Karlsruhe Institute of Technology

Institute for Beam Physics and Technology

A Python Framework for Integration of Measurements into the EPICS Control System

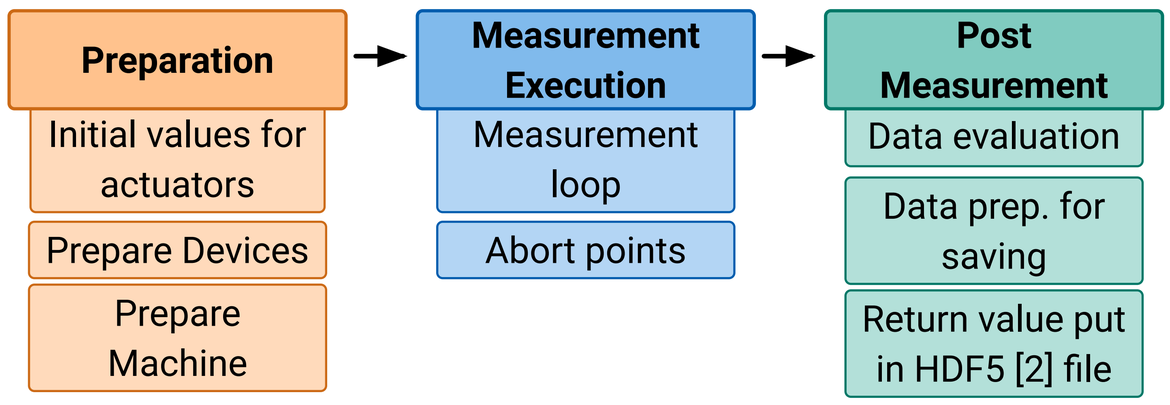
P. Schreiber, E. Blomley, J. Gethmann, M. Schuh, A.-S. Müller

Motivation

- Specialised measurements often developed by scientists and students, not control system experts
  $\Rightarrow$ scripts hard to find, reuse and maintain
- Measurements often rely on scans $\rightarrow$ scripts & routines
- Integration into control system not feasible for may people
- Automatic integration desired

Measurement Framework

- Python framework for easy creating of measurements with automatic integration into EPICS [1]
- Measurements are divided in 3 steps



- Each step has freely implementable part by user
- Python data types
- Measurements can be aborted
- Data is saved even in case of Python exceptions
- Monitors and Actuators including metadata
- 3 automatic interfaces allow control and configuration
- Automatic reset of actuators

Command Line Interface

- Can execute measurement
- Can start IOC and GUI interfaces
- Automatic parameter creation for configuration options of measurement
- Show help on parameters and configuration options

Summary

- Framework for pythonic creation of measurements with automatic EPICS integration
- No extensive knowledge of control system required
- Automatic data saving and graceful aborting
- Future work: closer integration into data management framework, automatic aggregation of multiple measurements and option to create measurement based on YAML files

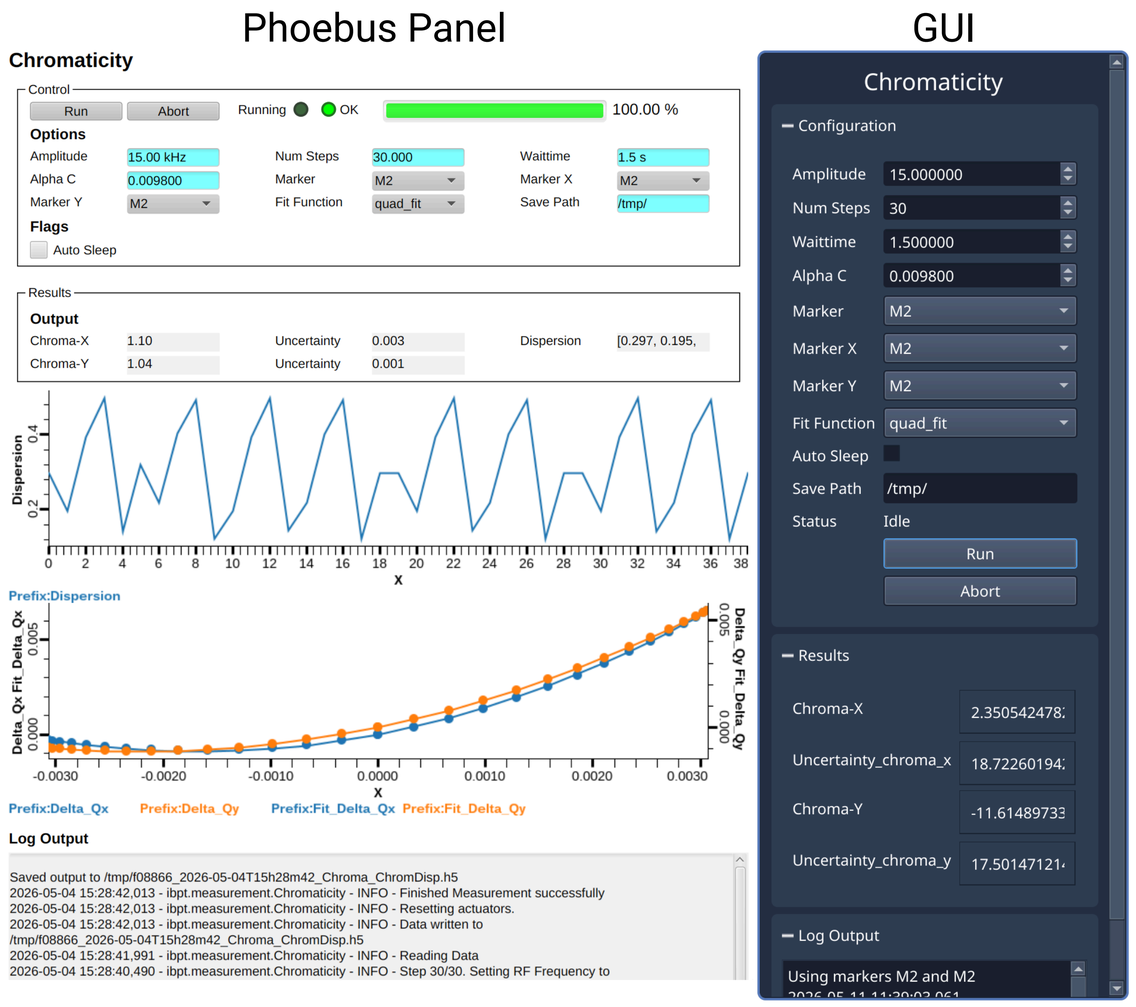
EPICS Integration

- **Automatically** created Python SoftIOC [3]
- **No extra code** required from user
- Long serving process
- Creates Process Variables (PV) for
  - Configuration parameters
  - Results and logs
  - Control
- Measurement routine has possibility to live update values
- Automatic panel creation for CSS and CSS-Phoebus with plots

GUI

- **Auto generated**, **Stand-alone GUI** for control
- Allows configuration of parameters
- Shows results and logs

Screenshots



Minimal Measurement Implementation

```python
class Meas(Measurement, Configurable):
  def __init__(self):
    super().__init__("Meas")
    self.setup_config({
      "iterations":
       CO(type=int, default=5)
    })
    self.set_monitors([
      EpicsMonitor("Mon", "Some:PV")
    ])
    self.set_actuators([
      EpicsActuator("A:PV", name="act")
    ])
```

```python
def prepare(self):
  epics.caput("S:PV", 3)
def measure(self):
  num = self.config.iterations
  for i in range(num):
    self.put_actuator("act", 3)
    self.sleep(1,
               abortable=True)
    self.read_data()

def post_measure(self):
   f = self.data.get("Mon")**2
   return { "q": f**2 }
```

Contact: patrick.schreiber@kit.edu

References

[1] EPICS Control System, https://epics-controls.org [2] HDF5, https://www.hdfgroup.org/HDF5/ [3] Python SoftIOC, https://github.com/DiamongLightSource/pythonSoftIOC

This project has received funding from the European Union's Horizon Europe Research and Innovation Programme under Grant Agreement no. 101057511 (EURO-LABS).