areaDetector: A module for EPICS area detector support

Jon Thompson

Beamline Controls Group

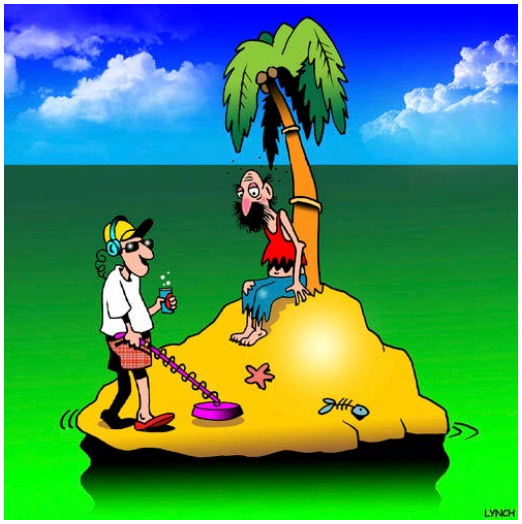
Diamond Light Source

Material primarily stolen from Mark Rivers and Ulrik Pedersen

Oxfordshire EPICS 2016

areaDetector Talk Outline

- Motivation & goals for areaDetector module
- Overview of architecture
- Drivers for detectors & cameras
- Plugins for real-time processing
- Viewers
- Writing software for a new detector or a new plugin
- Future ideas
- The collaboration



areaDetector Links

Take a look at these when you feel like you're about to drift off…

The docs:
http://cars9.uchicago.edu/software/epics/areaDetectorDoc.html
(or google areadetector epics – first hit)

The sources and collaboration, issues, pull requests, etc:
www.github.com/areaDetector

The wiki: www.github.com/areaDetector/areaDetector/wiki

The ultimate origin of these slides (a full 1 day course):
http://www.aps.anl.gov/epics/docs/APS2015.php

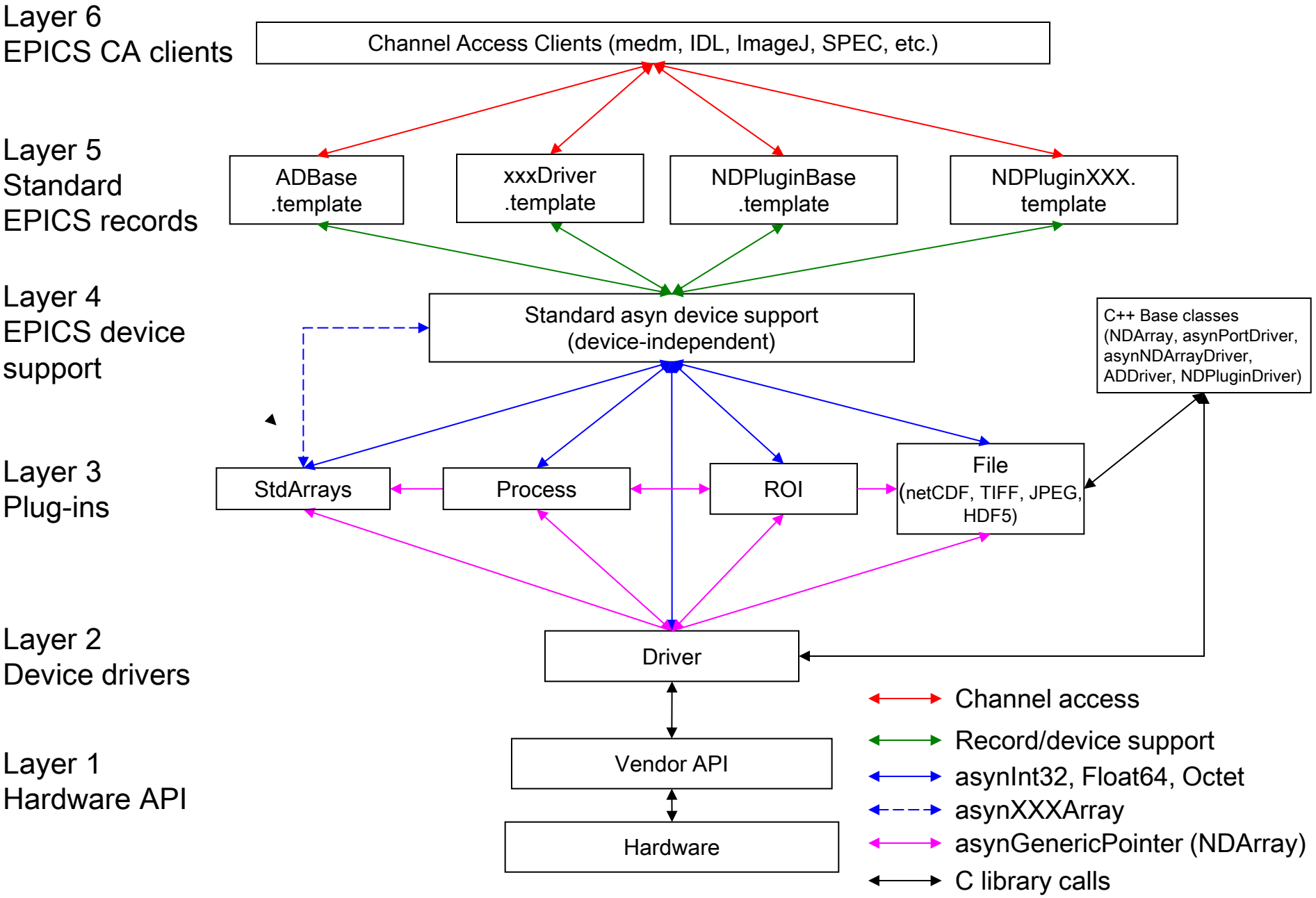
areaDetector - Motivation

- 2-D detectors are essential components of many end stations.
- Need to control the detectors from EPICS
- Clear advantages to an architecture that can be used on any detector, re-using many software components
- Providing EPICS control allows any higher-level client to control the detector and access the data
  - The client needs only to know how to talk to EPICS, not the details of the detector

areaDetector - Goals

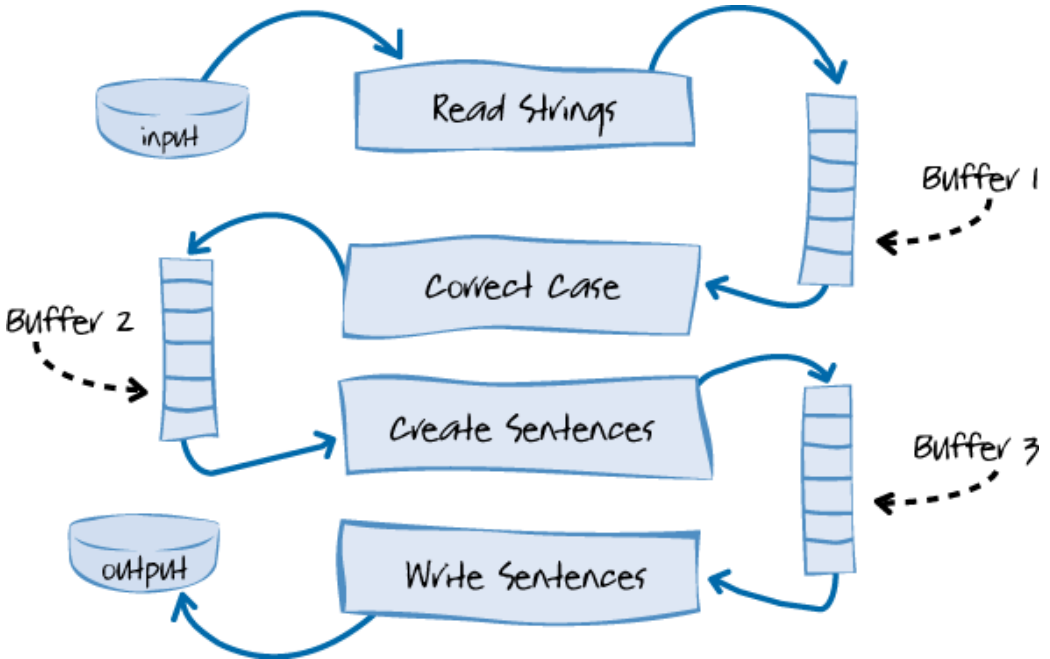
- Drivers for many detectors popular at synchrotron beamlines
  - Handle detectors ranging from >500 frames/second to <1 frame/second
- Basic parameters for all detectors
  - E.g. exposure time, start acquisition, etc.
  - Allows generic clients to be used for many applications
- Easy to implement new detector
  - Single device-driver C++ file to write. EPICS independent.
- Easy to implement detector-specific features
  - Driver understands additional parameters beyond those in the basic set
- EPICS-independent at lower layers.
- Middle-level plug-ins to add capability like regions-of-interest calculation, file saving, etc.
  - Device independent, work with all drivers
  - Below the EPICS database and Channel Access layers for highest performance

EPICS areaDetector Architecture



areaDetector – architecture

- Pipelining acquisition and processing framework
- “Driver”: acquisition of data and passing it into the framework – i.e. a producer in the framework
- “Plugin”: processing of data from a producer – i.e. a consumer
  - Data input in the form of N-dimensional arrays with associated meta-data (attributes)
  - Can effectively act as a filter: output a processed N-dimensional array (i.e. a consumer and a producer)
- Typically areaDetector application: one Driver followed by a chain of Plugins



R2-0 Organization

areaDetector
Top-level module
RELEASE files, documentation, Makefile

ADCore
Core module
Base classes, plugins, simDetector, documentation

ADBinaries
Binary libraries for Windows (HDF5, GraphicsMagick)

ADProsilica
Prosilica driver

ADPilatus
Pilatus driver

…

- Each box above is a separate git repository
- Can be released independently
- Hosted at http://github.com/areaDetector project
- Each repository is a submodule under areaDetector/areaDetector

Detector drivers

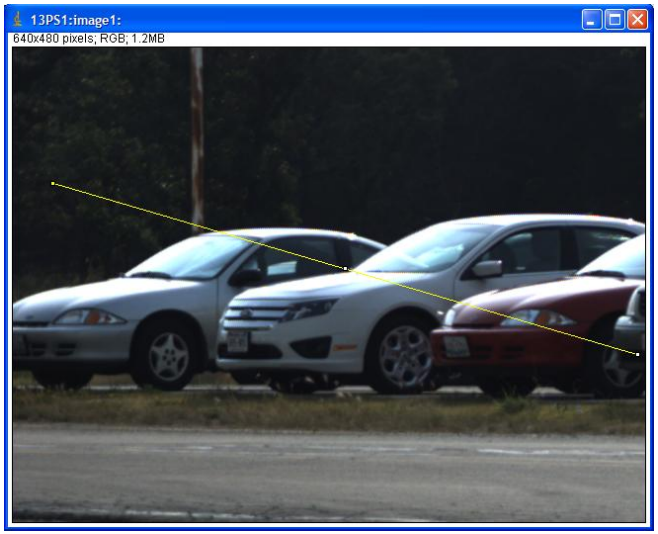
- ADDriver (in ADCore)
  - Base C++ class from which detector drivers derive. Handles details of EPICS interfaces, and other common functions.
- Simulation driver (in ADCore)
  - Produces calculated images up to very high rates. Implements nearly all basic parameters, including color. Useful as a model for real detector drivers, and to test plugins and clients.
- Detector Drivers
  - There's lots of these:
  - Prosilica, Firewire, Roper, PVCAM, Pilatus, marCCD, ADSC, mar345, Perkin-Elmer, Bruker, ….

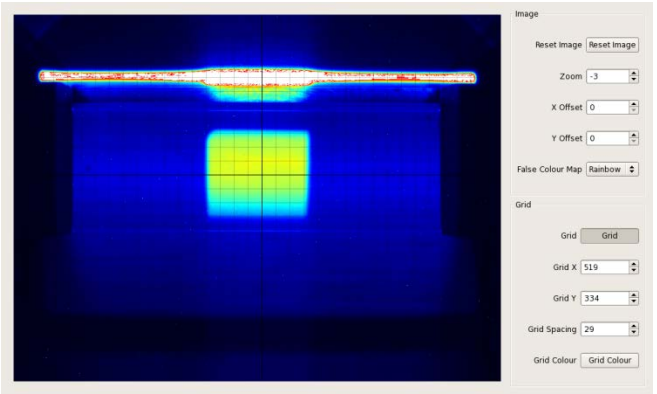
Plugins

- Designed to perform real-time processing of data, running in the EPICS IOC (not over EPICS Channel Access)
- Receive NDArray data over callbacks from drivers or other plugins
- Plug-ins can execute in their own threads (non-blocking) or in callback thread (blocking)
  - If non-blocking then NDArray data is queued
    - Can drop images if queue is full
  - If executing in callback thread, no queuing, but slows device driver
- Allows at runtime
  - Enabling/disabling
  - Throttling rate (no more than 0.5 seconds, etc)
  - Changing data source for NDArray callbacks to another driver or plugin
- Some plugins are also sources of NDArray callbacks, as well as consumers.
  - Allows creating a data processing pipeline running at very high speed, each in a different thread, and hence in multiple cores on modern CPUs.

Viewers

- areaDetector allows generic viewers to be written that receive images as EPICS waveform records over Channel Access
- Current viewers include:
  - ImageJ plugin
  - ffmpegViewer high-performance Qt-based viewer for MJPEG stream





Writing an areaDetector Driver

- Study the vendor API documentation
- Run the camera with the vendor GUI to understand how it works
- Does the vendor provide a simulation camera? If so you can start development before the hardware arrives
- Find an existing areaDetector driver to use as a model.
- Spend some time to figure out which existing driver is most similar to the one you will be writing
- Determine what features are most important to implement first
- Start with a simple version of the driver, add complexity one piece at a time

Future Ideas

- Extend areaDetector concepts to other types of detectors:
  - ADCs
  - Electrometers
  - Waveform digitizers
- They all produce 1-D (or 2-D for multi-channel inputs) arrays that could benefit from plugins for file saving, FFTs, ROI extraction, digital filtering, etc.
- Export NDArrays via EPICS V4

areaDetector Collaboration

- The move to GitHub has really helped areaDetector become a collaborative effort
- Many more people are contributing via additions and bug fixes.
- Make changes in their fork on github and then issue a “pull request”.
- Collaboration meeting ~monthly on Google Hangout
- In-person meetings ~2 times/year.
- Developed a road-map, following it pretty well.

Conclusions

- Architecture works well, easily extended to new detector drivers, new plugins and new clients
- Base classes, asynPortDriver, asynNDArrayDriver, asynPluginDriver actually are generic, nothing “areaDetector” specific about them.
- They can be used to implement any N-dimension detector, e.g. the XIA xMAP (16 detectors x 2048 channels x 512 points in a scan line)
- Can get documentation and pre-built binaries (Linux, Windows, Cygwin) from the Web site:
  - http://cars.uchicago.edu/software/epics/areaDetector
- Can get code from github
  - https://github.com/areaDetector

Acknowledgments

- Mark Rivers, Brian Tieman, Tim Madden, Tim Mooney, Arthur Glowacki, John Hammonds, Chris Roehrig (APS)
- Ulrik Pedersen, Tom Cobb, Nick Rees (Diamond)
- Alan Greer (Observatory Sciences)
- Matthew Pearson (ORNL)
- Emma Sheppard (Australian Synchrotron)
- Lewis Muir (IMCA CAT)
- Keith Brister (LS-CAT)
- Bruce Hill (SLAC)
- Many others for enhancements and bug fixes
- NSF-EAR and DOE-Geosciences for support of GSECARS where most of this work was done
- Thanks for your attention!!!