areaDetector: A new module for EPICS area detector support

Mark Rivers

GeoSoilEnviroCARS, Advanced Photon Source

University of Chicago

areaDetector Talk Outline

- Brief overview of areaDetector module
- Highlight changes since last TWG presentation (June 19, 2008)
  - Releases
    - 1-0 (11-Apr-2008)
    - 1-1 (10-May-2008)
    - 1-2 (24-Oct-2008)
    - 1-3 (24-Nov-2008)
    - 1-4 (30-Jan-2009)
    - 1-5 (23-August-2009)
- Demo with Prosilica camera

areaDetector - Motivation

- 2-D detectors are essential components of synchrotron beamlines
  - Sample viewing cameras, x-ray diffraction and scattering detectors, x-ray imaging, optical spectroscopy, etc.
- Need to control the detectors from EPICS (useful even on non-EPICS beamlines, since other control systems like SPEC etc. can talk to EPICS)
- Previously several packages available, each typically restricted to a small set of detectors (Flea, Pilatus, marCCD, etc.)
- 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 (SPEC, medm, IDL programs, Python scripts, etc)

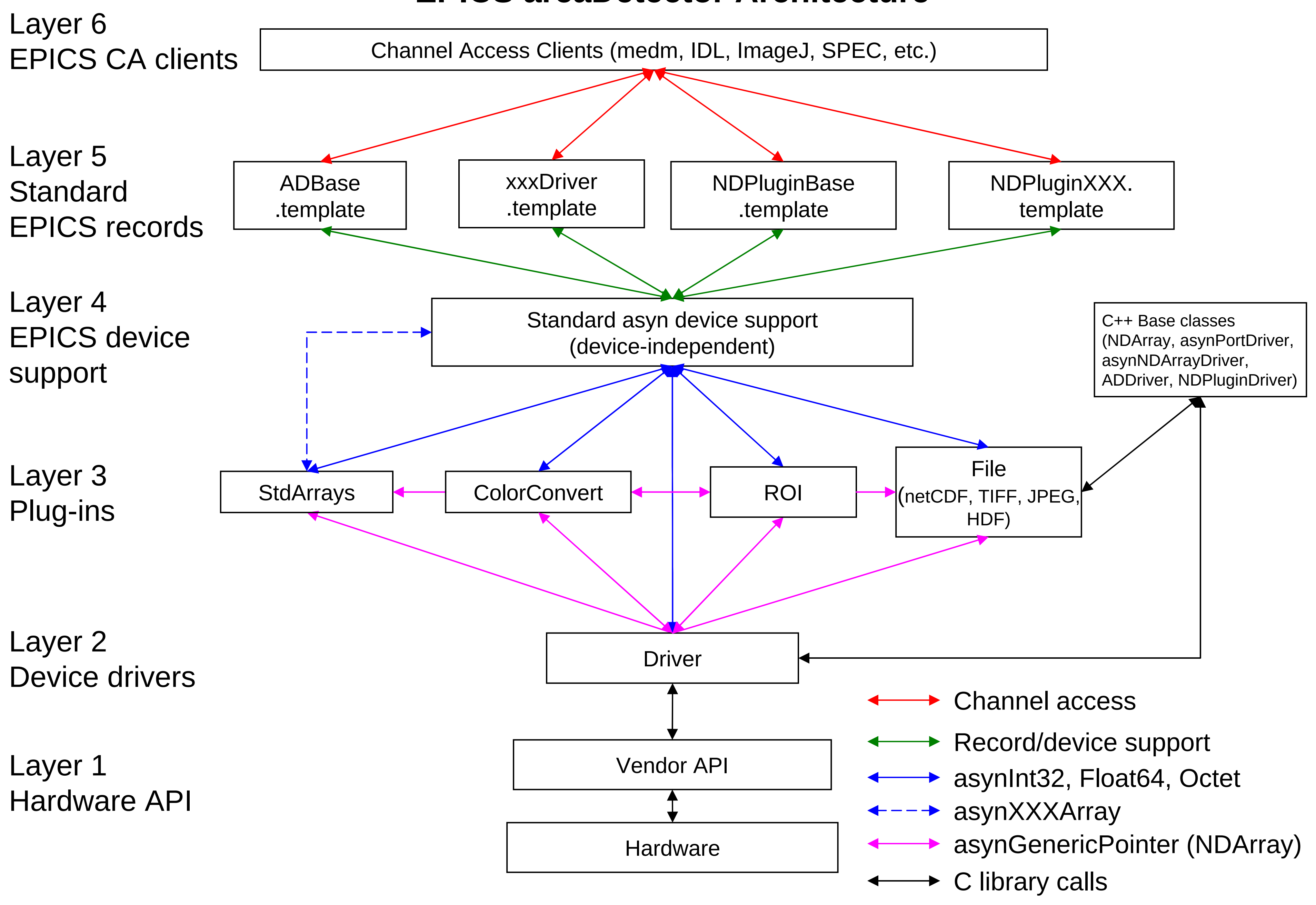
areaDetector - Goals

- 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 layer for highest performance

areaDetector – Data structures

- NDArray
  - N-Dimensional array.
    - Everything is done in N-dimensions (up to 10), rather than 2. This is needed even for 2-D detectors to support color.
  - This is what plug-ins callbacks receive from device drivers.
- NDAttribute (NEW)
  - Each NDArray has a list of associated attributes (metadata) that travel with the array through the processing pileline. Attributes can come from driver parameters or any EPICS PV; e.g. can store motor positions, temperature, ring current, etc. with each frame.
- NDArrayPool
  - Allocates NDArray objects from a freelist
  - Plugins access in readonly mode, increment reference count
  - Eliminates need to copy data when sending it to callbacks.

EPICS areaDetector Architecture



Details: NDArray

```
/** Enumeration of NDArray data types */
typedef enum
{
    NDInt8,     /**< Signed 8-bit integer */
    NDUInt8,    /**< Unsigned 8-bit integer */
    NDInt16,    /**< Signed 16-bit integer */
    NDUInt16,   /**< Unsigned 16-bit integer */
    NDInt32,    /**< Signed 32-bit integer */
    NDUInt32,   /**< Unsigned 32-bit integer */
    NDFloat32,  /**< 32-bit float */
    NDFloat64   /**< 64-bit float */
} NDDataType_t;


/** Structure defining a dimension of an NDArray */
typedef struct NDDimension {
    int size;
    int offset;
    int binning;
    int reverse;
} NDDimension_t;
```

Details: NDArray

```
class NDArray {
…
public:
    int             uniqueId;
    double          timeStamp;
    int             ndims;
    NDDimension_t dims[ND_ARRAY_MAX_DIMS];
    NDDataType_t  dataType;
    int             dataSize;
    void            *pData;
    NDAttributeList *pAttributeList;
};
```

Details: NDAttribute

```
class NDAttribute {
public:
 virtual int getValue(NDAttrDataType_t dataType, void *pValue, size_t
dataSize=0);
    virtual int setValue(NDAttrDataType_t dataType, void *pValue);
    virtual int updateValue();
    …
    char *pName;
    char *pDescription;
    char *pSource;
    NDAttrSource_t sourceType;
    NDAttrDataType_t dataType;

/** NDAttributeList class; this is a linked list of attributes. */
class NDAttributeList {

/** Attribute that gets its value from an EPICS PV. */
class PVAttribute : public NDAttribute {

/** Attribute that gets its value from an asynNDArrayDriver driver
parameter.
class paramAttribute : public NDAttribute {
```

Driver and PluginAttributes Defined via User-Created XML File

```
 <?xml version="1.0" standalone="no" ?>
<Attributes>
  <Attribute name="AcquireTime"
             type="EPICS_PV"
             source="13PS1:cam1:AcquireTime"
             dbrtype="DBR_NATIVE“
             description="Camera acquire time" />
  <Attribute name="CameraModel"
             type="PARAM"
             source="MODEL“
             datatype="STRING"
             description="CameraModel" />
  <Attribute name="FramesDropped"
             type="PARAM"
             source="PS_FRAMES_DROPPED"
             datatype="INT"
             description="FramesDropped" />
</Attributes>
```

Color Support (NEW)

- 4 colors models are supported
  - RGB1 (pixel interleave, e.g. RGB for each pixel are adjacent)
  - RGB2 (row interleave, e.g. R for row 1, then G, then B)
  - RGB3 (plane interleave, e.g. R for image, then G then B)
  - Bayer (color mask commonly used on color cameras)
- ColorConvert plugin converts between these color models
- Can reduce bandwidth 3X to read Bayer directly from camera (rather than RGB) and do color conversion in areaDetector
  - Tradeoff bandwidth for CPU usage
- Can stream Bayer to file and do color conversion later, reduce file size 3X.

Detector drivers

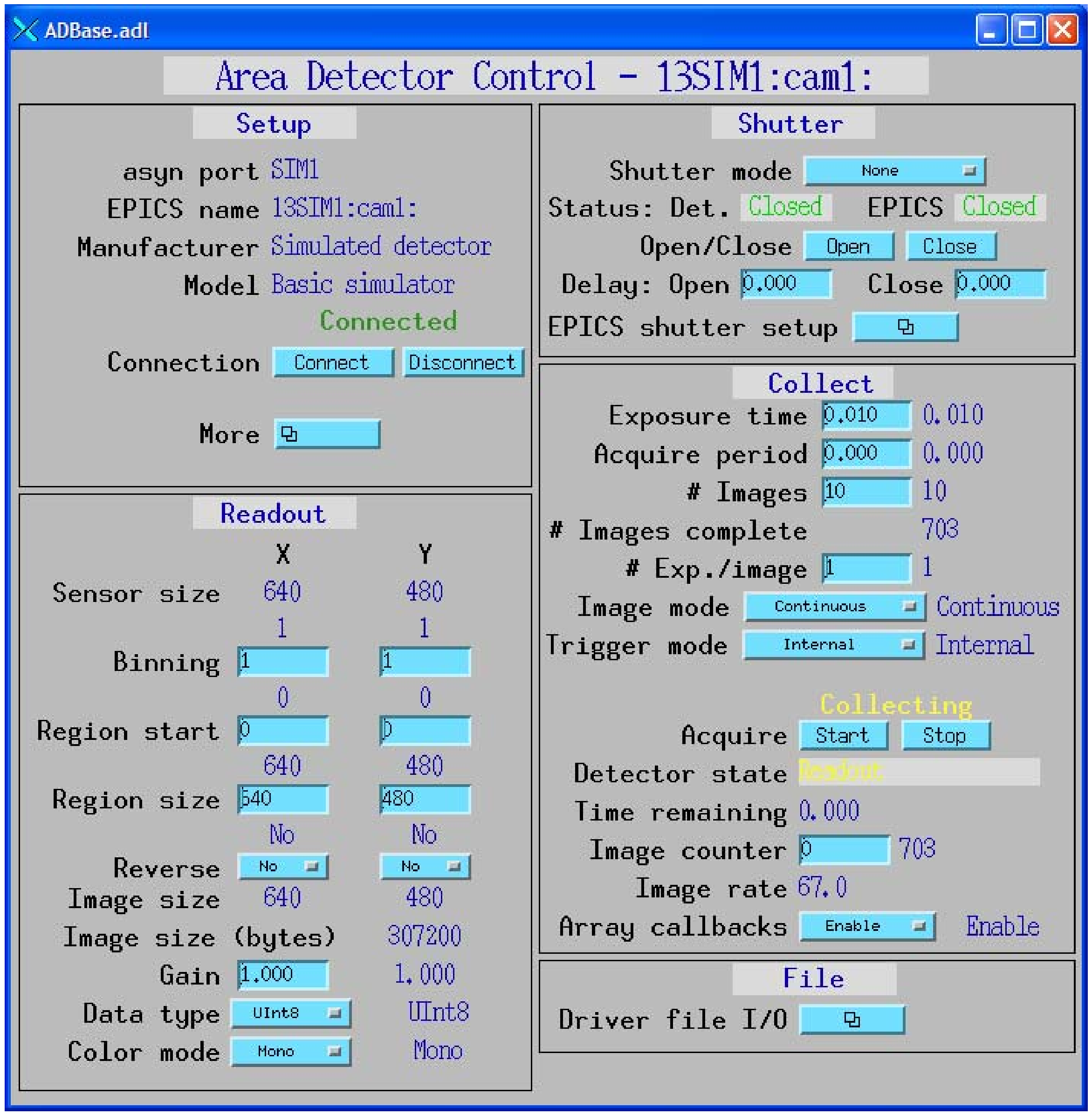
- ADDriver
  - Base C++ class from which detector drivers derive. Handles details of EPICS interfaces, and other common functions.
- Simulation driver
  - Produces calculated images up to very high rates. Implements nearly all basic parameters, including color. Useful as a model for real detector drivers, and to test plugins and clients.
- Prosilica driver
  - Gigabit Ethernet cameras, mono and color
  - High resolution, high speed, e.g. 1360x1024 at 30 frames/second = 40MB/second.
- Firewire (IEEE-1396 DCAM) (NEW)
  - Vendor-independent Firewire camera drivers for Linux and Windows
- Roper driver (NEW)
  - Princeton Instruments and Photometrics cameras controlled via WinView

Detector drivers (continued)

- PVCAM driver (NEW)
  - Princeton Instruments and Photometrics cameras controlled via PVCAM library
- Pilatus driver (NEW)
  - Pilatus pixel-array detectors.
- marCCD driver (NEW)
  - Rayonix (MAR-USA) CCD x-ray detectors
- ADSC driver (NEW)
  - ADSC CCD detectors
- mar345 driver (NEW)
  - marResearch mar345 online image plate
- Perkin-Elmer driver (NEW)
  - Perkin-Elmer amorphous silicon detectors

ADBase.adl – Generic control screen

- Works with any detector
- Normally write custom control for each detector type to hide unimplemented features and expose driver-specific features



Area Detector Control - 13SIM1:cam1:

Setup

asyn port SIM1
EPICS name 13SIM1:cam1:
Manufacturer Simulated detector
Model Basic simulator
Connected
Connection Connect Disconnect
More

Readout

	X	Y
Sensor size	640	480
	1	1
Binning	1	1
	0	0
Region start	0	0
	640	480
Region size	640	480
	No	No
Reverse	No	No
Image size	640	480
Image size (bytes)		307200
Gain	1.000	1.000
Data type	UInt8	UInt8
Color mode	Mono	Mono

Shutter

Shutter mode None
Status: Det. Closed EPICS Closed
Open/Close Open Close
Delay: Open 0.000 Close 0.000
EPICS shutter setup

Collect

Exposure time 0.010 0.010
Acquire period 0.000 0.000
Images 10 10
Images complete 703
Exp./image 1 1
Image mode Continuous Continuous
Trigger mode Internal Internal
Collecting
Acquire Start Stop
Detector state Readout
Time remaining 0.000
Image counter 0 703
Image rate 67.0
Array callbacks Enable Enable

File

Driver file I/O

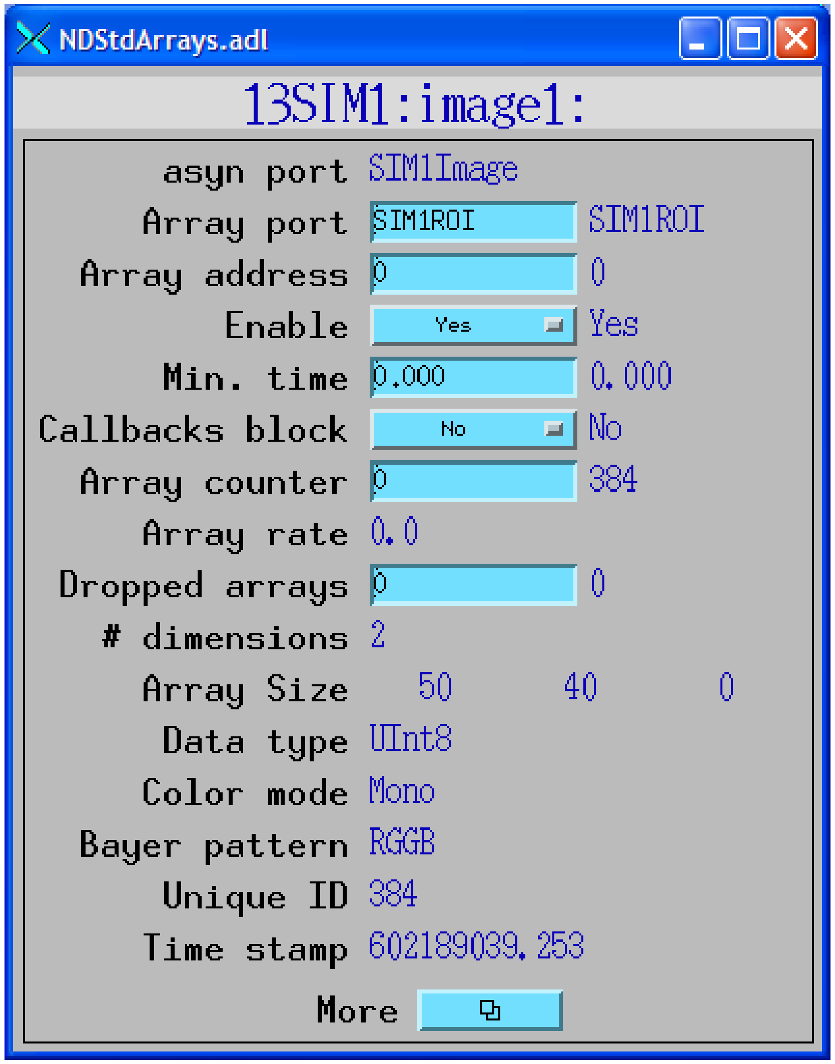
Plugins

- 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
  - 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.

Plugins (continued)

- NDPlugInStdArrays
  - Receives arrays (images) from device drivers, converts to standard arrays, e.g. waveform records.
  - This plugin is what EPICS channel access viewers normally talk to.
- NDPluginROI
  - Performs region-of-interest calculations
  - Select a subregion. Optionally bin, reverse in either direction, convert data type. Optionally highlight borders within other ROIs.
  - Optionally compute statistics on the region (min, max, mean, total, net, etc.)
  - Optionally compute histogram of the region, make available as waveform for client plotting.
- NDPluginColorConvert (NEW)
  - Convert from one color model to another (Bayer, RGB pixel, row or planar interleave)
- NDPluginMJPEG (NEW)
  - MJPEG server that allows viewing images in a Web browser.

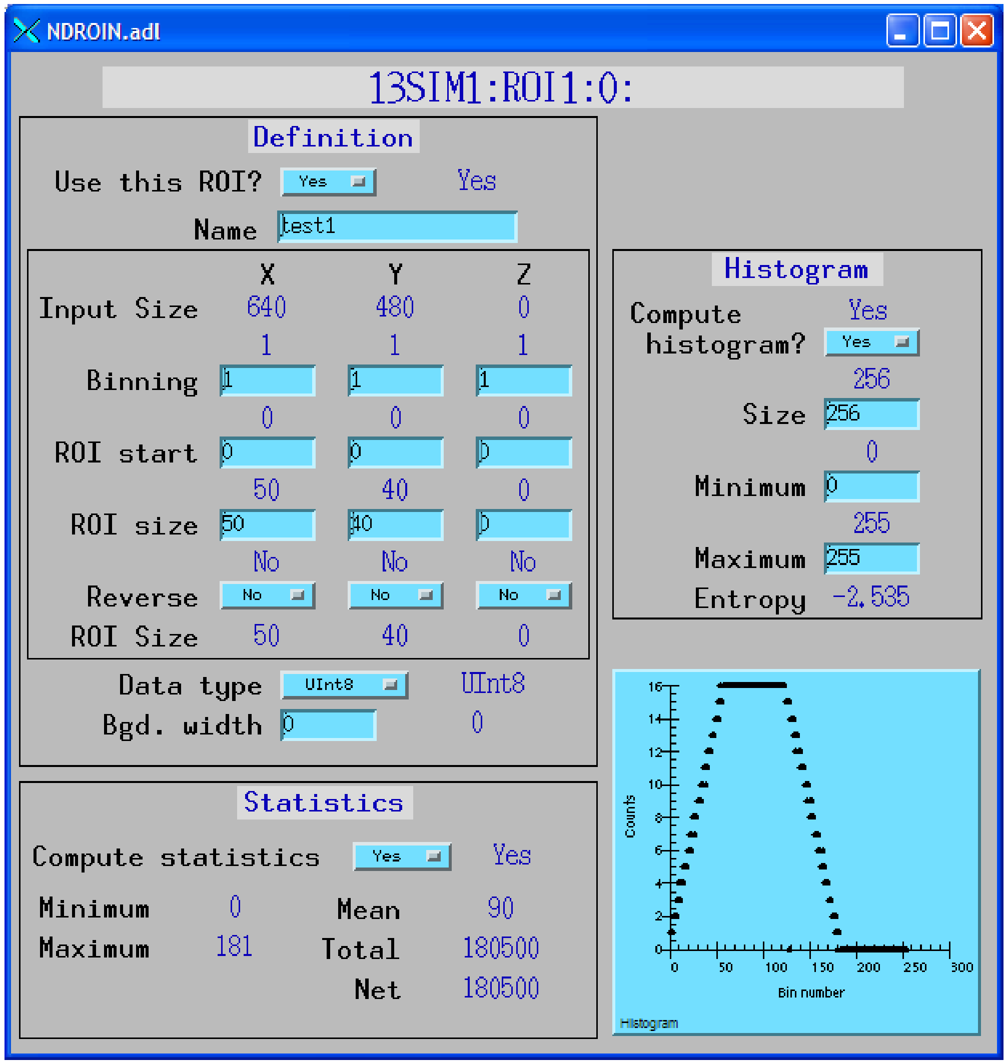
StdArrays and ROI plugin displays



NDStdArrays.adl

13SIM1:image1:

asyn port	SIM1Image	
Array port	SIM1ROI	SIM1ROI
Array address	0	0
Enable	Yes	Yes
Min. time	0.000	0.000
Callbacks block	No	No
Array counter	0	384
Array rate	0.0	
Dropped arrays	0	0
# dimensions	2	
Array Size	50 40 0	
Data type	UInt8	
Color mode	Mono	
Bayer pattern	RGGB	
Unique ID	384	
Time stamp	602189039.253	
More		



NDROIN.adl

13SIM1:ROI1:0:

Definition

Use this ROI? Yes Yes

Name test1

	X	Y	Z
Input Size	640	480	0
Binning	1 / 1	1 / 1	1 / 1
ROI start	0 / 0	0 / 0	0 / 0
ROI size	50 / 50	40 / 40	0 / 0
Reverse	No / No	No / No	No / No
ROI Size	50	40	0

Data type UInt8 UInt8

Bgd. width 0 0

Statistics

Compute statistics Yes Yes

Minimum	0	Mean	90
Maximum	181	Total	180500
		Net	180500

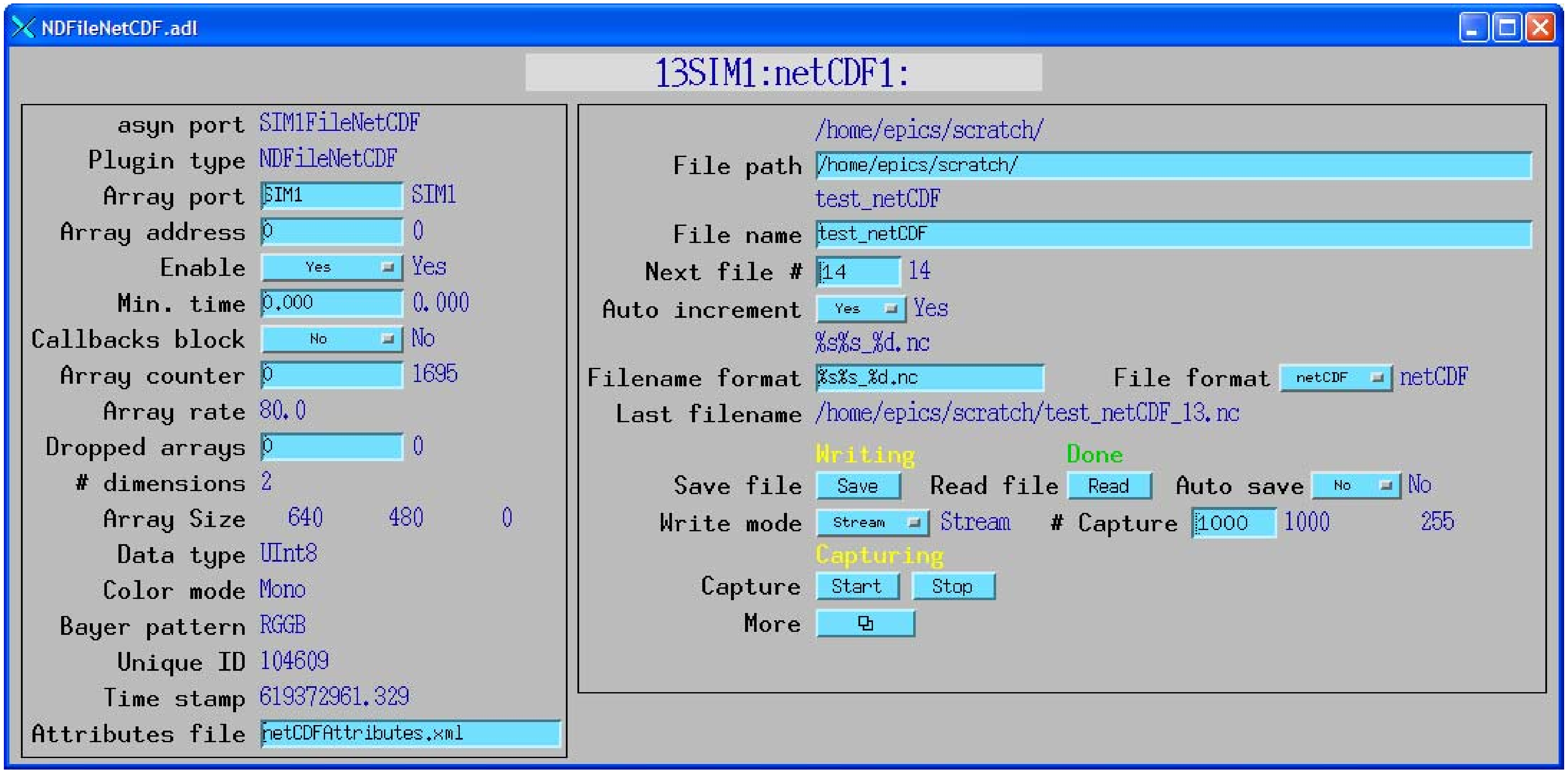
Histogram

Compute histogram?	Yes	Yes
Size	256	256
Minimum	0	0
Maximum	255	255
Entropy		-2.535

Plugins: NDPluginFile

- Saves NDArrays to disk
- 3 modes:
  - Single array per disk file
  - Capture N arrays in memory, write to disk either multiple files or as a single large file (for file formats that support this.)
  - Stream arrays to a single large disk file
- File formats currently supported
  - TIFF (NEW)
  - JPEG (with compression control) (NEW)
  - netCDF (popular self-describing binary format, supported by Unidata at UCAR)
  - NeXus (NEW) (standard file format for neutron and x-ray communities, based on HDF, which is another popular self-describing binary format, richer than netCDF).
- netCDF and HDF save NDAttributes as well as NDArray data to the file
- In addition to file saving plugins, many vendor libraries also support saving files (e.g. marCCD, mar345, Pilatus, etc.) and this is supported at the driver level.
- File saving plugin can be used instead of or in addition to vendor file saving
  - Can add additional metadata vendor does not support
  - Could write JPEGS for Web display every minute, etc.

NDPluginFile display



NDFileNetCDF.adl

13SIM1:netCDF1:

asyn port	SIM1FileNetCDF	
Plugin type	NDFileNetCDF	
Array port	SIM1	SIM1
Array address	0	0
Enable	Yes	Yes
Min. time	0.000	0.000
Callbacks block	No	No
Array counter	0	1695
Array rate	80.0	
Dropped arrays	0	0
# dimensions	2	
Array Size	640 480 0	
Data type	UInt8	
Color mode	Mono	
Bayer pattern	RGGB	
Unique ID	104609	
Time stamp	619372961.329	
Attributes file	netCDFAttributes.xml	

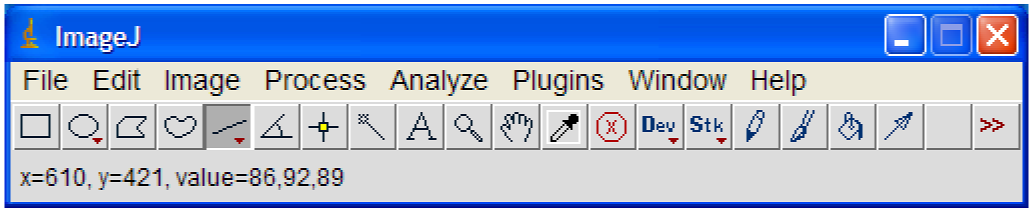
File path	/home/epics/scratch/	/home/epics/scratch/
File name	test_netCDF	test_netCDF
Next file #	14	14
Auto increment	Yes	Yes
Filename format	%s%s_%d.nc	%s%s_%d.nc
File format	netCDF	netCDF
Last filename	/home/epics/scratch/test_netCDF_13.nc	
Save file	Save	Writing
Read file	Read	Done
Auto save	No	No
Write mode	Stream	Stream
# Capture	1000	1000 255
Capture	Start Stop	Capturing
More		

Example: streaming 80 frames/second of 640x480 video to netCDF file, no dropped frames.

Viewers

- areaDetector allows generic viewers to be written that receive images as EPICS waveform records over Channel Access
- Current viewers include:
  - ImageJ plugin EPICS_AD_Display. (NEW) ImageJ is a very popular image analysis program, written in Java, derived from NIH Image.
  - IDL EPICS_AD_Display.
  - IDL areaVision (NEW) (Stephen Mudie from AS)
  - MJPEG server (NEW) allows image display in any Web browser

ImageJ Viewer



x=610, y=421, value=86,92,89

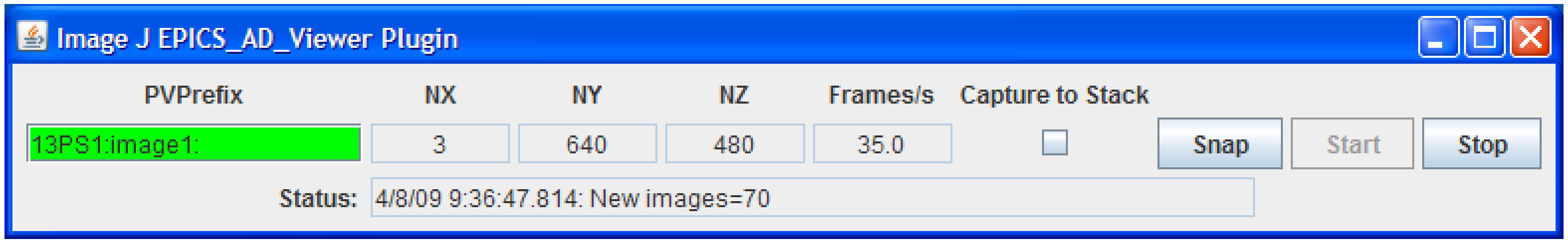
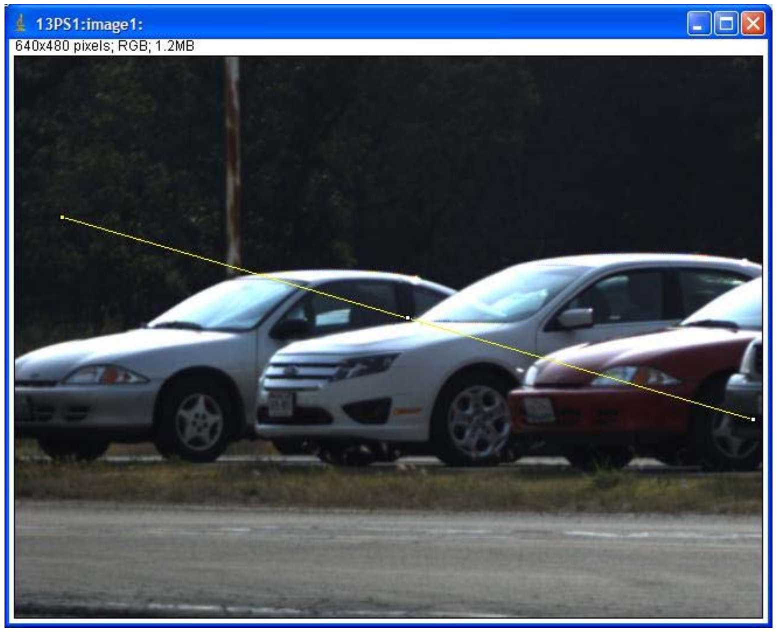


Image J EPICS_AD_Viewer Plugin

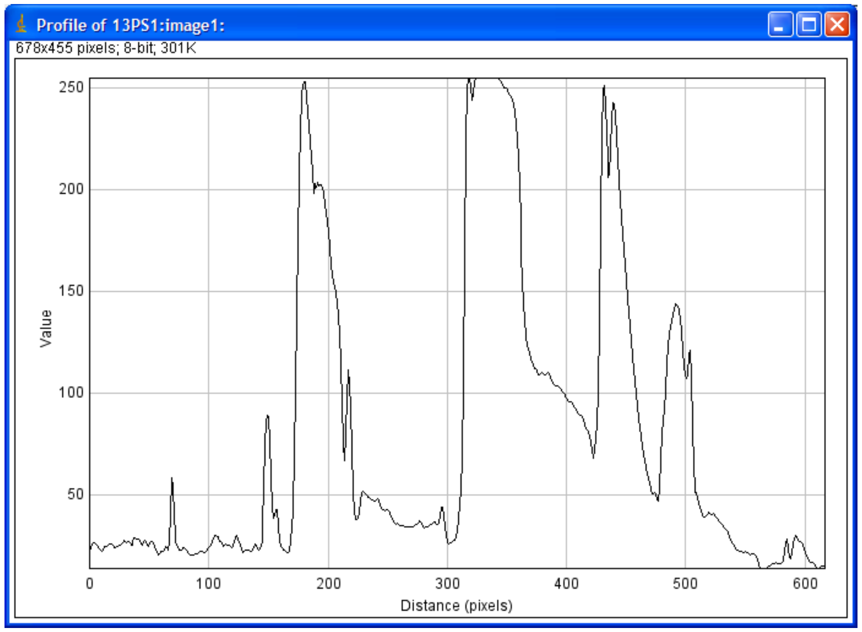
PVPrefix	NX	NY	NZ	Frames/s	Capture to Stack
13PS1:image1:	3	640	480	35.0	☐

Status: 4/8/09 9:36:47.814: New images=70



13PS1:image1:

640x480 pixels; RGB; 1.2MB

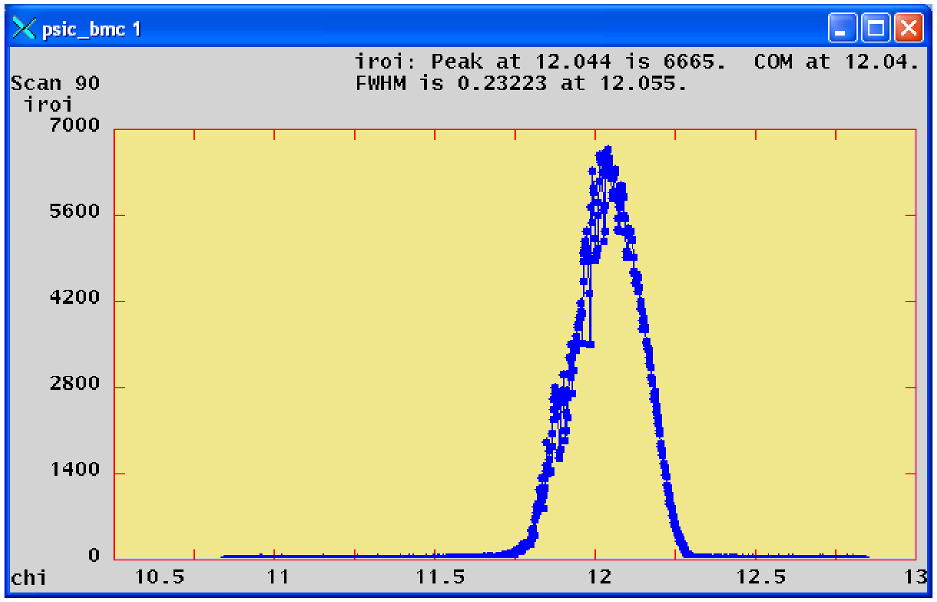


Profile of 13PS1:image1:

678x455 pixels; 8-bit; 301K

Performance Example with Pilatus driver

- SPEC used to collect 1000 points using trajectory scanning mode with the Newport XPS motor controller. Hardware trigger of Pilatus from XPS.
- Relative scan of the chi axis from -2 degrees to +2 degrees with 1000 points at .02 seconds/point
- Coordinated motion of the phi, kappa and omega axes.
- Theoretical time 20.0 second, actual time 20.8 seconds
- Includes time to save all 1000 images to disk (366 MB), Pilatus driver to read each file, correct bad pixels and flat field, compute ROIs, and post the ROIs and 1000 images to EPICS.



iroi: Peak at 12.044 is 6665. COM at 12.04.
FWHM is 0.23223 at 12.055.

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)
- Now installed on all APS detector pool 2D detectors.
- Can get source code and pre-built binaries (Linux, Windows, Cygwin) from our Web site:
  – http://cars.uchicago.edu/software/epics/areaDetector
- Can also get code on SourceForge
  – http://epics.svn.sourceforge.net/viewvc/epics/applications/trunk/areaDetector

Acknowledgments

- Brian Tieman (APS) Roper PVCAM driver
- John Hammonds (APS) Perkin-Elmer driver and NeXus file saving plugin
- Tim Madden (APS) initial version of ImageJ viewer
- Stephen Mudie (Australian Synchrotron) areaViewer IDL-based viewer
- Ulrik Pedersen and Tom Cobb (Diamond) MJPEG plugin, Linux Firewire driver
- Lewis Muir (U Chicago) ADSC CCD driver
- NSF-EAR and DOE-Geosciences for support of GSECARS where most of this work was done

- Thanks for your attention!!!