Fermilab LLRF Controls Integration

Pierrick Hanlet
PIP-II LLRF Final Design Review

17 July 2024

A Partnership of:
- US/DOE
- India/DAE
- Italy/INFN
- UK/UKRI-STFC
- France/CEA, CNRS/IN2P3
- Poland/WUST

Fermilab Control System Content

- Introduction
- Motivation
- Existing Fermilab code structure
  - Integrating SLAC/LBNL software with Fermilab software infrastructure
- Proposed software restructuring
- Client software
- Summary

Introduction

- At Fermilab since 1989 as visiting scientist; joined staff in 2018
- 18+ years of experience using EPICS
- Working in Accelerator Division's Front End controls team
  - Leading development and implementation of EPICS infrastructure at Fermilab
  - Goal is to simplify deployment of IOCs by non-experts
  - Goal is to modernize the client-side tools for operators in a seamless transition
  - Use modern computing methods; e.g. Continuous Integration/Continuous Deployment (CI/CD) for code management

- Brian Chase invited me to assist the LLRF team to support SLAC/LBNL software at Fermilab
- Recognize the value of the strong inter-lab LLRF collaboration
  - My task is to integrate the existing software into our framework *without breaking it*

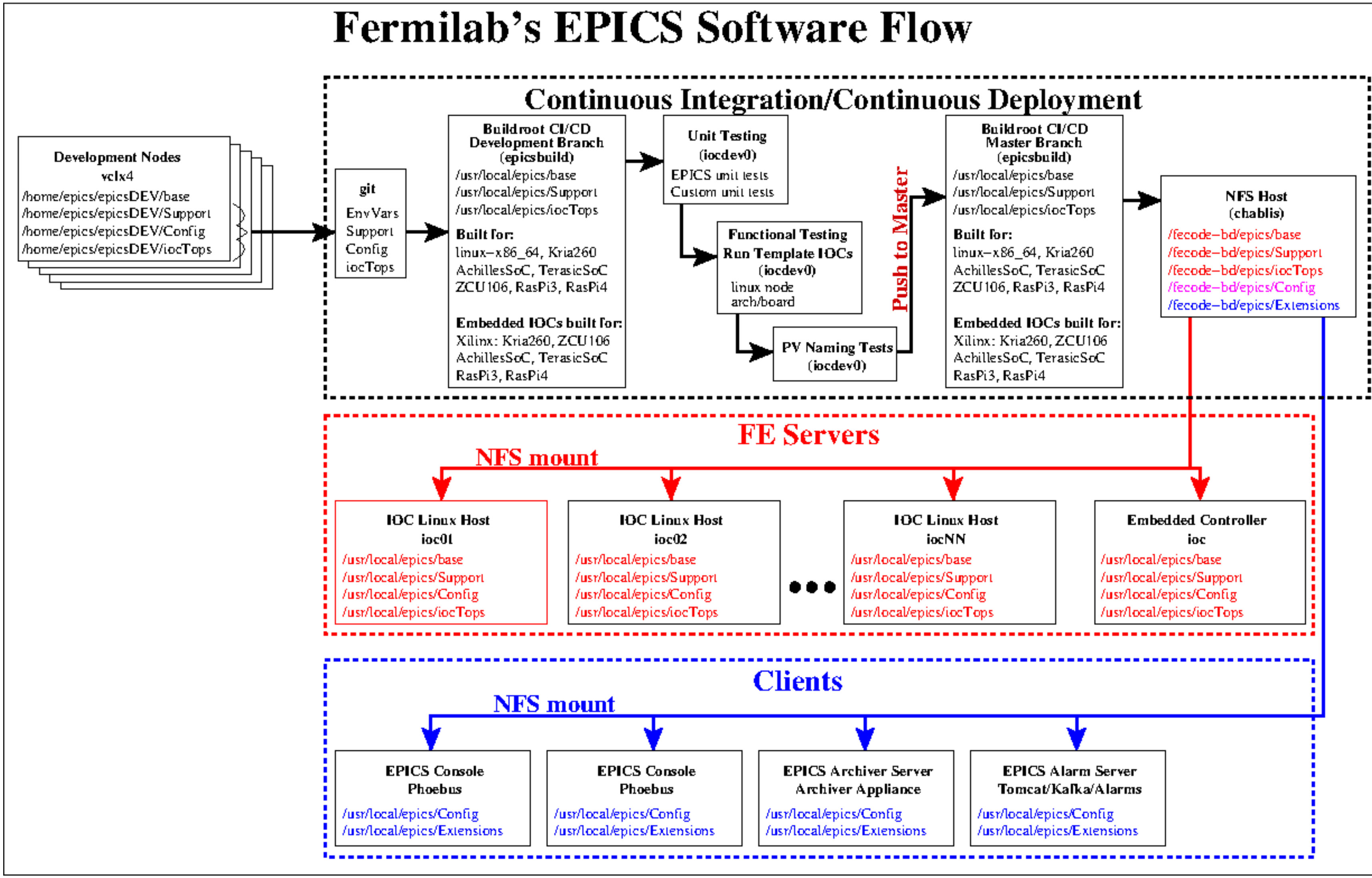
Motivation

- Treating EPICS deployment as a green field to simplify deployment for non-experts
  - https://ghe-pip2.fnal.gov/epics-controls/
- Small controls team, therefore we require:
  - robust build of infrastructure
  - automated build procedures
  - extensive testing
  - minimal functionality to automate deployment & production monitoring of IOCs
- Developed a standard EPICS infrastructure to simplify developing IOCs for new developers
  - “base” & “Support” software are built (on all supported platforms) and made available on controls network
  - developers start from template IOCs and build against production ./base and ./Support
  - template IOCs have built-in basic functionality required of all FNAL IOCs
- Standard deployment and automated build for:
  - robustness
  - ease in maintaining and debugging software
- Implement modern computing practices Continuous Integration/Continuous Deployment (CI/CD)
- Using PVXS api and disabling Channel Access (CA)
  - pvAccess protocol → structured data
  - already has ipV6 and multicast
  - new network security measures (zero-trust) are being implemented
- Client side
  - Update GUIs
  - Provide Save & Restore functionality
  - Provide data archiver
  - Provide alarms
  - Provide channel finder
- Strong collaboration, any changes in structure must remain in line with LLRF collaboration

FNAL EPICS code structure

- Structure follows conventional EPICS implementations
- The Fermilab “standard deployment” of EPICS IOC code assumes a 3-tier build:
  - EPICS base – main core of EPICS, comprising the build system and tools, common and OS-specific interface libraries, Channel Access and PV Access client and server libraries, static and run-time database access routines, the database processing code, and standard record, device, and driver support. Production code resides in ***/usr/local/epics/base***
  - EPICS Support – contains modules which are analogous to drivers one might add to the kernel for a computer to run specific functions and/or hardware drivers. We presently support ~50 support modules and expect this to grow. Production code resides in **/usr/local/epics/Support**
  - EPICS IOCs (**I**nput/**O**utput **C**ontroller) – specific front end servers for controls and monitoring; these are built by pulling in Support modules and adding application specific code. A template IOC is provided to developers and already has minimal Fermilab required functionality. Production code resides in **/usr/local/epics/iocTops**
- EPICS base, Support, & iocTops are built for different architectures/platforms
- EPICS base, Support, & iocTops are hosted by NFS server
- Code base is built and tested in Continuous Integration/Continuous Deployment pipeline
- Goal: robust EPICS code base – simplify IOC development for novices and to simplify expert debugging
- Builds exist for linux-x86_64, arm: Cyclone V, Arria-10, RasPi3, RasPi4, Kria260, ZCU106 (Xilinx)

Fermilab code structure – path to deployment



DOE Laboratories in LLRF Collaboration

LLRF Teams from FNAL, JLab, SLAC and LBNL have been collaborating for the past 6 years in the context of LCLS-II and now PIP-II

BERKELEY LAB

Fermilab

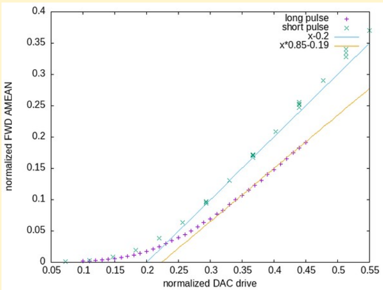
SLAC

Jefferson Lab

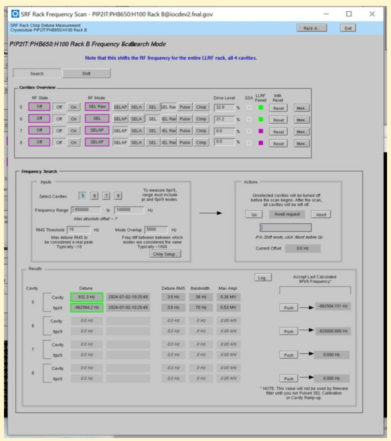
Successful collaboration for LCLS-II and after 9 years we continue to want to work together

SLAC/LBNL Software – pHB650 testing @ PIP2IT

calibrations

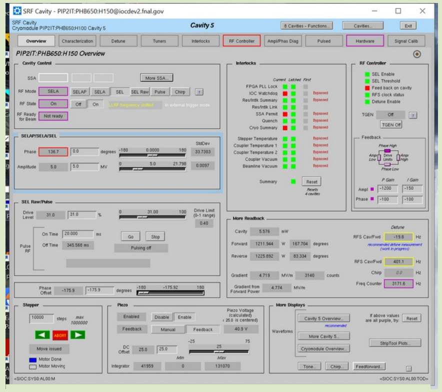


SSA

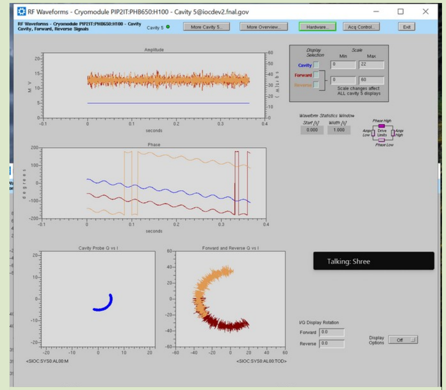


Frequency Tuning

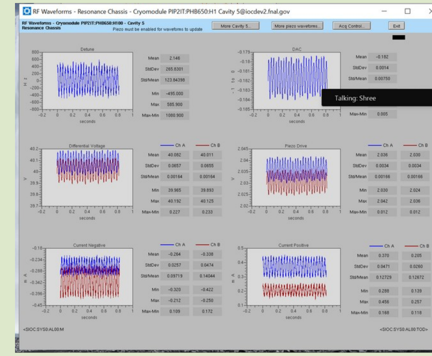
SELA Mode



Settings

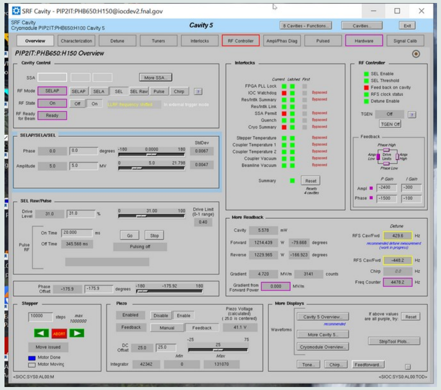


5MV/m

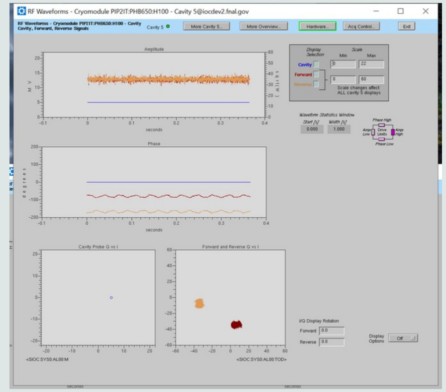


piezo waveforms

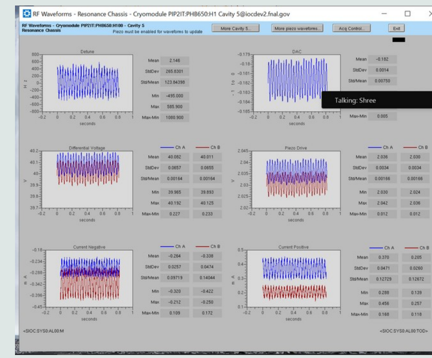
SELAP Mode



Settings



7MV/m



piezo waveforms

SLAC/LBNL Existing Software

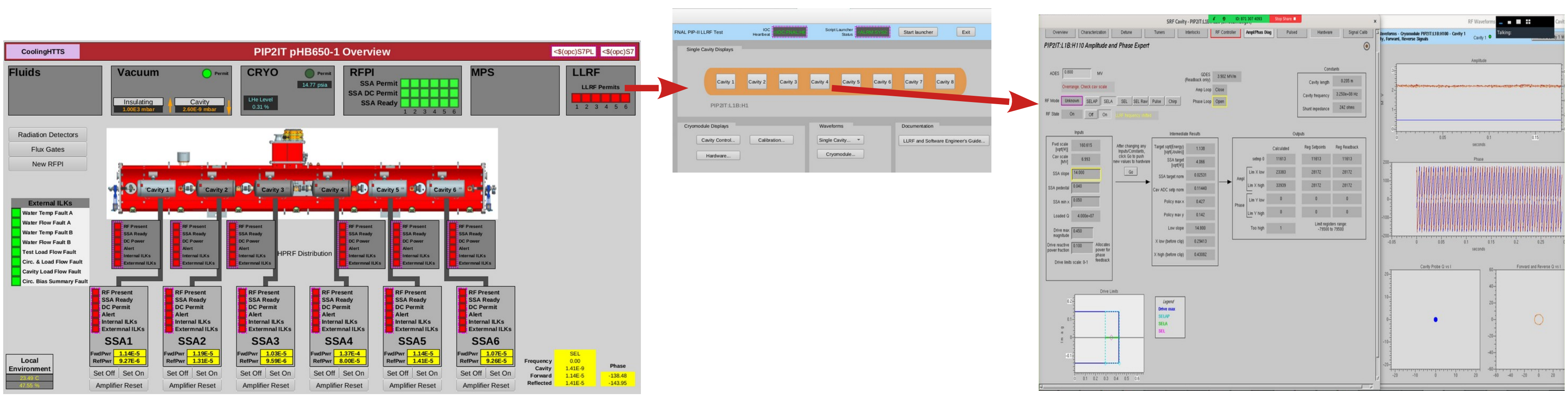
- SLAC/LBNL software is stable and mature
- Used at SLAC, LBNL, FNAL, and JLab
- Code is well documented
- Code base is appropriately versioned in git
- Most of the heavy lifting is done in python scripts – EPICS mostly serves as user interface & monitoring
- > 22k PVs for an 8 cavity cryomodule

Challenges for merging existing software within Fermilab infrastructure:

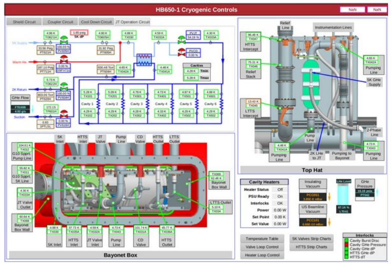
- Developed over long time, features added as needed, using different standards
  - Code is downloaded as two monoliths
  - Uses specific (older) version of EPICS base => cannot take advantage of EPICS v7 features
  - Different laboratories have hardware configurations which don't match LCLS-II requirements
  - Scripts and configuration files in a variety of sub-directories
  - Macro substitution is performed at run time
  - Structure presently incompatible with Fermilab deployment model
- Proposed code restructuring considerations:
  - Developed plan with Sonya Hoobler (SLAC) to restructure code
  - Code will be re-factored to have Common, LCLS-II specific, FNAL specific elements, etc.
  - Contract in place with Osprey Distributed Control Systems to perform refactoring
  - It is critical to maintain compatibility between labs
  - Restructure location of software, *not* change software
  - Existing code makes use of environment variables which will simplify restructuring
  - Include modernization of existing GUIs => convert EDM files to Phoebus bob files
- Have successfully built existing software using FNAL structure

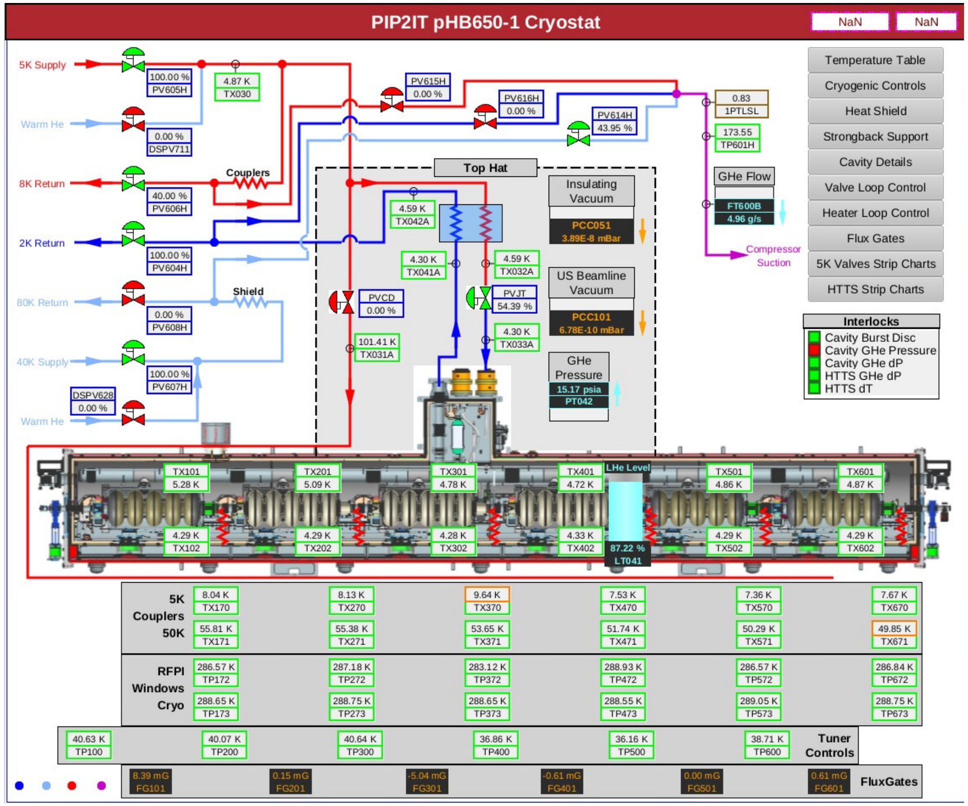
Client Side Support

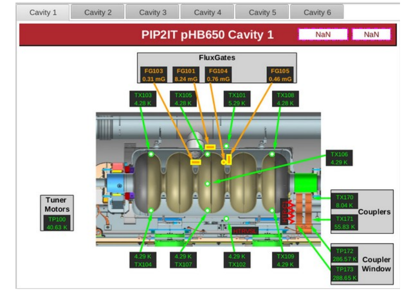
- Fermilab Controls team supports the following clients:
  - Archiver Appliance – operational at PIP2IT
  - Kafka Alarm server – operational at PIP2IT
  - Channel Finder – operational at PIP2IT
  - Save & Restore – in place and used for much of PIP2IT LLRF
  - Convert edm → Phoebus – in progress
    - Significant project as there are >100 screens
    - Phoebus has conversion capability, but not all features are automatically ported

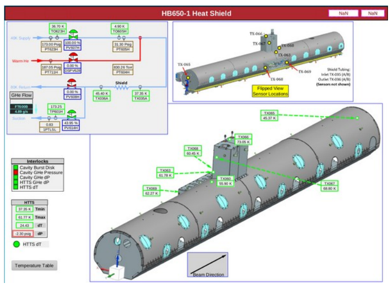


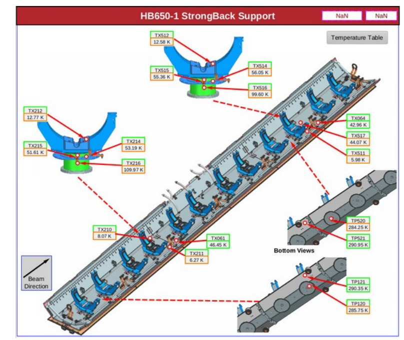
Client Side Support

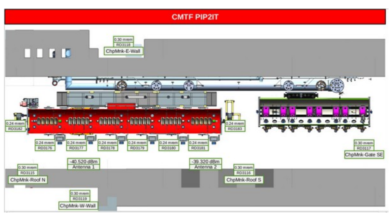


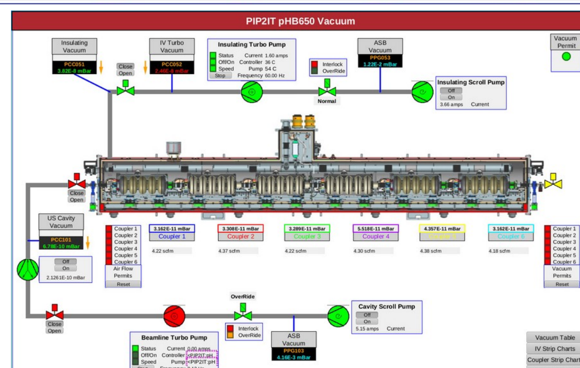


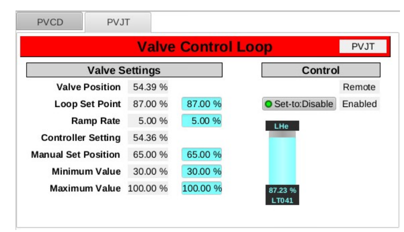


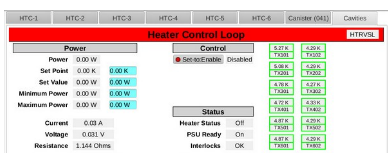








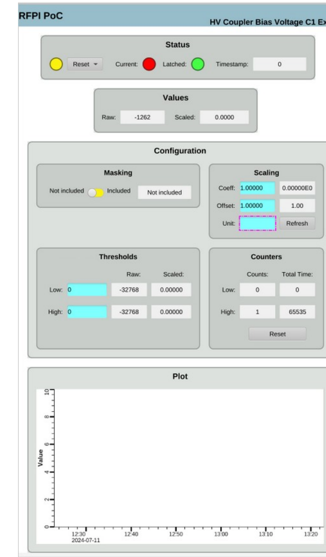




PIP-II

RFPI Final Prototype Testing

4 Cavity RFPI Module



- New module was installed last month at CMTF test stand
- All I/O were individually tested
- Testing is continuing with remote participation of LUT

Summary

- Standard SLAC/LBNL implementations are running at Fermilab in CMTS, PIP2IT, and LLRF test stands
- Strategy for creating a robust EPICS deployment at Fermilab is in place
- Strategy for modifying existing SLAC/LBNL code structure to fit diverse configurations is agreed upon
- Contract with Osprey for code refactoring in the the works
- Updating clients for PIP-II in progress