



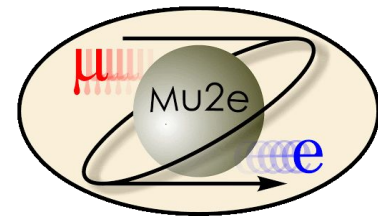
Trigger-DAQ and slow control systems in the Mu2e experiment

Antonio Gioiosa

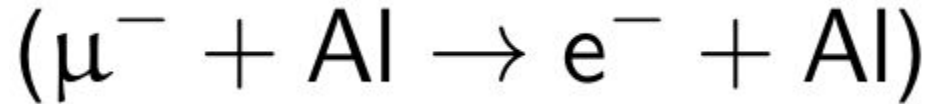
Università di Pisa, INFN Pisa

TIPP 2021

May 25, 2021

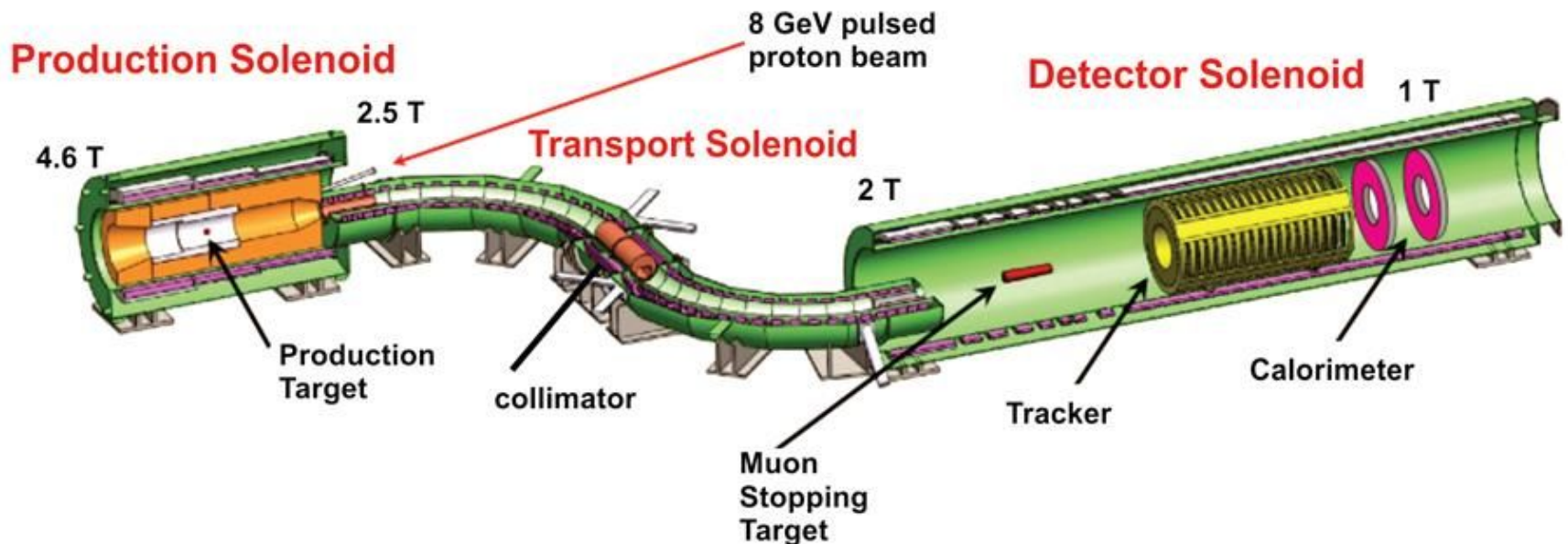


The Mu2e Experiment at Fermilab



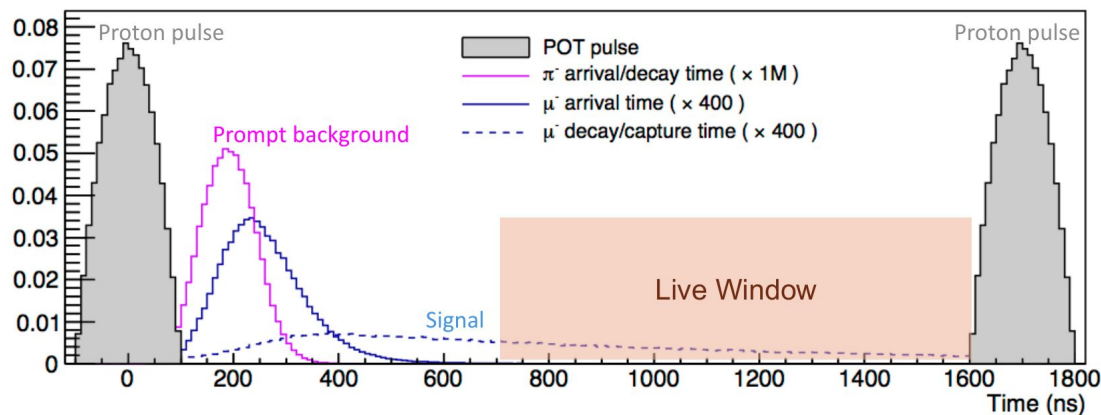
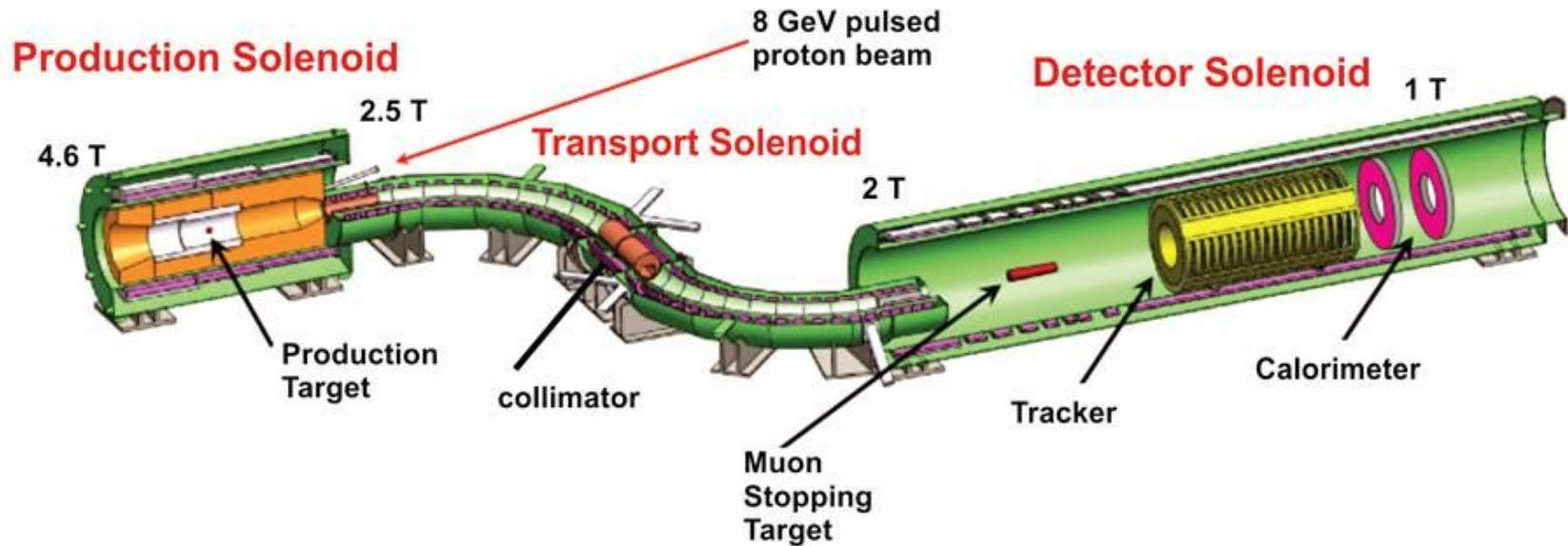
With the expected experimental sensitivity, Mu2e will improve the SINDRUM II limit ($7.0 \cdot 10^{-13}$) of four orders of magnitude

(Mu2e intends to reach a single event sensitivity of $3.0 \cdot 10^{-17}$, assuming we will run for three years, with $3.6 \cdot 10^{20}$ protons, with a run time of $6.0 \cdot 10^7$ s, requiring a background level below 1 event)



The Mu2e Experiment at Fermilab

The signal we are looking for is a delayed monoenergetic electron with an energy of just under 105 MeV (muon mass)

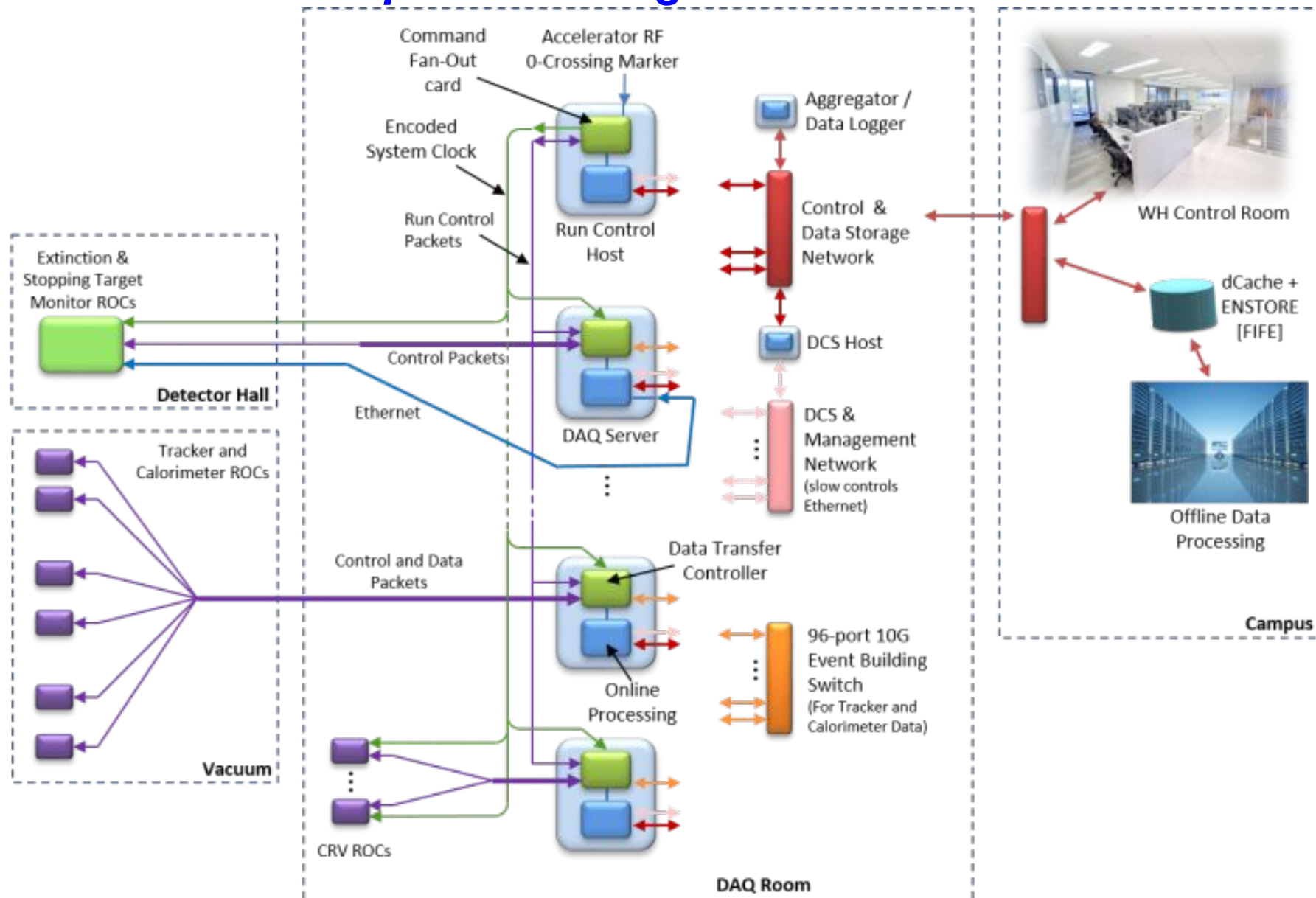


Mu2e TDAQ and Slow Control integration

Summary:

- Mu2e TDAQ components Diagram
- Mu2e Timing Distribution
- Mu2e TDAQ Readout scheme
- Online DAQ (***otsdaq***) overview
- Slow control and its integration in ***otsdaq***
 - **Monitoring** and Slow Controls GUI
 - Slow Controls **Integration** with ***otsdaq*** State Machine and Alarm handling
- Conclusions

Mu2e TDAQ components Diagram

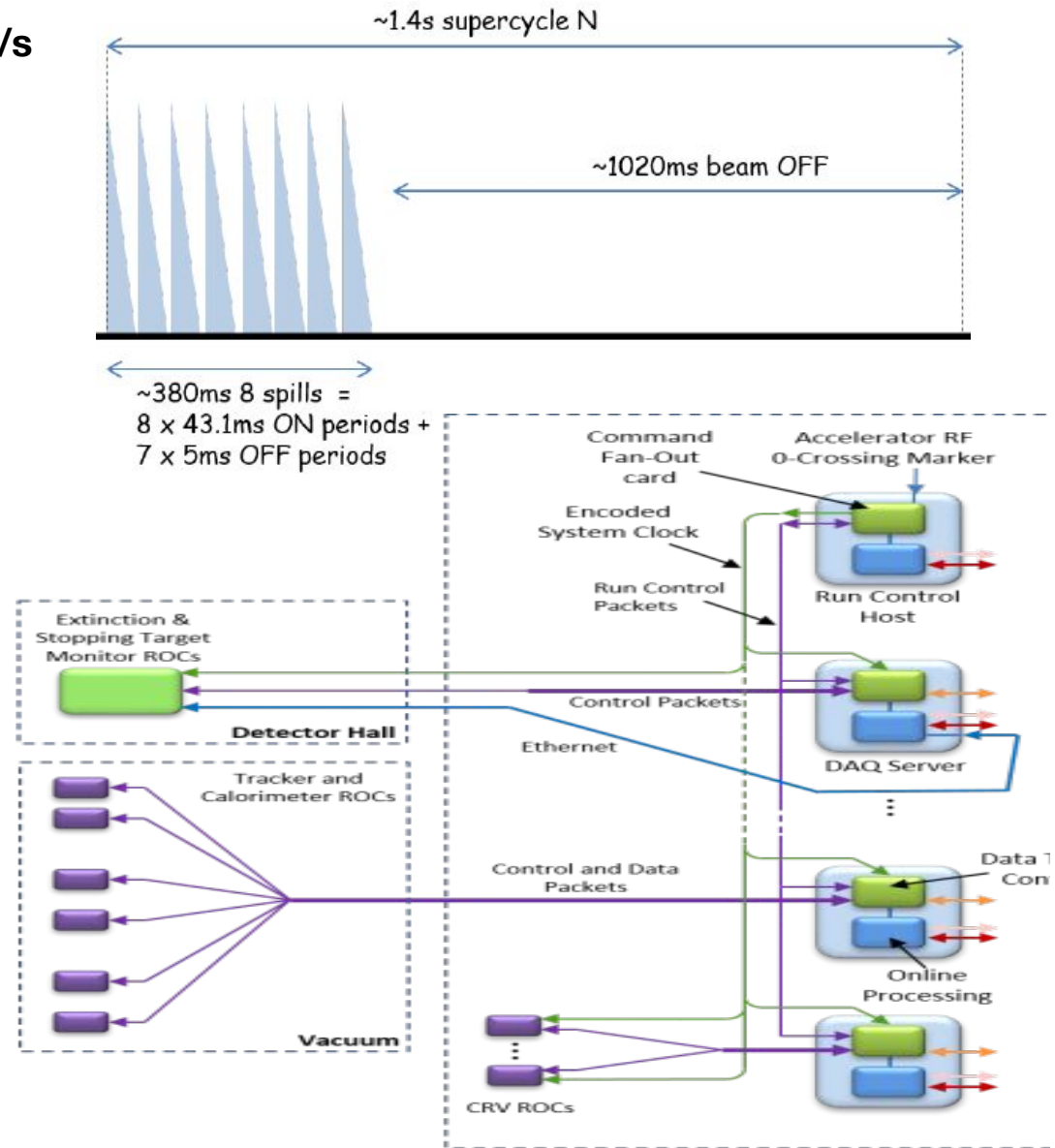


Mu2e Timing Distribution

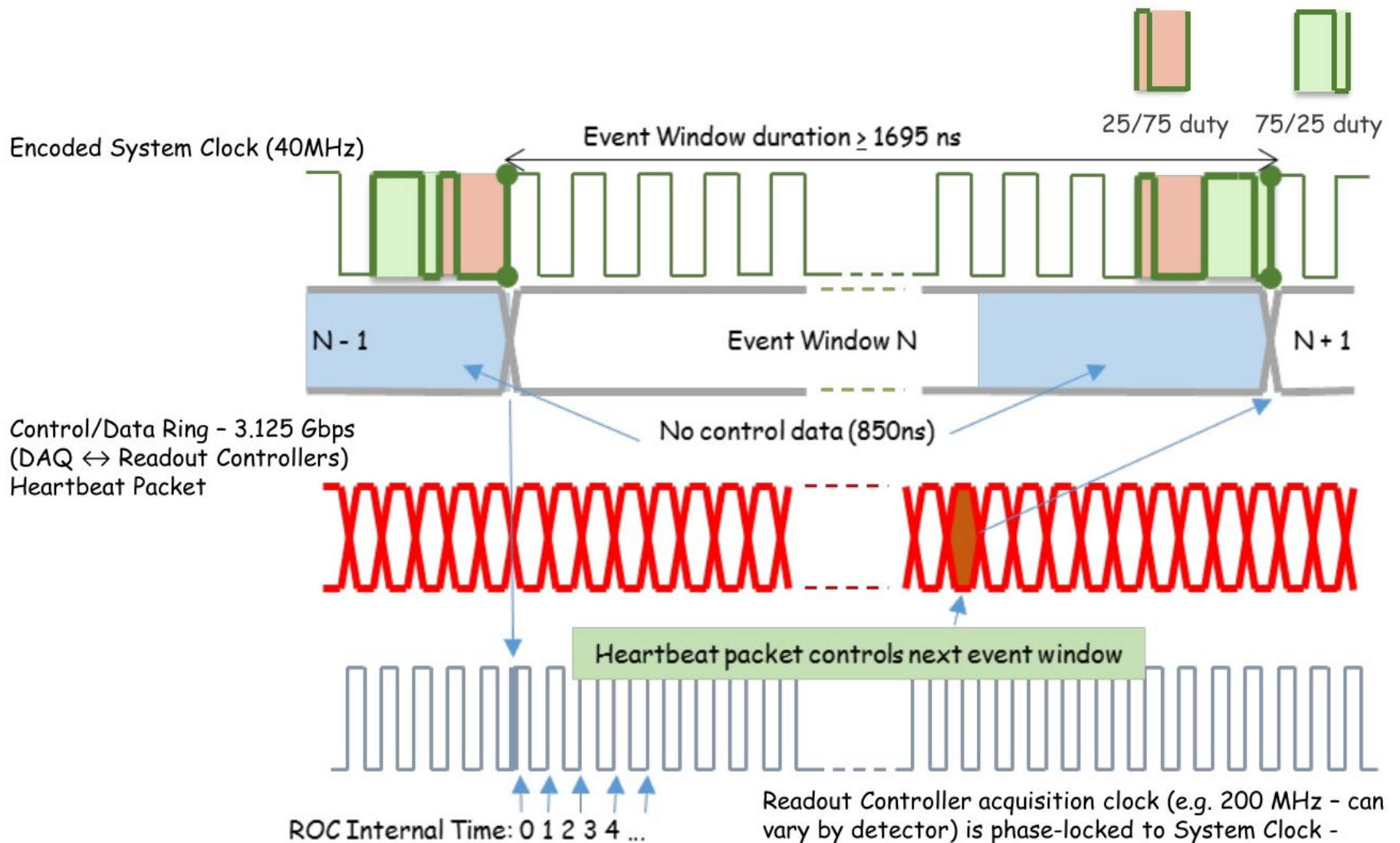
Requirement is to process 200K events/s

- Mu2e Runs are broken up into contiguous Event Windows
- Experiment defined Run Plan is coordinated by the Command Fan-Out Card (CFO)
- The System Clock (40MHz) and Event Window markers originate at the CFO ...and are distributed to ROCs:

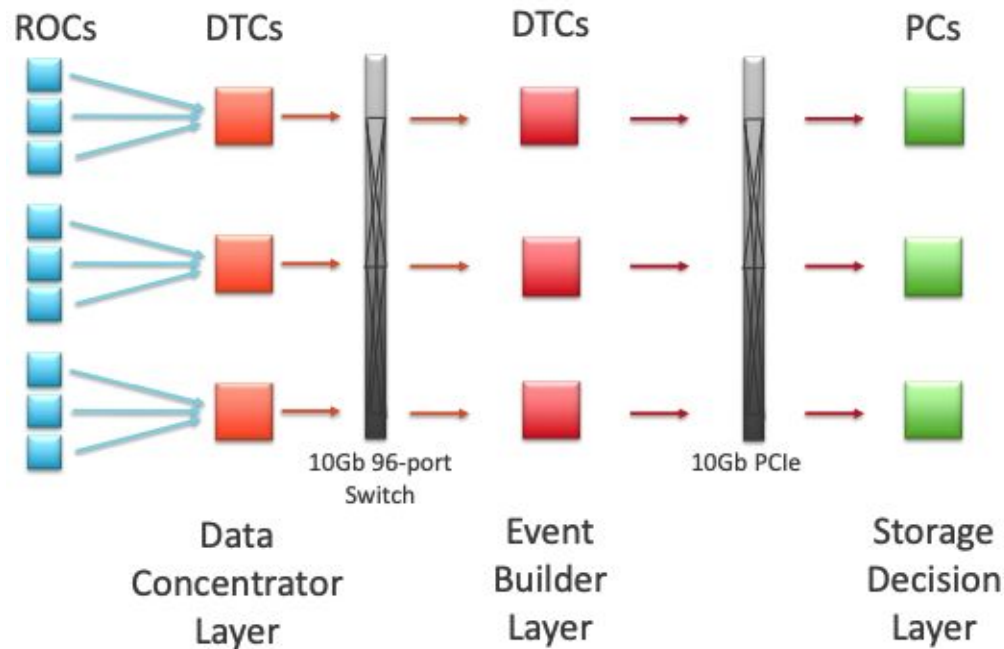
1. CFO distributes System Clock and Event Windows to DTCs with fixed latency
2. DTCs distribute System Clock and Event Windows to ROCs with fixed latency
3. ROCs respond to Data Requests



Mu2e Timing Distribution



TDAQ Readout scheme



- 396 ROCs 69 DTCs (Kintex-7) for data readout and event building
- Large front end buffers to average over long off-spill time
- 800 threads on 40 nodes for HLT → ~5 ms per event
- ~40 GB/s data read out to storage decision layer, ~280 MB/s written to disk

High Level Trigger Software



Mu2e Online DAQ solution: *otsdaq*



otsdaq overview

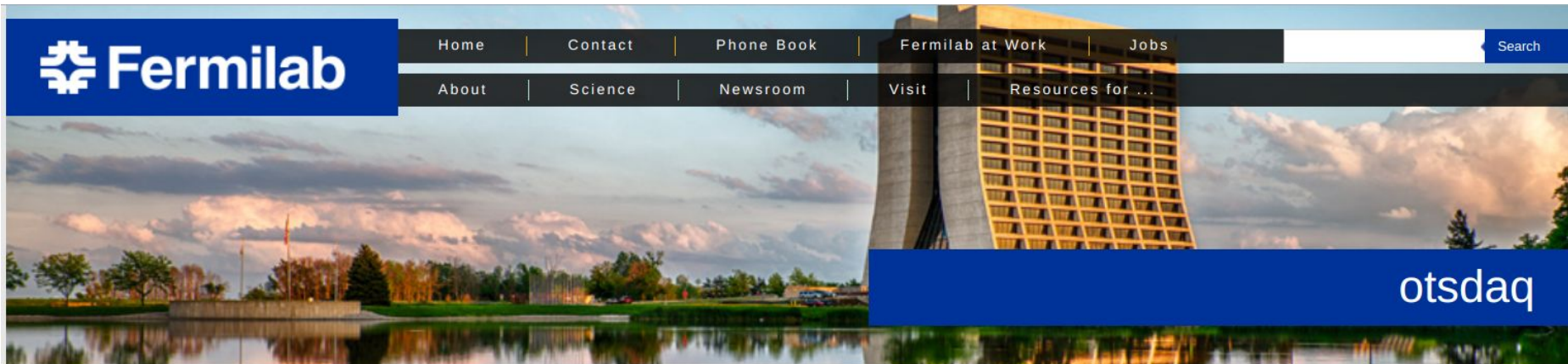
Acronym for “off-the-shelf data acquisition.”

- ***otsdaq*** is a Ready-to-Use data-acquisition (DAQ) solution aimed at test-beam, detector development, and other rapid-deployment scenarios
- it uses the ***artdaq*** DAQ framework under-the-hood, providing flexibility and scalability to meet evolving DAQ needs
- ***otsdaq*** provides a library of supported front-end boards and firmware modules which implement a custom UDP protocol
- Developments are in two directions: **server** side and **web** side.
- An integrated Run Control GUI and readout software are provided, preconfigured to communicate with ***otsdaq*** firmware

otsdaq overview



More info at **otsdaq** web page <https://otsdaq.fnal.gov/>



otsdaq

Project Homepage

Source Code Documentation

User Manual

Tutorials (User/Expert Training)

"First Demo" tutorial



otsdaq is a Ready-to-Use data-acquisition (DAQ) solution aimed at test-beam, detector development, and other rapid-deployment scenarios. *otsdaq* uses the *artdaq* DAQ framework under-the-hood, providing flexibility and scalability to meet evolving DAQ needs. *otsdaq* provides a library of supported front-end boards and firmware modules which implement a custom UDP protocol. Additionally, an integrated Run Control GUI and readout software are provided, preconfigured to communicate with *otsdaq* firmware.

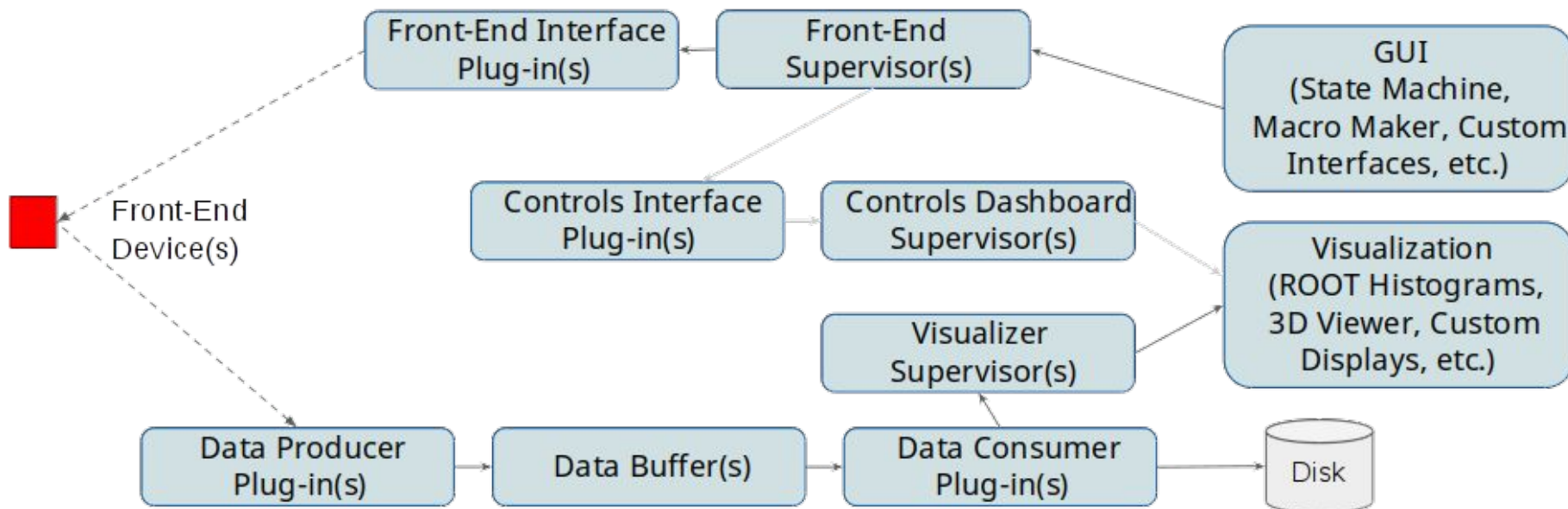
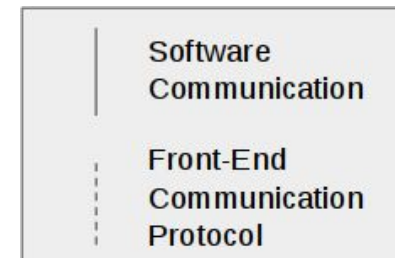
Last modified: 04/29/20 | email Fermilab

otsdaq overview



Data Flow Block Diagram

Server side is C++. User code is added through plugins (C++ classes inheriting from the appropriate class)

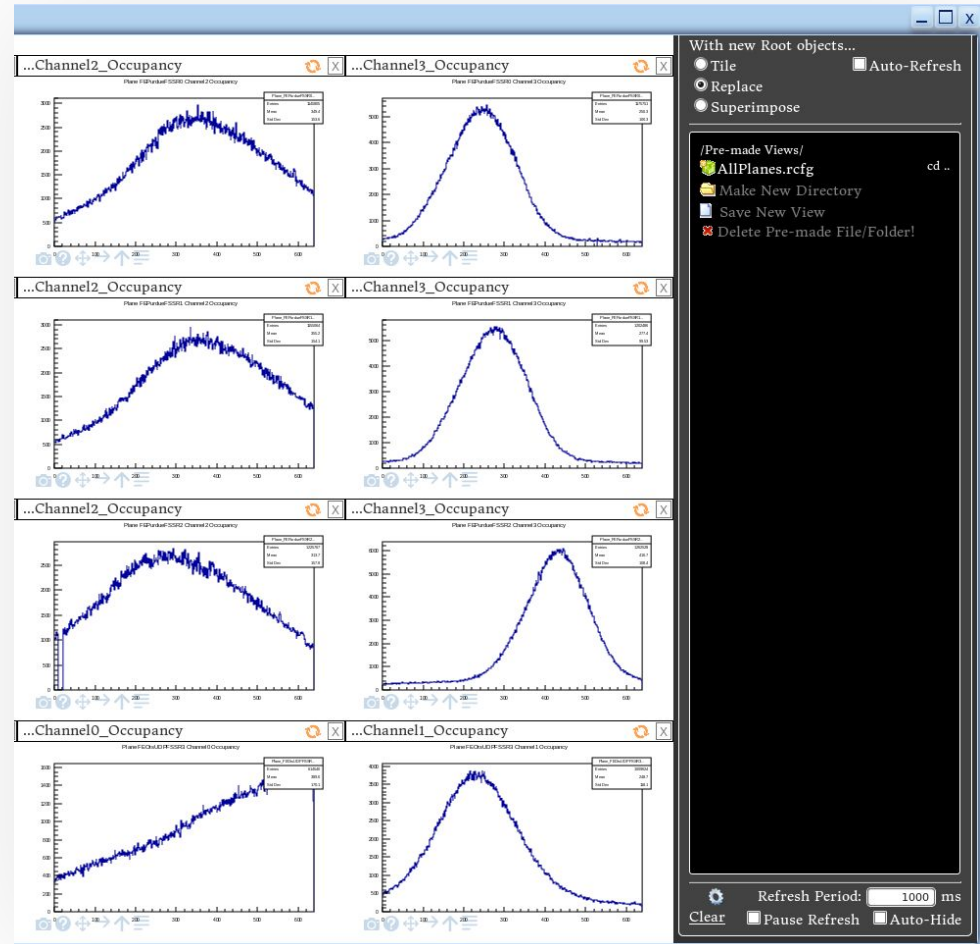


Web side is HTML and JavaScript. User code is added in the form of web-apps through .html files (including the appropriate .js and .css files)

otsdaq overview

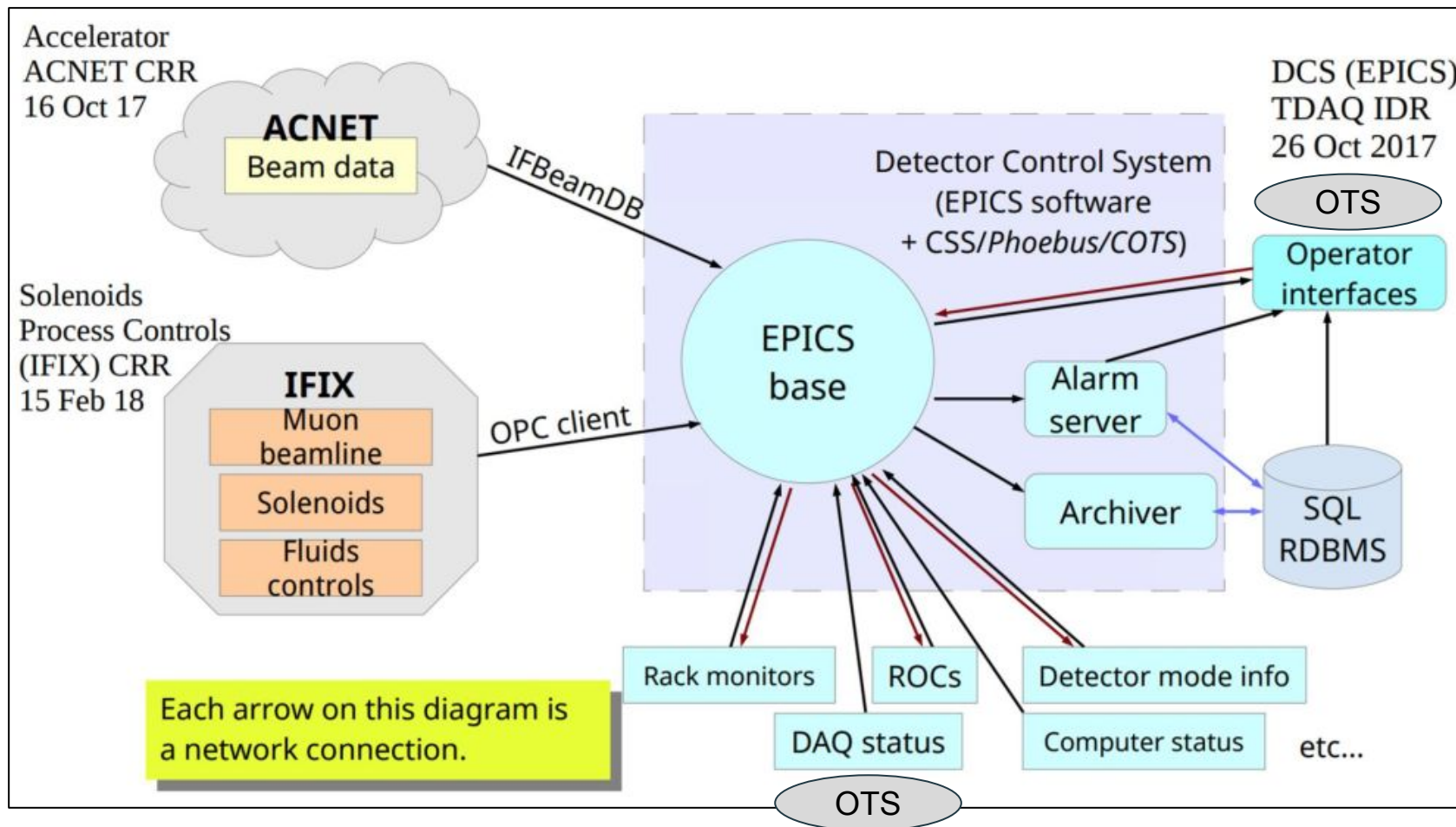
Data processing: Data Quality monitor GUI example

- Mu2e's event window data will be processed through artdaq modules
- Data processor and Data Quality Monitor **DQM** plugins are provided by otsdaq core
- **DQM** generates data products that are sent to an **artdaq Dispatcher**, which aggregates **DQM metrics** and presents them to a visualizer application

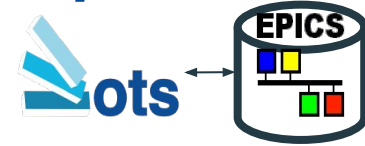


Slow Controls connection and **EPICS** plugin development in *otsdaq*

Experimental Physics and Industrial Control System



Slow Controls connection and **EPICS** plugin development in **otsdaq**



Channel subscription to **EPICS** uses Input Output Controller (**IOC**)

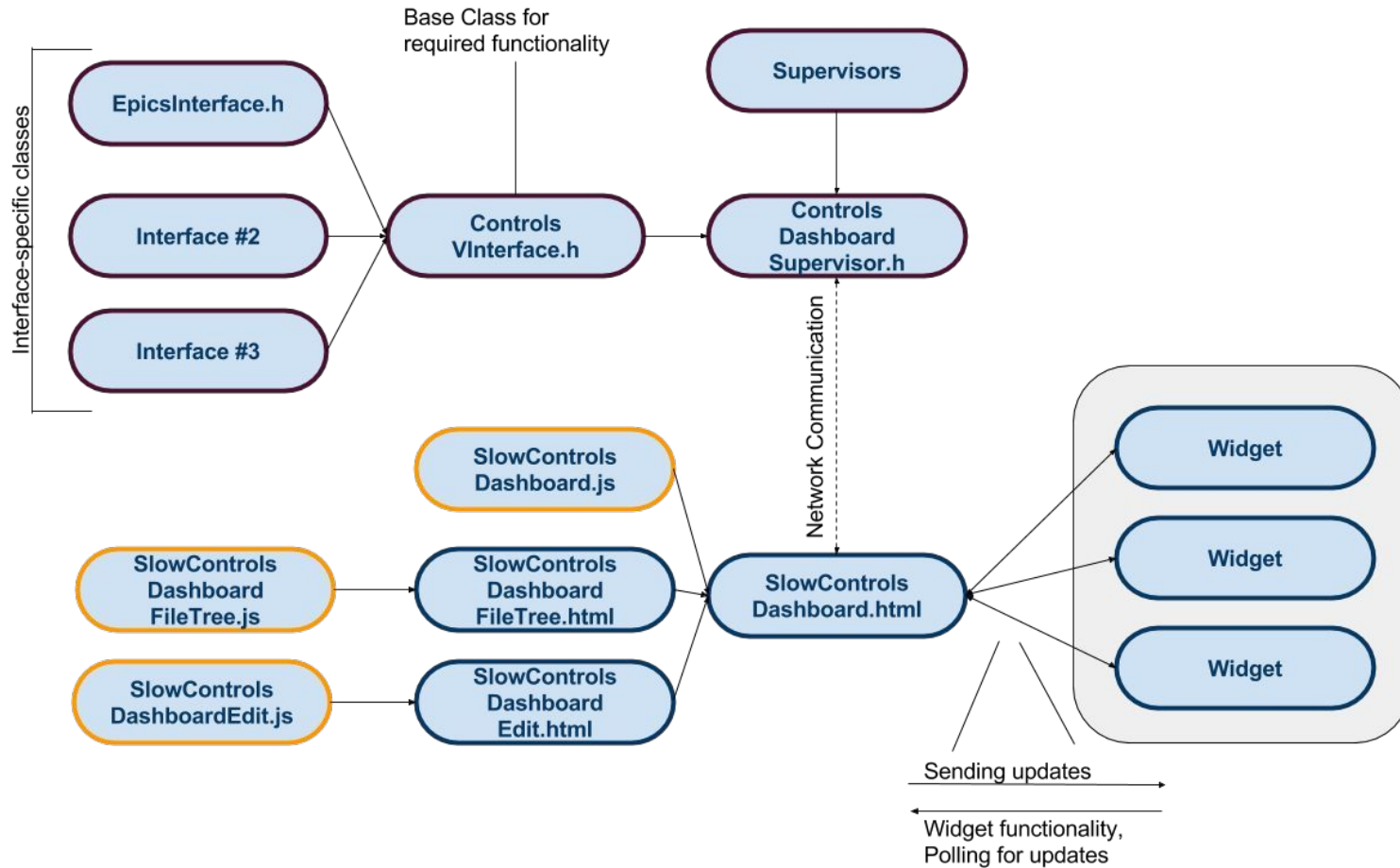
- integration of slow control in the online daq uses same Interface plugin for:
 - a. Monitoring of all mu2e slow control channels
 - b. Sending Process Variables (PVs) of DAQ hardware info as **EPICS** channels and PVs settings into **EPICS** databases
- The Interface plugin:
 - a. Performs channel subscription to **EPICS** using Channel Access **EPICS** C++ libraries to send and retrieve slow control data information like: Value, Alarm (Status, Severity), Settings
 - b. Uses Postgres database C++ libraries to set channels and retrieve channels and alarms histories from **EPICS** databases

Slow Controls Monitoring in otsdaq



— C++
— HTML
— Javascript

Slow Controls GUI Hierarchy

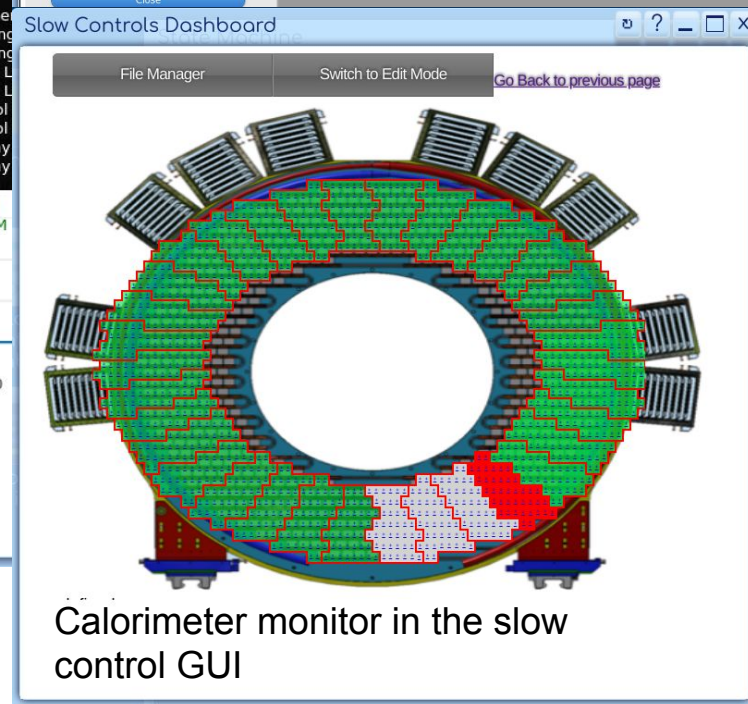
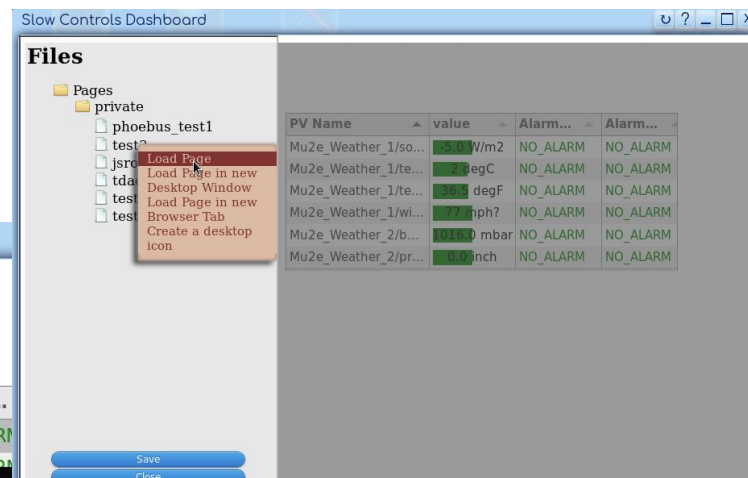
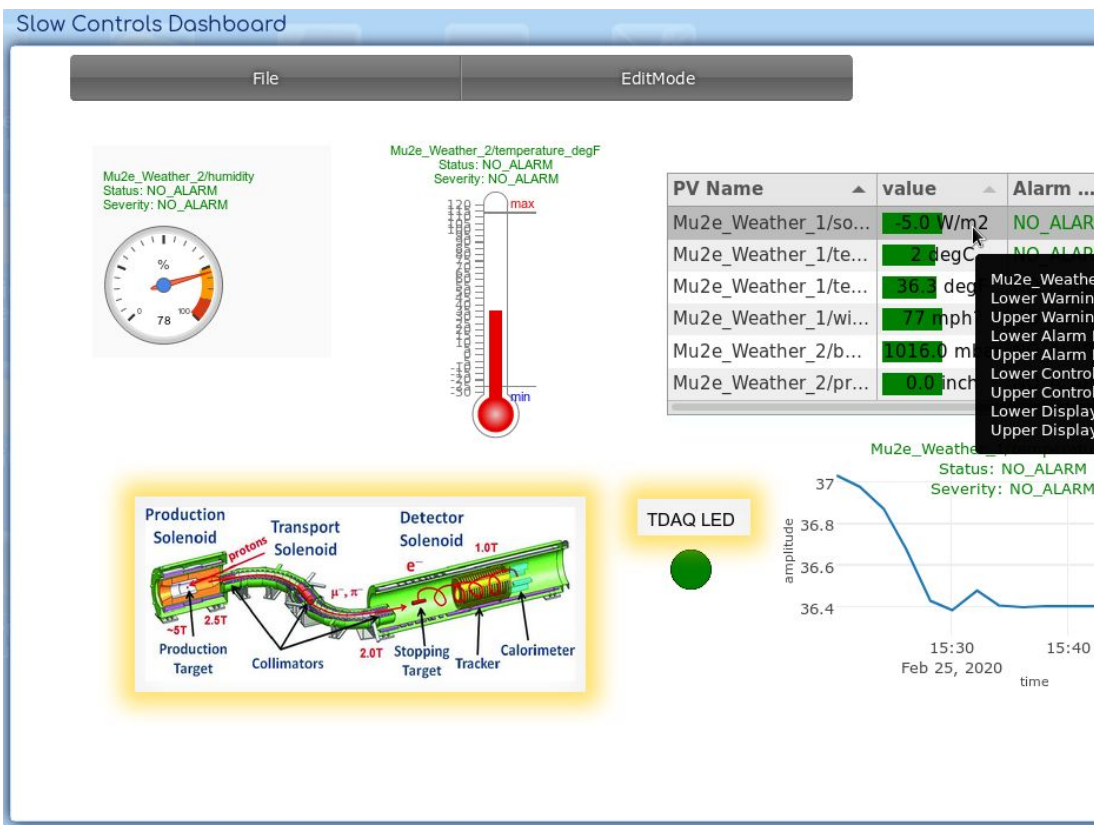


Slow Controls Monitoring GUI in otsdaq

Example of page loading

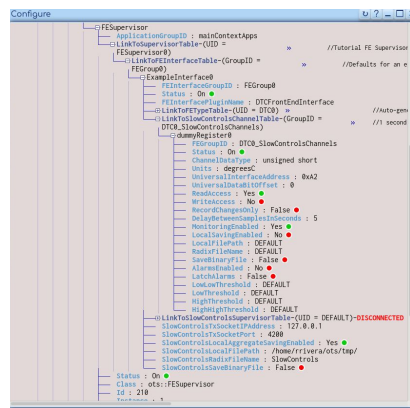
Examples

Example of loaded page

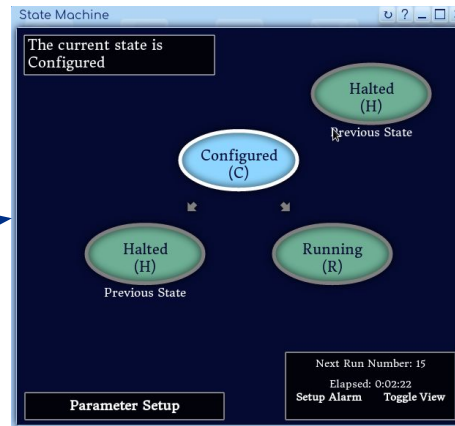


Integration with State Machine

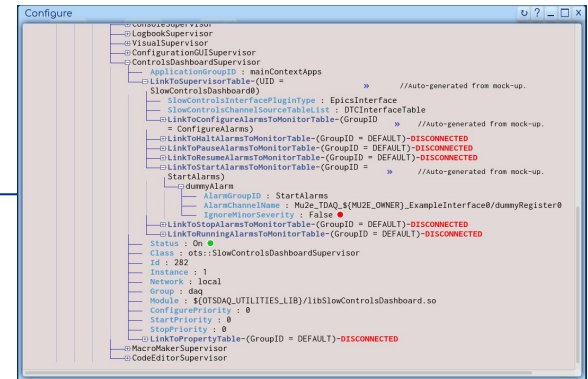
- **State Machine** Configuration and data subscription to **EPICS**
- Alarm propagation (from **EPICS**) and **otsdaq** State Machine handling
DAQ HW, artdaq and DQM metrics configuration



artdaq **EPICS** metrics
Plugin



Alarm Configuration



otsdaq **EPICS** Plugin

EPICS

Conclusions



- Mu2e Experiment is under construction at Fermilab and will be ready for data taking in two/three years
- Mu2e TDAQ and slow control are in large part developed according to the requirements (200K events/s for data taking) and hardware tests are going on
- Slow control integration in the online DAQ system, *otsdaq*, provides an advanced slow controls monitoring, an interface to send *otsdaq* front-end DAQ hardware, data processing, and DQM slow controls information to **EPICS**, and a real configuration and Integration with the *otsdaq* State Machine

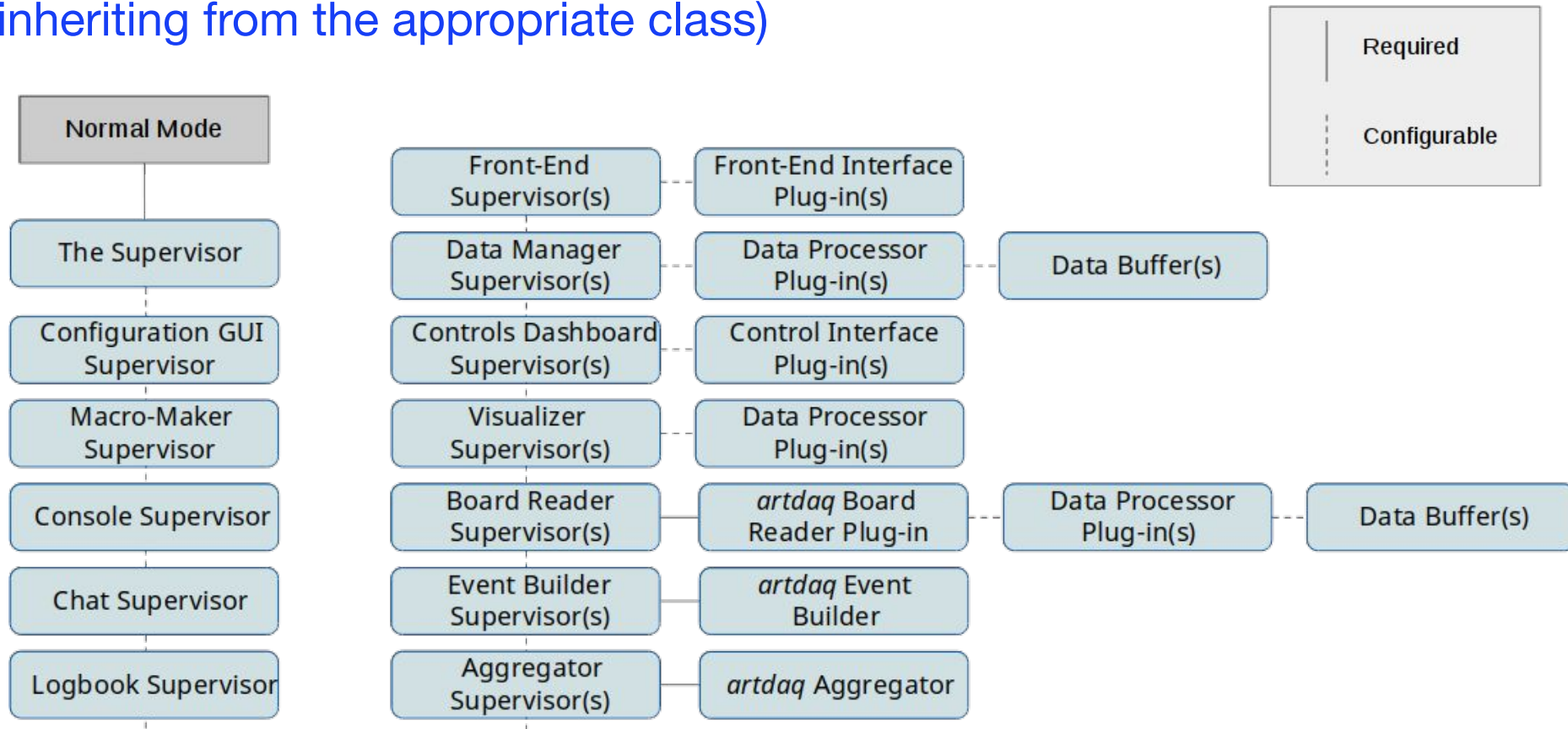
This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement no 734303, 822185, 858199, 101003460

Backup Slides

otsdaq overview



Server side is C++. User code is added through plugins (C++ classes inheriting from the appropriate class)

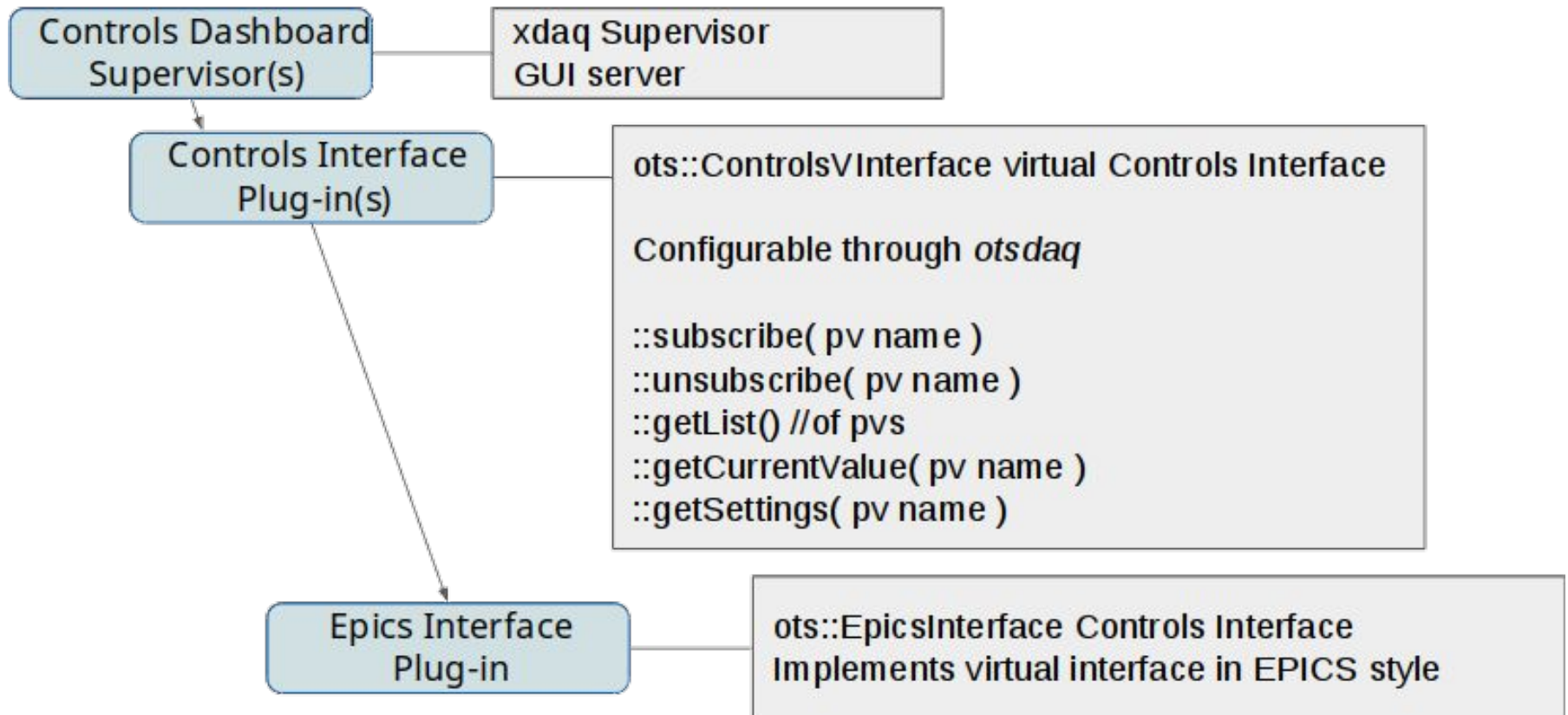


Web side is HTML and JavaScript. User code is added in the form of web-apps through .html files (including the appropriate .js and .css files)

Slow Controls Monitoring in otsdaq



Slow Controls C++ Hierarchy

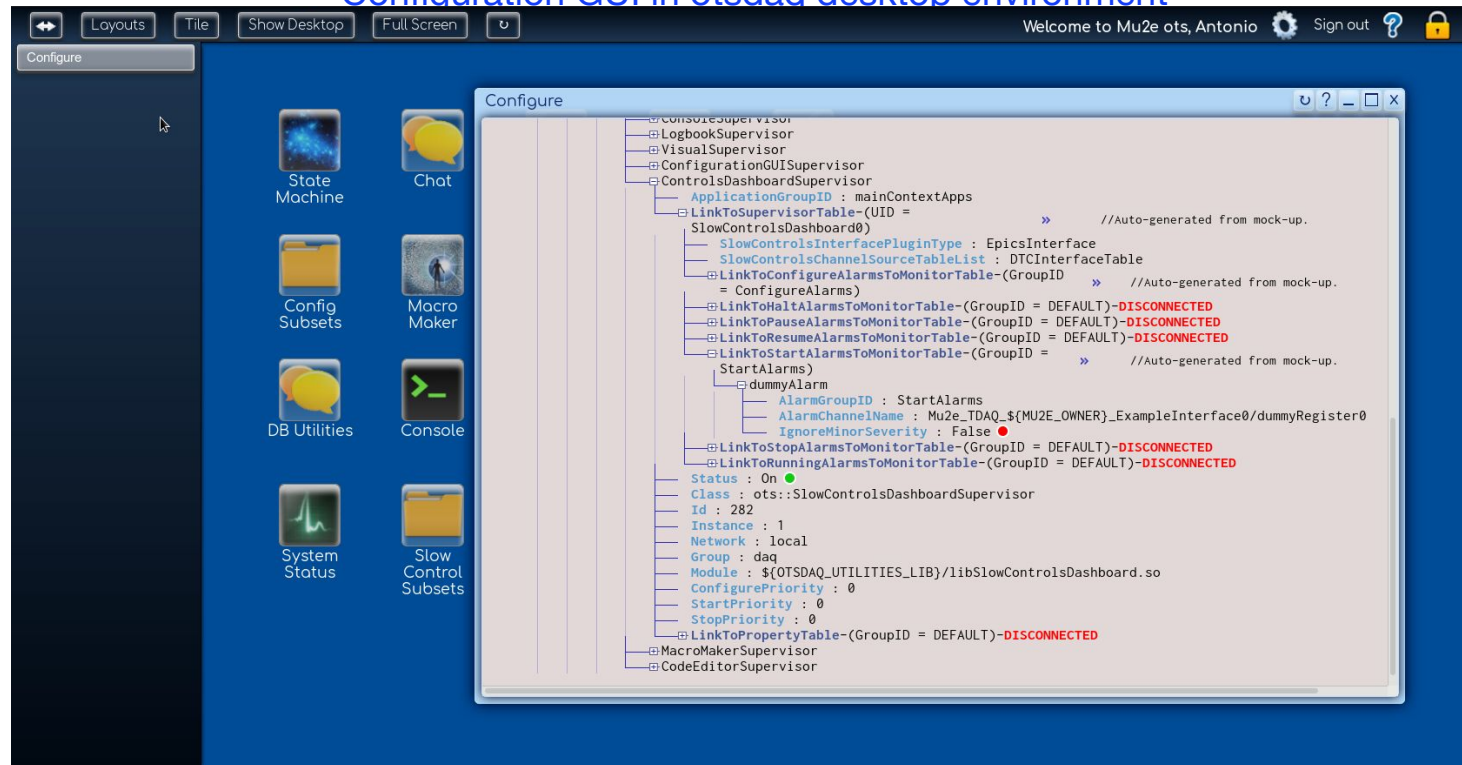


Slow Controls Monitoring in otsdaq

Configuring by specific tables in otsdaq

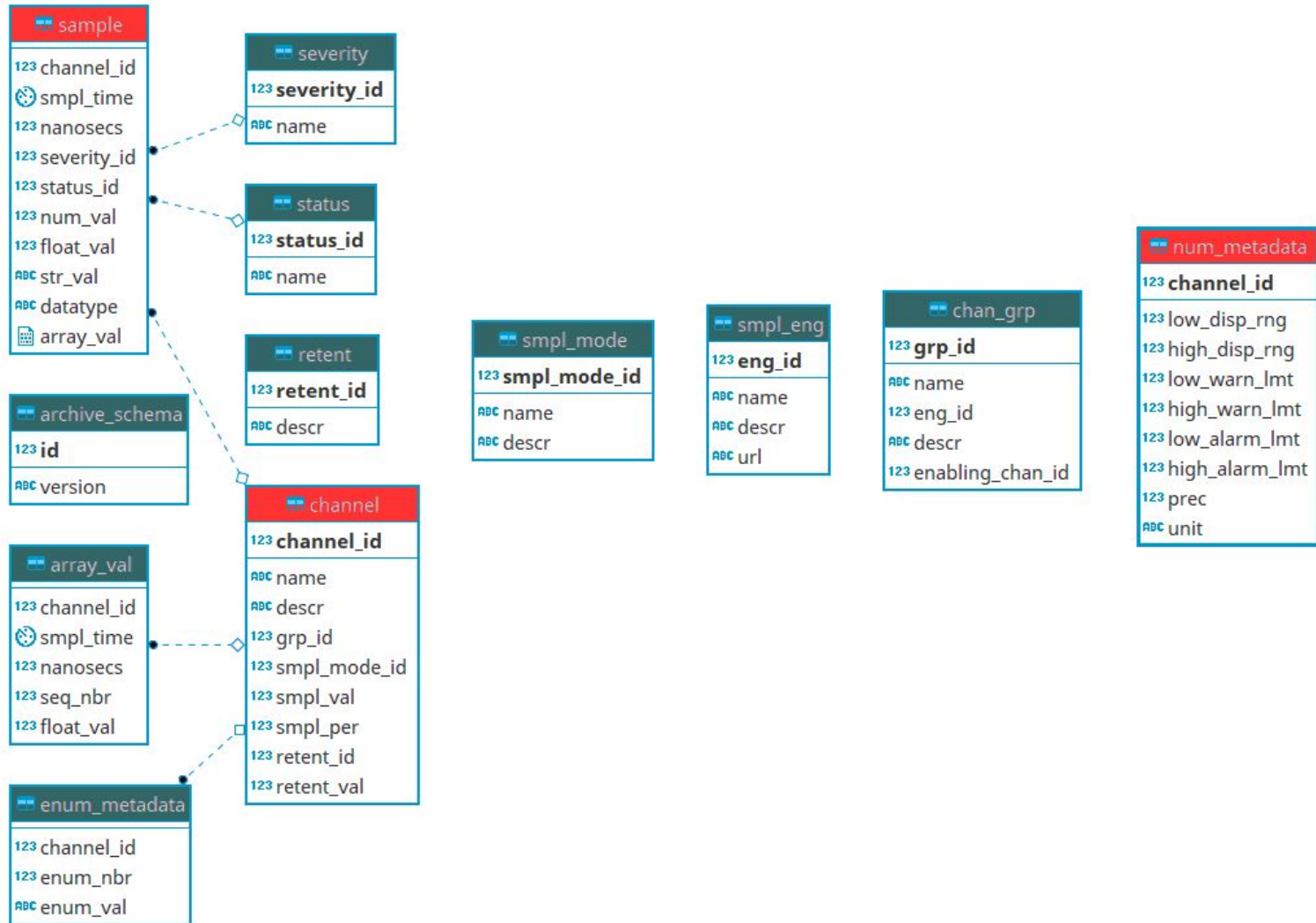
DesktopIconTable, XDAQApplicationPropertyTable, XDAQApplicationTable, XDAQContextTable

Configuration GUI in otsdaq desktop environment



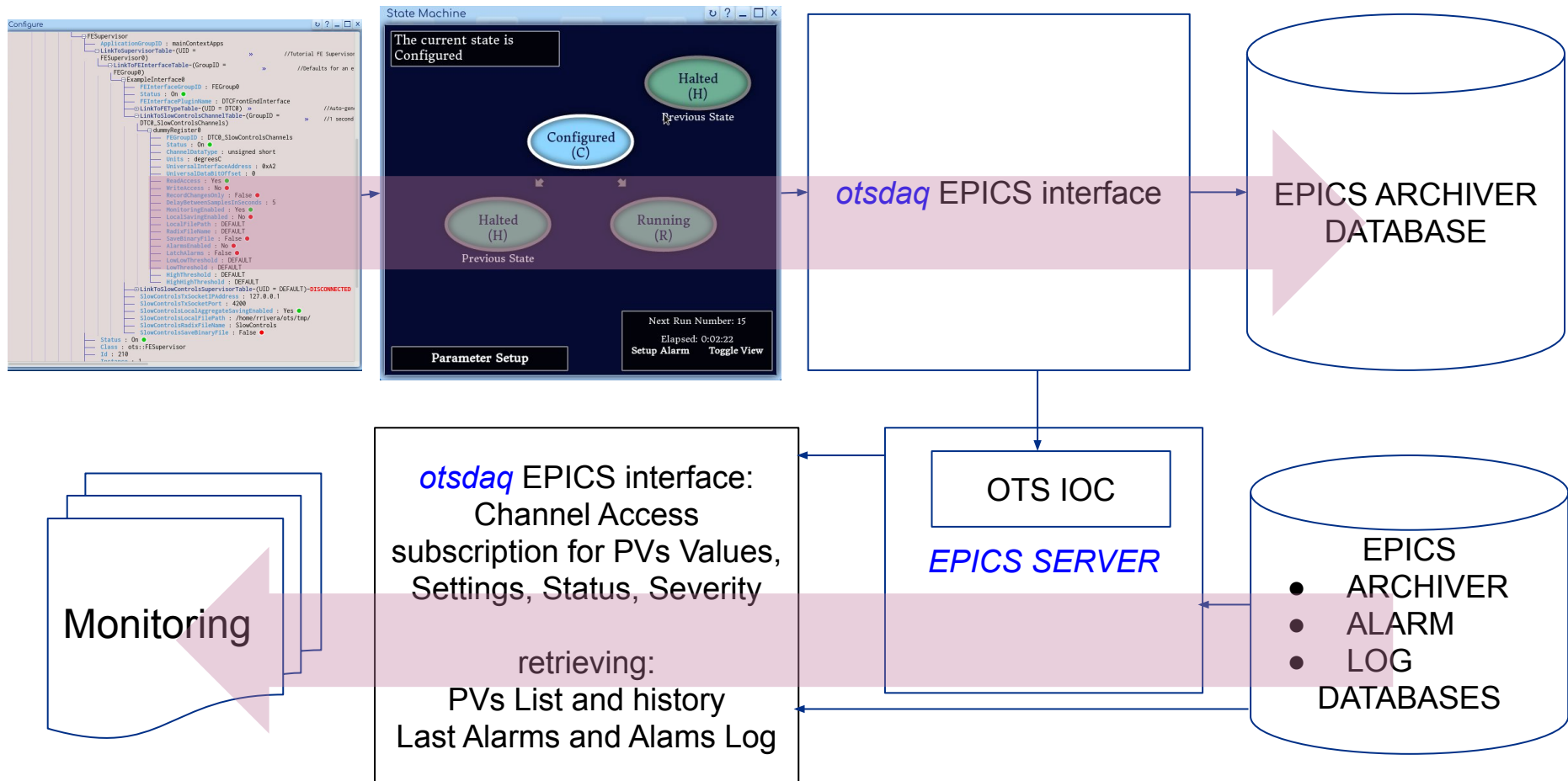
EPICS Database

- Postgres DBMS



Integration with State Machine

- *otsdaq* FE (DTC/ROC/CFO) / *artdaq* metric new channel or new slow control setting → configuring State Machine → EPICS DBs and IOC configuration
- *otsdaq* Interface → *otsdaq* CA subscription and DBs select → Monitoring



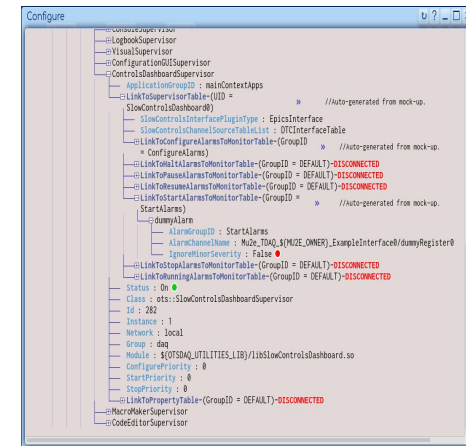
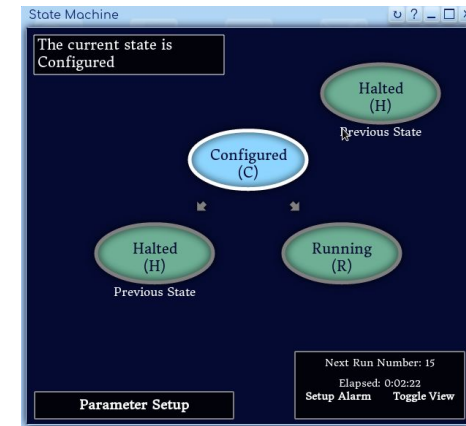
*Integration of **otsdaq** front-end DAQ hardware and artdaq metrics with **EPICS***

Actions designed and developed in *otsdaq*

1. *otsdaq* DCS channels Front End and tables configuration
2. *otsdaq* State Machine configuration implementation
3. add/update channels info for **IOC** and **Archiver** DB
4. software **IOC** restarting
5. **EPICS Archiver** restarting
6. new *otsdaq* epics_plugin channels subscriptions to EPICS
7. Sending configured channels values to **EPICS**:
*otsdaq DCS channels new values → artdaq Metric Manager
→ software **IOC** → **EPICS** → otsdaq DCS GUI*

Integration with State Machine

- **Alarm** propagation (from **EPICS**) and **otsdaq** state machine **handling** is available: needs just to identify which **PV alarms**, *status* and *severity* will be propagated
- *Tables and parameters designed for configuration*
 - SupervisorTable parameters:
 - *Slow Controls Interface Plugin Type*
 - *Slow Controls Channel Source Table List (HW list i.e. DTC Interface, CFO Interface)*
 - Alarms To Monitor Tables for transition to states:
 - *Configure*
 - *Halt*
 - *Pause*
 - *Resume*
 - *Start*
 - *Running*



Integration with State Machine

- **Alarm** propagation (from **EPICS**) and *otsdaq* state machine **handling**: Example on “Start” transition

Close Errors

Note: Newest messages are at the top.
(Press [ESC] to close and [SHIFT + ESC] to re-open)

↓

```

:GatewaySupervisor:otsdaq/otsdaq/GatewaySupervisor/GatewaySupervisor.cc [1550]
Received error from Supervisor instance =
'ControlsDashboardSupervisor' [LID=282] in Context
'mainContext' [URL=http://mu2eddaq12.fnal.gov:3075].

Error Message =
:SlowControlsDashboardSupervisor:ControlsDashboardSuperv
:otsdaq/otsdaq/CoreSupervisors/CoreSupervisorBase.cc [750]
Error was caught while Starting:
:EpicsInterface_slowcontrols.cc:otsdaq_epics/otsdaq-epics/ControlsInterfacePlugins/EpicsInterface_slowcontrols.cc
[1333]
During 'start'... Alarms monitoring (count=1):
    dummyAlarm

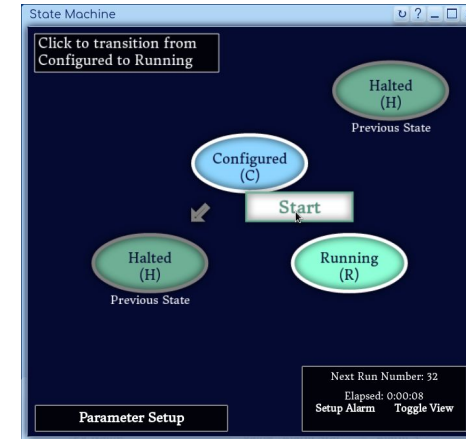
Found alarm for channel
'Mu2e_TDAQ_shift_ExampleInterface0/dummyRegister0' =
{time=1582678095, value=2020, status=HIHI, severity=MAJOR}

Total alarms found = 1
  
```

Slow Controls Dashboard

File EditMode

PV Name	value	Alarm Stat...	Alarm S...
Mu2e_TDAQ_shift_ExampleInte...	2020	HIHI	MAJOR



```

Configure
    -ControlSupervisor
    -LogbookSupervisor
    -VisualSupervisor
    -ConfigurationSupervisor
    -ControlsDashboardSupervisor
    ApplicationGroup : mainContextApps
    -LinkToControlsTable(UUID = ...) //Auto-generated from mock-up.
    -SlowControlDashboard
    -SlowControlInterfacePlugin : EpicsInterface
    -SlowControlChannelSourceTableList : DTCInterfaceTable
    -LinkToConfigureAlarmMonitorTable(GroupID = ...) //Auto-generated from mock-up.
    -ConfigureAlarms
    -LinkToStartAlarmMonitorTable(GroupID = DEFAULT)-DISCONNECTED
    -LinkToPassiveAlarmMonitorTable(GroupID = DEFAULT)-DISCONNECTED
    -LinkToActiveAlarmMonitorTable(GroupID = DEFAULT)-DISCONNECTED
    -LinkToStartAlarmMonitorTable(GroupID = ...) //Auto-generated from mock-up.
    StartAlarms
    -DummyAlarm
    -AlarmGroupID : StartAlarms
    -AlarmChannelName : Make_TDAQ_?MUSE_OWNER.ExampleInterface/dummyRegister0
    -EventNotifierSeverity : False
    -LinkToPassiveAlarmMonitorTable(GroupID = DEFAULT)-DISCONNECTED
    -LinkToActiveAlarmMonitorTable(GroupID = DEFAULT)-DISCONNECTED
    Status : On 0
    Class : SlowControlDashboardSupervisor
    Id : 282
    Instance : 1
    Network : local
    Group : daq
    Module : $(OTSDAQ_UTILITIES_LIB)/libSlowControlDashboard.so
    ConfigPriority : 0
    StartPriority : 0
    StopPriority : 0
    -LinkToPropertyTable(GroupID = DEFAULT)-DISCONNECTED
    -MacroMakerSupervisor
    -CodeEditorSupervisor

```

Slow Controls **WEB Monitoring GUI in otsdaq**

developed in JavaScript and HTML (client side) and C++ (server side)

Basic Widget Mechanism

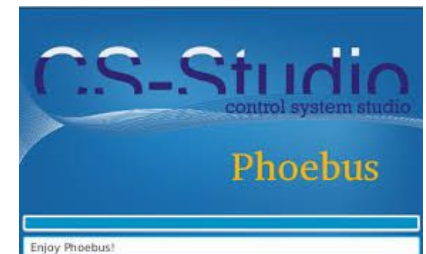
- All widgets have six required methods:
init(), getParameters(), setParameters(), setupPVs(), newWidget(), and newValue()

Widget properties

- Dynamic sizing
- Proper handling of setups
- Value error, warning and alarm handling
- Disconnection handling

Load and save dashboard page in XML

Cs-Studio Phoebus (EPICS GUI) compatible format



Slow Controls Monitoring in otsdaq

Examples

Editor

Slow Controls Dashboard

File EditMode

Mu2e_Weather_2/humi

Mu2e_Weather_2/temper: Status: NO_ALAF Severity: NO_ALA

PV Name	value	Alarm
Mu2e_Weather_1/so...	5.0 W/r	
Mu2e_Weather_1/te...	2 degC	
Mu2e_Weather_1/te...	36.3 de	
Mu2e_Weather_1/wi...	17 mph	
Mu2e_Weather_2/b...	1016.0	
Mu2e_Weather_2/pr...	0.0 inch	

Mu2e_Weather_1/temperatur Status: NO_ALARM Severity: NO_ALARM

Editor Panel

Choose your widgets:

Grid Color Background

Default Values

Name: Basic Root file viewer
Type: Root file

Example of widget settings window

Slow Controls Dashboard

widget-0

Parameter	Value
class	undefi
border	false
text	TDAQ
text_position	left
font	arial

Slow Controls Dashboard

widget-0

Chose PV names

Mu2e:TDAQ_hwdev_DTC0_BurstDataCount

Mu2e:TDAQ_shift:ExampleInterface0:dummyR

Mu2e:TDAQ_shift:ExampleInterface0:dummyR

Mu2e:TDAQ_shift:ExampleInterface0:dummyR

Mu2e:TDAQ_shift:ROC0:dummyRegister0

Add Remove

PV names chosen

Mu2e:TDAQ_shift:ExampleInterface0:dummyR

Mu2e:TDAQ_shift:ExampleInterface0:dummyR

Mu2e:TDAQ_shift:ExampleInterface0:dummyR

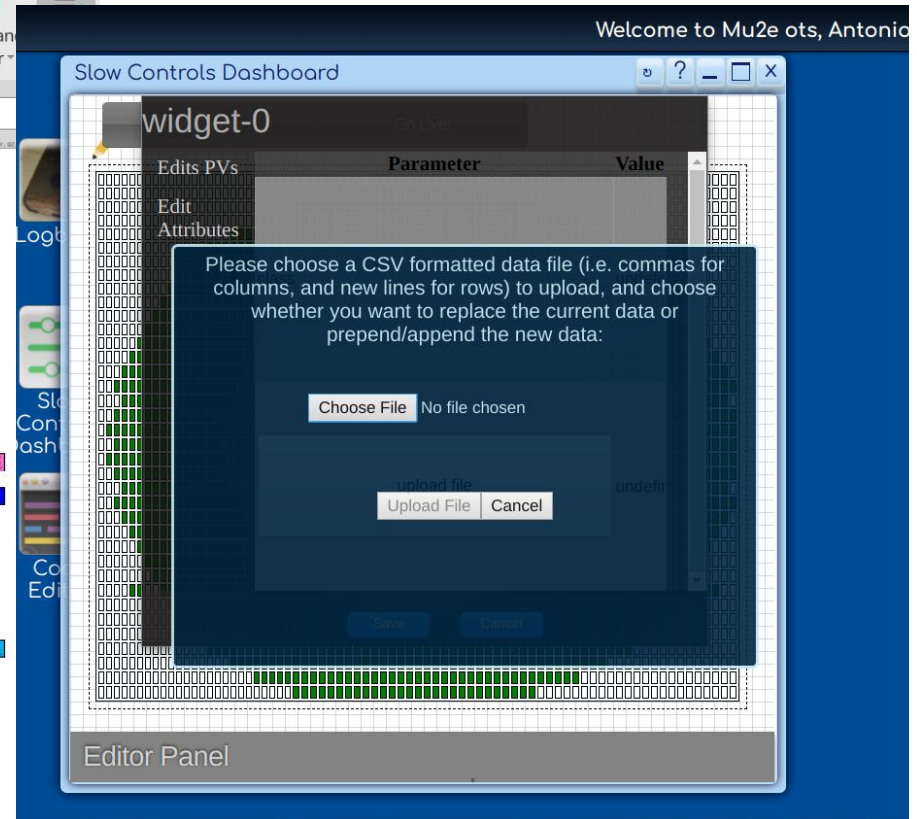
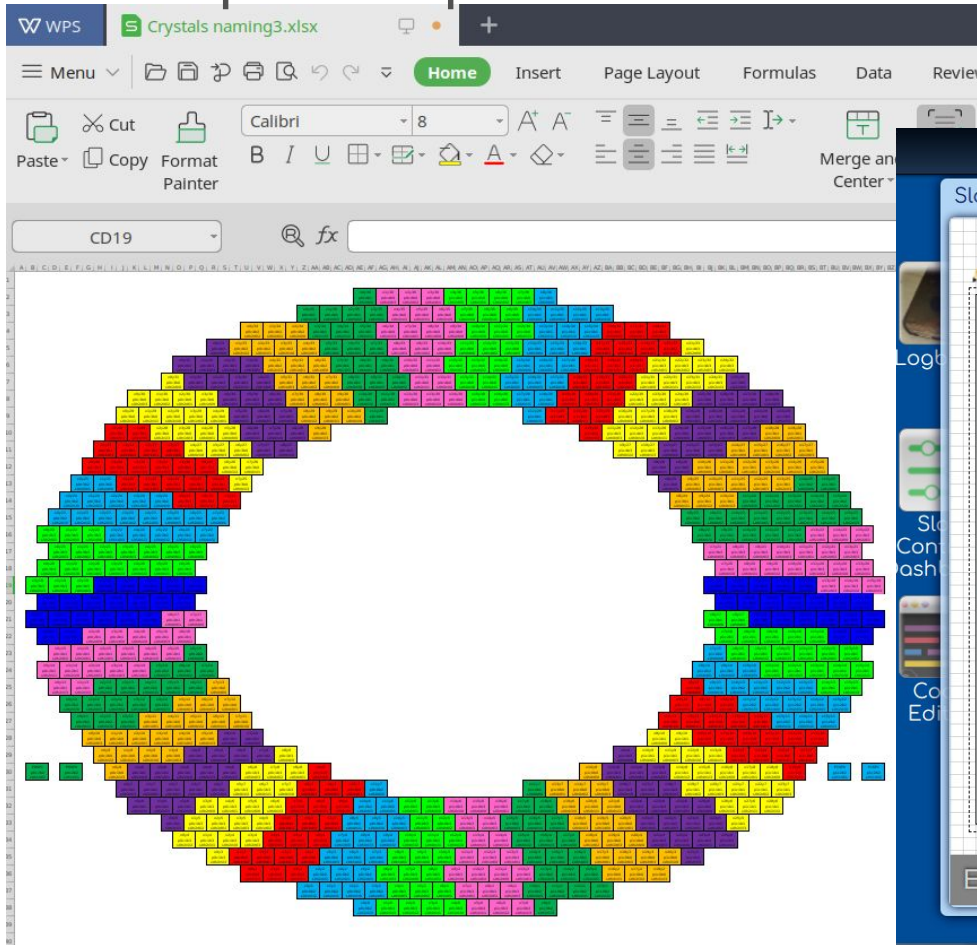
Save Cancel

Editor Panel

widget attributes editor

Calorimeter monitoring and the Slow Controls GUI

Examples: Import an xls file in a 2D-stop light widget



Slow Controls alarm notification by System Message

System message alarm notification example

The screenshot displays the 'Slow Controls Dashboard' interface. A system message notification is shown in a blue box on the left, stating: 'System Message Received at 20:01:19', 'Slow Control Alarm Notification: PV: Mu2e_TDAQ_shift_ExampleInterface0/dummyRegister0', 'at time: Mon Mar 30 13:01:07 2020 value: 1233 stouts: HIHI severity: MAJOR'. A 'Dismiss' button is visible at the bottom right of the notification box.

The main dashboard area features a table titled 'Slow Controls Dashboard' with the following columns: 'PV Name', 'Alarm Status', 'Alarm Severity', and 'Last Update'. The table contains one row of data:

PV Name	Alarm Status	Alarm Severity	Last Update
Mu2e_TDAQ_shift_ExampleInterface0/dummyRegister0	HIHI	MAJOR	03/30/20

The dashboard also includes a sidebar with icons for 'Config Subsets', 'Macro Maker', 'DB Utilities', 'Console', 'System Status', and 'Slow Control'. The top navigation bar includes buttons for 'Layouts', 'Tile', 'Show Desktop', and 'Full Screen', along with a 'Welcome to Mu2e ots, Antonio' message and a 'Sign out' button.