The new lock system has been called "generic" because of a few key features.

Server Architecture

The main component of the Generic Lock System is the Lock Server. It is implemented in C++ using as a CDEV (Common DEVice) Generic Server [5,6]. Generic programming techniques have been used throughout (OO, templates, STL, Design Patterns, etc.). Each lock object is a virtual device that exposes a set of attributes containing the lock's operating parameters. Each lock type completely defines its own characteristics, such as how many inputs and outputs it has, how it is calibrated or tuned, whether it can be created or deleted on the fly, etc. All lock types share a common interface, so the same server is used for all types and can host multiple types simultaneously.

The Generic Lock Server presently handles general purpose Proportional, Integral, Derivative (PID) locks as well as the specialized current locks and several types of helicity-correlated asymmetry locks. Several new types are being added.

User Interface

The GUI for the slow locks is scripted in Tcl/Tk using the TclCdev package. The GUI is not preconfigured. It discovers all the locks dynamically. A Tk package has been created to give all related GUI's a consistent look and feel.

Configuration Files

All configuration information is stored in an eXtensible Markup Language (XML) file with a flexible structure, and can be entered directly through the GUI. The structure of the XML file is described in [4]. The format is simple and flexible enough that there has been no need to modify it as new lock types have been added. The XML files are both parsed and written using the Xerces Document Object Model (DOM) parser freely available from the Apache Project [7].

*Work supported by the U. S. DOE, contract DE-AC05-84-ER40150.
†bevins@jlab.org

The Proxy IOC

The Proxy IOC (PIOC) is an EPICS Channel Access (CA) server running on a back-end computer [8]. It allows other programs to dynamically create ("sponsor") soft PV's, which the PIOC then serves to CA clients. The Generic Lock Server optionally sponsors all of its virtual device attributes as PV's in the PIOC. This makes them available to any CA aware client, e.g. display managers, archivers, save and restore tools, etc. One drawback is that communicating with the lock server only through PV's lacks the full expressiveness of CDEV's device/message model. Figure 1 shows the communications process as the Lock Server uses the PIOC.

The use of CDEV as an underlying technology had imposed another limitation on the system which has now been addressed though the use of the PIOC. Previously, dynamically created locks either had to have predefined names (e.g., PIDLock01) or the server had to rewrite and recompile the Device Definition Language (DDL) files each time a lock was created on the fly. With CDEV configured to fall through to Channel Access, the PIOC allows dynamic discovery by clients of the new devices.

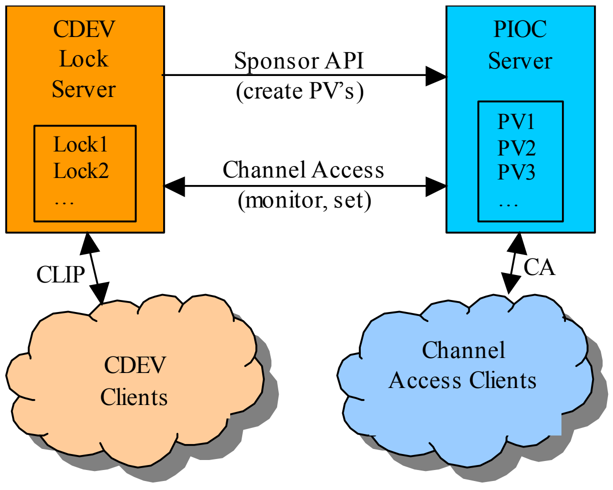


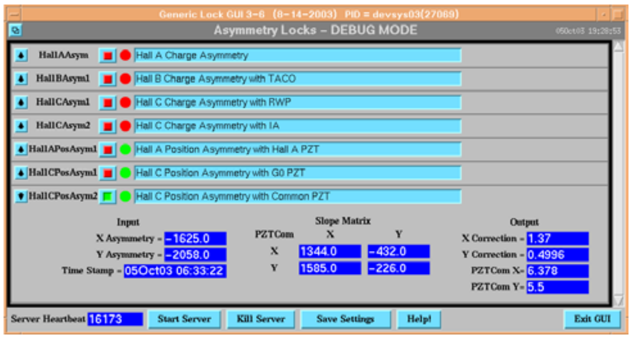
Figure 1: Use of the Proxy IOC

NEW AND ENHANCED LOCK TYPES

Multiple Input/Multiple Output Locks

The MIMO locks implemented so far are for managing the helicity-correlated asymmetries of the electron beam for parity experiments. 2x2 locks are in use now, with 3x3 and 5x5 locks being tested. The 2x2 local orbit locks and the highly over-determined energy locks are currently being integrated into the Generic Lock framework as well.

The Asymmetry Lock GUI is shown in Figure 2. It presents a view nearly identical to that of the PID Lock GUI in [4]. The lock shown expanded has a vector input, a 2x2 gain matrix, and a vector output.


Generic Lock GUI 3-6 (8-14-2003) PID = devsys03(27069)

Asymmetry Locks – DEBUG MODE   05Oct03 19:28:53

HallAAsym	🟥	●	Hall A Charge Asymmetry
HallBAsym1	🟥	●	Hall B Charge Asymmetry with TACO
HallCAsym1	🟥	●	Hall C Charge Asymmetry with RWP
HallCAsym2	🟥	●	Hall C Charge Asymmetry with IA
HallAPosAsym1	🟥	●	Hall A Position Asymmetry with Hall A PZT
HallCPosAsym1	🟥	●	Hall C Position Asymmetry with G0 PZT
HallCPosAsym2	🟩	●	Hall C Position Asymmetry with Common PZT

Input
X Asymmetry - -1625.0
Y Asymmetry - -2058.0
Time Stamp - 05Oct03 06:33:22

Slope Matrix

PZTCom	X	Y
X	1344.0	-432.0
Y	1585.0	-226.0

Output
X Correction - 1.37
Y Correction - 0.4996
PZTCom X- 6.378
PZTCom Y- 5.5

Server Heartbeat 16173

Figure 2: The Asymmetry Lock GUI

The PID Locks

The functionality of the PID locks is derived from the EPICS CPID record, extended to allow both the input and output signals to be arbitrarily specified at runtime. Any variable in the control system accessible through CDEV can be used. This includes all CA signals as well as signals from other CDEV servers, including model servers and other locks.

For the lock input, the user can enter an algebraic expression involving the names of up to 12 channels and all the functions and operators available to the EPICS calc record with the exception of the C conditional "?:". This is disallowed since colons frequently appear in PV names, creating a parsing ambiguity. For example the string *"1–0.5*exp(CTD1242.VAL–CTD1248.VAL)"* could be entered as the lock input "variable". A second such expression can be entered that will disable the lock when it evaluates to logical false.

An addition to the PID Lock functionality is the ability to designate the input as periodic with a specified period. The error between setpoint and input value is calculated modulo this period. For example, if an input is specified to have period 360 and a setpoint of 175, then an input value of -175 gives an error (setpoint – input) of –10 instead of +350.

Automatic Tuning of PID Locks

Determining appropriate gains for PID controllers is a non-trivial task. To make the PID Lock class more usable by non-experts it is desirable to automatically determine appropriate PID gains for optimum stability, if they exist. Automatic PID tuning is still experimental and not in production use. Two methods are provided for testing.

For processes with simple dynamics where the main goal is disturbance rejection, a static gain measurement is performed. Most slow locks used by the accelerator are of this type since accelerator dynamics are typically much faster than the time scale over which the slow locks operate, giving them effectively instantaneous proportional response.

For processes with richer dynamics on the slow lock time scale, the Ziegler–Nichols frequency response method is used [9]. A relay feedback experiment is performed to determine the ultimate point on the Nyquist curve, with relay hysteresis set automatically from the process+measurement noise. This works reasonably well for process dynamics up to second order. This tuning method is provided primarily for use by the Cryogenics Group.

FUTURE DIRECTIONS

Additional Lock Types

The legacy orbit and energy locks are presently being integrated into the Generic Lock framework. This extends the system to allow for locks to be calibrated at runtime and to use the on-line accelerator model server for calibration. A spin-off of this effort is the creation of general purpose MIMO locks of arbitrary size than can be created and configured on the fly.

PID Lock Tuning

Automatic tuning of the PID Lock gains will be improved and become the standard way of configuring a newly created lock. This will require better characterization of the process dynamics than the simple relay feedback experiment provides. This could be achieved by simple modifications to the procedure to obtain additional points on the Nyquist curve.

Server Security

There is still a security issue with the Generic Lock Server. The CDEV Generic Server has no built-in security model. This means that any user connected to the accelerator network can write to the virtual attributes of the lock devices and create a lock that writes to any channel in the system, effectively circumventing CA security. The strategy to attack this problem has changed. Rather than add a security layer to the server itself, tighter integration with the PIOC will allow its CA security to protect the lock server.

Dynamic Linking

Dynamic Linking of lock types with the server is still a goal. Following the model that CDEV uses with its service classes, the code for individual lock types could be dynamically loaded as needed. This will allow completely new types of locks to be added to a running server without even restarting it, much less rebuilding it.

Other Extensions of the Framework

The Generic Lock System framework will be gradually extended to include non-feedback accelerator setup tools to bring the benefits of common look and feel and easier maintainability.

INFRASTRUCTURE

Though some of the lock types are specialized for Jefferson Lab, the general purpose PID Locks and the forthcoming general purpose MIMO locks are flexible enough to be useful elsewhere. Developing new lock types to add to the framework is also quite easy. Building and using the system requires the following elements, all of which are freely available:

- CDEV including the Generic Server extension
- A CDEV service supporting the underlying control system. (EPICS, TINE, ADO, CLD/SLD, etc.)
- Xerces, the XML DOM parser from Apache
- Tcl/Tk with TclCdev
- The Proxy IOC (optional) form Jefferson Lab

REFERENCES

[1] M. Bickley, B. A. Bowling, D. A. Bryan, J. van Zeijts, K. S. White and S. Witherspoon, “Using Servers to Enhance Control System Capability”, Proceedings of PAC 1999, New York.

[2] J. van Zeijts, S. Witherspoon and W. A. Watson, “Design and Implementation of a Slow Orbit Control Package at Thomas Jefferson National Accelerator Facility”, Proceedings of PAC 1997, Vancouver.

[3] A. Hofler, D. Bryan, L. Harwood, M. Joyce and V. Lebedev, “Empirically Determined Response Matrices for On-line Orbit and Energy Correction at Thomas Jefferson National Accelerator Facility”, Proceedings of PAC 2001, Chicago.

[4] B. Bevins and A. Hofler, “A Distributed Feedback System for Rapid Stabilization of Arbitrary Process Variables”, Proceedings of ICALEPCS2001, San Jose, CA.

[5] J. Chen, G. Heyes, W. Akers, D. Wu and W. Watson, “CDEV: An Object-Oriented Class Library for Developing Device Control Applications”, Proceedings of ICALEPCS95, Chicago.

[6] W. Akers, “An Object-Oriented Framework for Client-Server Applications”, Proceedings of ICALEPCS97, Beijing.

[7] The Apache Project, “The Apache XML Project”, http://xml.apache.org.

[8] J. Sage, “Proxy IOC & CA NameServer Update”, EPICS Collaboration Meeting, Fall 2002, Berlin.

[9] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed., Instrument Society of America, 1995.