Epics Applications using PyEpics

Release 1.0

Matthew Newville

January 23, 2012

CONTENTS

PyEpics Apps contains several Epics Applications written in python, using the pyepics module. Many of these are GUI Application for interacting with Epics devices through Channel Access. The programs here are meant to be useful as end-user applications, or at least as examples showing how one can build complex applications with PyEpics. Many of the applications here rely on wxPython, and some also rely on other 3rd party modules (such as Image and SQLAlchemy).

The list of applications should be expanding, but currently include:

CHAPTER

ONE

AREADETECTOR DISPLAY

A simple, live display for an Epics Area Detector.

CHAPTER

TWO

STRIP CHART

A simple “live plot” of a set of PVs, similar to the common Epics StripTool.

CHAPTER

THREE

EPICS INSTRUMENTS

This application helps you organize PVs, by grouping them into instruments. Each instrument is a loose collection of PVs, but can have a set of named positions that you can tell it. That is, you can save the current set of PV values by giving it a simple name. After that, you can recall the saved values of each instrument, putting all or some of the PVs back to the saved values. The Epics Instrument application organizes instruments with tabbed windows, so that you can have a compact view of many instruments, saving and restoring positions as you wish.

CHAPTER

FOUR

SAMPLE STAGE

A GSECARS-specific GUI for moving a set of motors for a sample stage, grabbing microscope images from a webcam, and saving named positions.

CHAPTER

FIVE

ION CHAMBER

This non-GUI application is synchrotron-beamline specific. It reads several settings for the photo-current of an ion chamber and calculates absorbed and transmitted flux in photons/sec, and writes these back to PVs (an associated .db file and medm .adl file are provided). The script runs in a loop, updating the flux values continuously.

CHAPTER

SIX

XRF COLLECTOR

This non-GUI application interacts with a small epics database to save data from a multi-element fluorescence detector. The script runs as a separate process, watching PVs and saving data from the detector on demand. Associated .db file and medm .adl file are provided.

6.1 Overview

An *Epics Instrument* is simply a grouping of low-level parameters (Process Variables) as exposed through Epics Channel Access. At first, this may not seem very interesting. However, Epics Instruments allows you to

The Epics Instruments application allows you to group these components into a logical group – an Instrument. Once an Instrument has been defined, you can then save and restore settings for this Instrument. Furthermore, these settings are automatcally saved in a single, portable file for later use.

Epics Channel Access gives a simple and robust interface to its lowest common unit – the Process Variable or PV. The Epics control system also provides sophisticated ways to express and manipulate complex devices, both physical and virtual. Creating such devices and defining their behavior is generally done by well-trained programmers. The application here uses a much simpler approach that can expose some categories of "Settings" that may need to changed en masse, and returned to at a later time.

As defined here, An Epics Instrument is simply a named collection of Epics Process Variables (PVs). The PVs do not need to be physically related to one another nor be associated with a single Epics Record or Device. Rather, an Instrument is defined at the level of the Epics Channel Access client, allowing a station scientist or engineer to use their own grouping of PVs as an abstract "Instrument". A simple example would be a pair of motors that work together to move some device.

In addition to a name and a set of PVs, an Instrument has a set of "named Positions". At any point, the current values of an Instruments PVs can be saved as its Position. And, of course, the named Positions can then be restored simply by selecting the Position.

6.2 Downloading and Installation

Many Epics Applications are available as GUI Application. For Windows, you can download and install from a binary installers below.

For Linux and Mac OS X, building and installing from source are currently the principle options (a Mac OS X App will be available soon).

6.2.1 Downloads

The latest source kits and installers will be at

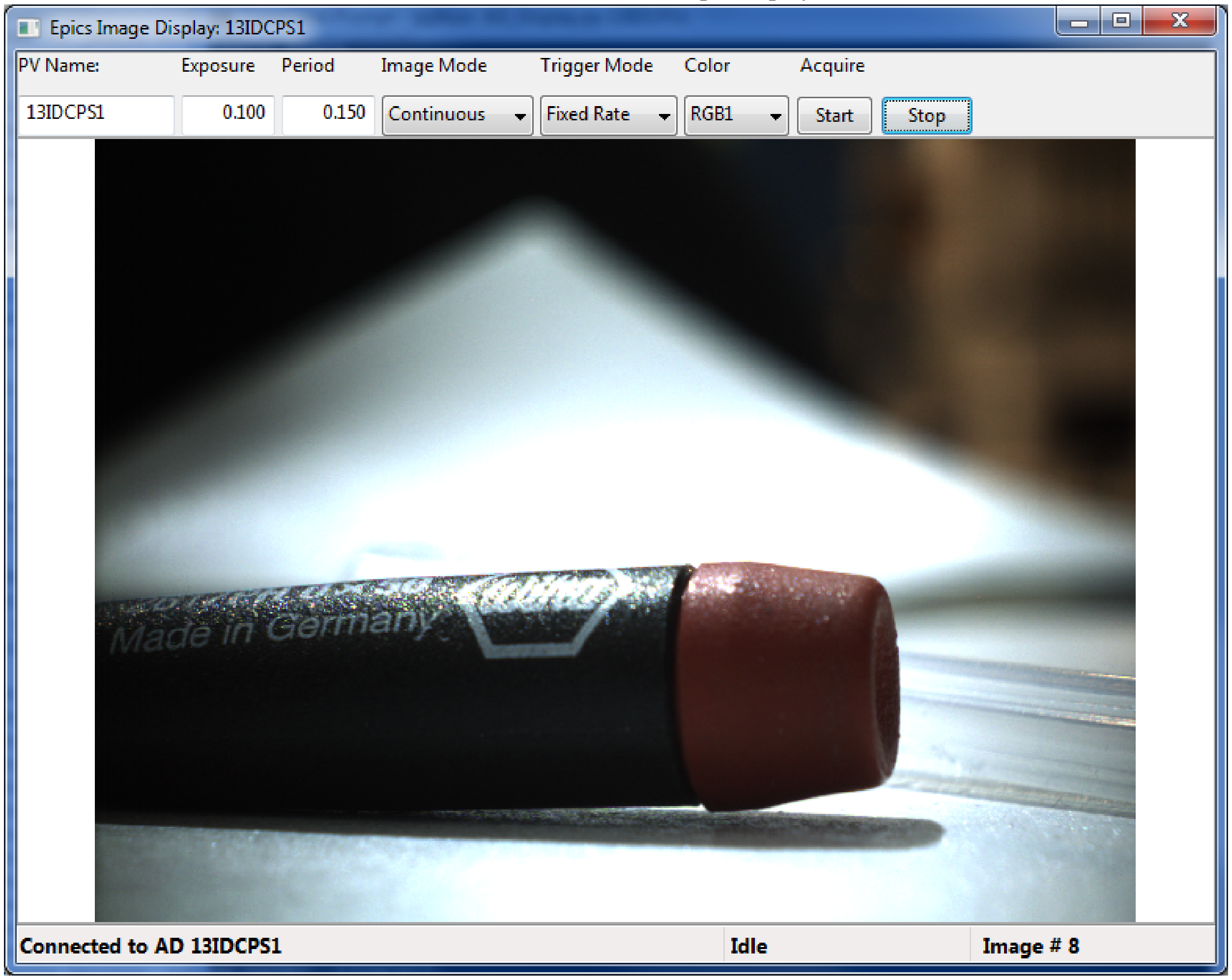
Download Option	Location
Windows Installers	cars_downloads
Source Kit	github_repo

6.3 Area Detector Display

Epics Area Detector Display is a wxPython GUI application for viewing images from an Epics Area Detector. This application requires wxPython, pyepics, numpy, and the Python Image Libary.

To run this application, simply run AD_Display.py at the command line:: python AD_Display.py

and enter the base name of the AreaDetector in the PV box. A sample display would look like this:



Epics Image Display: 13IDCPS1

PV Name: 13IDCPS1
Exposure: 0.100
Period: 0.150
Image Mode: Continuous
Trigger Mode: Fixed Rate
Color: RGB1

Connected to AD 13IDCPS1 Idle Image # 8

The fields and buttons on the top control several Area Detector settings, such as starting and stopping the acquisition.

6.4 Strip Chart Display

StripChart is a wxPython GUI application for viewing time traces of PVs as a strip chart. This is inspired by the classic Epics Stripchart application written with X/Motif.

6.4.1 Dependencies, Installation

This application needs pyepics, numpy, matplotlib, and wxPython.

It also the wxmplot plotting library, which can be found at http://pypi.python.org/pypi/wxmplot/, with development versions at http://github.com/newville/wxmplot/ and may be installed wth:

```
easy_install -U wxmplot
```

Installation of the striphart can be done with:

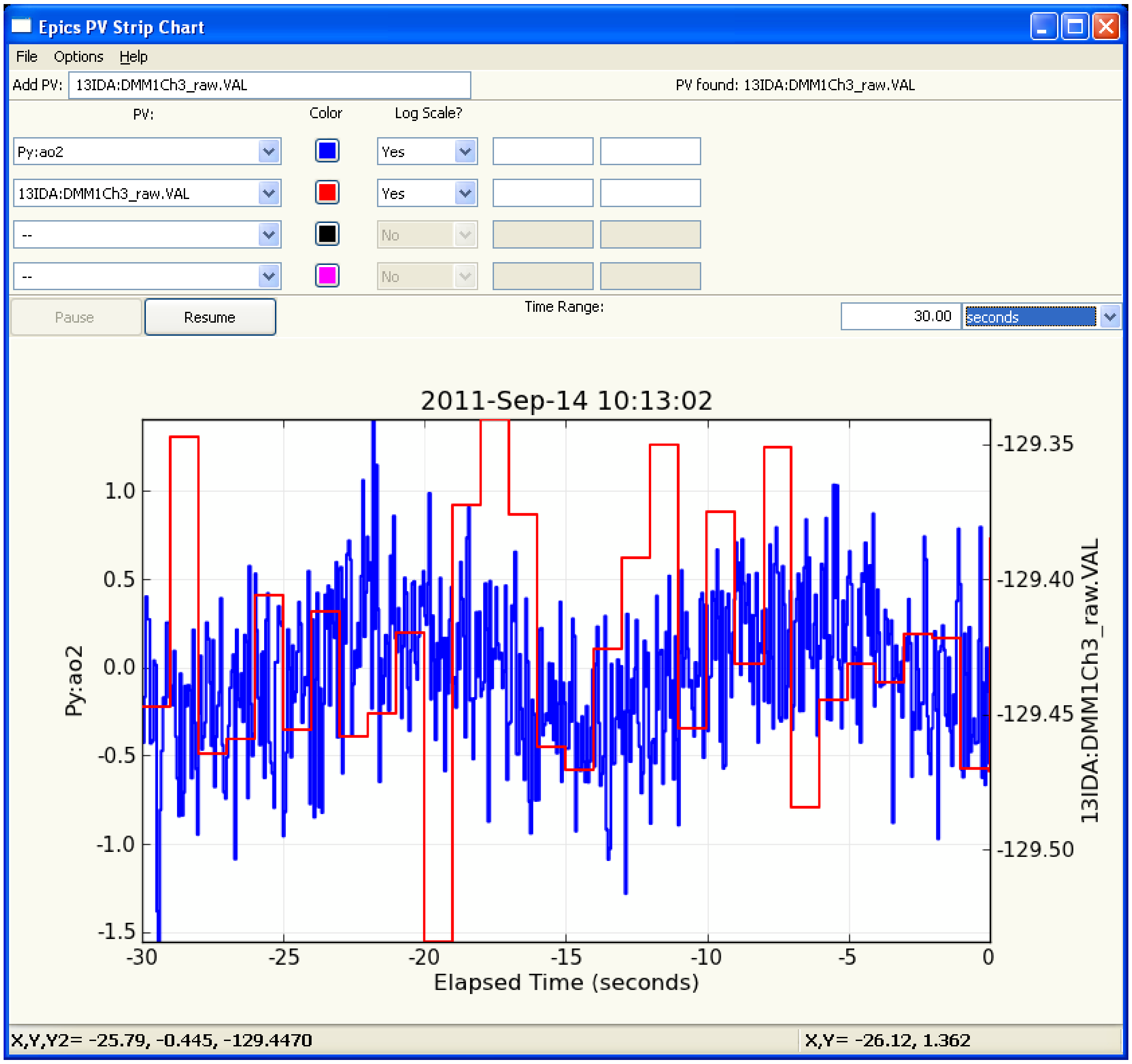
```
python setup.py install
```

6.4.2 Running Stripchart

To run this application, simply run stripchart.py at the command line:

```
python pyepics_stripchart.py
```

and enter the base name of the PVs to follow. A sample display would look like this:



Add PV: 13IDA:DMM1Ch3_raw.VAL

PV found: 13IDA:DMM1Ch3_raw.VAL

PV:	Color	Log Scale?
Py:ao2	🟦	Yes
13IDA:DMM1Ch3_raw.VAL	🟥	Yes
--	⬛	No
--	🟪	No

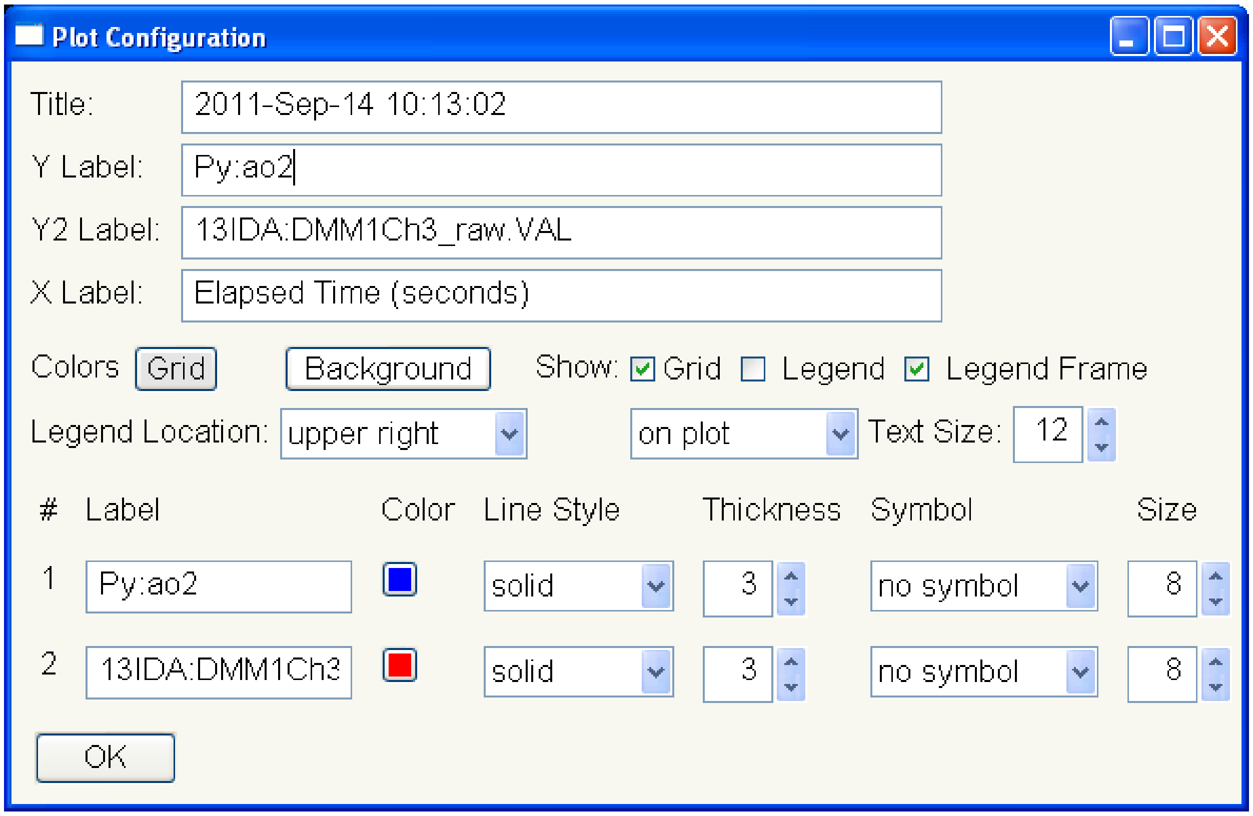
Time Range: 30.00 seconds

X,Y,Y2= -25.79, -0.445, -129.4470

X,Y= -26.12, 1.362

6.4.3 Usage

Plot details such as line colors, thicknesses, labels, etc can be adjusted from the configuration screen, available from the Menu.


Plot Configuration

Title: 2011-Sep-14 10:13:02

Y Label: Py:ao2

Y2 Label: 13IDA:DMM1Ch3_raw.VAL

X Label: Elapsed Time (seconds)

Colors Grid Background Show: [✓ Grid] Legend [✓ Legend Frame]

Legend Location: upper right on plot Text Size: 12

#	Label	Color	Line Style	Thickness	Symbol	Size
1	Py:ao2	🟦	solid	3	no symbol	8
2	13IDA:DMM1Ch3	🟥	solid	3	no symbol	8

OK

6.5 Using Epics Instruments

6.5.1 Getting Started

To run Epics Instruments, click on the icon. You will see a small window to select an Epics Instrument File. If this is your first time using the application, choose a name, and hit return to start a new Instrument File. The next time you run Epics Instruments, it should remember which files you've recently used, and present you with a drop-down list of Instrument Files.

6.5.2 Defining an Instrument

6.5.3 The Instrument File

All the information for definitions of your Instruments and their Positions are saved in a single file – the Instruments file, with a default extension of '.ein' (Epics INstruments). You can use many different Instrument Files for different domains of use.

The Instrument File is an SQLite database file, and can be browsed and manipulated with external tools. Of course, this can be a very efficient way of corrupting the data, so do this with caution. A further note of caution is to avoid having a single Instrument file open by multiple applications – this can also cause corruption. The Instrument files can be moved around and copied without problems.

6.6 XRF Collector

XRF Collector provides a simple Epics interface control of an x-ray fluorescence detector. The main point of showing it here is as an example of using a python process to interact with a custom Epics database to provide a customized way to acquire data.

6.7 Ion Chamber

This application, like the XRF Collector, provides a simple Epics interface control of a non-trivial device. Again, the main point of showing it here is as an example of using a python process to interact with a custom Epics database to provide a customized way to acquire data. While the standard Epics solution would be to write a state-notation-language program that runs in an IOC, the approach here minimizes Epics coding to writing a simple database, and putting all the logic and control into a python script that runs as a long-running process along-side an IOC process.