A Distributed Feedback System for Rapid Stabilization of Arbitrary Process Variables

Brian Bevins and Alicia Hofler

ICALEPCS 2001

Outline

- The Problem – Making Feedback Easy
- Our Solution – The Generic Lock Server
- The PID Locks
- Implementation
- Initial Results
- Future Directions
- Conclusions

The Problem

- Feedback control loops (“locks”) are an essential part of any process control system.
- In control systems for **experimental** equipment and facilities, new problems and new ideas are always changing the requirements.
- Shift operators, physicists, and system experts would like to create new locks on the fly without the need for additional programming effort.
- The goal is to allow for quick prototyping of new control ideas and easy accommodation to temporary operating conditions.

Requirements

- **Rapid:** New locks can be created on the fly with no disruption to the underlying control system or the operating machine.
- **Distributed:** All the locks must be accessible from any operations console. No information is hidden.
- **Arbitrary:** Any process variable from any front end computer can be used for input or output of a lock.

The Environment

- The Continuous Electron Beam Accelerator Facility (CEBAF) at Jefferson Lab is a 6 GeV electron linear accelerator for nuclear physics.
- The control system for CEBAF is based on EPICS with over 250,000 database records on approximately 100 IOC front-end computers.
- The “Slow Locks” are a collection of programs running on back-end hosts that implement closed loop feedback for various parts of the system at speeds ≤ 1 Hz.

The Solution

- A “Generic” Lock Server has been developed to consolidate the slow locks into a unified framework to improve performance, extensibility, and maintainability.
- Three lock types have been integrated thus far:
  - beam currents
  - beam polarization-correlated asymmetries
  - general purpose Proportional-Integral-Derivative (PID) locks.
- The server allows a user to create PID locks on the fly for any process variable using any other process variable.

PID Lock Features

- The PID functionality is based on the EPICS cpid record.
- The locks reside in a central server that enforces consistency. A new lock cannot be activated using an output variable that is in use by another lock.
- A GUI allows all the locks to be viewed from any X display and controlled by any authorized user.
- All configuration information for the locks is stored in a human readable eXtensible Markup Language (XML) file.

Input Expressions

- Instead of specifying a channel name for the lock input, the user can specify an expression involving the values of up to 12 channels. For example:

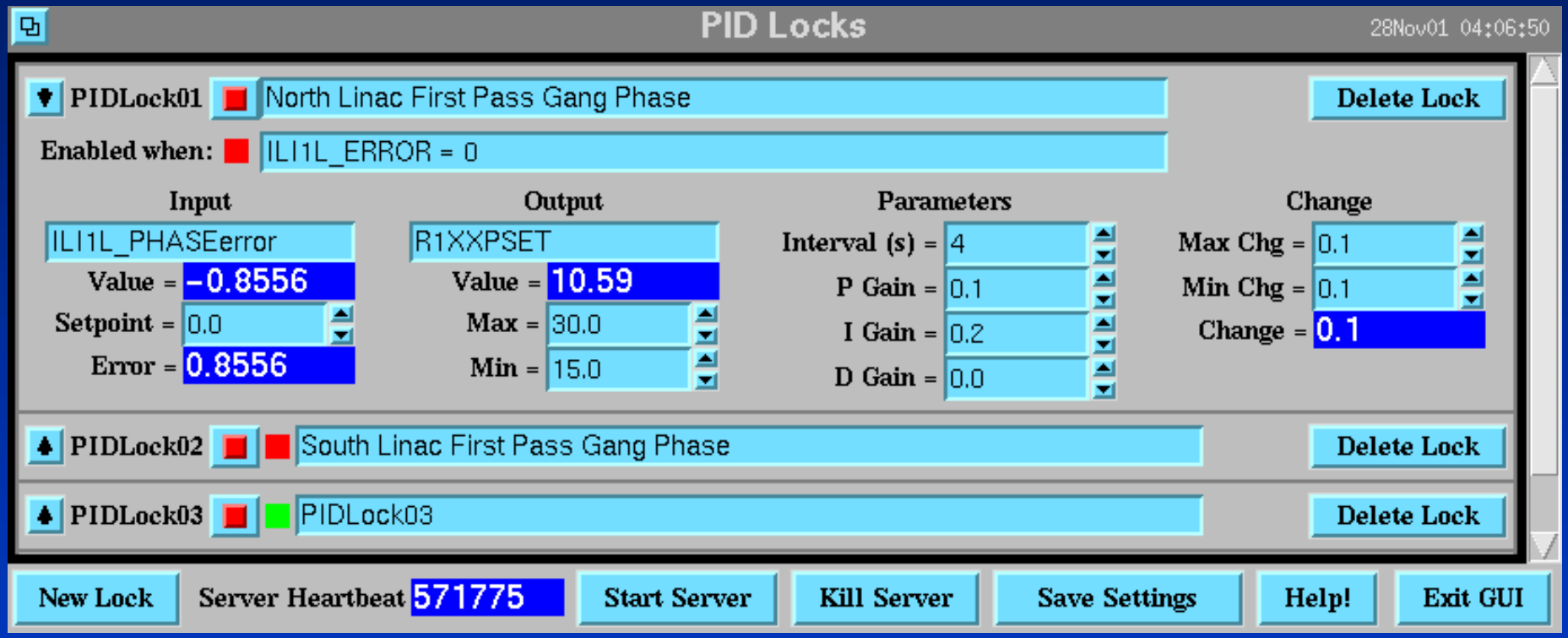
  1 – 0.5*exp(CTD1242.VAL – CTD1248.VAL)

- The full range of functions available to the EPICS calc record can be used.
  - Exception: the C conditional operator “?:”
- A second expression can be entered that will disable the lock when it evaluates to false.

PID Lock Usage

1. Have a clever idea.
2. Start the PID Lock GUI.
3. Button to create a new PID lock.
4. Enter a name for the output (control) variable and a name or expression for the input (locked) variable.
5. Optionally enter an enabling expression.
6. Enter the desired set point, min/max output, etc.
7. Enable the lock to start it operating.
8. Adjust the gains as needed.

PID Lock GUI


PID Locks
28Nov01 04:06:50
PIDLock01
North Linac First Pass Gang Phase
Delete Lock
Enabled when:
ILI1L_ERROR = 0
Input
ILI1L_PHASEerror
Value = -0.8556
Setpoint = 0.0
Error = 0.8556
Output
R1XXPSET
Value = 10.59
Max = 30.0
Min = 15.0
Parameters
Interval (s) = 4
P Gain = 0.1
I Gain = 0.2
D Gain = 0.0
Change
Max Chg = 0.1
Min Chg = 0.1
Change = 0.1
PIDLock02
South Linac First Pass Gang Phase
Delete Lock
PIDLock03
PIDLock03
Delete Lock
New Lock
Server Heartbeat 571775
Start Server
Kill Server
Save Settings
Help!
Exit GUI

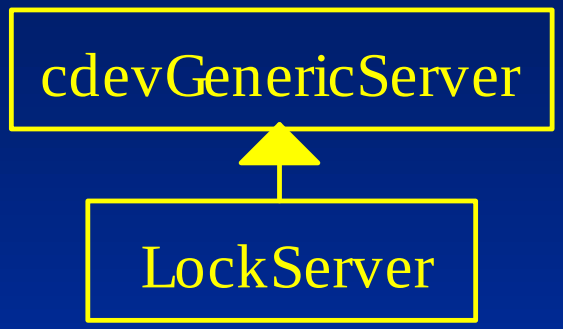
Implementation

- The server is implemented in standard C++ and is highly object oriented.
- Nearly all memory management is handled invisibly by the standard library.
- The Common DEVice (CDEV) Generic Server framework is used for all network activity.
- Each lock exists as a virtual CDEV device whose attributes can be accessed by other programs like StripTool and MEDM.

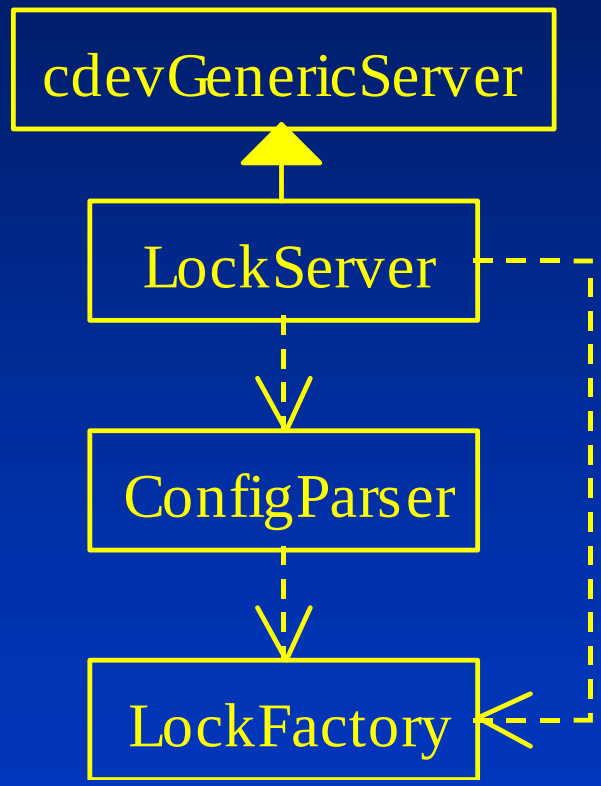
Implementation Cont'd

- Input signals are accessed through the CDEV client library.
  - all EPICS channel access signals are available
  - also values from other CDEV servers such as the on-line model server Art++ (THAP012, Y. Roblin and T. Larrieu).
- The XML configuration file is read and written using the Document Object Model (DOM) parser in the Qt toolkit from Trolltech, AS.
- The GUI is implemented in Tcl/Tk using the TclCdev package.

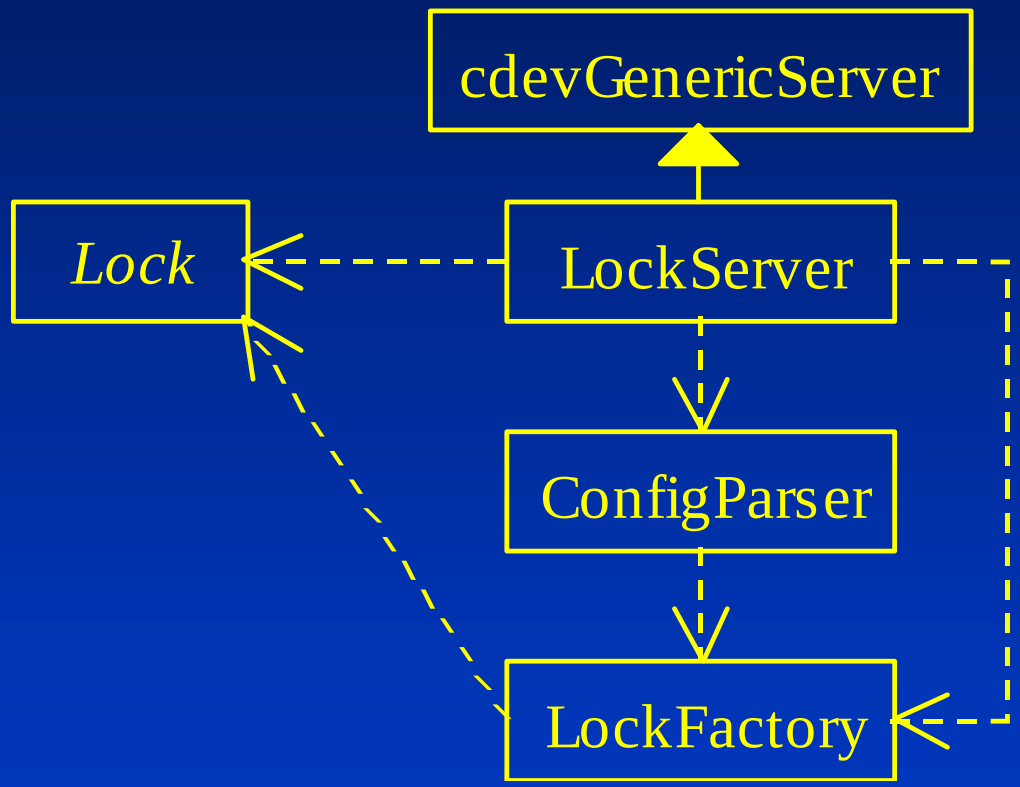
Server Class Diagram



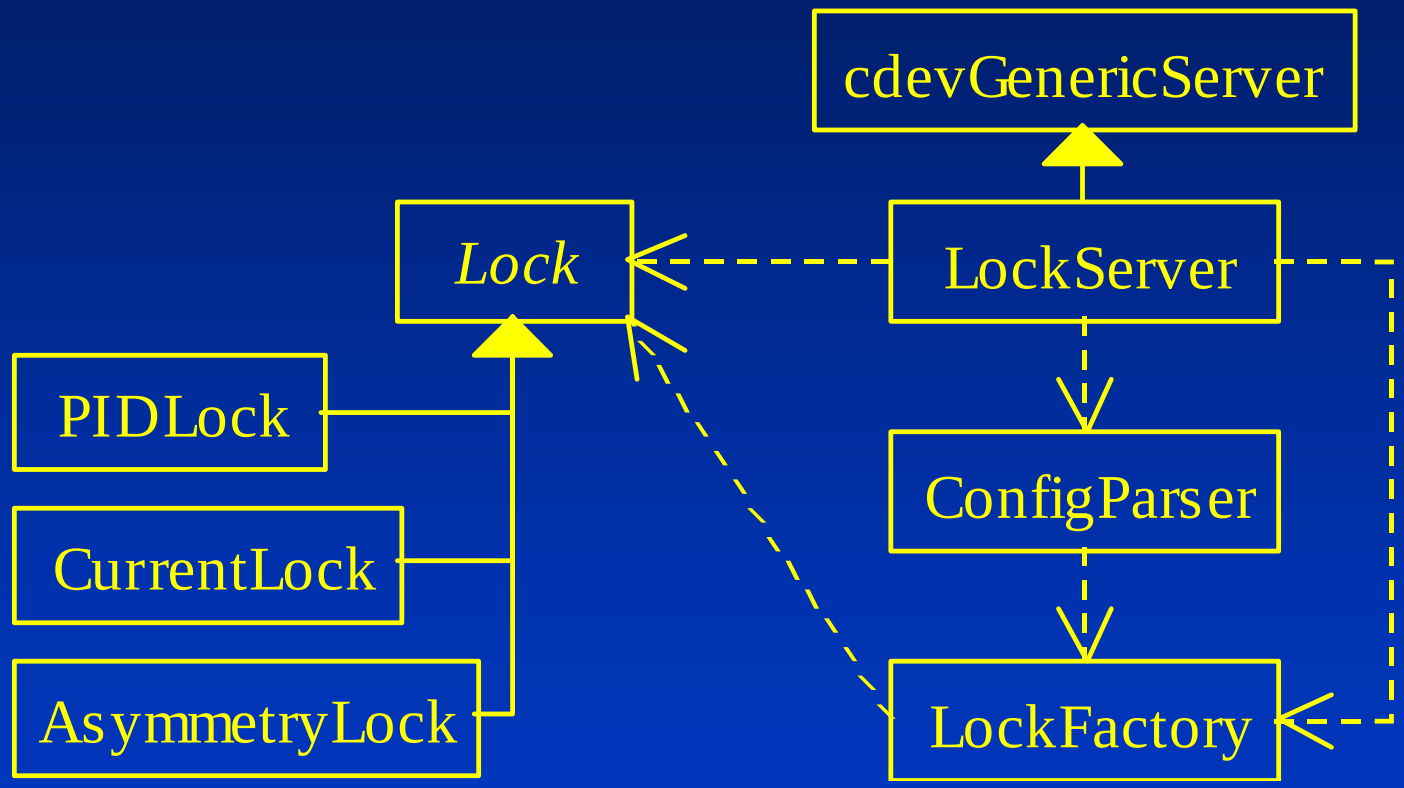
Server Class Diagram



Server Class Diagram



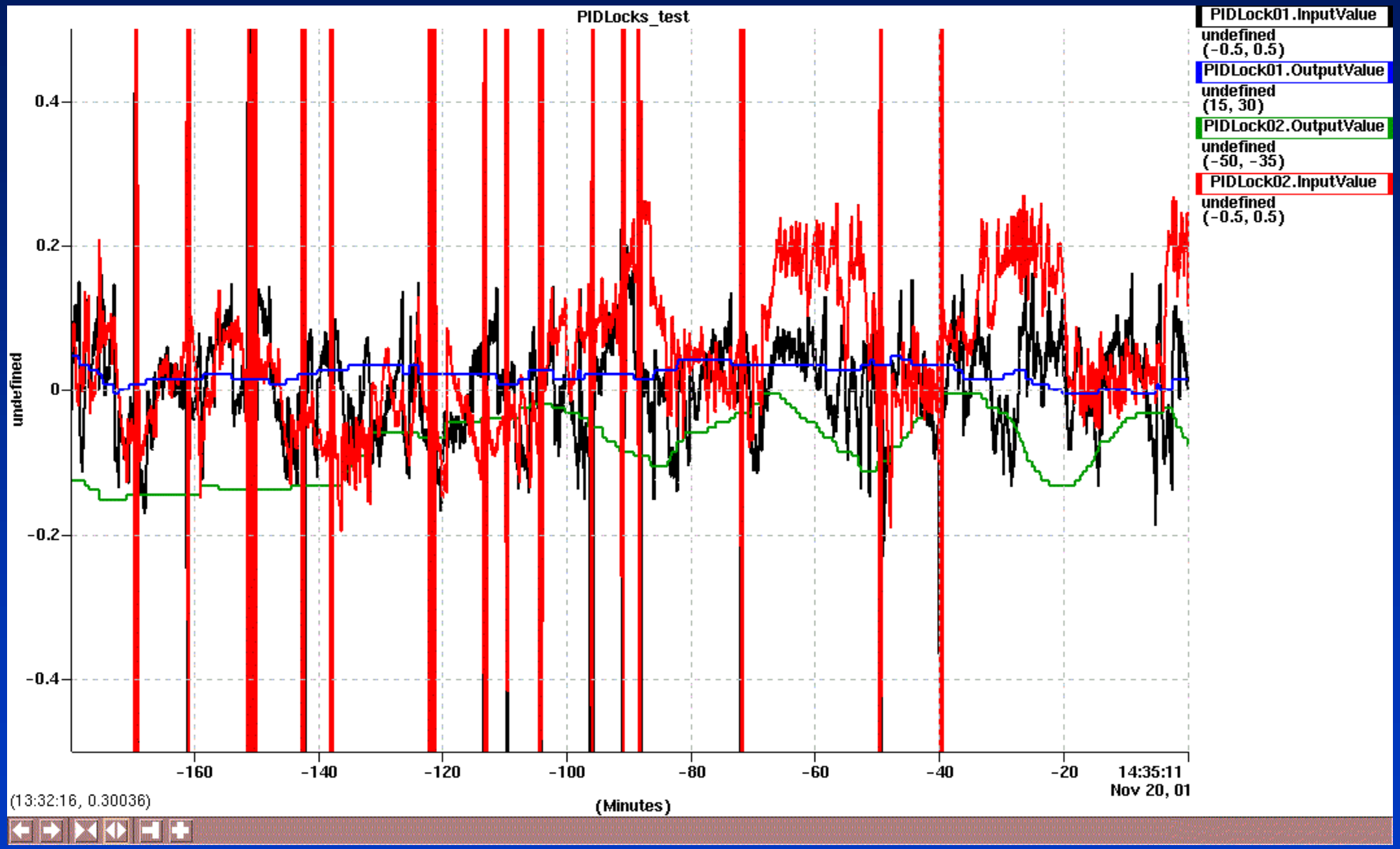
Server Class Diagram



XML Configuration

```
<lockConfig>
    <Lock type="PIDLock" name="PIDLock02" />
    <device name="PIDLock02" >
        <attribute value="0" name="GainD" />
        <attribute value="0.2" name="GainI" />
        <attribute value="0.1" name="GainP" />
        <attribute value="ILI1L_PHASEerror" name="InputName" />
        <attribute value="4" name="Interval" />
        <attribute value="0.1" name="MaxChange" />
        <attribute value="0.1" name="MinChange" />
        <attribute value="30" name="MaxPos" />
        <attribute value="15" name="MinPos" />
        <attribute value="R1XXPSET" name="OutputName" />
        <attribute value="0" name="SetPoint" />
        <attribute value="North Linac First Pass Gang Phase"
          name="Description" />
        <attribute value="ILI1L_ERROR = 0" name="EnableString" />
    </device>
</lockConfig>
```

Results


PIDLocks_test
PIDLock01.InputValue
undefined
(-0.5, 0.5)
PIDLock01.OutputValue
undefined
(15, 30)
PIDLock02.OutputValue
undefined
(-50, -35)
PIDLock02.InputValue
undefined
(-0.5, 0.5)
0.4
0.2
0
-0.2
-0.4
undefined
-160
-140
-120
-100
-80
-60
-40
-20
14:35:11
Nov 20, 01
(13:32:16, 0.30036)
(Minutes)

Future Work – Near Term

- **Flexibility:** Other types of control loops will be added.
  - multiple inputs/outputs
  - model based feedback
- **Security:** The CDEV Generic Server engine does not have a built-in security model. A security layer will be added using Access Security.
- **Usability:** Auto-tuning of PID loop gains and/or gain scheduling will be added.

Future Work – Longer Term

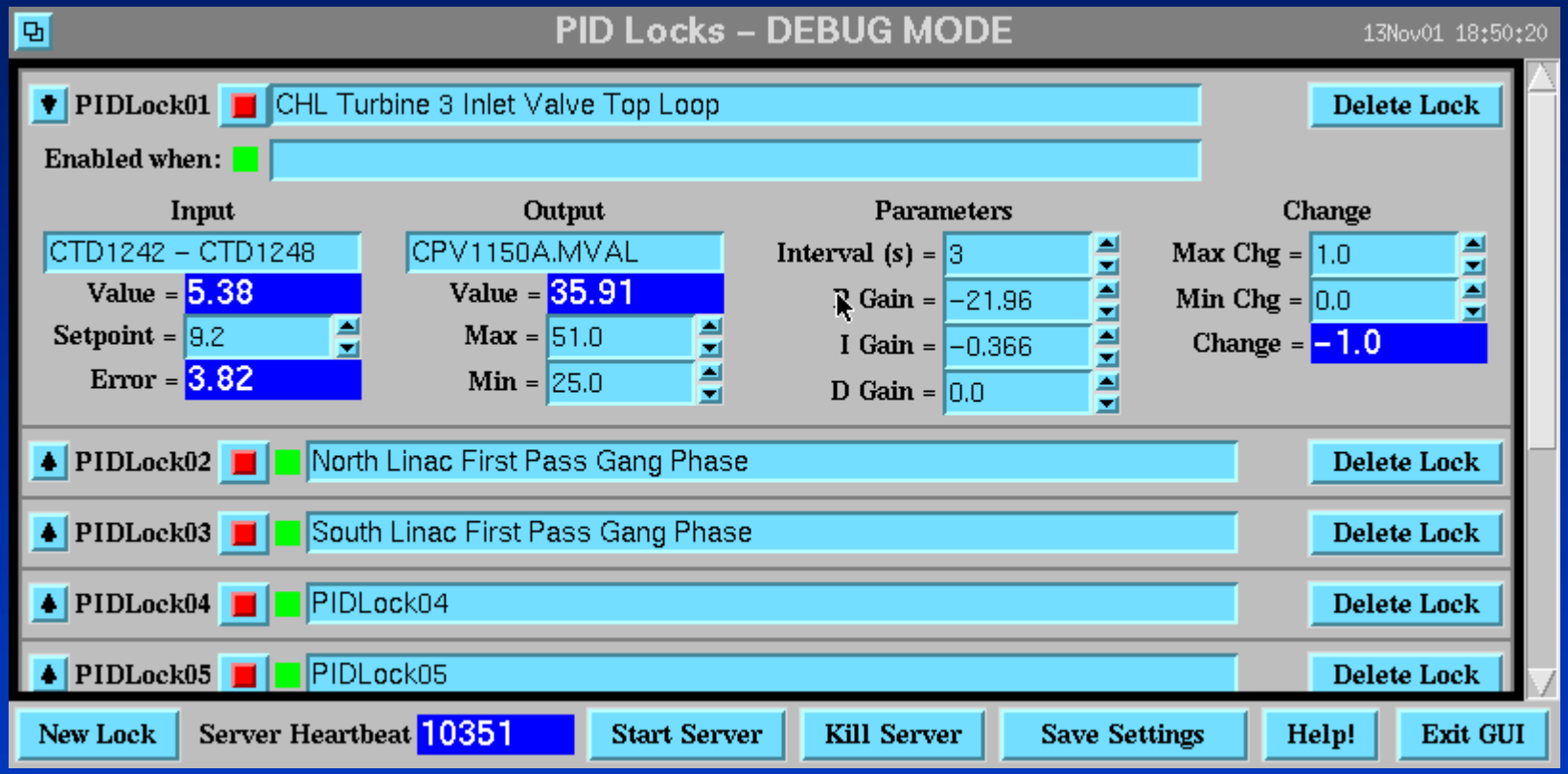
- **Output Functions:** A user specified calculation could be applied to the lock output before writing to the output channel.

- **Dynamic Linking:** Each lock type could be compiled into a shared object that would be loaded only if needed, similar to the way CDEV services are handled. This would allow completely new lock types to be added to a running server.

Conclusions

- The Generic Lock Server is already reducing the burden of maintaining and extending the CEBAF slow locks, with greater benefits still to come.
- The ability to create new PID locks on the fly adds a unique capability to the accelerator operator's toolbox.
- Implementation of the PID Locks went surprisingly quickly thanks to the flexibility of the Generic Lock Server architecture and reliance on existing code from:
  - C++ Standard Library
  - CDEV Generic Server
  - EPICS Calculation Record
  - Qt DOM XML Parser

PID Lock GUI Cont'd



Overview

- In most systems, adding new locks is difficult:
  - reboot the front end computers ➔ highly disruptive
  - restart the back end lock server ➔ somewhat disruptive
  - create new locks on the fly ➔ non-disruptive