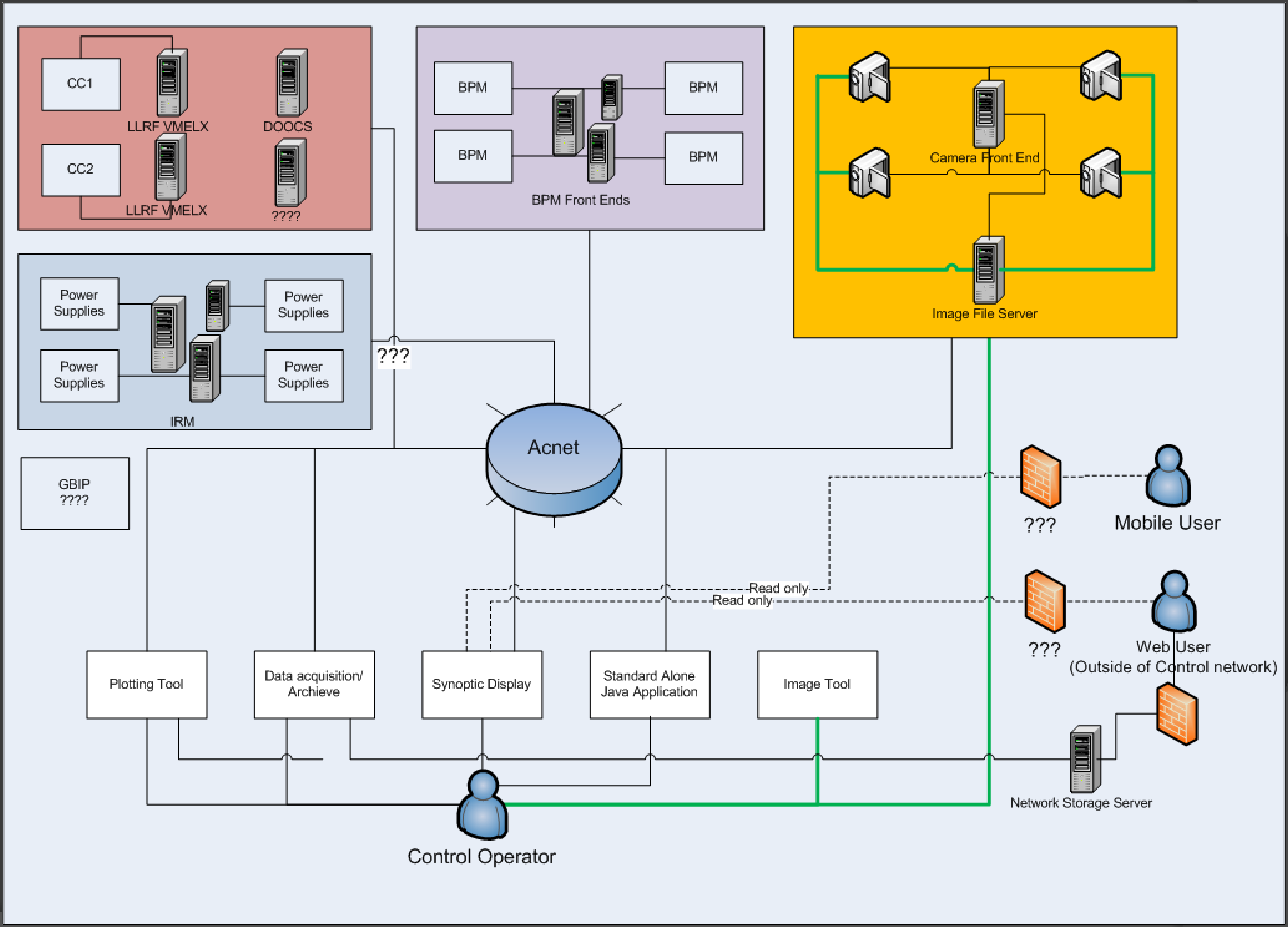
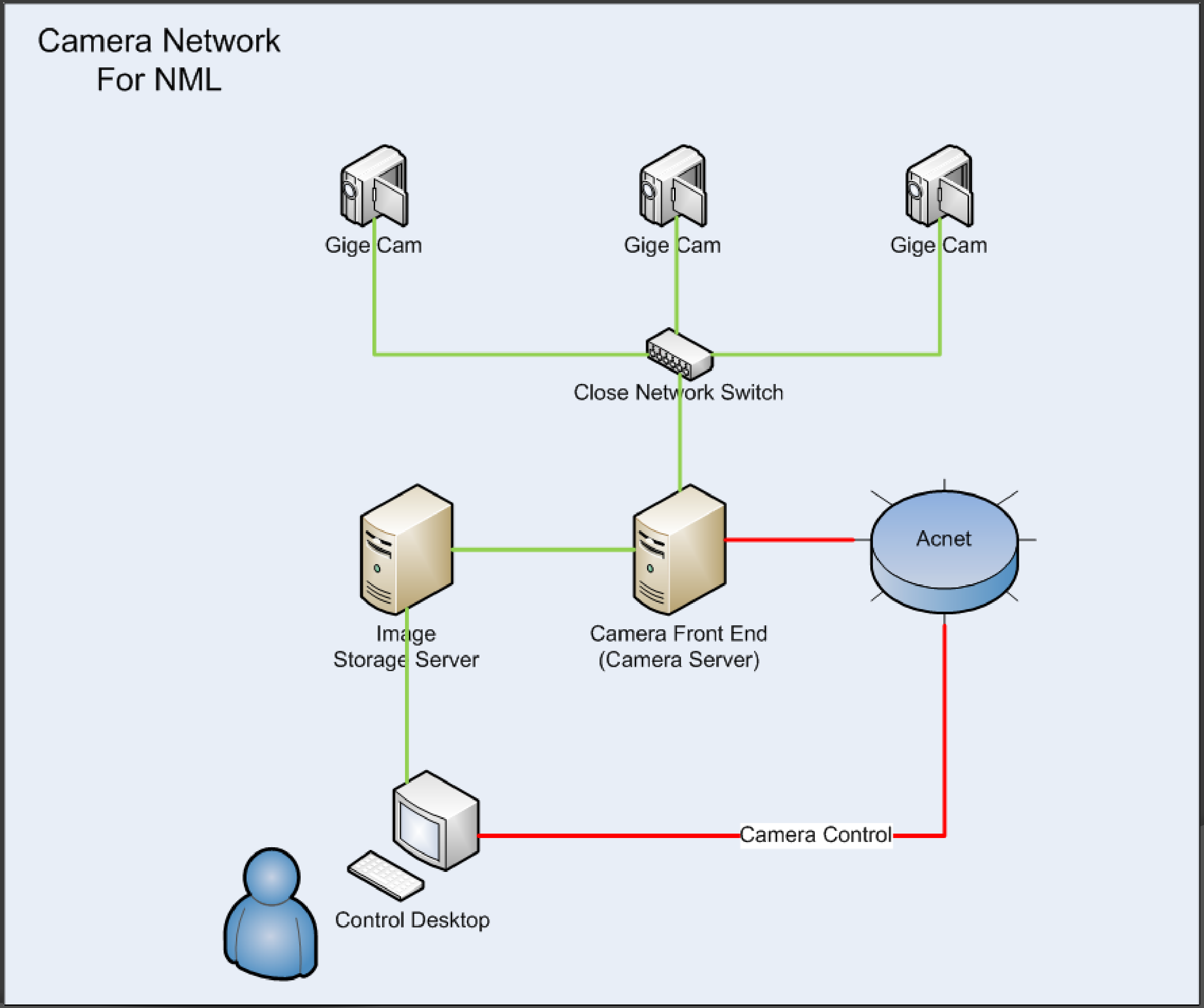
Experimental implementation with Prosilica GE 1350

NEW MUON LAB CAMERA SERVER

Proposed Structure



Proposed Camera network in Detail



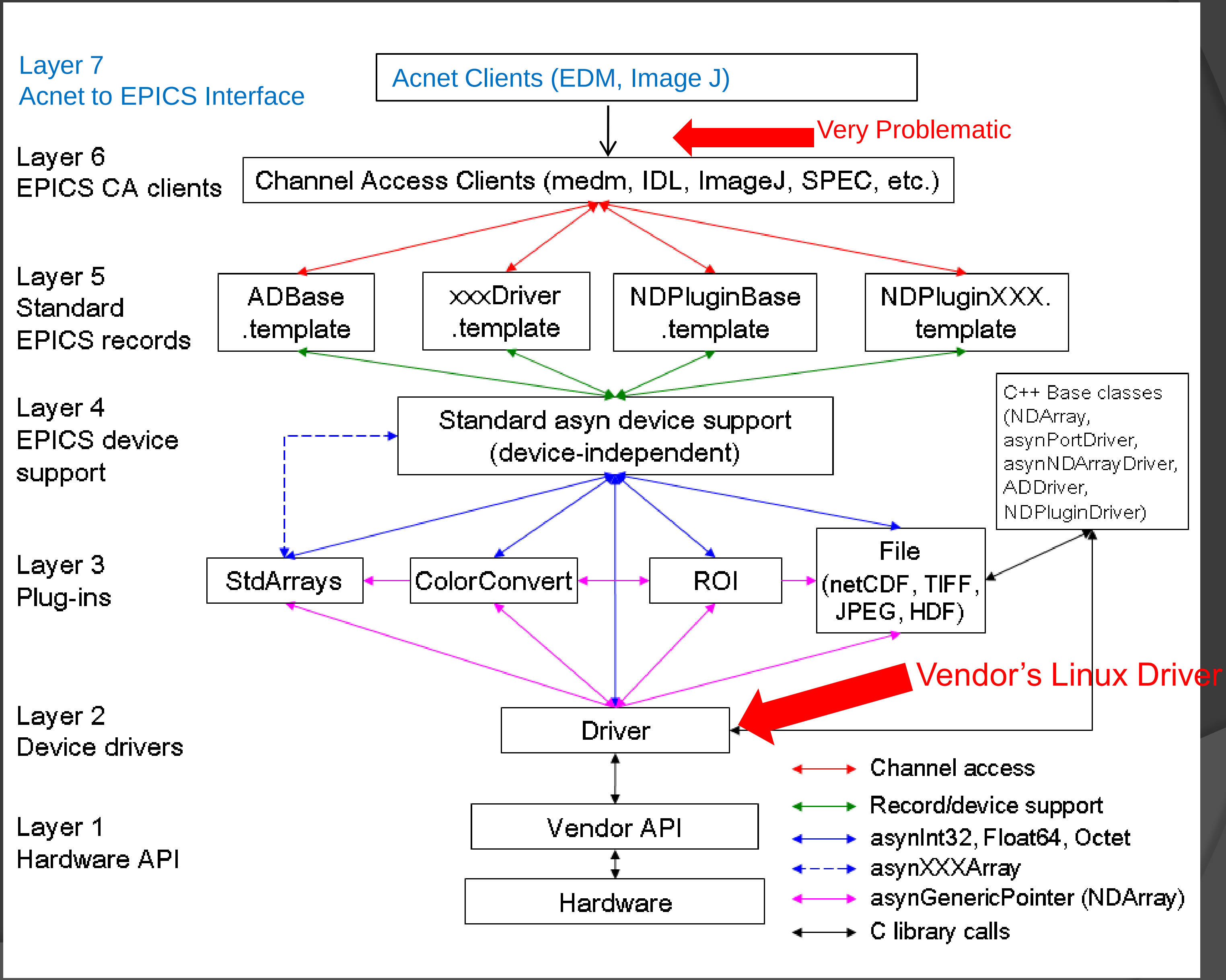
Components

- Switch
  - Cisco Catalyst 2960 G Switch
    - Connectivity
      - 24 or 48 ports of Gigabit Ethernet desktop connectivity
      - Industry first PoE+ with up to 30W per port to support the latest PoE+ capable devices
      - Optional four 1 Gigabit Ethernet SFP or two 10 Gigabit Ethernet SFP+ Uplinks
- Camera Server PC
  - Dell Precision T3400
    - Dual Ethernet cards
      - Connect to both control network and camera network

Camera Front End

- Area Detector – Developed by CAR9 at U of C
  - An extended module on top of EPICS system
  - Provide a standard interface defining the functions and parameters that a detector driver must support.
  - Provide a set of base EPICS records that will be present for every detector using this module. This allows the use of generic EPICS clients for displaying images and controlling cameras and detectors.
  - Allow easy extensibility to take advantage of detector-specific features beyond the standard parameters. Have high-performance.
  - Applications can be written to get the detector image data through EPICS, but an interface is also available to receive the detector data at a lower-level for very high performance.
  - Provide a mechanism for device-independent real-time data analysis such as regions-of-interest and statistics.

Camera Front End –cont.



Layer 7
Acnet to EPICS Interface

Acnet Clients (EDM, Image J)

Very Problematic

Layer 6
EPICS CA clients

Channel Access Clients (medm, IDL, ImageJ, SPEC, etc.)

Layer 5
Standard
EPICS records

ADBase .template

xxxDriver .template

NDPluginBase .template

NDPluginXXX. template

Layer 4
EPICS device
support

Standard asyn device support (device-independent)

C++ Base classes (NDArray, asynPortDriver, asynNDArrayDriver, ADDriver, NDPluginDriver)

Layer 3
Plug-ins

StdArrays

ColorConvert

ROI

File (netCDF, TIFF, JPEG, HDF)

Vendor's Linux Driver

Layer 2
Device drivers

Driver

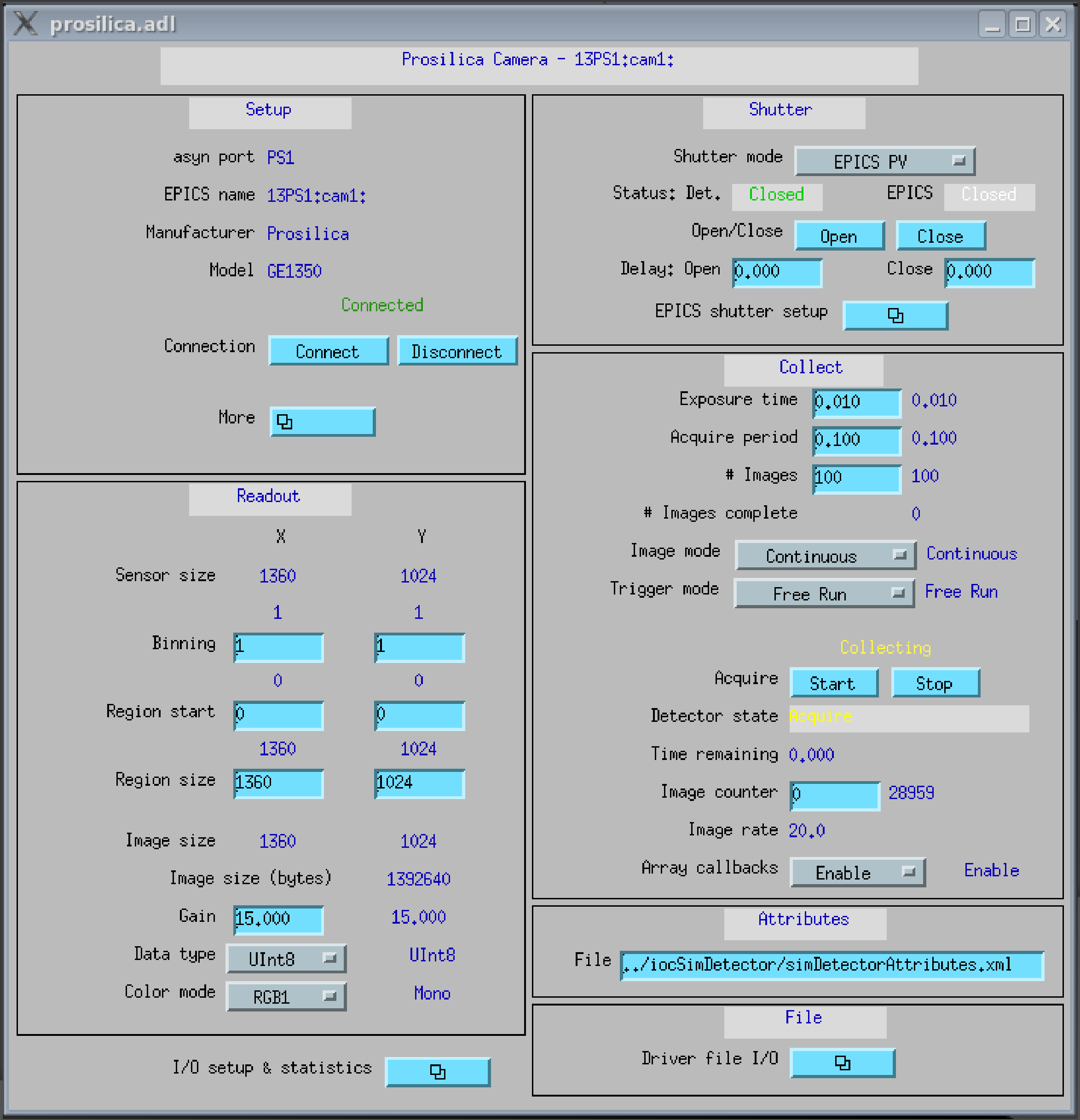
Layer 1
Hardware API

Vendor API

Hardware

↔ Channel access
↔ Record/device support
↔ asynInt32, Float64, Octet
⇠⇢ asynXXXArray
↔ asynGenericPointer (NDArray)
↔ C library calls

Camera Main Control



prosilica.adl

Prosilica Camera - 13PS1:cam1:

Setup

asyn port PS1

EPICS name 13PS1:cam1:

Manufacturer Prosilica

Model GE1350

Connected

Connection Connect Disconnect

More

Readout

	X	Y
Sensor size	1360	1024
	1	1
Binning	1	1
	0	0
Region start	0	0
	1360	1024
Region size	1360	1024
Image size	1360	1024

Image size (bytes) 1392640

Gain 15.000 15.000

Data type UInt8 UInt8

Color mode RGB1 Mono

I/O setup & statistics

Shutter

Shutter mode EPICS PV

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 0.100 0.100

Images 100 100

Images complete 0

Image mode Continuous Continuous

Trigger mode Free Run Free Run

Collecting

Acquire Start Stop

Detector state Acquire

Time remaining 0.000

Image counter 0 28959

Image rate 20.0

Array callbacks Enable Enable

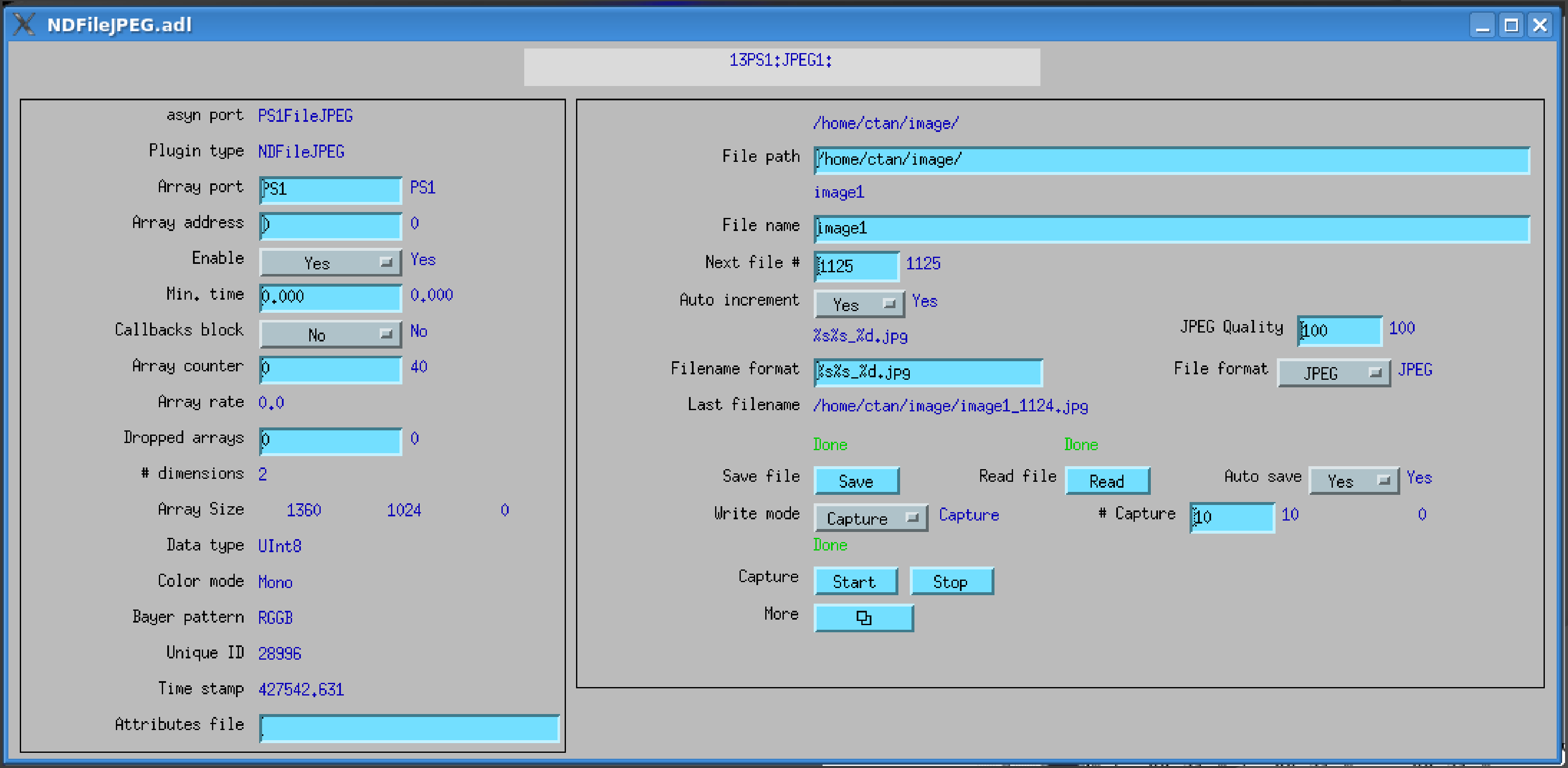
Attributes

File ../iocSimDetector/simDetectorAttributes.xml

File

Driver file I/O

Image Acquisition



NDFileJPEG.adl

13PS1:JPEG1:

asyn port	PS1FileJPEG		
Plugin type	NDFileJPEG		
Array port	PS1	PS1	
Array address	0	0	
Enable	Yes	Yes	
Min. time	0.000	0.000	
Callbacks block	No	No	
Array counter	0	40	
Array rate	0.0		
Dropped arrays	0	0	
# dimensions	2		
Array Size	1360	1024	0
Data type	UInt8		
Color mode	Mono		
Bayer pattern	RGGB		
Unique ID	28996		
Time stamp	427542.631		
Attributes file			

File path: /home/ctan/image/ (readback /home/ctan/image/)

File name: image1 (readback image1)

Next file #: 1125 1125

Auto increment: Yes Yes

JPEG Quality: 100 100

Filename format: %s%s_%d.jpg (readback %s%s_%d.jpg)

File format: JPEG JPEG

Last filename: /home/ctan/image/image1_1124.jpg

Save file: Save — Done

Read file: Read — Done

Auto save: Yes Yes

Write mode: Capture Capture

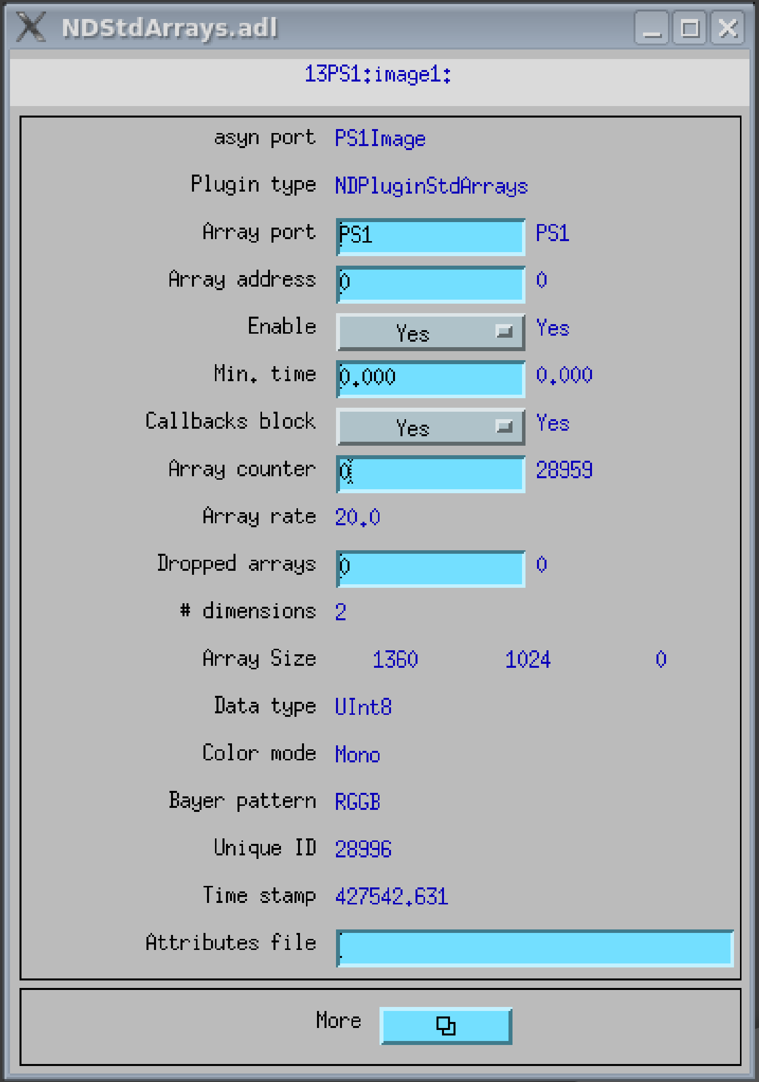
Capture: 10 10 0

Done

Capture: Start Stop

More

Image Viewer – Image J (Java based)



13PS1:image1:

asyn port	PS1Image	
Plugin type	NDPluginStdArrays	
Array port	PS1	PS1
Array address	0	0
Enable	Yes	Yes
Min. time	0.000	0.000
Callbacks block	Yes	Yes
Array counter	0	28959
Array rate	20.0	
Dropped arrays	0	0
# dimensions	2	
Array Size	1360 1024 0	
Data type	UInt8	
Color mode	Mono	
Bayer pattern	RGGB	
Unique ID	28996	
Time stamp	427542.631	
Attributes file		

DEMO

Start and stop

Single, Continuous, Stream

Image J viewer launch from apc-con01.fnal.gov

Can’t dump image to clx system right…

At 20 Hz (free run)
1.3 MB X 20 per second = 26 MB per second
1 min consume = 1560 MB = 1.5 GB
30 min lunch without shutting off the image acquisition will cost 46.8 GB.

At 5 Hz
1.3 X 5 = 6.5 MB per second
1 min consume = 390 MB
30 min lunch without shutting off the image acquisition will cost 11.7 GB

Major Problems

- EPICS Getaway
  - BUSY module (allows CA clients to indicate completion in a way that works with EPICS putNotify/ca_put_callback mechanism) -- the CA gateway converts ca_put() into a ca_put_callback() to the IOC
    - ".. for most records the ca_put_callback() will complete almost immediately and you won't notice the difference. However the busy record type is deliberately designed to not report completion until all of the underlying operations kicked off by your put have completed, and in your case that probably means until the camera's exposure has completed, which I'm guessing is taking longer than the default 1 second time-out that caput() waits for the CA operation to complete.."
- Workaround
  - Go through non-gatewayed path (disabled a lot other features that need use BUSY)
- Proposed Solutions
  - Upgrade Fermi lab EPICS version to 3.14.11 that supports latest BUSY module.

Major Problems

- MEDM
  - Lab only support EDM
  - MEDM to EDM convector tool is not bullet proof
- Storage
  - What is the best mechanize for image retriever?
  - Storage space?

Future Action

- Camera
  - Resolve compatibility issues between EPICS and Acnet
  - Ensure all EDMs running from CLX system
  - Deploy 2nd camera to the network and analysis the network traffic
  - Move existing camera setting to A0 south cave
  - Obtain images, use MatLab to determine the image quality
- Define the constraints for camera use.
  - Rules for how operator should use the camera
    - Image limits, concurrency limits, continuous run limits