Canadian Light Source
Centre canadien de rayonnement synchrotron

Qt at the CLS

Glen Wright
Glen.Wright@lightsource.ca
EPICS Collaboration Spring 2012

Overview

- SAL and CLS GUI development history
- CLS epicsQt – Version 4 (current)
- acquaman
- pyEdm
- Another Approach
- Other Issues
- Screenshots

A brief and focused history

Before the Canadian Light Source, there was the Saskatchewan Accelerator Laboratory

- GUI – Amiga 1000, custom widgets
- Sun Workstations – OpenLook, Motif
- NeXTstation - NeXTstep

History Part 2

CLS – since 1999

- GTK+ 1.2 & Glade (0.6.2) – common handler to simplify attaching PV's to widgets
- Borland Builder – simple interface to Channel Access
- QT 3.3, 4.X – multiple methods to interface to Channel Access

CLS epicsQt

EpicsQt Version 4

EpicsQt 4 – C++, Python 2.6

- Qt 4.5, 4.8
- Uses “epicsConnect” library from GTK+ work
- Support Classes – displayAlias, QEpicsAcqClass (factory base), QEpicsColorRule, QEpicsConnect, QEpicsConnectList, QEpicsPlotStream, genericFactory

EpicsQt Version 4

New Widgets

QEpicsAcq, QEpicsAcqWidget, QEpicsButtonGroup, QEpicsComboBox, QEpicsDial, QEpicsHist1D, QEpicsLabel, QEpicsMButton, QEpicsProgressBar, QEpicsShape, QEpicsSlider, QEpicsSpinBox, QEpicsText, QEpicsWheel

EpicsQt Version 4

Common Properties

- PVname
- ColorPVname
- FgColorRule
- BgColorRule
- FGalarmSensitive
- BGalarmSensitive

EpicsQt Version 4

- Consistent set of Qt properties for all widgets
- Separate Interface and Implementation classes
- Factory method for data acquisition: easily add support for a new data acquisition framework to an existing application

EpicsQt Version 4

```
#if !defined QEPICSLABEL_H
#define QEPICSLABEL_H 1

#include "qepicsbase.h"
#include <QLabel>

class QEpicsLabel : public QLabel
{
	Q_OBJECT

	Q_PROPERTY( QString Format READ getFormat WRITE setFormat)
#include "qepicscommon.inc"
public:
	QEpicsLabel(QWidget *parent = 0);
	QString getFormat() const ;
	void setFormat(const QString &f) ;
  protected:
	virtual void mousePressEvent( QMouseEvent *);

} ;

#endif
```

EpicsQt Version 4

- Support for ‘middle-click’ to set cursor to PV name

```
// monitor the mouse clicks, and put the PV name in the drag buffer
void QEpicsLabel::mousePressEvent( QMouseEvent *ev)
{
	if( ev->button() == Qt::MidButton)
		dynamic_cast<QEpicsLabelImpl*>(Impl)->getQEpicsConnector().dragName(this);
	else
		QLabel::mousePressEvent(ev);
}
```

EpicsQt Version 4

Drawbacks of current design:

- Not Thread Friendly
- Exposes channel access too easily

Acquaman

Acquaman & Dataman

- A framework for providing acquisition and data management for Big Science
- C++, Qt 4.7
- Custom classes geared specifically to managing experiments, controlling a beamline, and recording, storing, and displaying data

Acquaman

- AMcontrol – abstract class, QObject with signals & Slots
- AMPVcontrol (and others) inherit AMcontrol – direct calls (and callbacks) of Channel Access functions

pyEdm

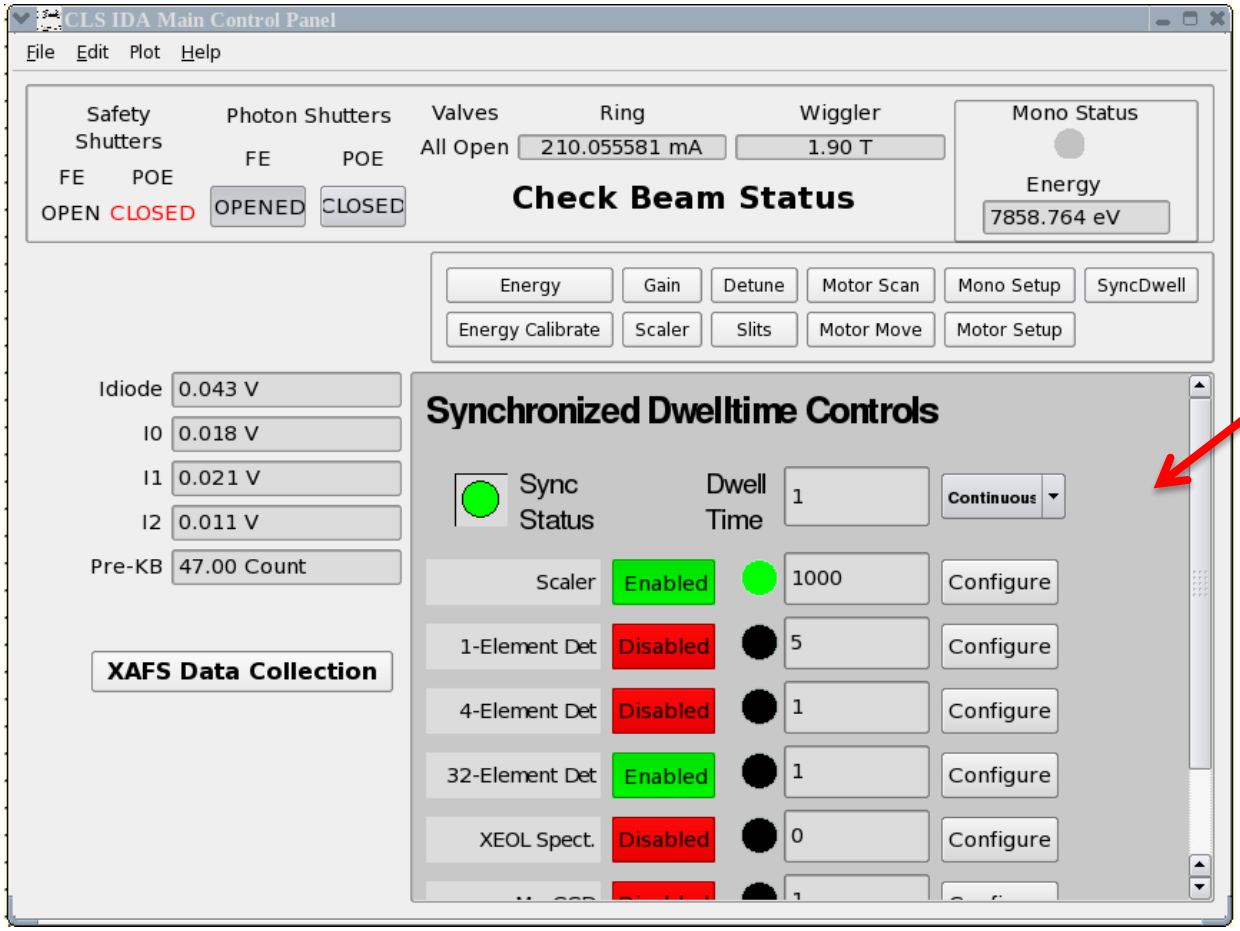
Canadian Light Source Centre canadien de rayonnement synchrotron

pyEdm

- Reads .edl files, produces display screens using standard Qt Widgets
- Uses a base class similar to edm for PV access
- EPICS Channel Access via pyEpics 3
- Further info “EDM Screen Display using Python” EPICS Collaboration Fall 2011

pyEdm embedded

Main Window: Qt UI file displayed using PyQt4



Embedded pyEdm display

Another Approach

Another Approach

Another Approach

pyEpics (direct channel access function calls) and PyQt4 used with standard Qt widgets – used when few PV's or an extremely custom display

Other Issues

Other Issues

Other Issues:

- Common Channel Access QT Class
- Common Plot Tools
- Common Display Format

Common Channel Access Qt Class

Right now…

- Python: pyEpics, callbacks to custom display code
- C++ Channel Access, callbacks to custom display code
- QEpicsConnect, custom widgets, signals and slots
- AMPVcontrol – Acquaman only

Common Plot Tools

- CERN ROOT – powerful, available Qt widget, but too slow for immediate feedback use (trying to run 2D scan and update display)
- Qwt – powerful, but some learning curve. Better performance than ROOT in CLS use cases
- Mplot – Acquaman unique – written for speed improvement

Display File Format

- Why aren't we making more use of XML?
- .adl, .edl, .ui – geared towards a specific application or framework
- pyEdm might make more sense as a .edl to .ui file translator than as a display generator
- AMcontrol, configured for Acquaman use

Time to Collaborate?

- Is it time to collaborate?
  - How much?
  - On What? Base Class? Widgets? File Format?
- EPICS and Qt “too simple” to require collaboration?
- Is it time to drop Qt and go with CSS?

Questions?

Questions?

(answers are also appreciated)