

Data AcQuisition Overview & Online Analysis

Photon Controls Data Systems Data Acquisition Group

**Chris Ford, Wilfred Ghonsalves,
Philip Hart, Chris O'Grady,
Jack Pines, Jana Thayer,
Tomy Tsai, Matt Weaver**

Outline

■ DAQ Overview

- Architecture

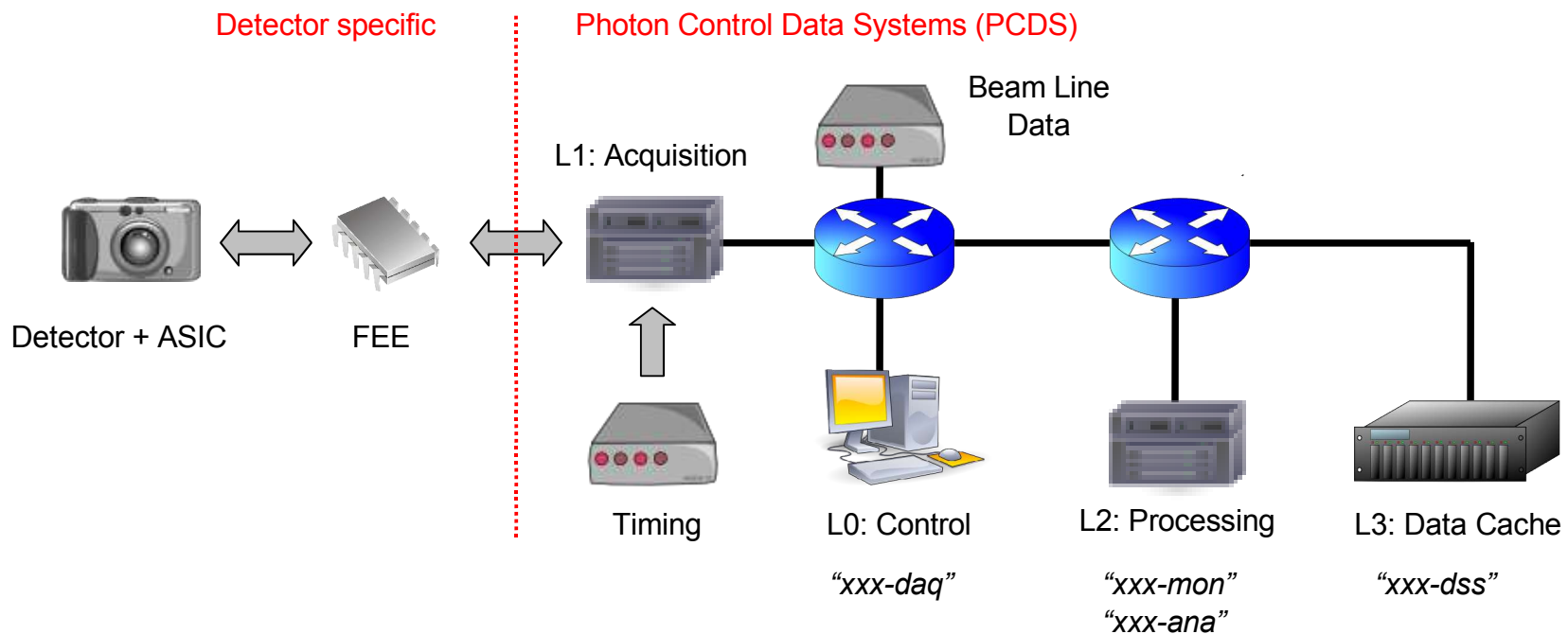
- Operation

■ Online Analysis

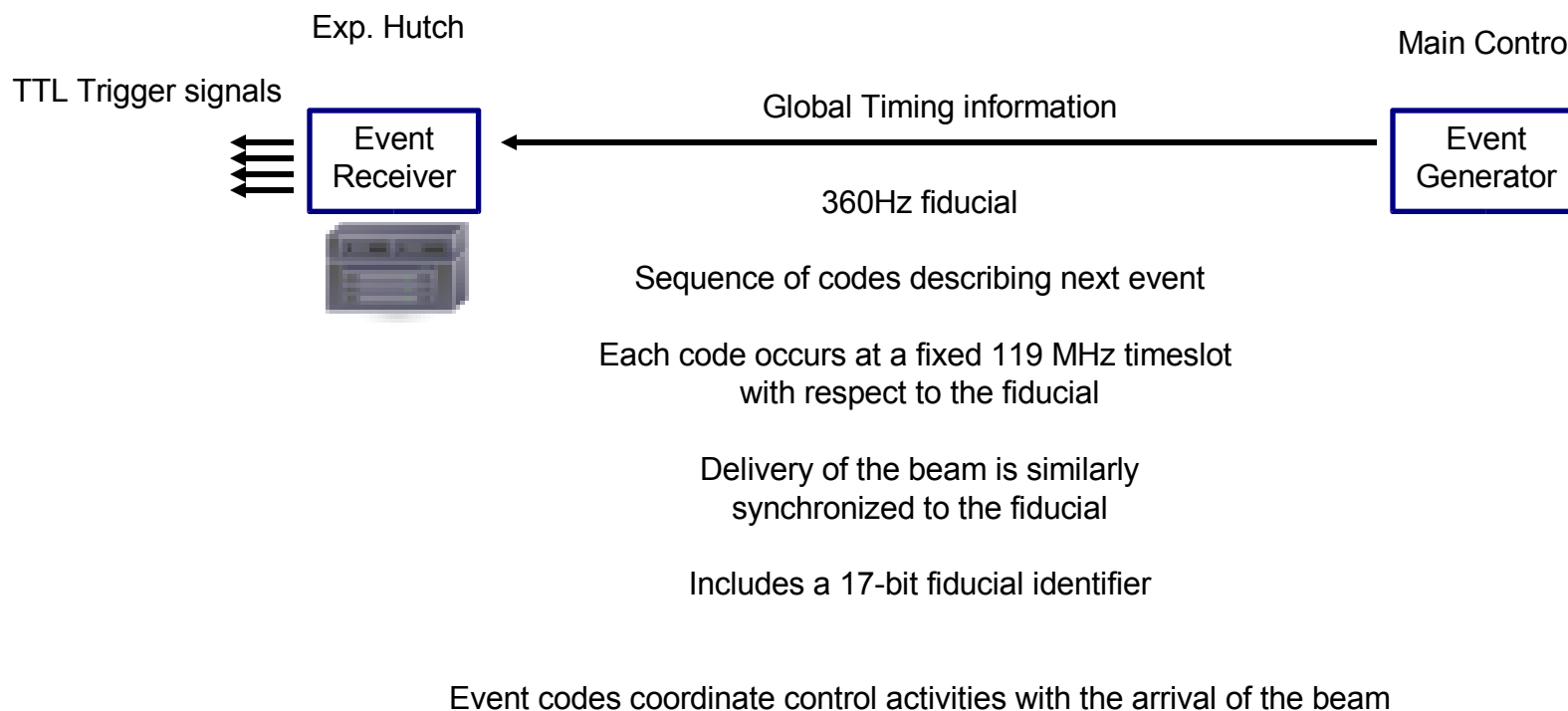
- Core monitoring

- User analysis

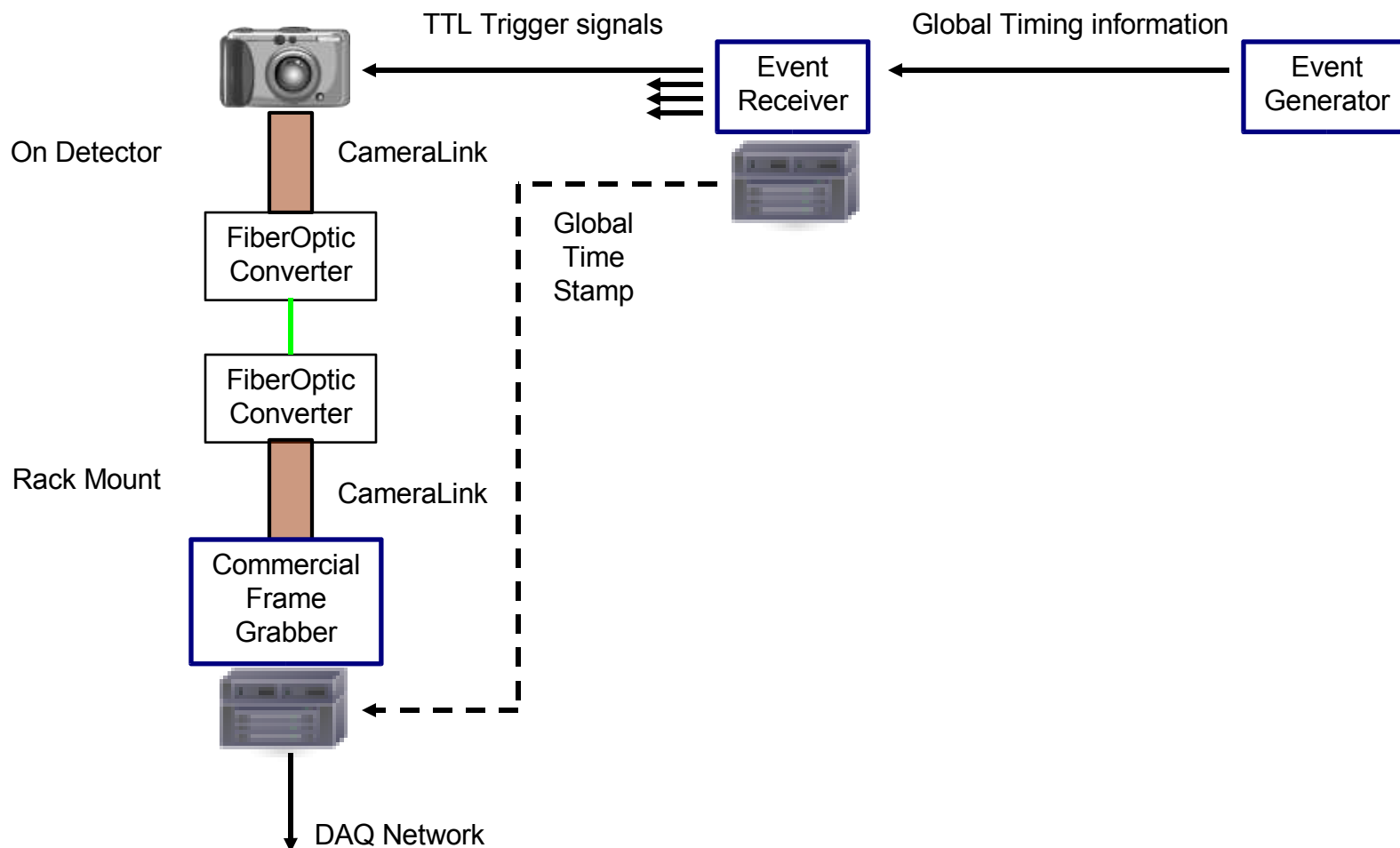
DAQ Architecture



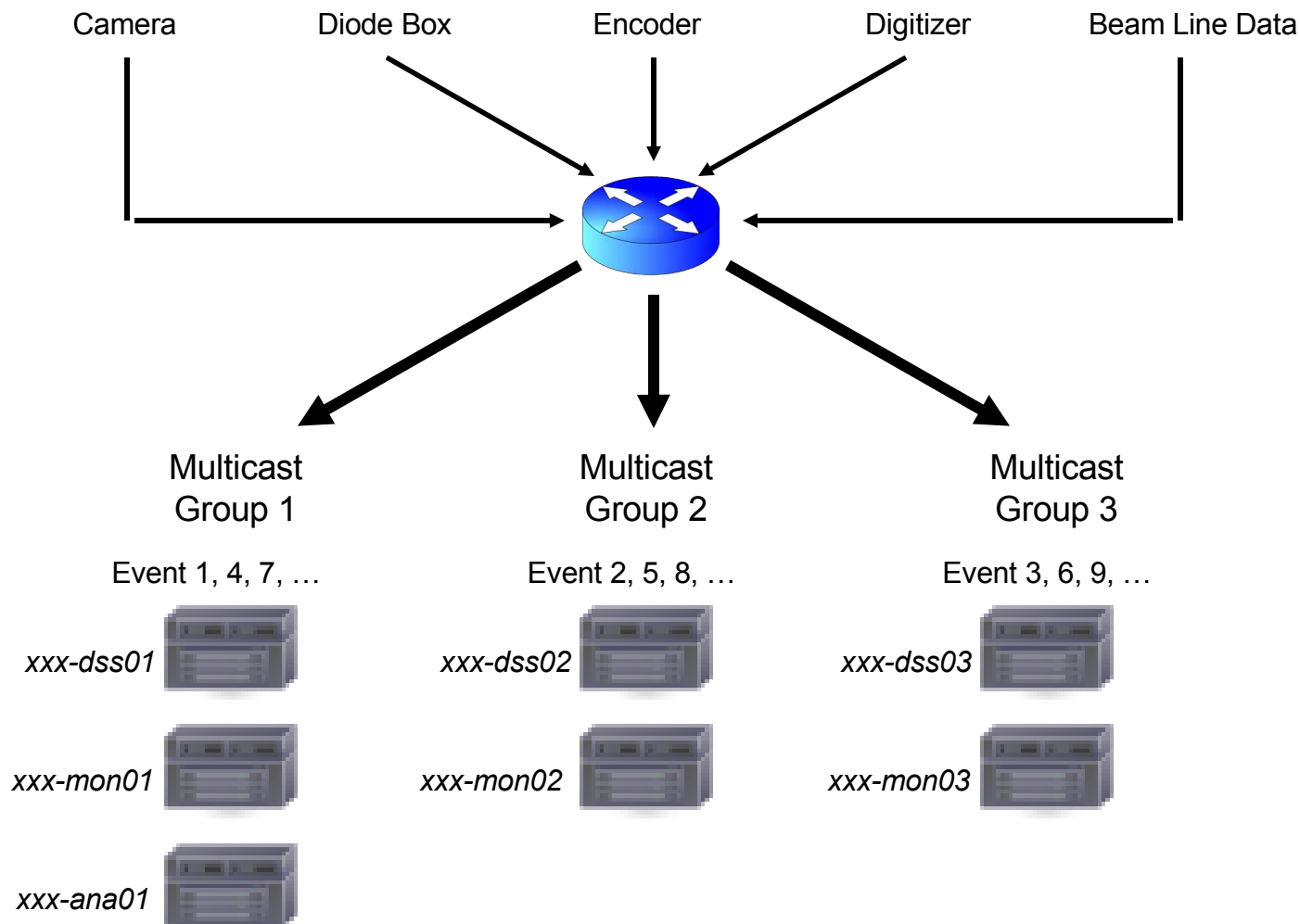
Timing System : Generating Triggers



Example Detector Readout : Commercial Cameras



DAQ Network : Event Building



DAQ Control : Launching a “Run”



L0: Control

“xxx-daq”

Configure Triggers and
Detector Readout

Go!

Watch Progress

The screenshot shows the DAQ Control window with the following sections:

- Configuration:** Type: BEAM, Edit, Scan buttons.
- Partition:** Select, Display buttons.
- Control:** Record Run (checked), Target State dropdown, Last Transition yellow field.
- Detector:** Control State: NOT READY (red bar).
- Run Statistics:** Duration, Events, Damaged, Size, Damage Stats button.
- Log:** 16:19:50: Run control started. Connected to platform.

DAQ Control : Configuring Triggers



L0: Control

"xxx-daq"

Event Enable	Pulse Code	Pulse Polarity	Pulse Delay (sec)	Pulse Width (sec)	CSPAD-1	CSPAD-2	SB1-IPM-1	SB2-IPM-1	END-OPAL-1	END-OPAL-2		
<input checked="" type="checkbox"/>	140	Pos	0.00089	1e-05	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	140	Pos	0.000905	1.0084e-07	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	140	Pos	0.0007	0.0003	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>												
<input type="checkbox"/>												
<input type="checkbox"/>												
<input type="checkbox"/>												
<input type="checkbox"/>												

Detector readout is triggered by arrival of an event code
 (140 = beam on this fiducial)
 to gate the sampling/digitization with a TTL signal

DAQ Control : Configuring Readout



L0: Control

“xxx-daq”

Run Delay	2			
Event Code	40			
Inact Run Mode	RunButDrop			
Activ Run Mode	RunAndSendTriggeredByTTL			
Test Data Indx	0			
Bad ASIC Mask (hex)	0			
Sector Mask (hex)	c0000000			

Quad	0	1	2	3
Quad Registers				
Shift Sel	4 4 4 4	4 4 4 4	4 4 4 4	4 4 4 4
Edge Sel	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
Read Clk Set	2	2	2	2
Read Clk Hold	1	1	1	1
Data Mode	2	2	2	2
PRst Sel	1	1	1	1
Acq Delay	280	280	280	280
Int Time	1500	1500	1500	1500
Dig Delay	960	960	960	960
Amp Idle	0	0	0	0
Inj Total	0	0	0	0
Row/Col Shift	5	5	5	5
Digital Pots Fields				
Vref	186	186	186	186
Vin	186	186	186	186
RampCurrR1	4	4	4	4
RampCurrR2	37	37	37	37
RampCurrRef	0	0	0	0
RampVoltRef	97	97	97	97
CompBias1	255	255	255	255

--

Most detector readout configurations are simple.
Few parameters require changing.

Data Processing

- Monitor node processing (“xxx-mon”)
 - Nodes register for event data multicasts
 - Receive each detector’s data and assemble complete events
 - Copy events to shared memory and push event pointer into a queue for application consumption
 - If the queue is full (processing bottleneck), events are dropped
- Data cache node processing (“xxx-dss”)
 - Nodes register for event data multicasts
 - Receive each detector’s data and assemble complete events
 - Record each event to online disk cache
 - Report event statistics to user console (#, size, health)
- Offline transfer
 - Run data is transferred from data cache nodes to offline storage
 - Transfer can start during recording as resources allow

Data Formats

- XTC (eXtended Tagged Container)
 - Online format : custom
 - Event-based (data belonging to same event are contiguous)
 - Serial access (but could be indexed)
 - Defined in “pdsdata” C++ package

- HDF5
 - Translated from XTC after the offline transfer
 - Random access
 - Standard

Online Analysis

- Core Online Monitoring
- User Shared Memory Application
- User Disk-Based Application

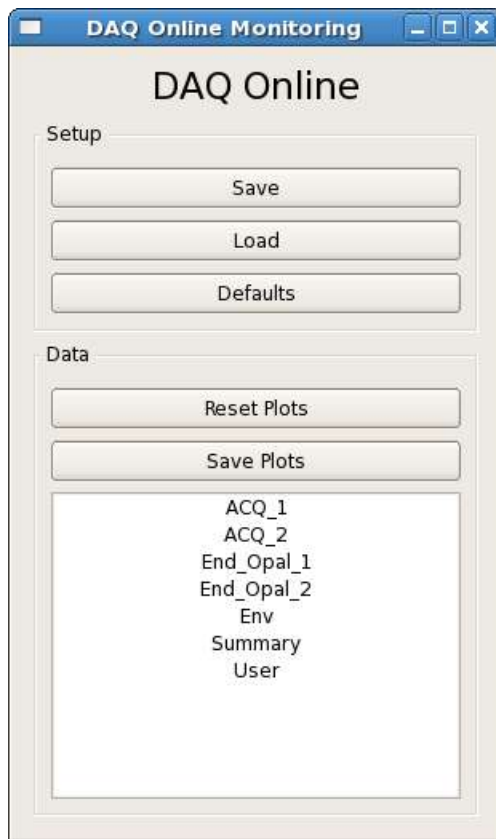
Core Online Monitoring

- Graphical display of raw data
 - Scalar data { diodes, encoders, beamline data }
 - Waveform data { sampling digitizers }
 - Image data { commercial and custom cameras }
- User configuration of fast, useful processing
 - Background subtraction
 - Sample/range selection
 - Projection
 - Filtering
 - Detector correlations
- Perform even more processing
 - User coding of specializations
 - Dynamic linking

Core Online Monitoring

Archive display setup (many plots)

List of primary event displays



Reset/Capture data for all plots (text format)

“Env” Display (scalars)

Channel selection

BeamLine Data
 { gas det.,
 e- bunch msmt }

Detector Scalars
 { diodes, encoders }

Slow controls
 { motor positions,
 voltages, ... }

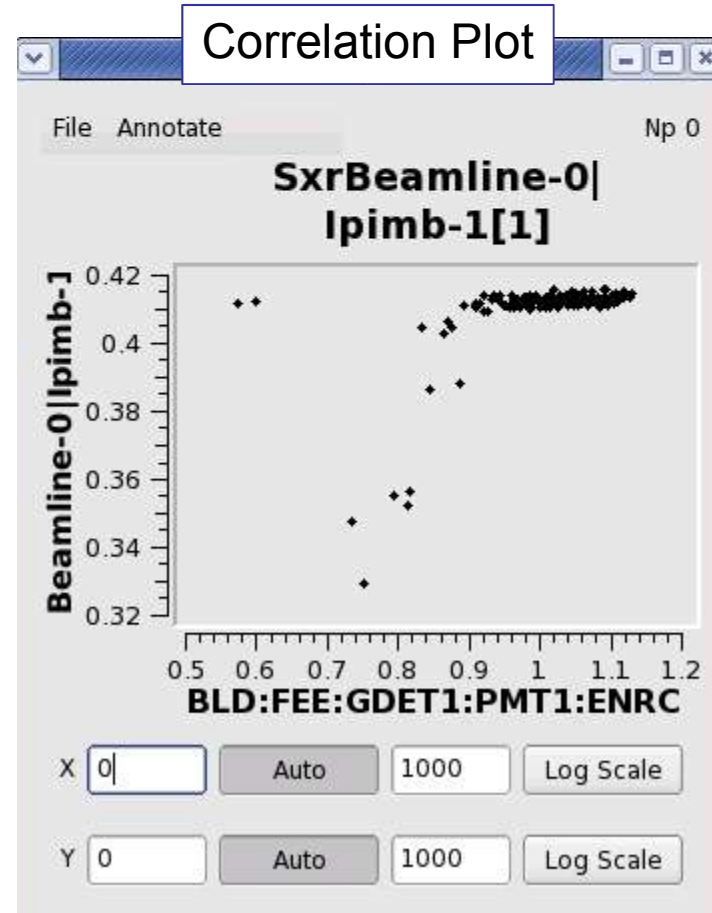
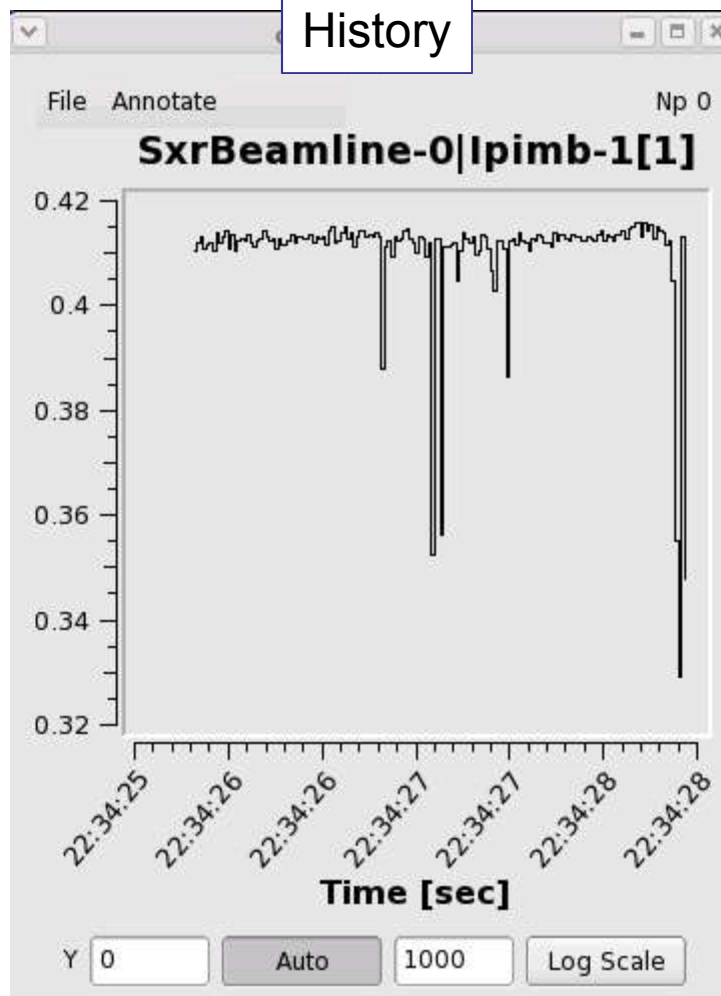
Plot Types

Frequency histogram

History stripchart

Correlation plot
 (vs other scalars)

“Env” Display (scalars)



Waveform Display (Acqiris digitizer)

Multiple channels

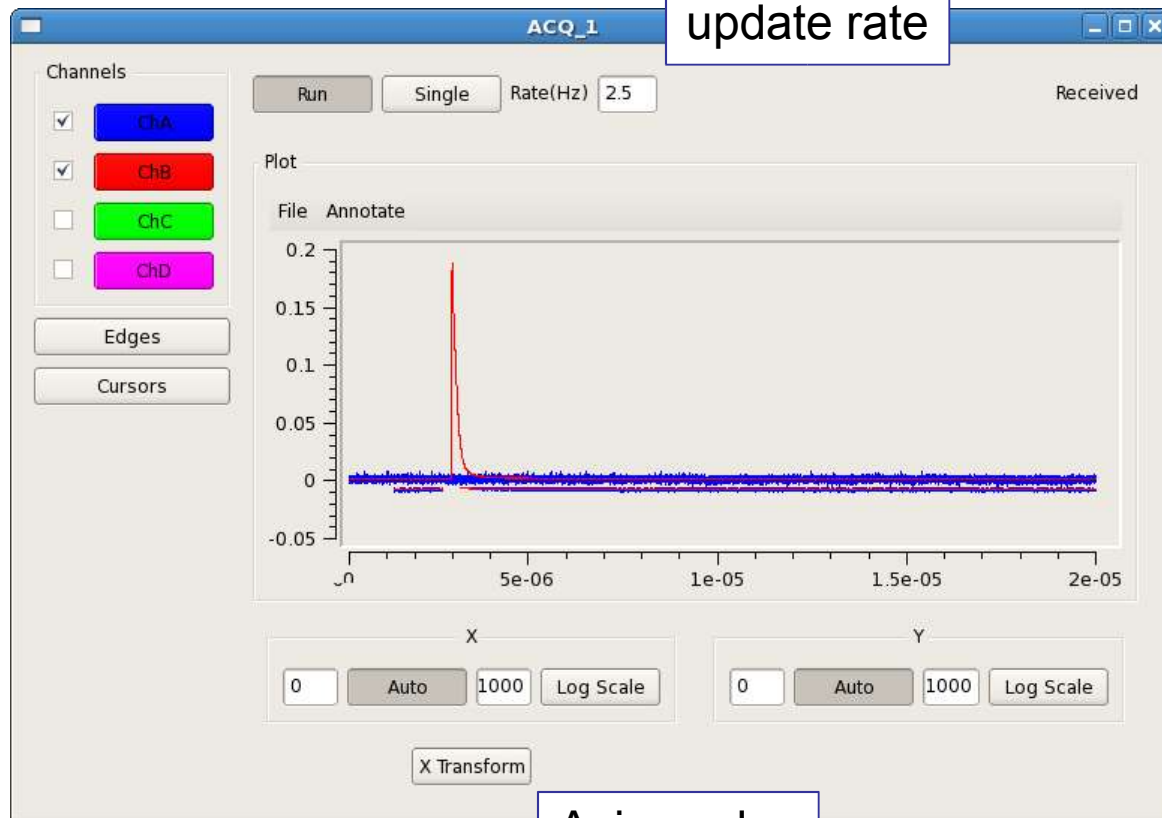
Superimpose reference, average, channel math, filter

Operations

Constant fraction discriminator

Sample/range selection

Display update rate



Axis scales

Channel Definitions

- Single event
- Average last N events
- Arithmetic expr.
- Reference wf

The screenshot shows two overlapping windows from a data acquisition software interface.

Channel Math Calculator: This window is titled "Channel Math" and contains a text input field with the expression $ACQ_1_ChA - ACQ_1_ChB + 0.01$. Below the input are buttons for "DEL", "CLR", and "RST". A numeric keypad follows, with channel names (ACQ_1_ChA, ACQ_1_ChB, ACQ_1_ChC, ACQ_1_ChD) and digits (0-9, ., ^, x, /, +, -, (), =) as buttons. At the bottom are "Apply" and "Cancel" buttons.

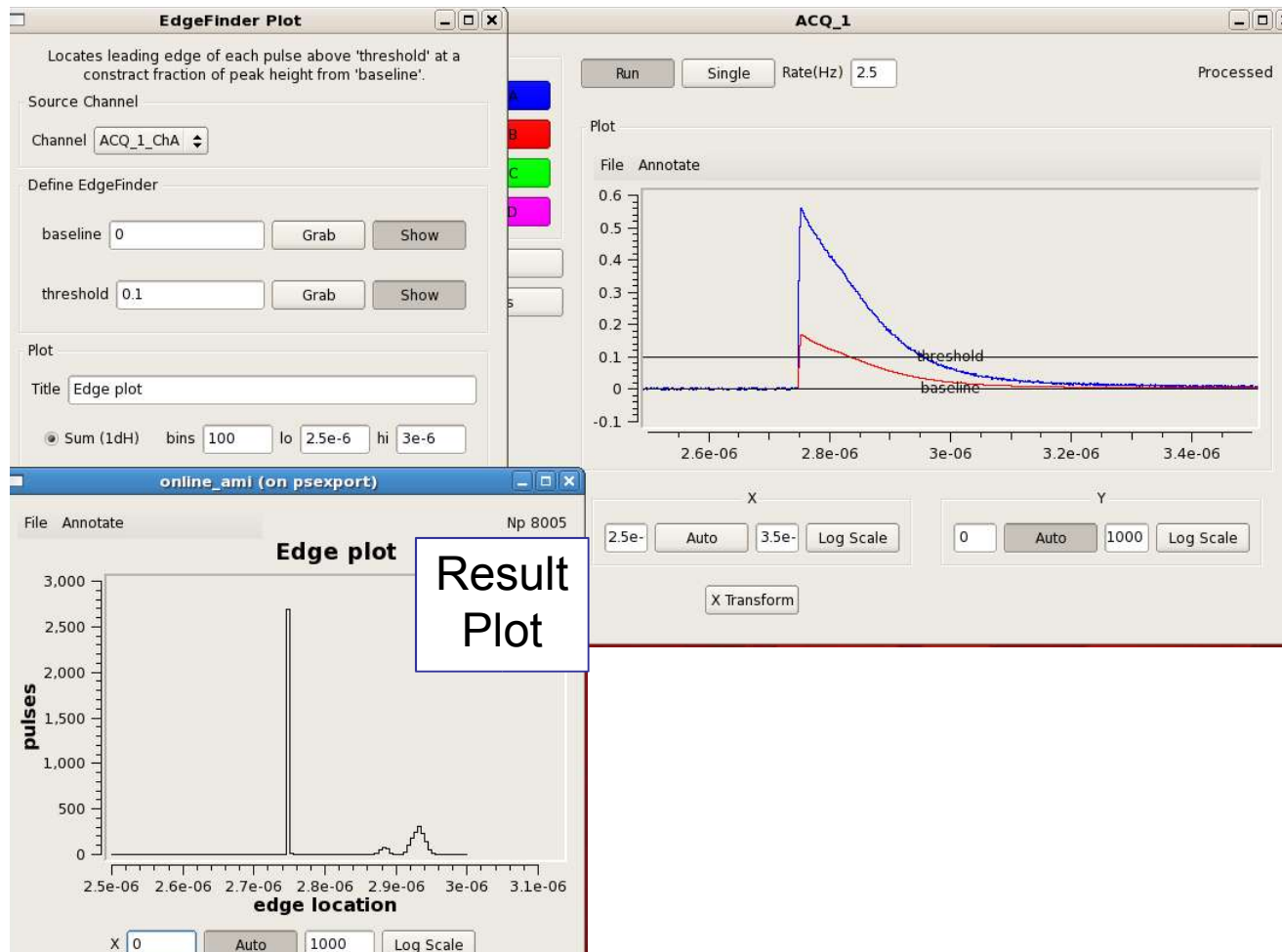
Multi-Channel Display: This window is titled "ACQ_1_ChC" and shows a plot of data. On the left, there is a "Channels" list with checkboxes for ChA (checked), ChB (checked), ChC (checked), and ChD (unchecked). Below the list are buttons for "Edges" and "Cursors". The plot area shows a noisy signal with three distinct traces: a green trace (ChC) with the highest amplitude, a blue trace (ChA), and a red trace (ChB) with the lowest amplitude. The x-axis is labeled "X" and "Y" with values from 0 to $1e-06$. The y-axis ranges from -0.01 to 0.02. At the bottom of the plot are "File" and "Annotate" buttons.

Operations – Constant Fraction Discr.

Channel selection

Discriminator settings

Result plot definition



Operations - Cursors

Define Cursors

Generate Expression

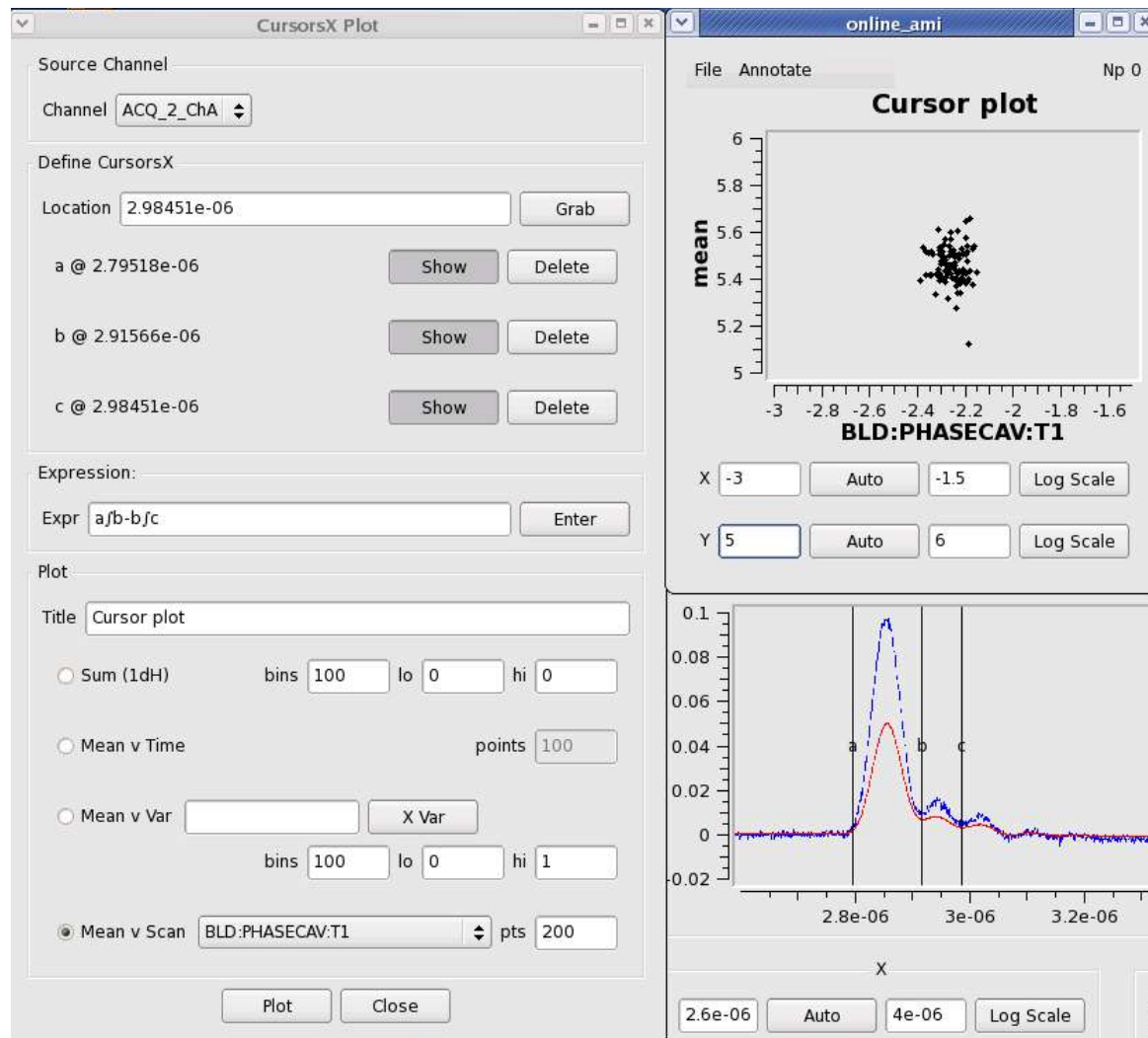
f (Value@cursor,
integral btwn)

Creates a scalar
result

Choose Plot

Same types as
“Env” plots

Can correlate
against any other
“Env” value



Operations – Cursors (contd)

- Scalar values which are generated by some operation can be correlated against the original set of scalar variables.
- Important application for scans.
- Appears in several places in the core monitoring interface.

Operations – Channel Filters

Can filter on logical combinations of “Env” scalar ranges

Example: eventcode indicating presence of optical laser

The screenshot displays the ACQ_2 software interface with several windows:

- ACQ_2 (Main):** Shows channel selection (ChA, ChB, ChC, ChD), a plot of data, and control buttons (Run, Single, Rate(Hz) 2.5).
- online_ami (ChA):** Shows a plot of ChA data with a peak around 2.8e-06.
- online_ami (ChB):** Shows a plot of ChB data with a peak around 2.8e-06.
- ACQ_2_ChA (Define Conditions):** Shows the condition $A := 0 \leq \text{DAQ:EVR0:Evt142} \leq 0$.
- ACQ_2_ChB (Define Conditions):** Shows the condition $A := 1 \leq \text{DAQ:EVR0:Evt142} \leq 1$.
- ACQ_2_ChB (Expression):** Shows the expression A .

An information box at the bottom states: "An EXPRESSION is a set of CONDITIONS separated by the operators A n B : logical AND of A and B".

Camera Display (custom CCD)

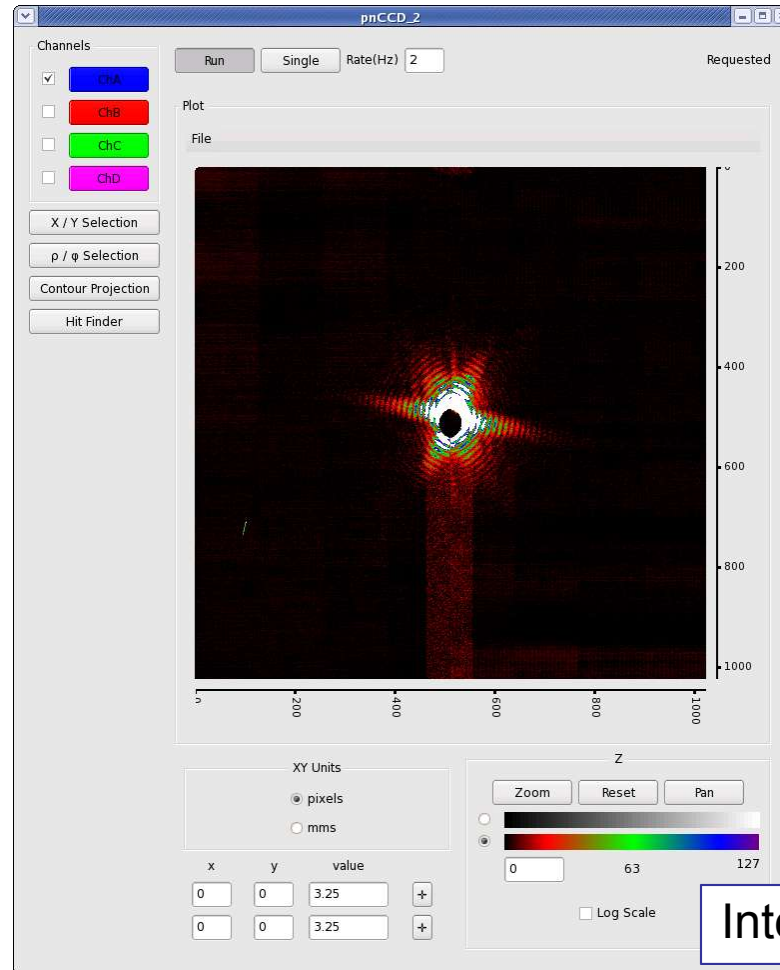
Operations

Rectangular
selection/projection

Annular
selection/projection

Contour
projection

Hit
mapping



Intensity scale

Operations – X/Y projection

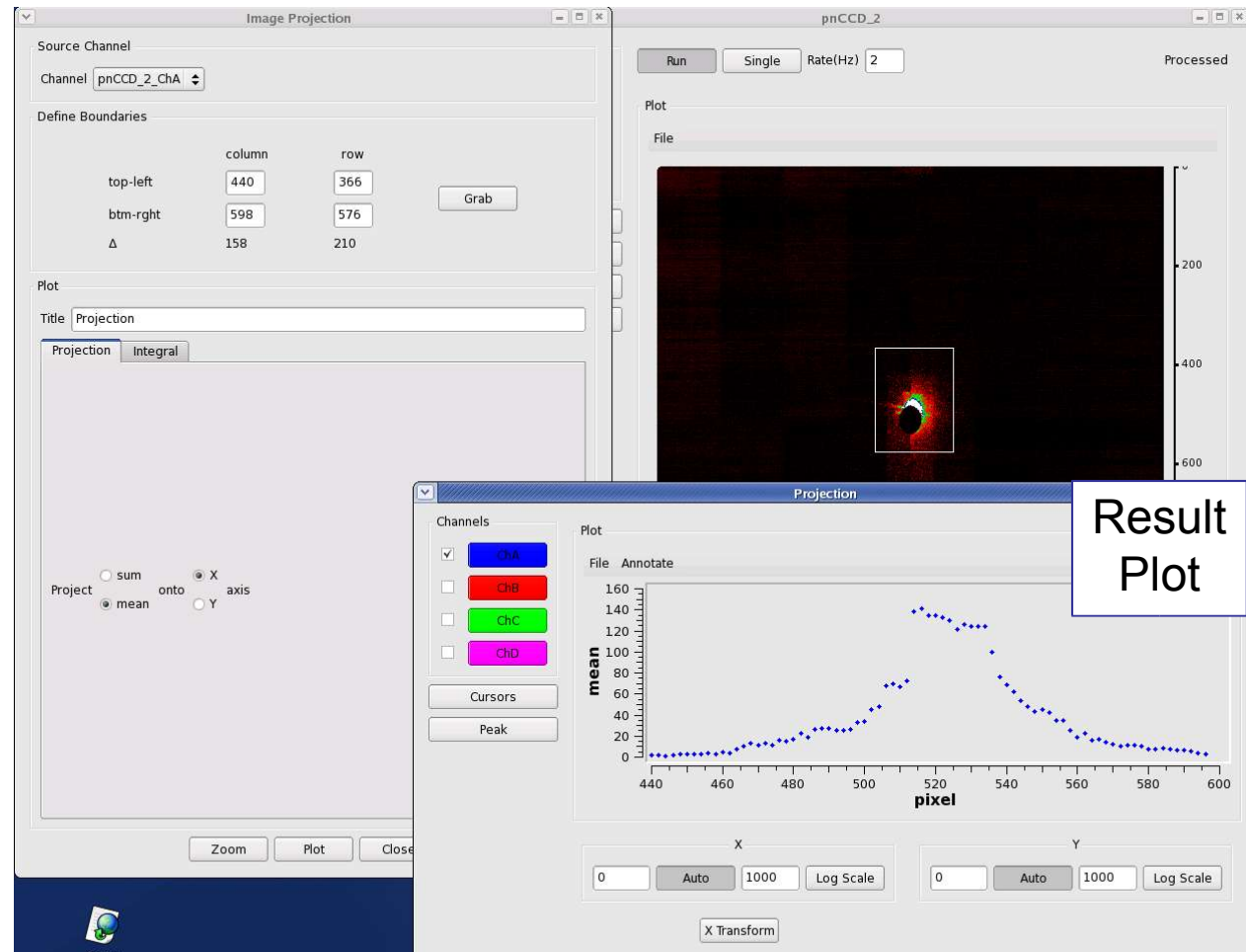
Define Region

Rectangular coordinates

Choose Projection

Direction/Axis

Sum or Average (normalize by area)

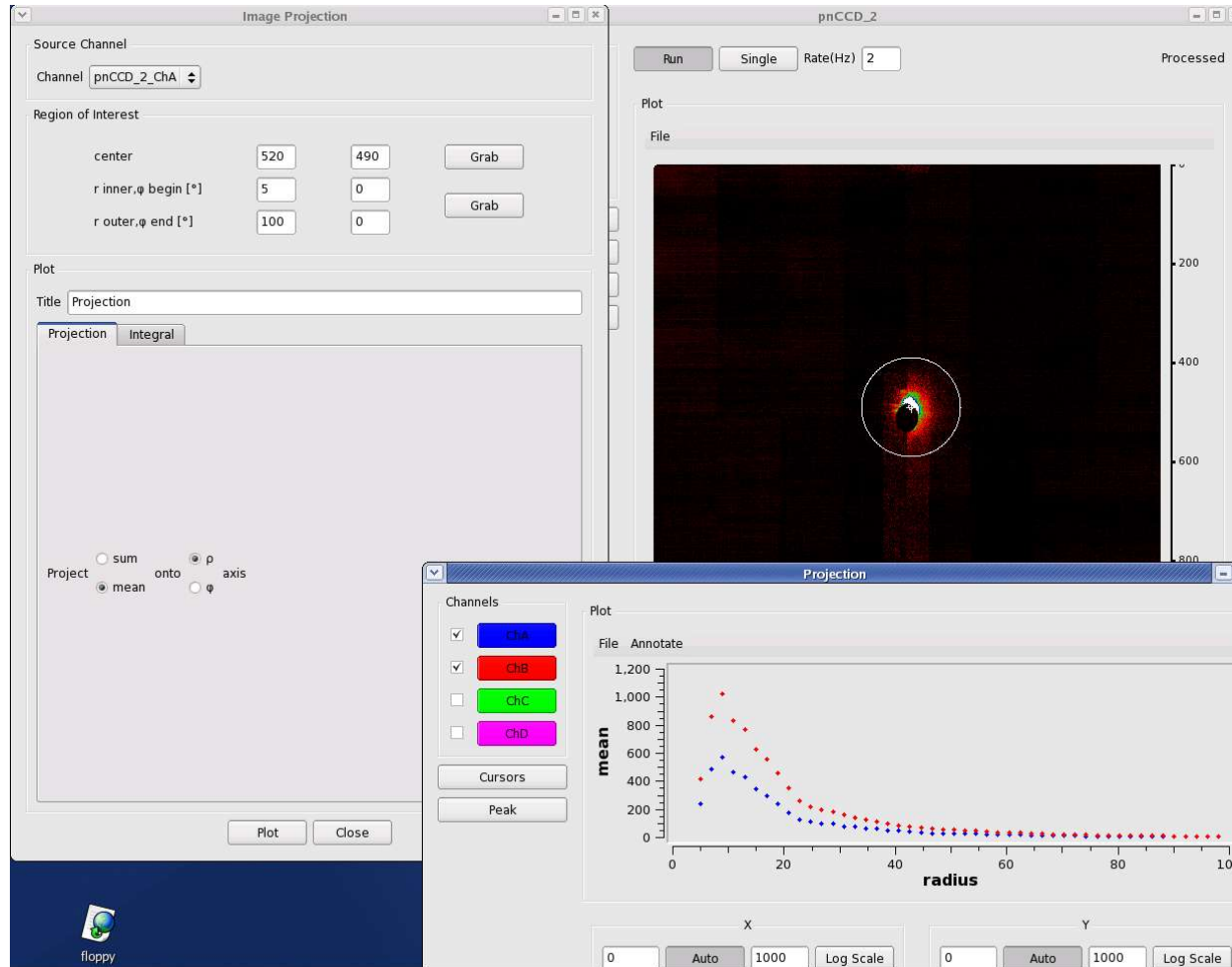


Result Plot

Operations – X/Y projection

- The resulting projections can be treated like original waveforms :
channels {averaging, reference},
operations {cursors, peak finder}
- Similarly for other projections
- Can also integrate over rectangular region to generate a scalar value and plot ...

Operations – radial/azimuthal projection



Core Online Monitoring – User Specialization

- Simple C++ API defined for writing a plug-in module (*amiuser*)
 - *User code called for each detector event received*
 - *Defined set of plot types are available for generating displays*
 - *Some accomodation of plot arrangement on pages*

- User writes code and compiles a dynamic library

- Core monitoring application links in user library when available

Core Online Monitoring – User Specialization (2)

(<https://confluence.slac.stanford.edu/display/PCDS/Adding+plots+to+the+Core+Online+Monitoring>)

Copy /reg/g/pcds/package/amiuser to your area
`cp -rf /reg/g/pcds/package/amiuser ~/.`

Edit the ExampleAnalysis.{hh,cc} files
`cd ~/amiuser; gedit ExampleAnalysis.cc`

Build the libamiuser.so library
`make`

Copy the libamiuser.so to your experiment's home area
`cp libamiuser.so ~xxxopr/.`

User Shared Memory Application

- C++ API defined for receiving event data in shared memory (*pdsdata/app*)
- User writes code and builds an executable in their development environment
 - User adopts graphical display package of their choosing
 - May integrate with other analysis tools
- Application runs on monitoring nodes {“xxx-mon”}
- Recorded data files can also be played back through the shared memory interface
- CAMP pnCCD analysis example { XOnline, CASS }

User Disk-Based Application

- One node (“xxx-ana”) in each experiment records an extra copy of the data for prompt readback (even while writing)
 - *No risk to official data cache*
 - *File remains for brief period (1-10hrs)*
 - *First opportunity for offline-style analysis*
 - *Provides only data access when offline transfer is backlogged*

- Data format is “XTC”

- Initial analysis support provided by “myana” package
 - C++ framework (mostly simple C calls)

Getting Started with *myana*

(<https://confluence.slac.stanford.edu/display/PCDS/Data+Analysis>)

Nice instructions on Confluence now, but basic steps are:

ssh psexport	(login to a machine data access)
cp -r -d /reg/g/pcds/package/ana .	(copy package to your diskspace)
cd ana	(edit source files; e.g. myana.cc)
./comp	(build the executable and run)
./myana -f /reg/g/pcds/package/anatestdata/acqiris.xtc -n 100	
./root	(view <i>ROOT</i> histogram output)

Several example uses of *myana* are described on Confluence.

Conclusion

The core developers try to make the system intuitive and easy to use, but users will benefit considerably from early preparation of their analysis code and monitoring plans.