The RCDAC Data Acquisition System

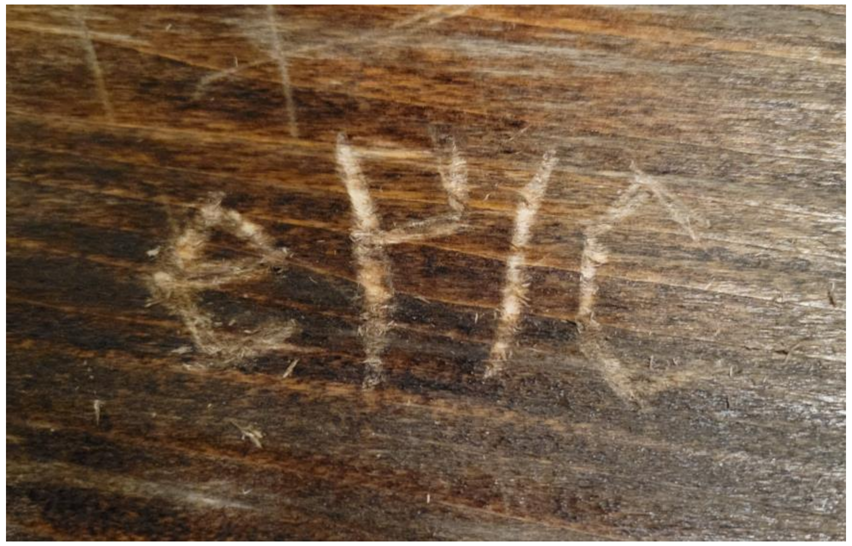
Martin L. Purschke

DAQ manager of sPHENIX

Previous DAQ/Electronics WG co-convener for what is now ePIC

I have been a DAQ (and notably, a calorimeter) guy since my early days at CERN

Brookhaven National Laboratory



The new ePIC logo now immortalized at the F&A Grog House,
117 East 3rd St, Bethlehem, PA 18015

What I'll be talking about today

RCDAQ is DAQ system that has been around since about 2012

It started out as your swiss army knife-type DAQ system to quickly read out whatever you need for your R&D project

It was used in pretty much all R&D campaigns for sPHENIX, but already much earlier in several EIC-themed test beams and other measurements, typically at the Fermilab Test Beam Facility

To the best of my knowledge (some I know, some I learn about when I get questions), it is in use in about 25-30 places around the world

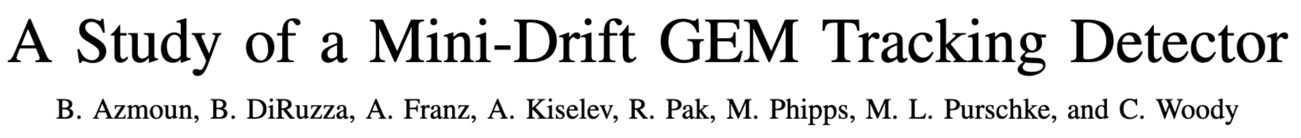
I have step-by-step manuals with examples that seem to work for groups that set up completely autonomously. Just the other day I got a question from an outside group that I didn't know about. They had been using RDAQ for more than a year and finally had a question.

RCDAQ was chosen to be the main DAQ system of sPHENIX, with several higher-level additions ("Run Control") that are irrelevant here

I will mostly focus on lab-test / R&D-style / test beam setups today

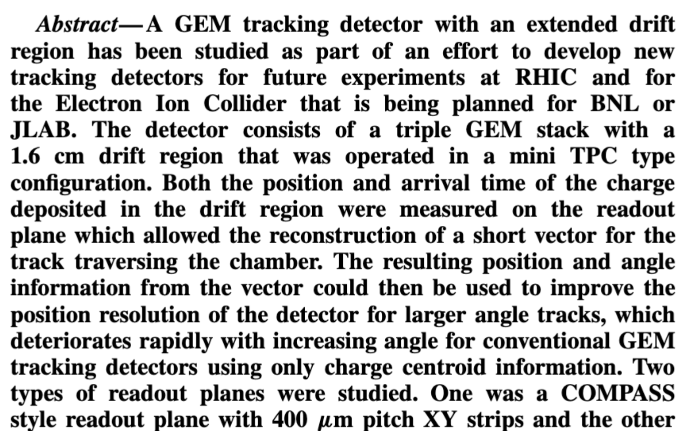
Go to https://www.phenix.bnl.gov/~purschke/rcdaq for the manuals and sample data files etc

One of the early EIC test beam campaigns with RCDAQ - The Minidrift GEM tracking detector (2014)

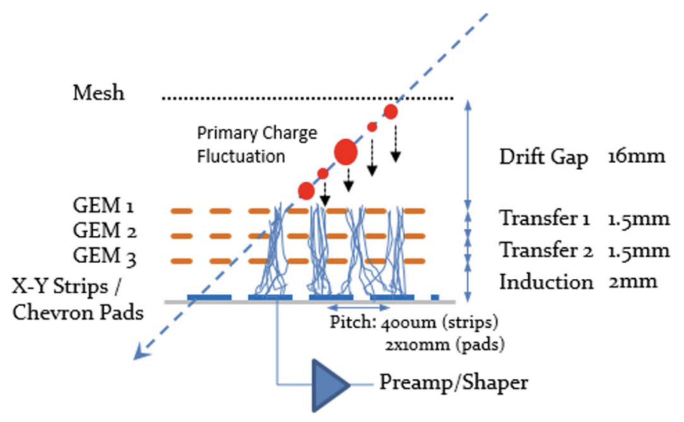


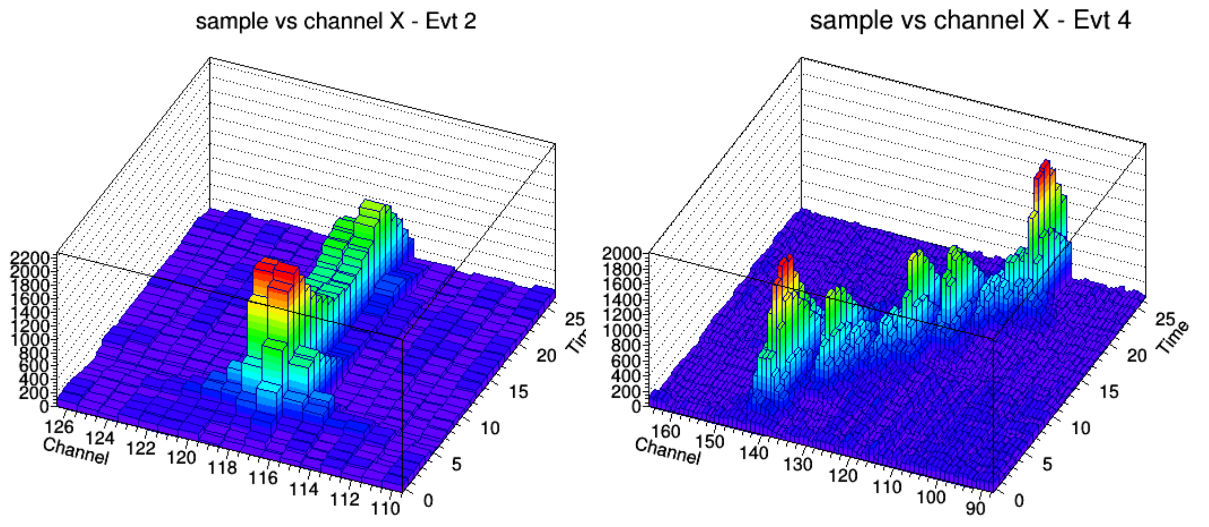
A Study of a Mini-Drift GEM Tracking Detector

B. Azmoun, B. DiRuzza, A. Franz, A. Kiselev, R. Pak, M. Phipps, M. L. Purschke, and C. Woody

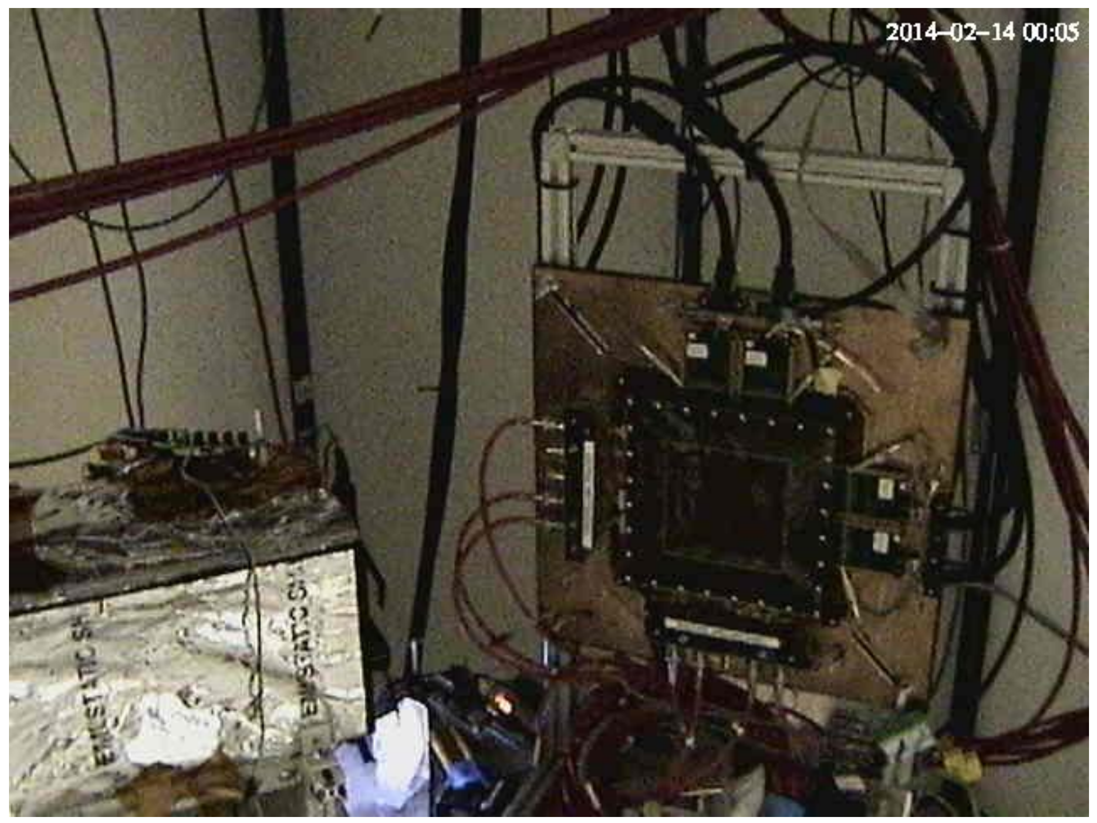


Abstract—A GEM tracking detector with an extended drift region has been studied as part of an effort to develop new tracking detectors for future experiments at RHIC and for the Electron Ion Collider that is being planned for BNL or JLAB. The detector consists of a triple GEM stack with a 1.6 cm drift region that was operated in a mini TPC type configuration. Both the position and arrival time of the charge deposited in the drift region were measured on the readout plane which allowed the reconstruction of a short vector for the track traversing the chamber. The resulting position and angle information from the vector could then be used to improve the position resolution of the detector for larger angle tracks, which deteriorates rapidly with increasing angle for conventional GEM tracking detectors using only charge centroid information. Two types of readout planes were studied. One was a COMPASS style readout plane with 400 μm pitch XY strips and the other



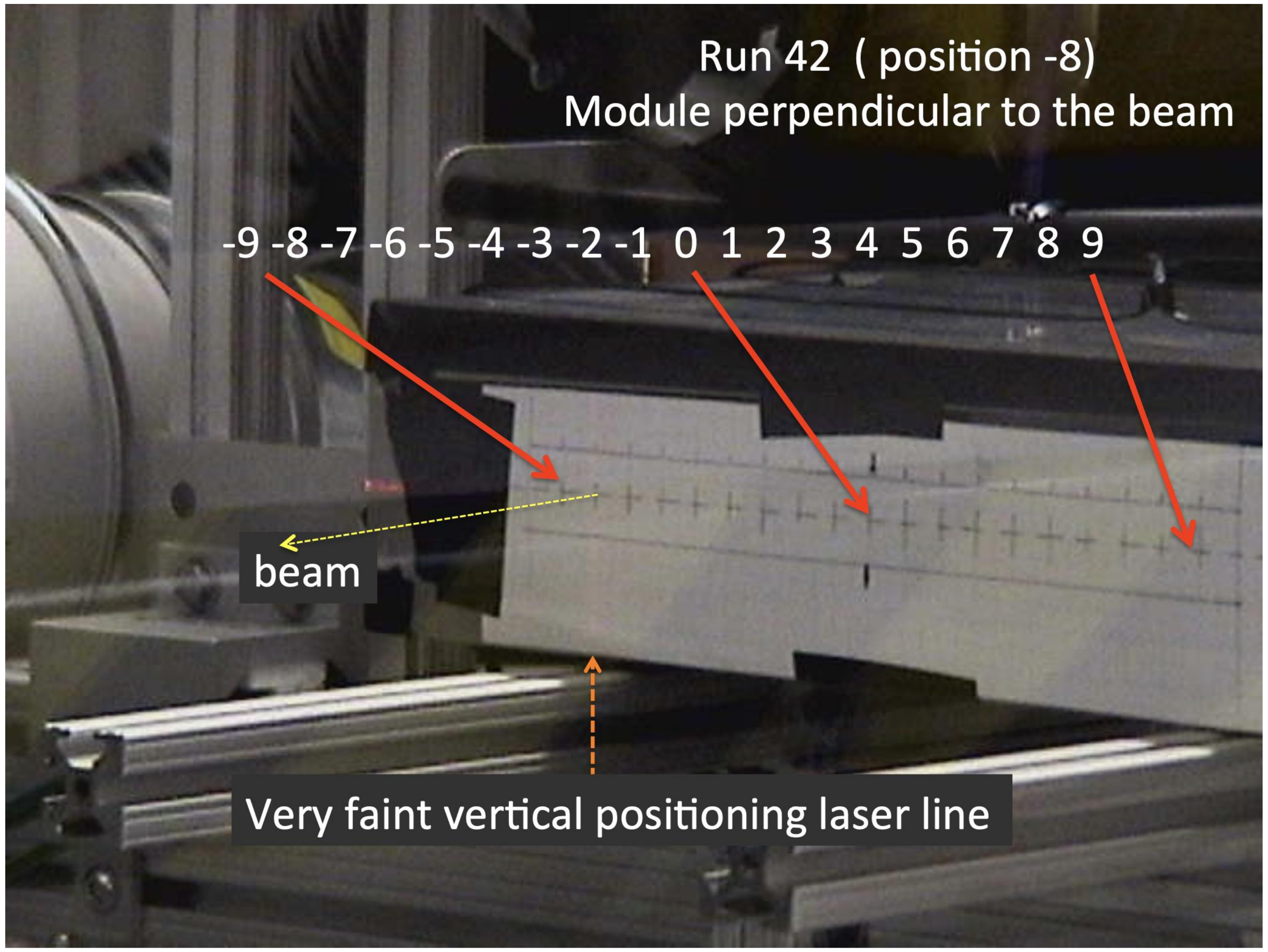


IEEE Transactions on Nuclear Science, vol. 63, no. 3, pp. 1768-1776, June 2016

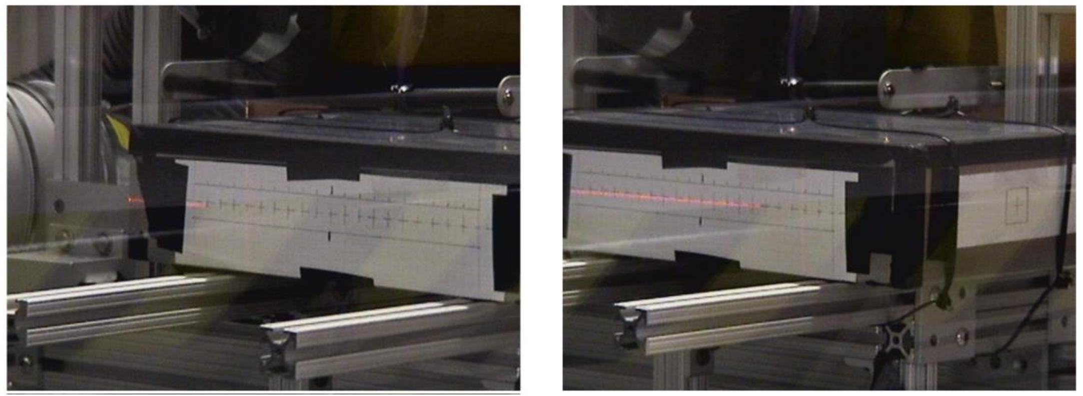


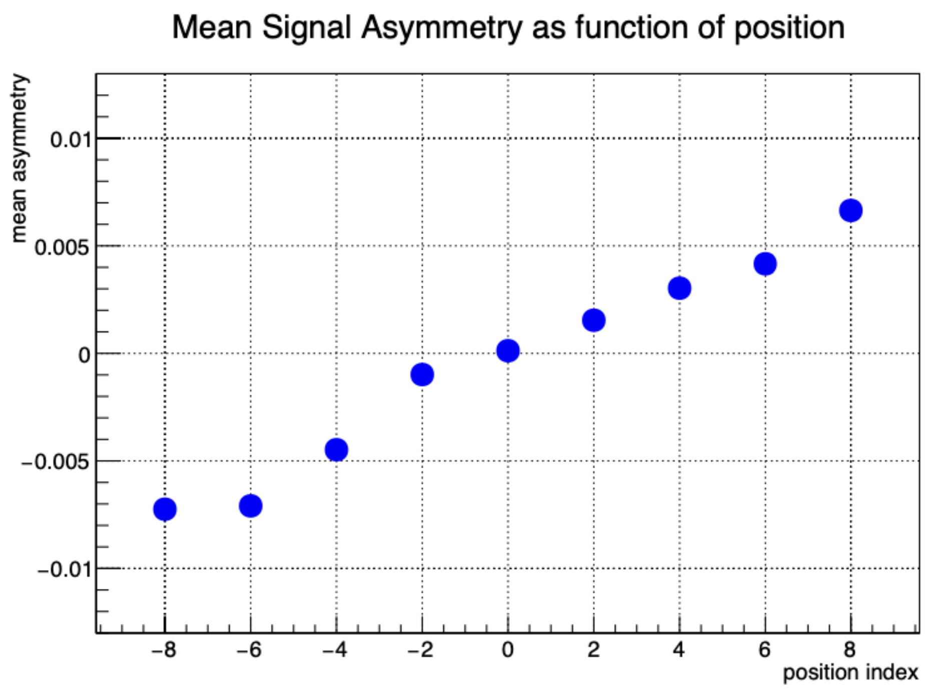
Another one from the EIC calo orbit – Oleg, Craig, myself

”Can we get more calorimeter position information by adding a dual readout and measure time?”


Run 42 (position -8)
Module perpendicular to the beam
-9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9
beam
Very faint vertical positioning laser line

Moving the calorimeter in the beam for “more left” or “more right” incidence





In the end we found that it’s too small an effect to pursue, but it was worth checking!

What does one need?

Pretty much any Linux machine and distro will do. (I do lots of development on a Linux VM on my M2 Mac here)

I'm striving to be distro-agnostic:

- RedHat and derivatives (RHEL, Fedora, CentOS, Alma… all good)
- Debian and similar (Ubuntu, Mint, …)
- Arch Linux

And yes, it does run on a Raspberry Pi – sometimes you want to take a few weeks worth of cosmics with some detector module without tying up a more expensive PC

The main platforms currently at work in sPHENIX are 96-core AMD EPYC PCs.

For practicing/going through the manuals, you can run everything without any actual readout hardware – RCDAQ provides "pretend-devices" that behave like an ADC

RCDAQ - The High Points

Each interaction with RCDAQ is a **shell command**. There is no “starting an application and issuing internal commands” (think of your interaction with, say, root)

RCDAQ out of the box doesn’t know about any particular hardware. All knowledge how to read out something, say, a FELIX card, comes by way of a **plugin** that teaches RCDAQ how to do that.

That makes RCDAQ highly portable and also **distributable** – some sPHENIX FEMs use commercial drivers for the readout; I cannot re-distribute CAEN software, etc etc

RCDAQ has **no proprietary configuration** files. (huh? In a minute).

Support for different **event types**

Built-in support for standard **online monitoring**

Built-in support for **electronic logbooks** (Stefan Ritt’s Elog)

Network-transparent control interfaces

Everything is a shell command…

One of the most important features. Any command is no different from “ls –l“ or “cat”

Everything is inherently scriptable

You have the full use of the shell’s capabilities for if-then constructs, error handling, loops, automation, cron scheduling, and a myriad of other ways to interact with the system

In that sense, there are no proprietary configuration files – only configuration *scripts.*

This is quite different from “my DAQ supports scripts”!

I do not want to be trapped within the limited command set of any application!

With shell commands, the DAQ is fully integrated into your existing work environment

(And yes there are GUIs – they usually just trigger the appropriate command)

On Autopilot - Scripts at work

Very often – especially in your R&D days – you want to step through a range of values of a configuration parameter and see what your detector prototype has to say

- Bias voltage scans (we characterized gazillions of SiPMs)
- Position scans
- Temperature scans
- And on and on

Such a measurement is best done in a script that reads predetermined positions / voltage settings / what have you and performs the measurement

I picked an example: What is the response uniformity of a calorimeter module when a shower develops in different places? (We were very worried about this)

We were simulating different shower positions by “writing light with a light fiber” on the module front face

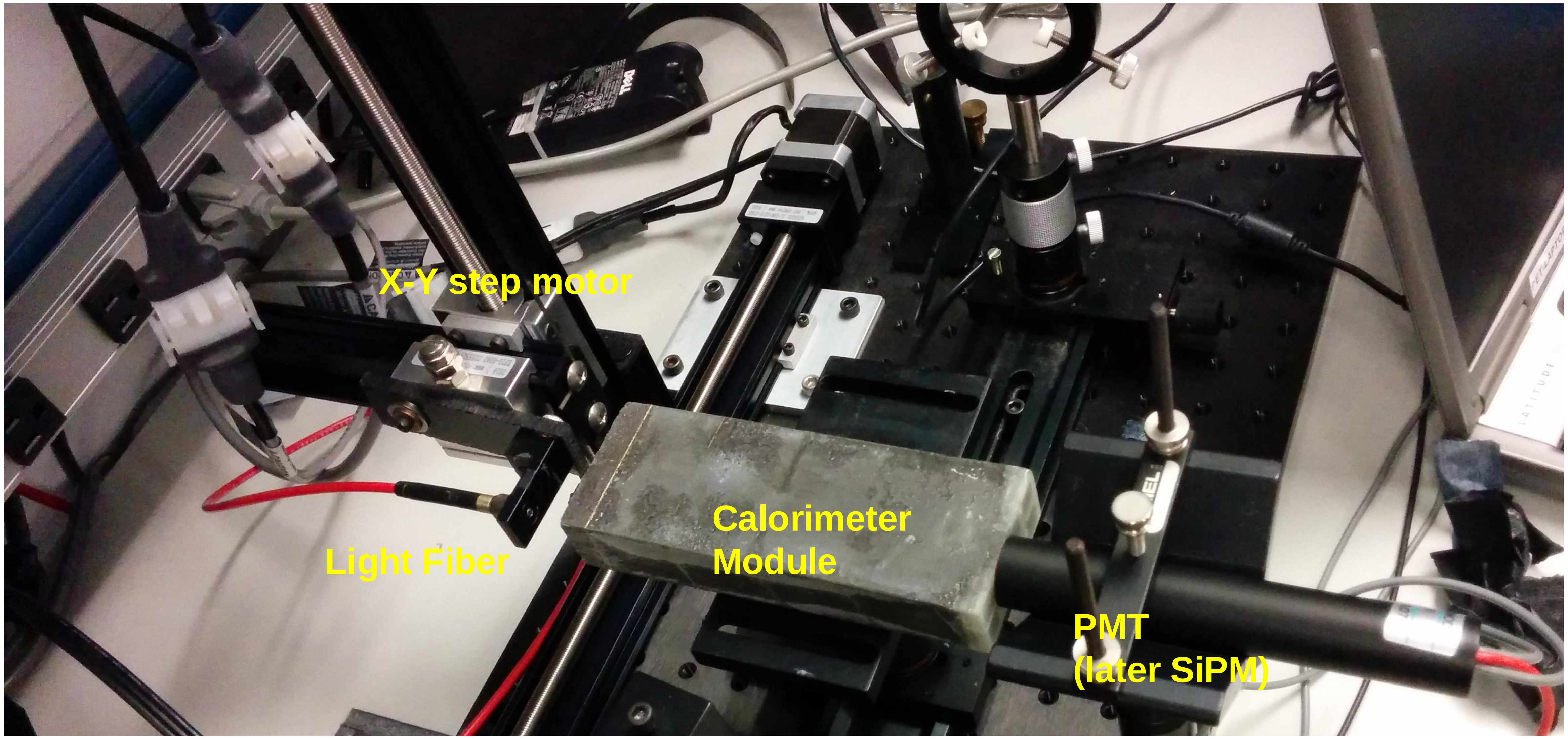
Measurements on autopilot through scripting

Simulate shower incidence positions by moving a light fiber in x and y

take a run for each position w/ 4000 events

50 x 25 = 1250 positions (later we had 60x60, you really want to automate that)

Let it run overnight, come back in the morning, look at the data



The Script

The DAQ operation becomes an integral part of your shell environment

Automatic end after 4000 events

25 positions in y

move the Y motor

50 positions in x

move the x motor

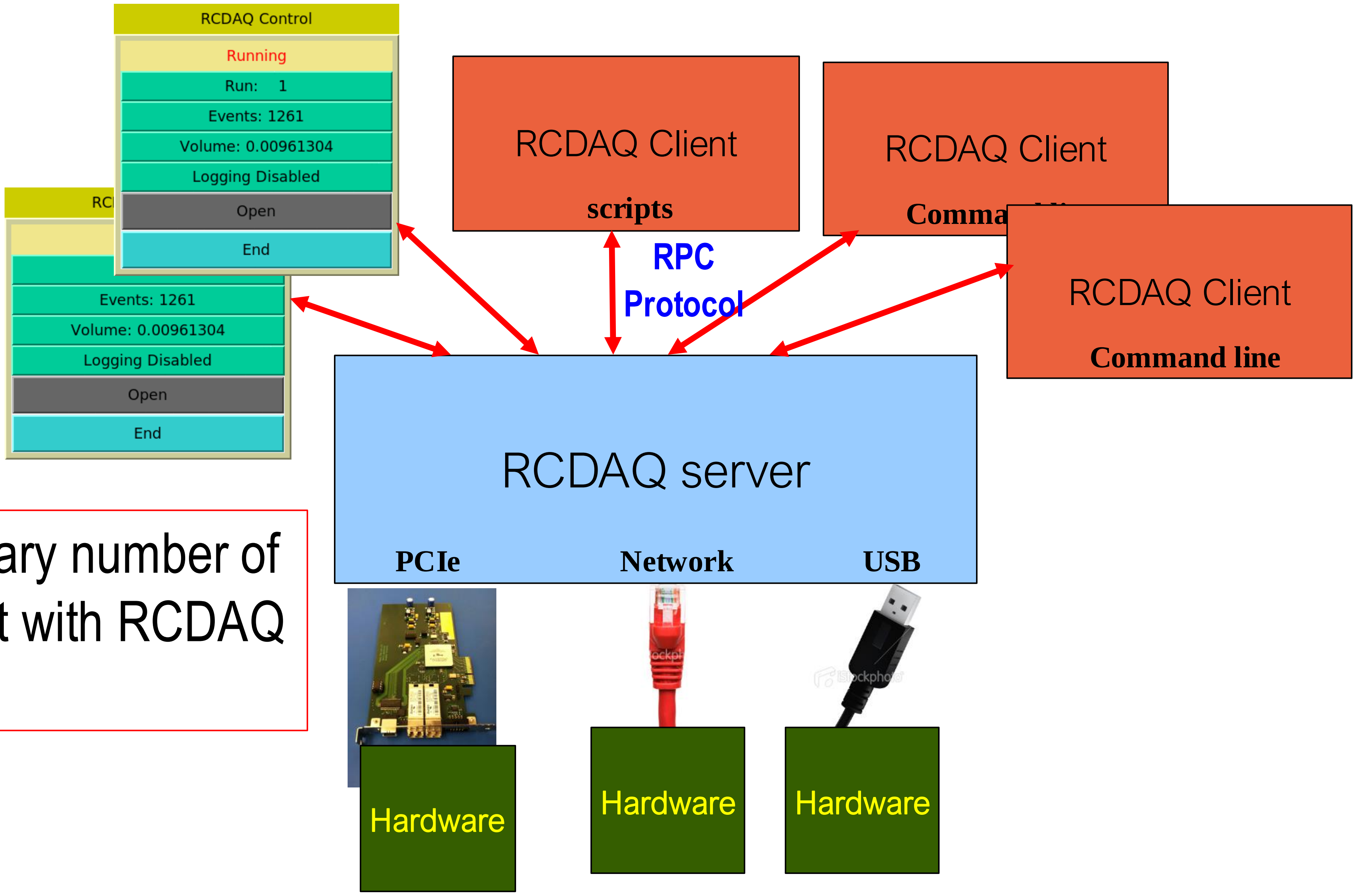
start the DAQ

next x

next y

```
#! /bin/sh
STARTPOSX=0
STARTPOSY=9900
INCREMENTX=200
INCREMENTY=-200

CURRENTPOSY=$STARTPOSY
rcdaq_client daq_set_maxevents 4000

for posy in $(seq 25) ; do

    quickmove.sh $CURRENTPOSY 2
    sleep 5
    CURRENTPOSY=$( expr $CURRENTPOSY + $INCREMENTY)
    CURRENTPOSX=$STARTPOSX

    for posx in $(seq 50) ; do

      echo "moving to $CURRENTPOSX"
      quickmove.sh $CURRENTPOSX 1
      sleep 5

      rcdaq_client daq_begin
      wait_for_run_end.sh

      CURRENTPOSX=$( expr $CURRENTPOSX + $INCREMENTX)
    done
done
```

The RCDAQ client-server concept

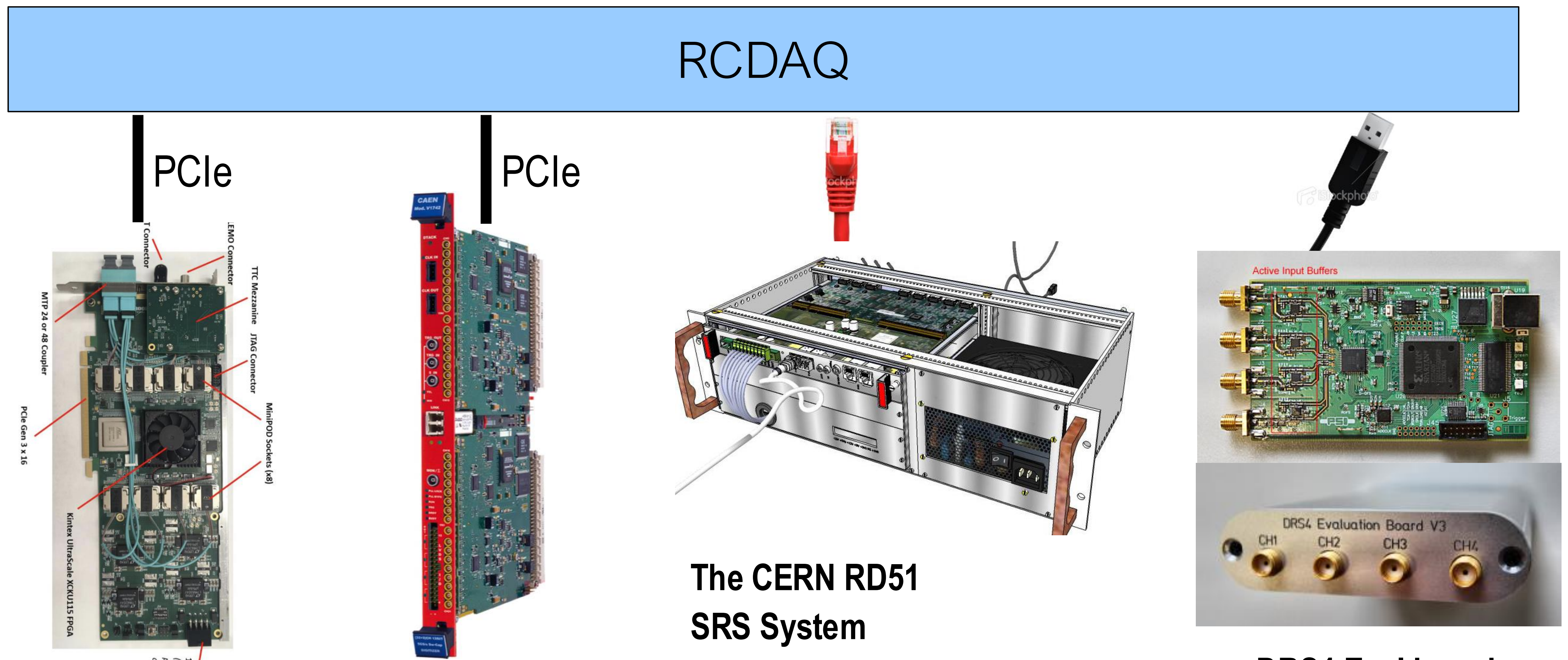


This allows an arbitrary number of processes to interact with RCDAQ concurrently

These are the 3 fundamental ways for data to get into a PC –

- PCIExpress
- Network
- USB

Some workhorse devices implemented in RCDAQ



FELIX Card

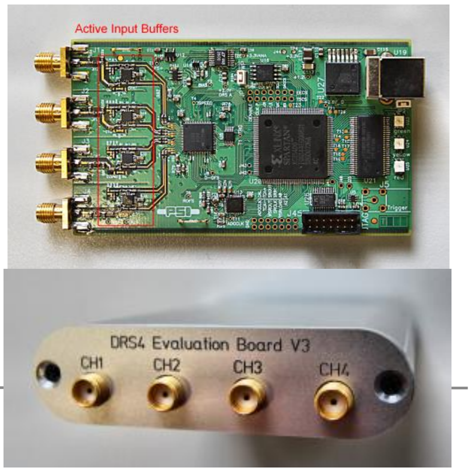
The CAEN V1742 waveform digitizer

The CERN RD51 SRS System

DRS4 Eval board
"USB Oscilloscope"

There are *many* more not shown (all told, there are plugins for about 60)
Many devices that you can often find in your institute already, or in the CAEN catalog

Example: reading out a DRS4 Eval board



```
$ rcdaq_client load librcdaqplugin_drs.so

$ rcdaq_client create_device device_drs -- 1 1001 0x21 -150 negative 140 3

$ daq_open

$ daq_begin

  # wait a while…

$ daq_end
```

You see, each interaction is a separate shell command.

Meta Data Capturing

In the “real” experiment that’s running for a few years (think sPHENIX, ATLAS, what have you) you are embedded in an environment that supports all sorts of record keeping

At a test beam or you in your lab needs a different kind of “record keeping support”

What was the temperature? Was the light on? What was the HV? What was the position of that X-Y positioning table?

We capture this information in the raw data file itself and **the data cannot get lost**

I often add a webcam picture to the data so we have a visual confirmation that the detector is in the right place, or something

A picture captures everything…

Let me show you how we always capture the RCDAQ setup itself

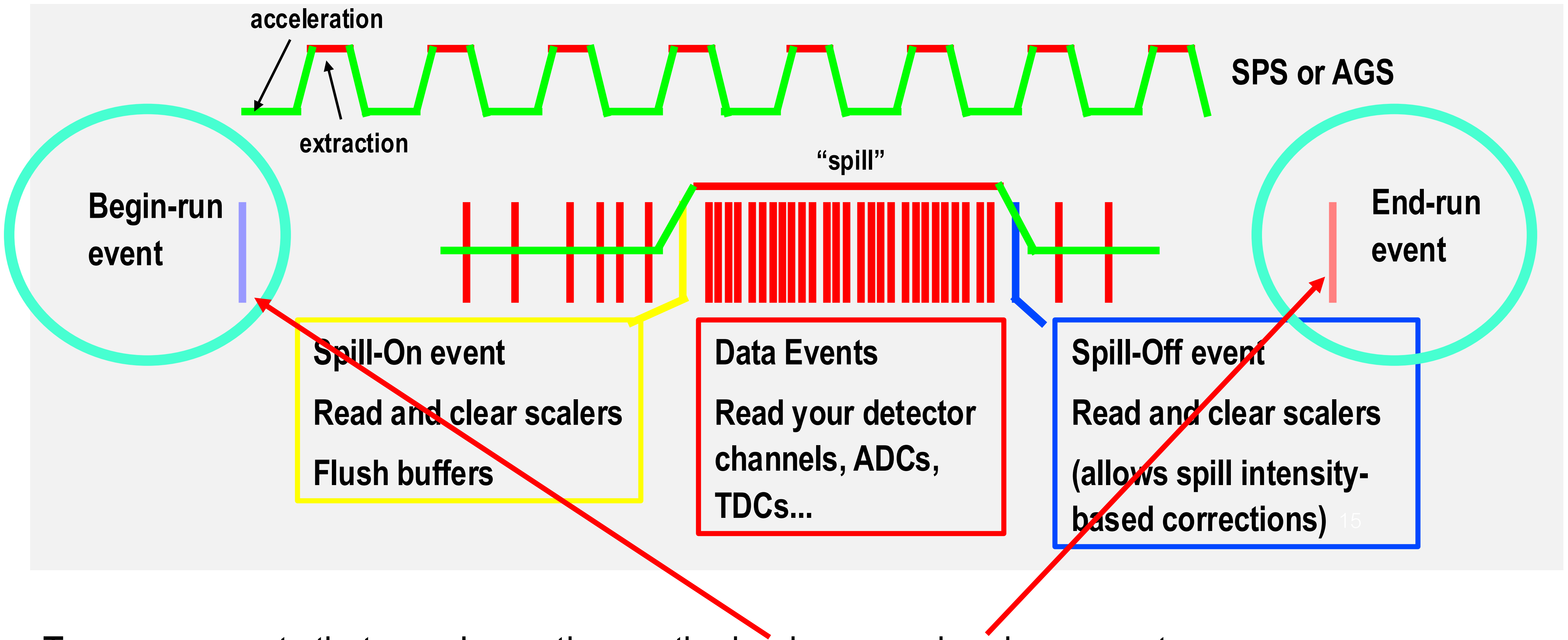
Reading different things with different Event Types

You would think of the DAQ as "reading out your detector"

Very often, it is necessary to read out different things at different times.

Let's go to the CERN-SPS (or the Fermilab Test Beam facility) for an example

One needs to read out "meta-info" about each spill for spill-by-spill corrections



Two more events that are always there – the begin-run and end-run events

Remember this?

This was our typed-in example from before

```
$ rcdaq_client load librcdaqplugin_drs.so

$ rcdaq_client create_device device_drs -- 1 1001 0x21 -150 negative 140 3
```

Now you put this into a script so you always get the same setup:

```
#! /bin/sh

rcdaq_client load librcdaqplugin_drs.so

rcdaq_client create_device device_drs -- 1 1001 0x21 -150 negative 140 3
```

Capturing the setup script itself for posterity

We add this very setup script file into our begin-run event for posterity

This “device” captures a file as text into a packet

This “9” is the event type of the beg-run

And this refers to the name of the file itself

```
#! /bin/sh

rcdaq_client create_device device_file 9 900 ”$0”

rcdaq_client load librcdaqplugin_drs.so

rcdaq_client create_device device_drs -- 1 1001 0x21 -150 negative 140 3
```

So this gets added as packet with id 900 in the begin-run

It’s not quite right yet - $0 is usually just “setup.sh”, so the server may not be able to find it.

Let me show the “end product”:

A typical RCDAQ Setup Script

```
#! /bin/sh
# this sets up the DRS4 readout with 5GS/s, a negative
# slope trigger in channel 1 with a delay of 140

if ! rcdaq_client daq_status > /dev/null 2>&1 ; then
    echo "No rcdaq_server running, starting..."
    rcdaq_server > $HOME/rcdaq.log 2>&1 &
    sleep 2
fi
MYSELF=$(readlink -f $0)
rcdaq_client daq_clear_readlist

rcdaq_client create_device device_file 9 900 "$MYSELF"

rcdaq_client load librcdaqplugin_drs.so
rcdaq_client create_device device_drs -- 1 1001 0x21 -150 negative 140 3
```

We comment a lot as a way of documentation

If no server is running, we start one here.

We convert the script filename into a full path

We clear all existing definitions

We load the plugin(s) and define the device(s)

Here is the actual setup script for our FELIX readout

Abridged version, just the essentials

```
#!/bin/bash

RunType=beam
H=$RCDAQHOST
[ -z "$H" ] &&  H=$(hostname)

MYSELF=$(readlink -f $0)
rcdaq_client daq_clear_readlist
rcdaq_client create_device device_file 9 900 "$MYSELF"

rcdaq_client load   librcdaqplugin_dam.so
rcdaq_client create_device device_dam 1 4${H:4:2}1 1 128

rcdaq_client daq_set_runcontrolmode 1
```

More about capturing your environment

Sometimes you add meta-info to make the analysis easier, especially in the aforementioned "scanning something" setups - example later

Many times you capture things only "just in case"

I usually add a camera picture to the begin-run, especially when the detector moves in the beam for some position scan

You don't routinely look at them in your analysis, but it's good to have that info

If you have some inexplicable feature, you can use the meta-data to do "forensics"

Find out what, if anything, went wrong

The more data you capture, the better this gets

Think of it as "black box" on a plane…

Forensics (FermiLab test beam for the future ePIC HCal)

“It appears that the distributions change for Cherenkov1 at 1,8,12,and 16 GeV compared to the other energies. It seems that the Cherenkov pressures are changed. […] Any help on understanding this would be appreciated.”

Martin: “Look at the info in the data files:”

```
$ ddump -t 9 -p 923 beam_00002298-0000.prdf
S:MTNRG   = -1     GeV
F:MT6SC1 =  5790        Cnt
F:MT6SC2 =  3533        Cnt
F:MT6SC3 =  1780        Cnt
F:MT6SC4 =  0           Cnt
F:MT6SC5 =  73316       Cnt
E:2CH     =  1058  mm
E:2CV     =  133.1 mm
E:2CMT6T =  73.84 F
E:2CMT6H =  32.86 %Hum
F:MT5CP2 =  .4589 Psia
F:MT6CP2 =  .6794 Psia
```

```
$ ddump -t 9 -p 923 beam_00002268-0000.prdf
S:MTNRG   = -2     GeV
F:MT6SC1 =  11846       Cnts
F:MT6SC2 =  7069        Cnts
F:MT6SC3 =  3883        Cnts
F:MT6SC4 =  0           Cnts
F:MT6SC5 =  283048      Cnts
E:2CH     =  1058  mm
E:2CV     =  133   mm
E:2CMT6T =  74.13 F
E:2CMT6H =  37.26 %Hum
F:MT5CP2 =  12.95 Psia
F:MT6CP2 =  14.03 Psia
```

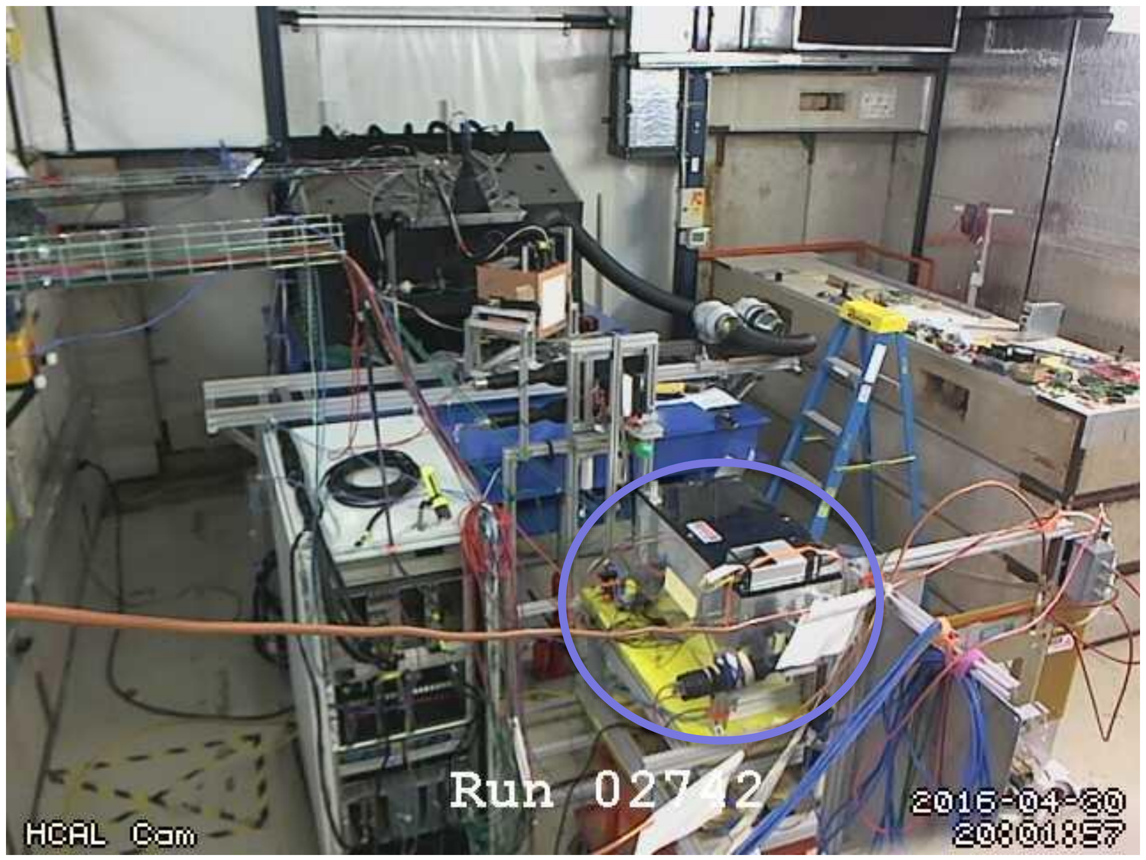
Among many other things, we capture the most relevant beamline parameters

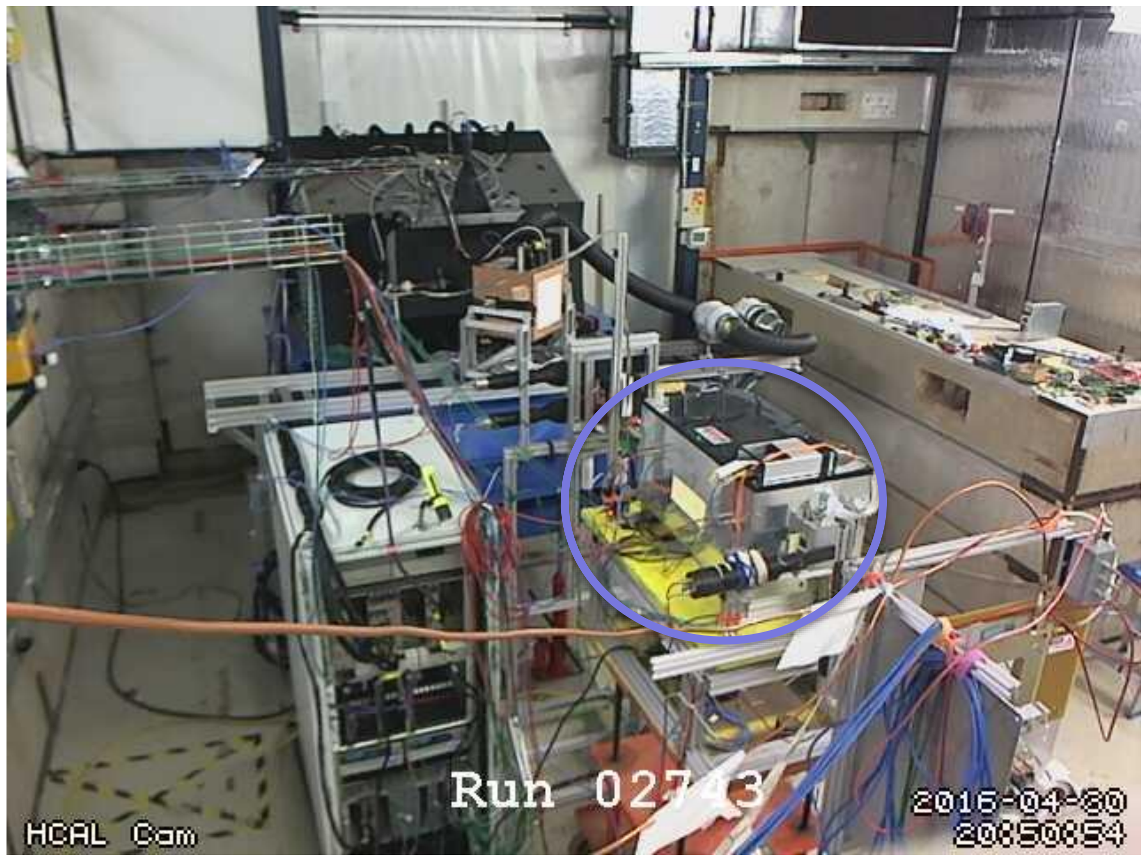
More Forensics (HCal at the Fermilab test beam again…)

"There is a strange effect starting in run 2743. There is a higher fraction of showering than before. I cannot see anything changed in the elog."

Look at the cam pictures we automatically captured for each run:

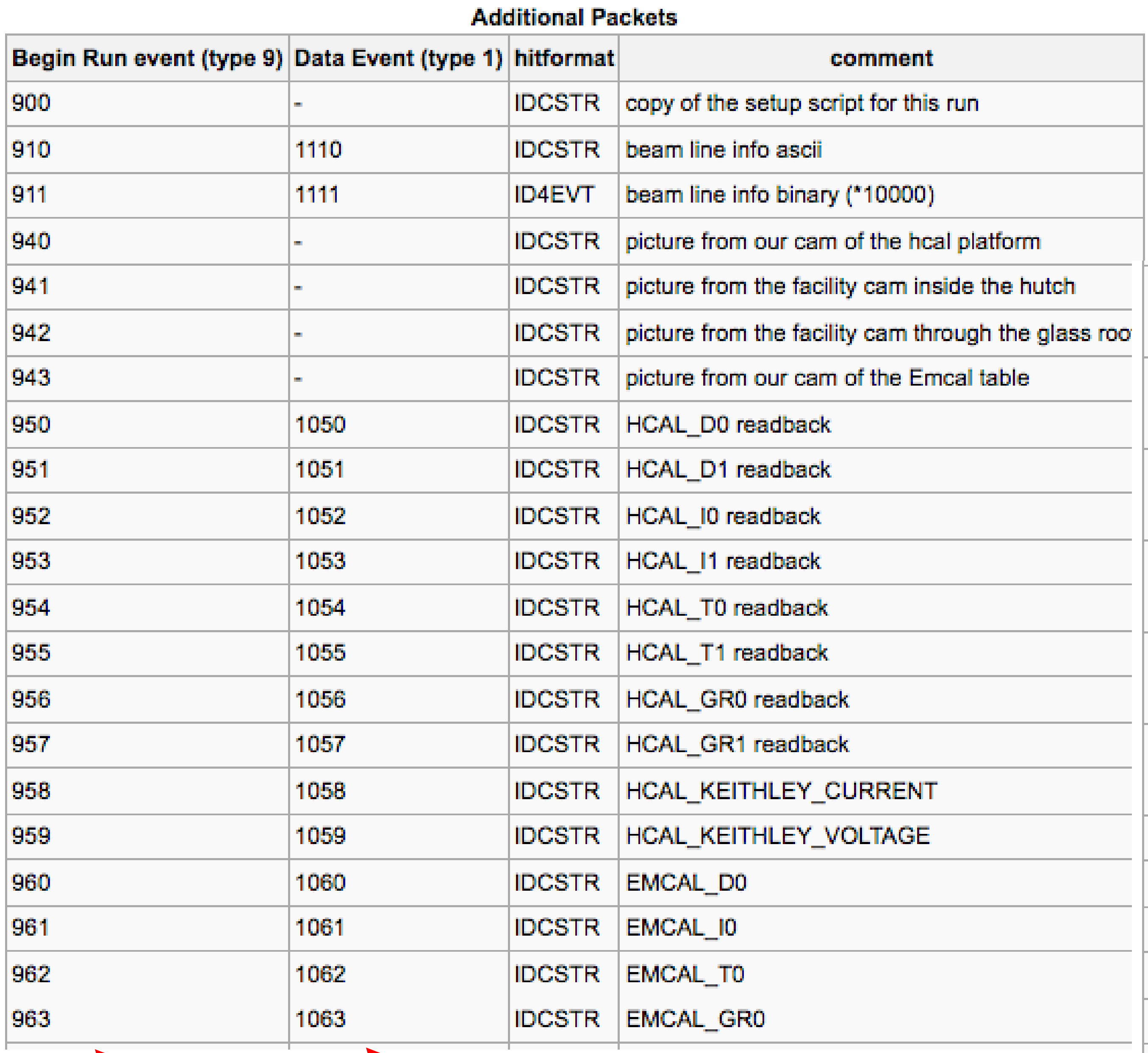
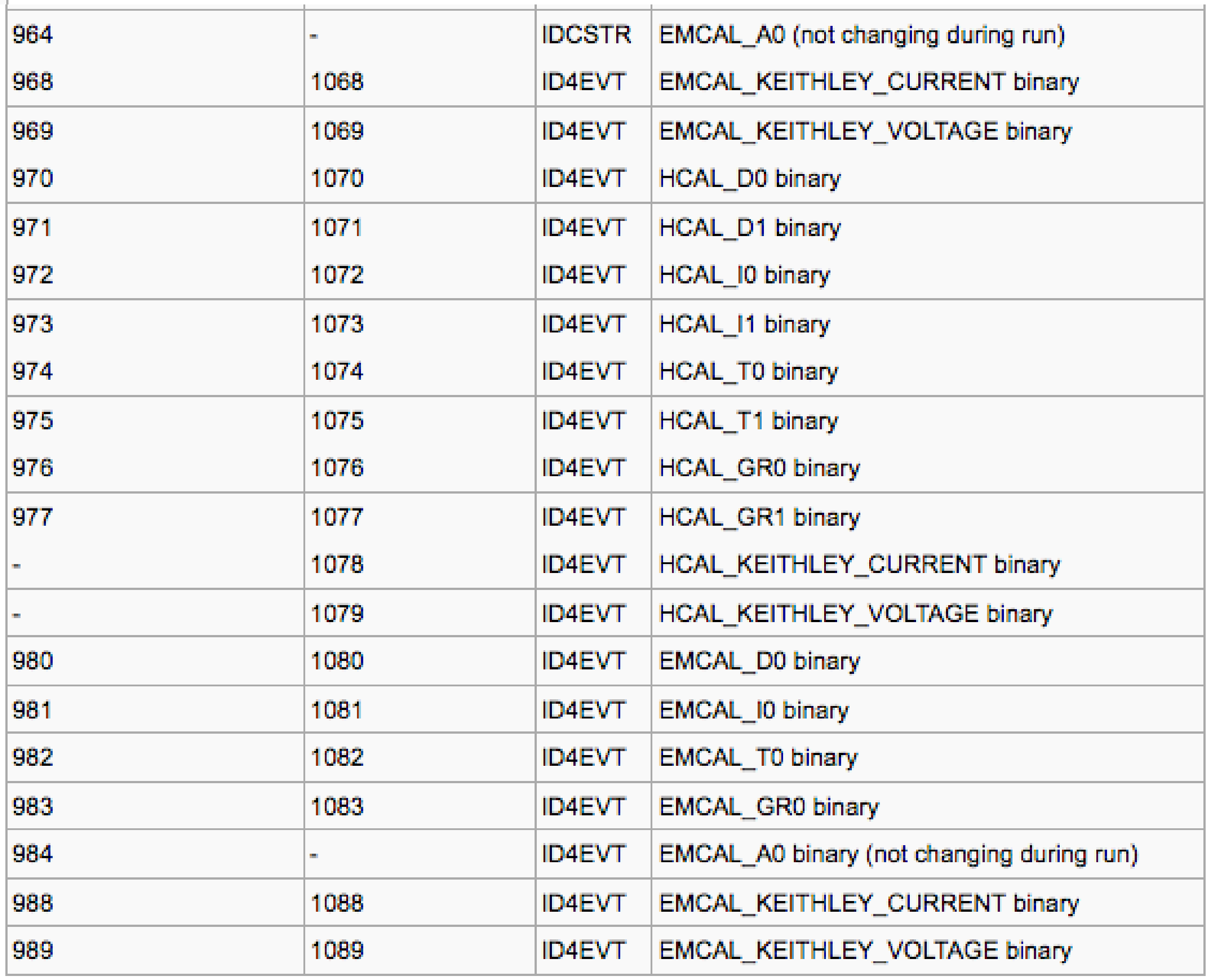
```
$ ddump -t 9 -p 940 beam_00002742-0000.prdf > 2742.jpg
$ ddump -t 9 -p 940 beam_00002743-0000.prdf > 2743.jpg
```





“Meta Data” Packet list from that test beam

More than 72 environment-capturing packets (accelerator params, voltages, currents, temperatures, pictures, …)

Additional Packets

Begin Run event (type 9)	Data Event (type 1)	hitformat	comment
900	-	IDCSTR	copy of the setup script for this run
910	1110	IDCSTR	beam line info ascii
911	1111	ID4EVT	beam line info binary (*10000)
940	-	IDCSTR	picture from our cam of the hcal platform
941	-	IDCSTR	picture from the facility cam inside the hutch
942	-	IDCSTR	picture from the facility cam through the glass roo
943	-	IDCSTR	picture from our cam of the Emcal table
950	1050	IDCSTR	HCAL_D0 readback
951	1051	IDCSTR	HCAL_D1 readback
952	1052	IDCSTR	HCAL_I0 readback
953	1053	IDCSTR	HCAL_I1 readback
954	1054	IDCSTR	HCAL_T0 readback
955	1055	IDCSTR	HCAL_T1 readback
956	1056	IDCSTR	HCAL_GR0 readback
957	1057	IDCSTR	HCAL_GR1 readback
958	1058	IDCSTR	HCAL_KEITHLEY_CURRENT
959	1059	IDCSTR	HCAL_KEITHLEY_VOLTAGE
960	1060	IDCSTR	EMCAL_D0
961	1061	IDCSTR	EMCAL_I0
962	1062	IDCSTR	EMCAL_T0
963	1063	IDCSTR	EMCAL_GR0
964	-	IDCSTR	EMCAL_A0 (not changing during run)
968	1068	ID4EVT	EMCAL_KEITHLEY_CURRENT binary
969	1069	ID4EVT	EMCAL_KEITHLEY_VOLTAGE binary
970	1070	ID4EVT	HCAL_D0 binary
971	1071	ID4EVT	HCAL_D1 binary
972	1072	ID4EVT	HCAL_I0 binary
973	1073	ID4EVT	HCAL_I1 binary
974	1074	ID4EVT	HCAL_T0 binary
975	1075	ID4EVT	HCAL_T1 binary
976	1076	ID4EVT	HCAL_GR0 binary
977	1077	ID4EVT	HCAL_GR1 binary
-	1078	ID4EVT	HCAL_KEITHLEY_CURRENT binary
-	1079	ID4EVT	HCAL_KEITHLEY_VOLTAGE binary
980	1080	ID4EVT	EMCAL_D0 binary
981	1081	ID4EVT	EMCAL_I0 binary
982	1082	ID4EVT	EMCAL_T0 binary
983	1083	ID4EVT	EMCAL_GR0 binary
984	-	ID4EVT	EMCAL_A0 binary (not changing during run)
988	1088	ID4EVT	EMCAL_KEITHLEY_CURRENT binary
989	1089	ID4EVT	EMCAL_KEITHLEY_VOLTAGE binary

Captured at begin-run

Captured again at spill-off

Moving Detector Example: “HCal Tile Mapping” at the Fermi Test Beam Facility

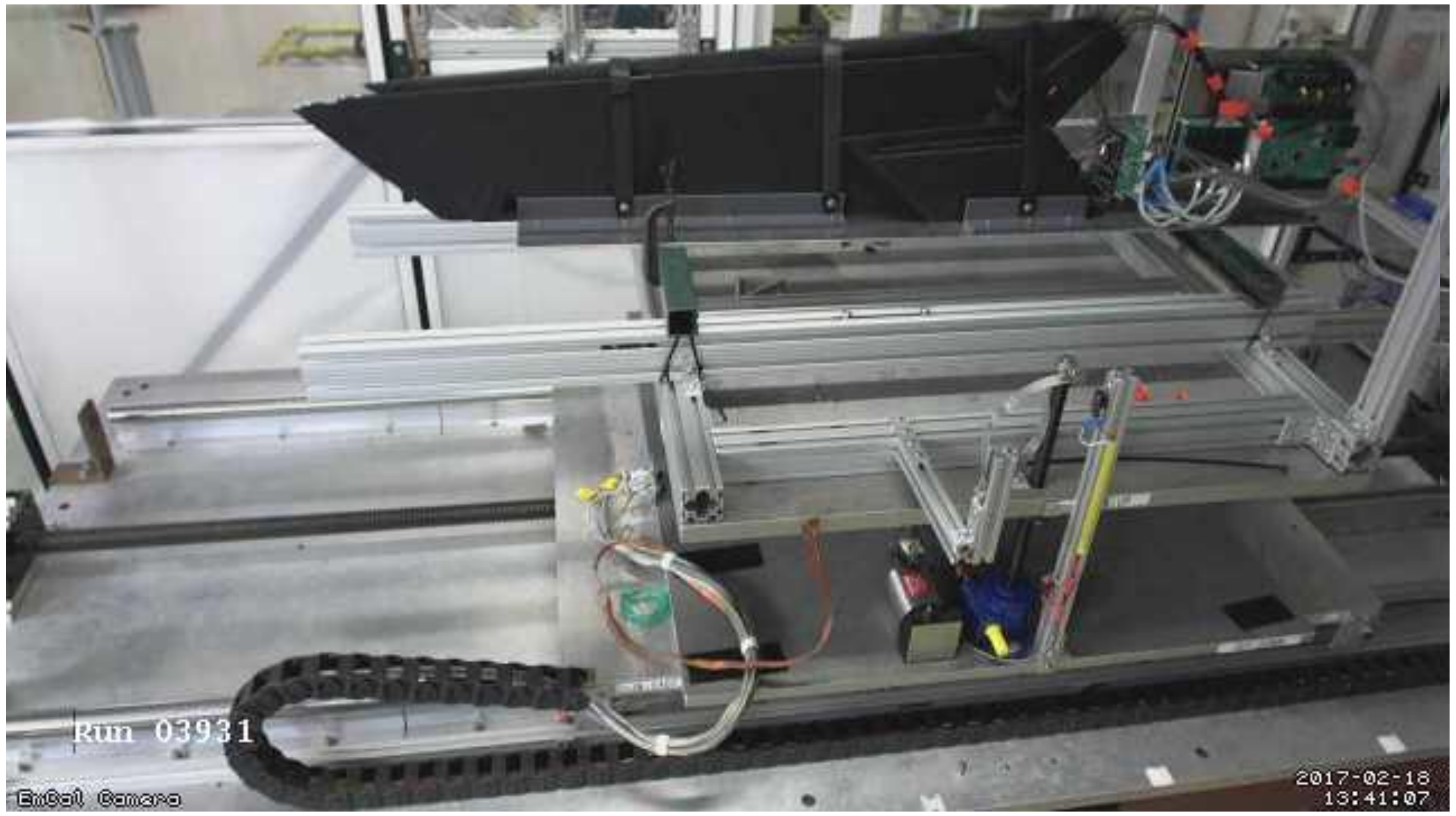
“Tile mapping” refers to mapping the position-dependent response of a hadronic calorimeter tile.

About 200 individual positions of the tile relative to the beam – you’d go nuts doing all that manually, and you are bound to make mistakes

So all is done on autopilot as explained before

And all on camera for feel-good value – does that thing move right when we think it should?

This setup exercises many of the aforementioned features: scripting and reacting to the FTBF spill, network transparency (we cannot access the table positioning from our DAQ machine, but a FTBF-owned machine can control our DAQ)

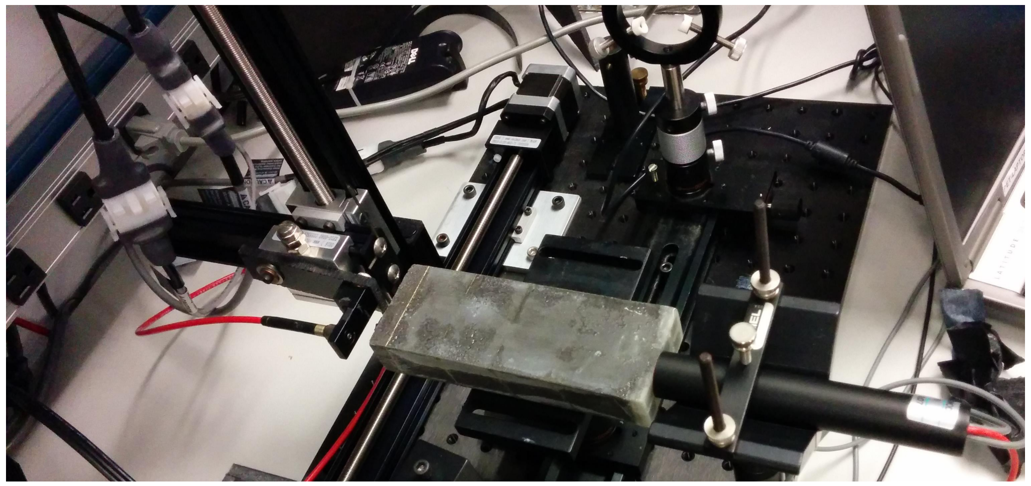


One last point: Using RCDAC features to streamline your analysis

Going back to the “calo module mapping” for a moment

You are ending up with several thousand individual data files, one per position

In each begin-run event we capture additional information, especially the x/y position this data point is from



RCDAQ has a “command device” that doesn’t read out anything but executes a command when the event is triggered – here we execute “getmotorpositions.sh” at each begin-run

This script reaches out to the positioning system and writes a file “positions.txt” with the two numbers

We then absorb the file into the begin-run event

```
# we add a command to capture the positions to a file
rcdaq_client create_device device_command 9 0 "$HOME/getmotorpositions.sh"
rcdaq_client create_device device_file 9 920 "$HOME/positions.txt"
rcdaq_client create_device device_filenumbers_delete 9 921 "$HOME/positions.txt"
```

And we can retrieve this in the analysis (here shown with a command-line utility):

```
 $ ddump -t 9 -p 920 scan10_0000201800-0000.evt
4600
-4400
```

Using RCDAC features to streamline your analysis

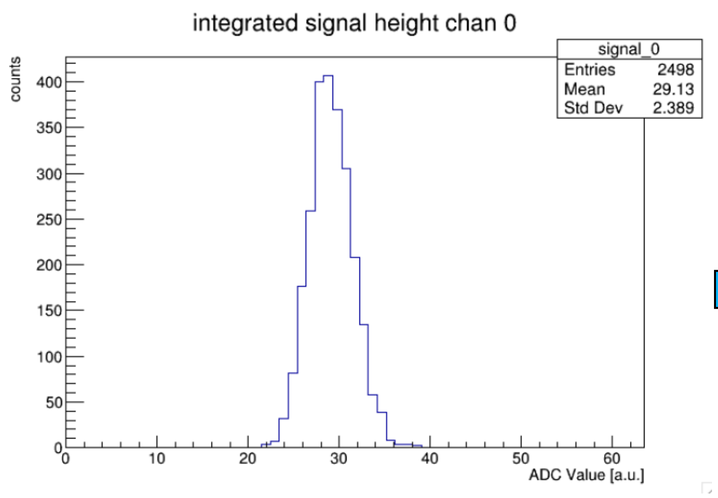
Then we make a list of all files that belong to this scan, and throw them all at the at the analysis process

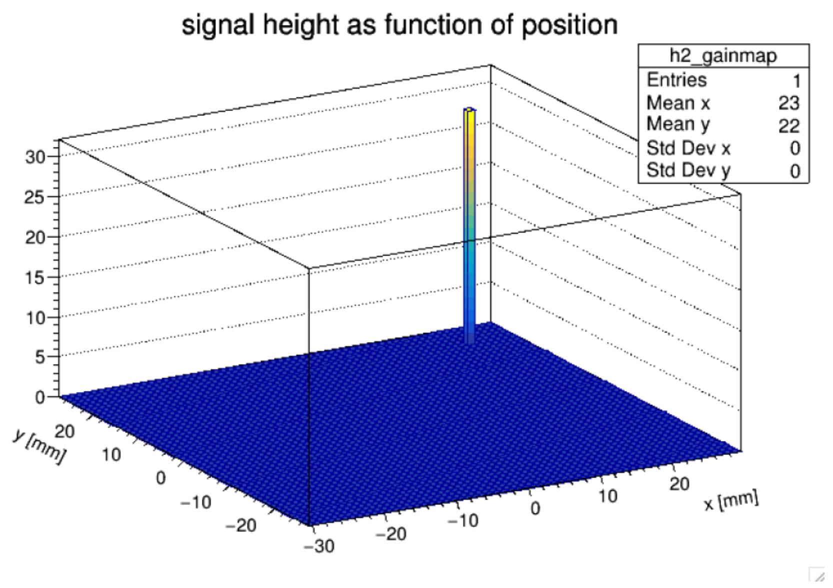
We get the begin-run event and extract and remember the x-y positions

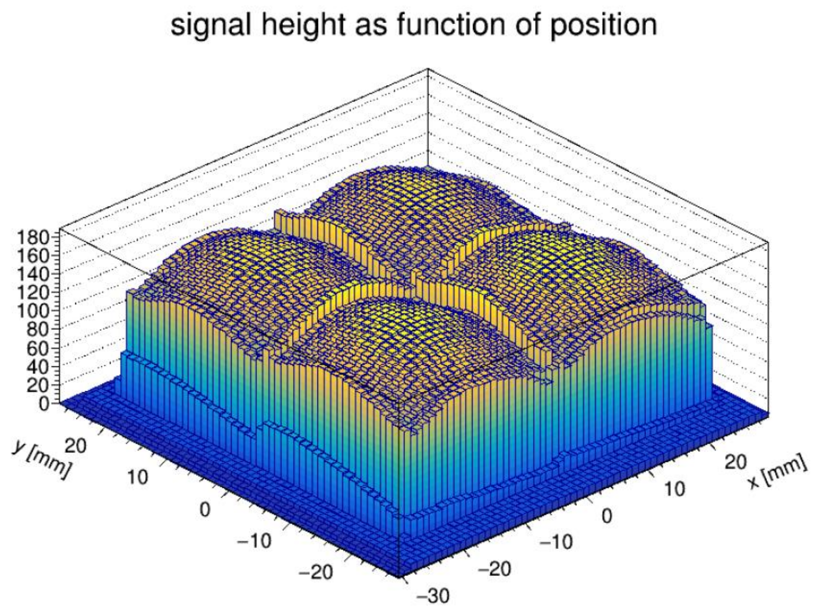
We then histogram the signal with the actual data events

Eventually we hit end end-run event, know that this position is done. Get the mean from the histogram, and fill it in the map at the right position

Run through all files in the set, have a coffee, and see the results:







In this way you can run the analysis without burdensome additional bookkeeping

All you need to analyze the data in one fell swoop is contained in the data themselves

What’s in it for ePIC

I take pride in having one of the easiest-to-use and most versatile DAQ systems out there (from scratch on a brand-new PC to seeing a histogram: one hour)

RCDAQ can read out our detector-specific devices (think FELIX), and many commercial devices that are often used in lab tests (like CAEN V1742, SRS, DRS4, …)

The latest addition is Nalu’s ASocV3 (in progress)

We have been (and are, all BNL/Yale/SBU, test beams) using RCDAQ for our R&D, ample operational experience among ePIC members

Superb support for automated measurements that we will need for many tests

Support for analysis and online monitoring (not enough time today, maybe another time)

(I brought a DRS4 with me if you want a demo in a coffee break)

BTW: We maintain a permanent “bridgehead” at the Fermi FTBF and can spin up a readout system in an hour after you are through the front gate

There is one thing I haven’t told you yet…

What does “RCDAQ” stand for?

The “Really Cool Data Acquisition”

Thank you!

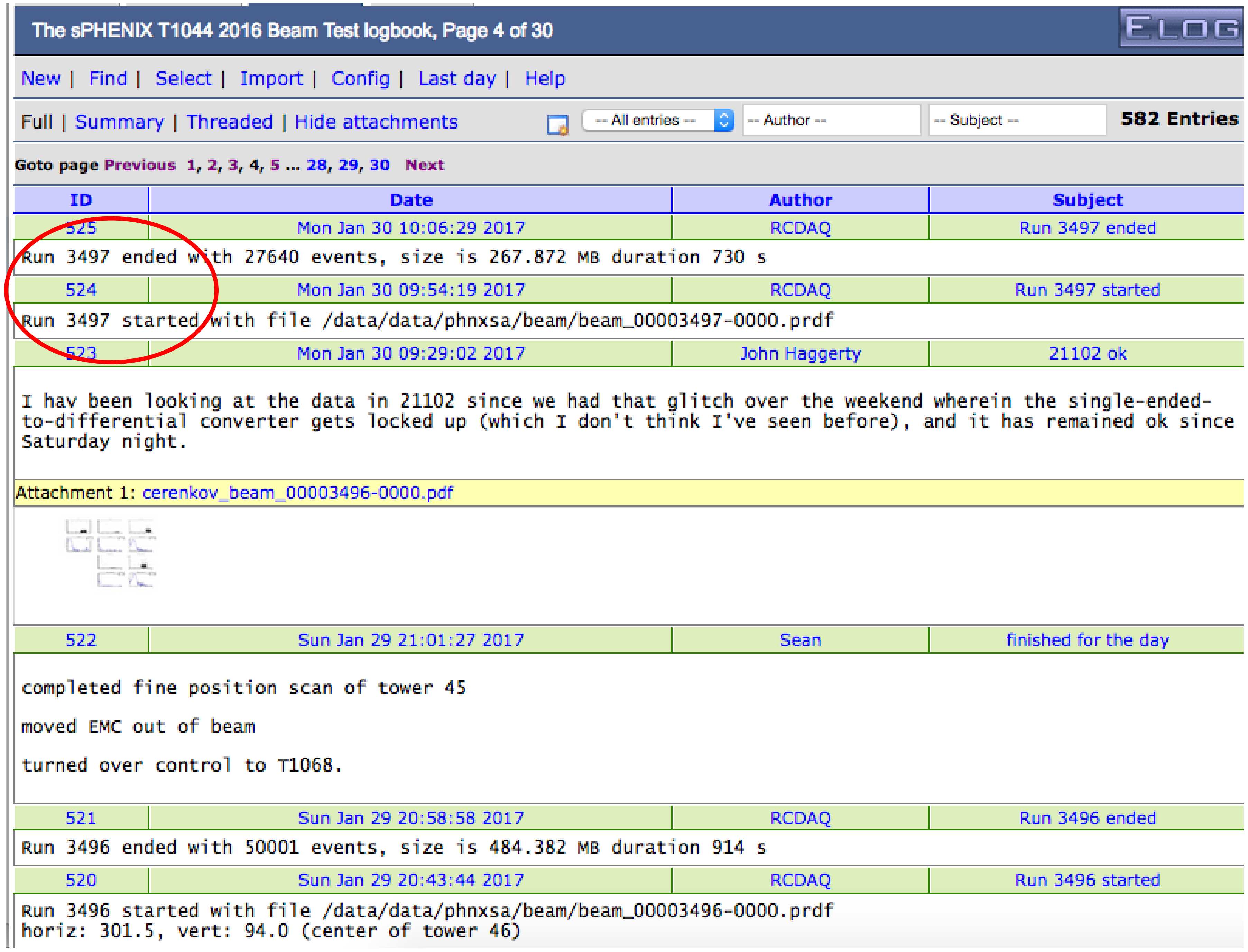
This page is intentionally left blank

Automated Elog Entries

RCDAQ can make automated entries in your Elog

Of course you can make your own entries, document stuff, edit entries

Gives a nice timeline
and log



The sPHENIX T1044 2016 Beam Test logbook, Page 4 of 30

ELOG
New | Find | Select | Import | Config | Last day | Help
Full | Summary | Threaded | Hide attachments
-- All entries --
-- Author --
-- Subject --

582 Entries

Goto page Previous 1, 2, 3, 4, 5 ... 28, 29, 30 Next

ID	Date	Author	Subject
525	Mon Jan 30 10:06:29 2017	RCDAQ	Run 3497 ended

```
Run 3497 ended with 27640 events, size is 267.872 MB duration 730 s
```

ID	Date	Author	Subject
524	Mon Jan 30 09:54:19 2017	RCDAQ	Run 3497 started

```
Run 3497 started with file /data/data/phnxsa/beam/beam_00003497-0000.prdf
```

ID	Date	Author	Subject
523	Mon Jan 30 09:29:02 2017	John Haggerty	21102 ok

```
I hav been looking at the data in 21102 since we had that glitch over the weekend wherein the single-ended-
to-differential converter gets locked up (which I don't think I've seen before), and it has remained ok since
Saturday night.
```

Attachment 1: cerenkov_beam_00003496-0000.pdf

ID	Date	Author	Subject
522	Sun Jan 29 21:01:27 2017	Sean	finished for the day

```
completed fine position scan of tower 45

moved EMC out of beam

turned over control to T1068.
```

ID	Date	Author	Subject
521	Sun Jan 29 20:58:58 2017	RCDAQ	Run 3496 ended

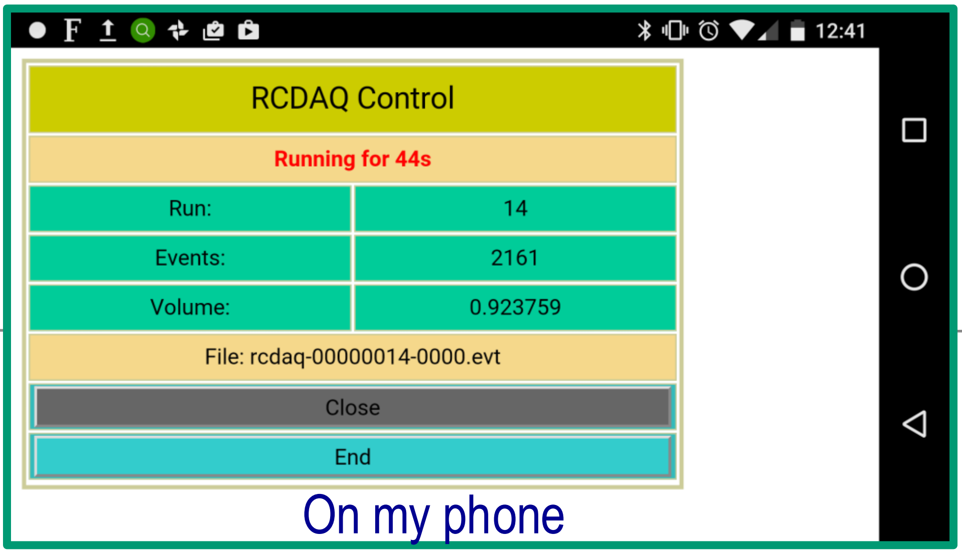
```
Run 3496 ended with 50001 events, size is 484.382 MB duration 914 s
```

ID	Date	Author	Subject
520	Sun Jan 29 20:43:44 2017	RCDAQ	Run 3496 started

```
Run 3496 started with file /data/data/phnxsa/beam/beam_00003496-0000.prdf
horiz: 301.5, vert: 94.0 (center of tower 46)
```

GUIs

- **GUIs must not be stateful!**
- Statelessness allows to have multiple GUIs at the same time
- And allows to mix GUIs with commands (think scripts)
- (all state information is kept in the rcdaq server)
- My GUI approach is to have perl-TK issue standard commands, parse the output
- Slowly transitioning to Web-based controls (web sockets + Javascript)


RCDAQ Control

Running for 44s

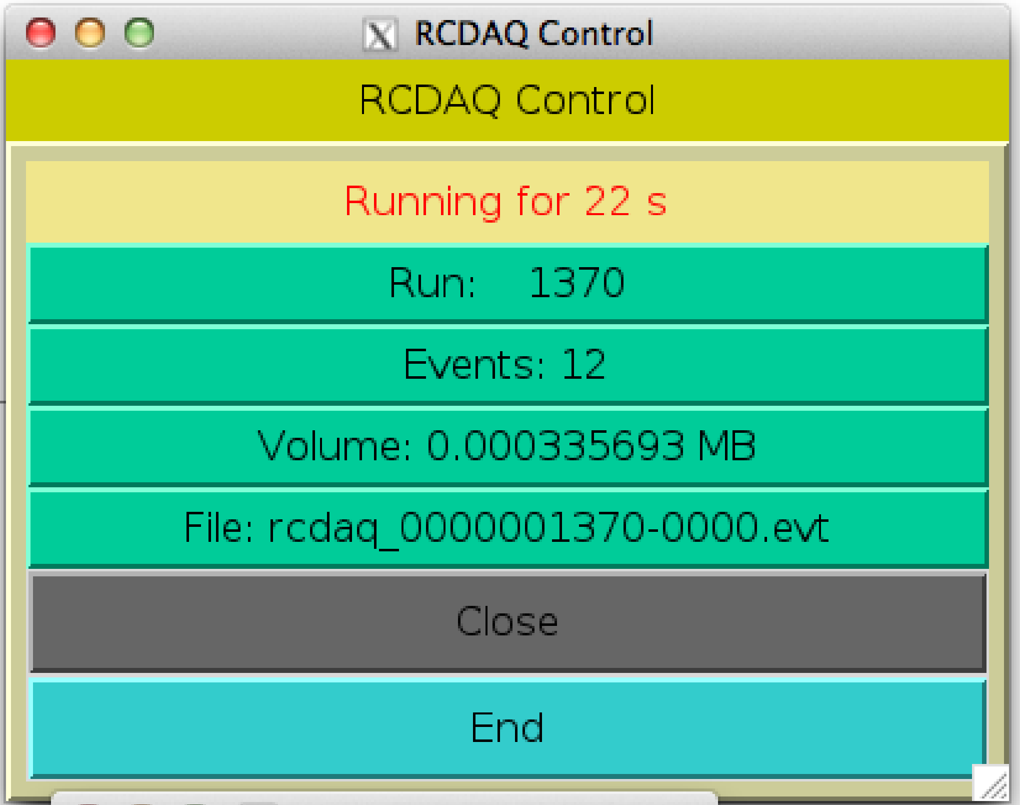
Run:	14
Events:	2161
Volume:	0.923759

File: rcdaq-00000014-0000.evt

Close

End

On my phone


RCDAQ Control

RCDAQ Control

Running for 22 s

Run: 1370

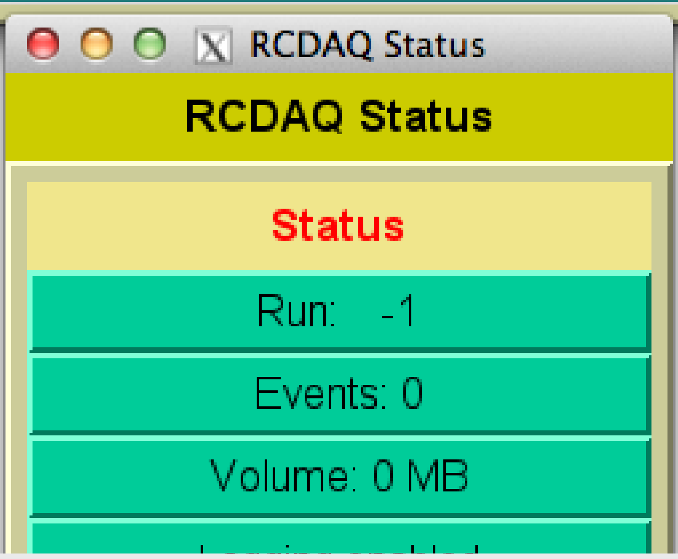
Events: 12

Volume: 0.000335693 MB

File: rcdaq_0000001370-0000.evt

Close

End


RCDAQ Status

RCDAQ Status

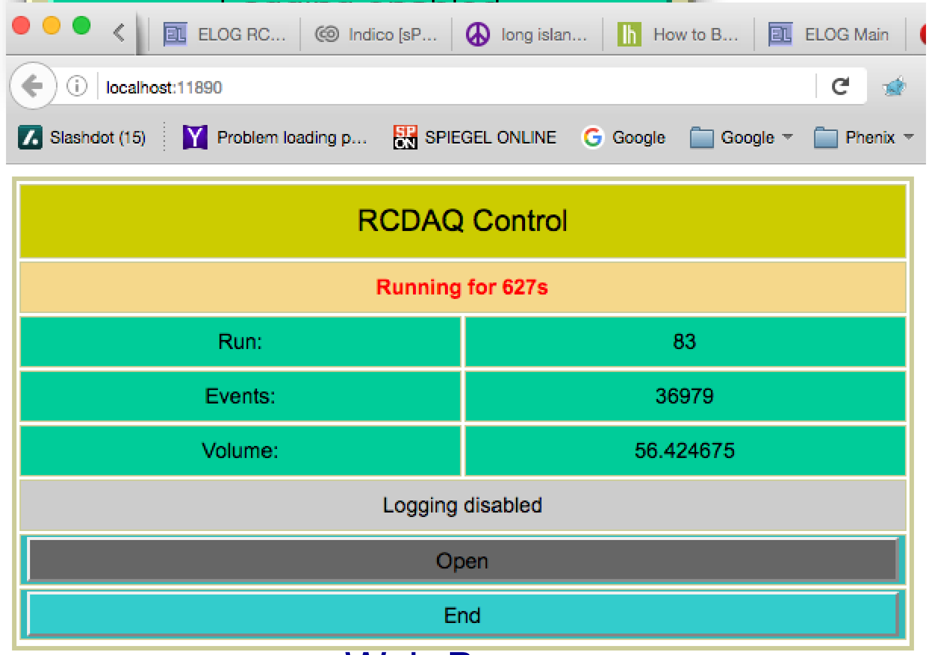
Status

Run: -1

Events: 0

Volume: 0 MB

Perl-TK



RCDAQ Control

Running for 627s

Run:	83
Events:	36979
Volume:	56.424675

Logging disabled

Open

End

Web Browser

Ok, that was a lot about test beams… what about sPHENIX

I harped on lab tests / test beams etc a bit because that’s what ePIC will be busy with for some time

What do we do in sPHENIX?

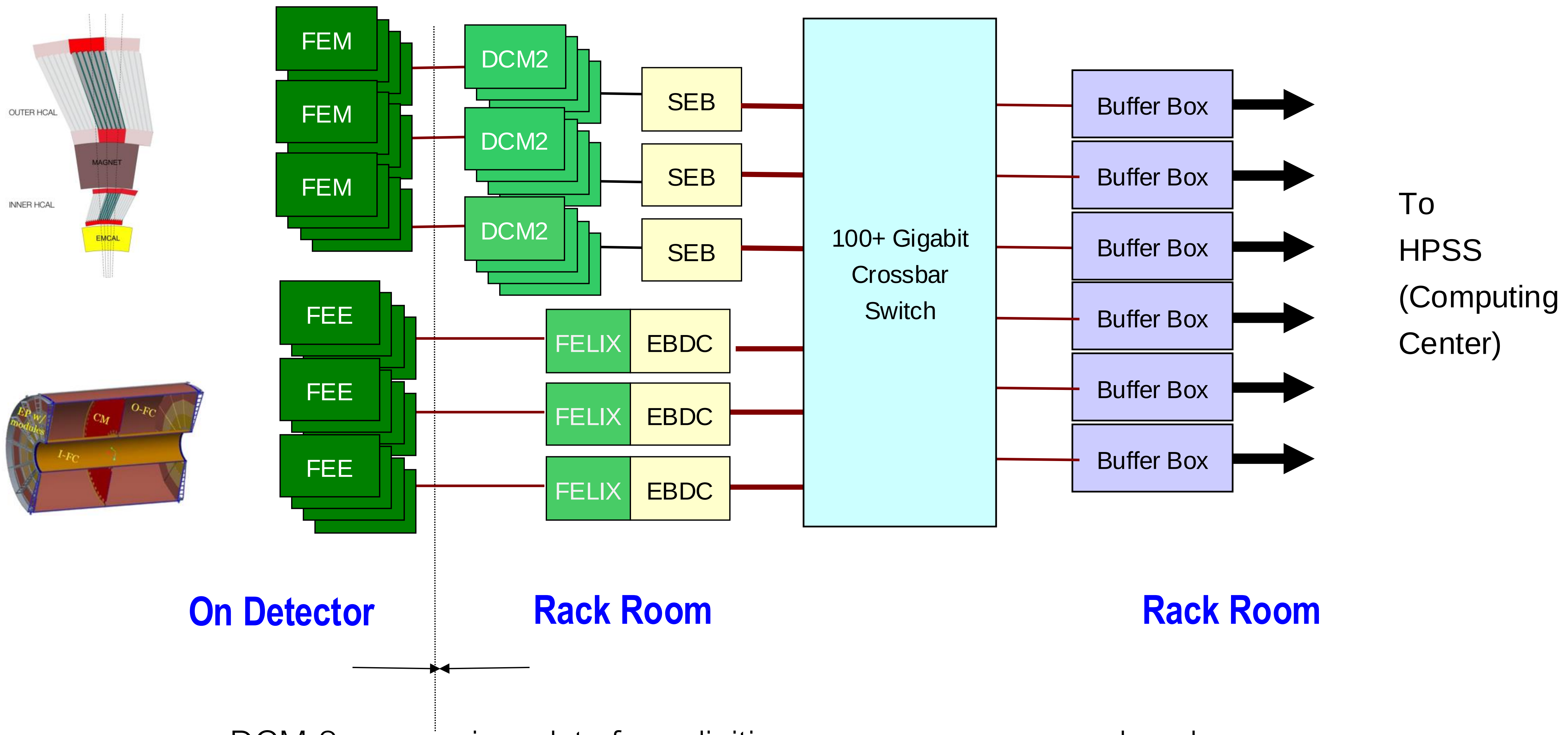
In short, each detector element connects to a PC that runs its own instance of RCDAQ

It’s all “glued” together with the timing system

And our (in 2023) 52 RCDAQ instances are controlled by a meta-process “RunControl”

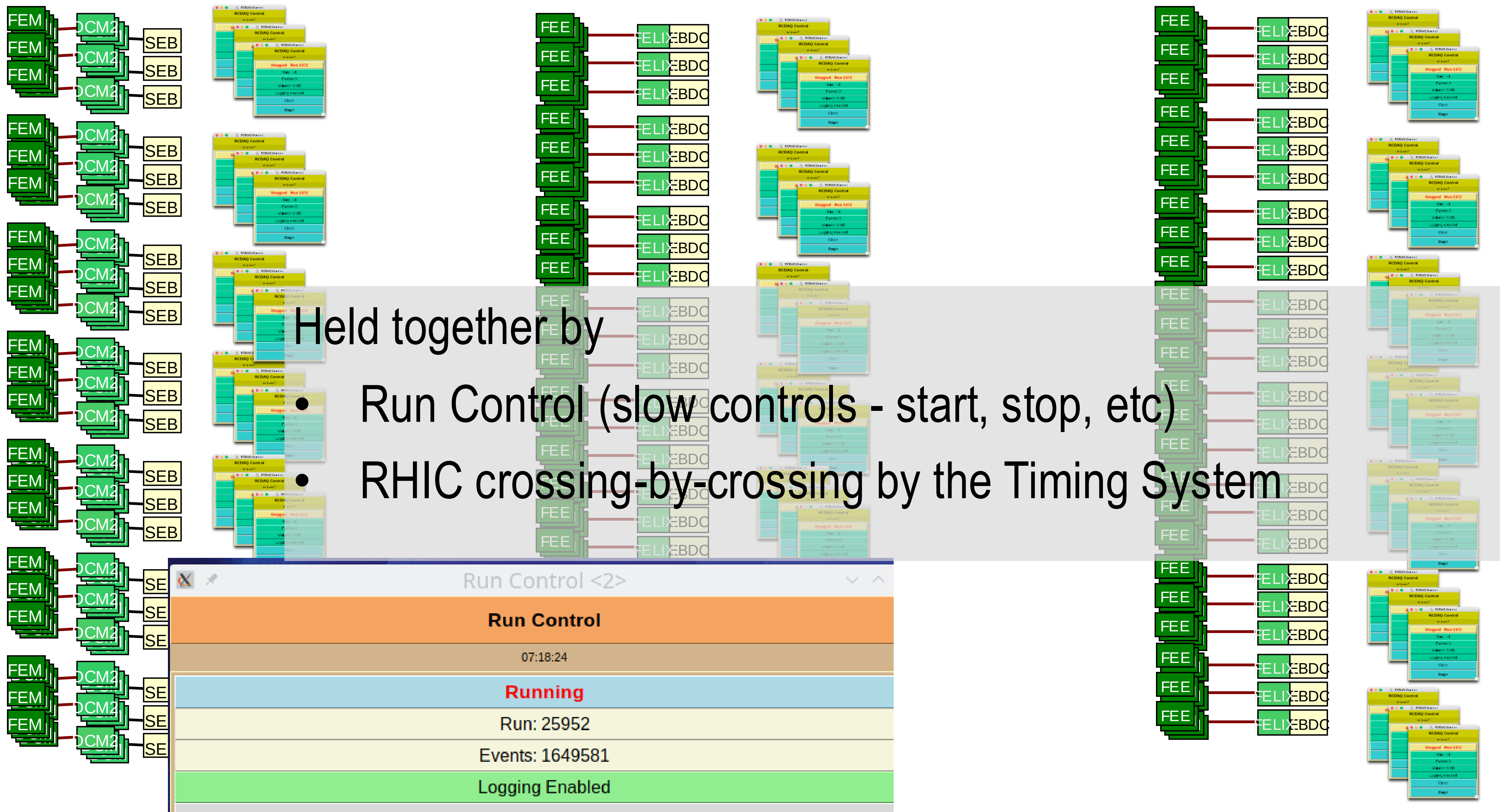
The RCDAQs run in a “run control mode” that reduces some of their autonomy so that Run Control is, well, in control

sPHENIX DAQ Bird's eye view



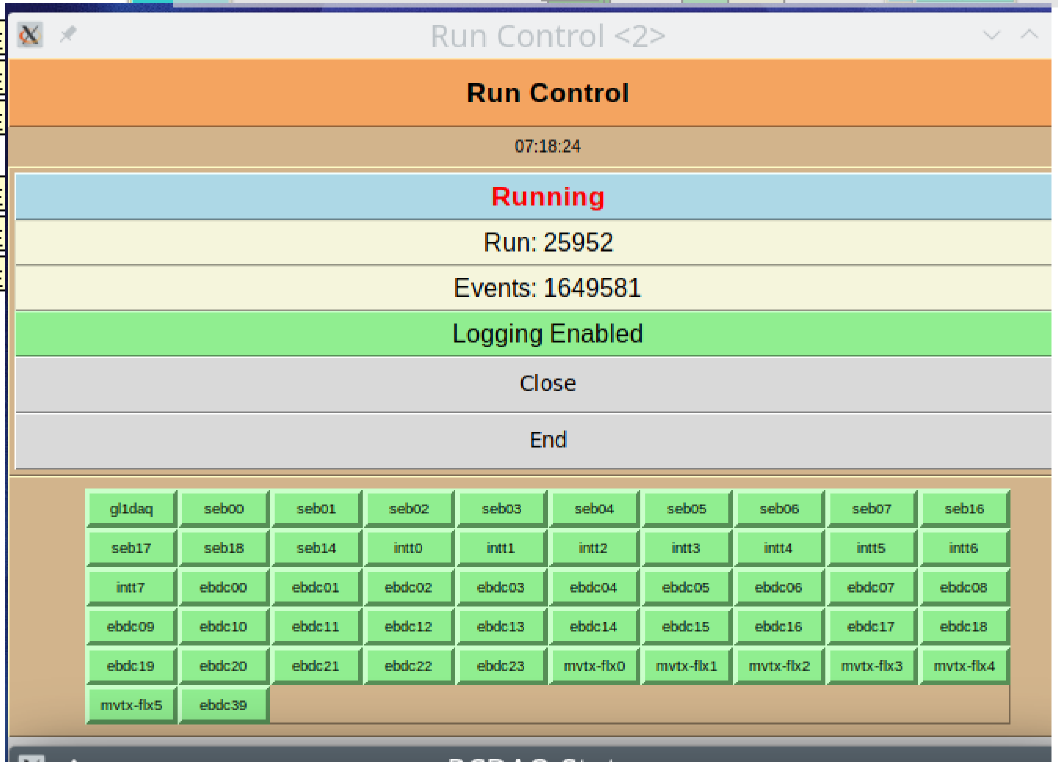
• DCM-2	receives data from digitizer, zero-suppresses and packages
• SEB	collects data from a DCM group (~20)
• EBDC	Event Buffer and Data Compressor (~40)
• Buffer Box	data interim storage before sending to the computing center (6)

many, many RCDAQs in sPHENIX



Held together by

- Run Control (slow controls - start, stop, etc)
- RHIC crossing-by-crossing by the Timing System


Run Control <2>

Run Control

07:18:24

Running

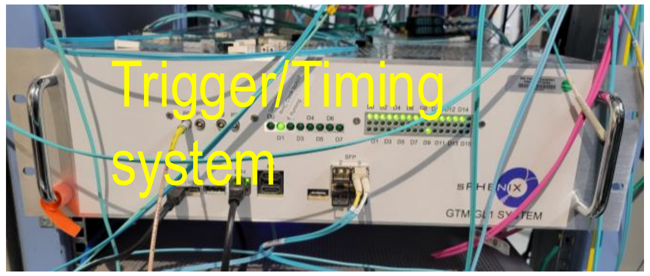
Run: 25952

Events: 1649581

Logging Enabled

Close

End



Coming back to the “shell command” feature

For the last 3 minutes, I want to harp some more on the superiority of that “everything is a shell command” approach

Often I’m learning of a new ingenious way to use this aspect for something cool

A real good tool gets used in ways that the designer did not envision… but it works!

A group needed to test a few thousand pads on a plane if they a) work and b) are connected right.

Inject charge into the pads one by one... **but you can't take your eyes (or the probe) off the pad plane or you lose your position**

They came up with…

Data Formats in general…

One of the trickiest parts when developing a new application is defining a data format

It can take up easily half of the overall effort – think of Microsoft dreaming up the format to store this very PowerPoint presentation you are seeing in a file. We used to have ppt, now we have pptx – mostly due to limitations in the original format design

A good data format takes design skills, experience, but also the test of time

The tested format usually comes with an already existing toolset to deal with data in the format, and examples – nothing is better than a working example

Case in point: We could easily accommodate the sPHENIX Streaming Readout data in this format, event though no one had ever heard the term when I designed this

I have no time today to talk about the analysis end / online monitoring, etc of this, maybe another time

Modularity and Extensibility

No one can foresee and predict requirements of a data format 20 years into the future.

Must be able to grow, and be extensible

The way I like to look at this:

FedEx (and UPS) cannot possibly know how to ship every possible item under the sun

But they know how to ship a limited set of box formats and types, and assorted weight parameters and limits

Whatever fits into those boxes can be shipped

During transport, they only look at the label on the box, not at what's inside

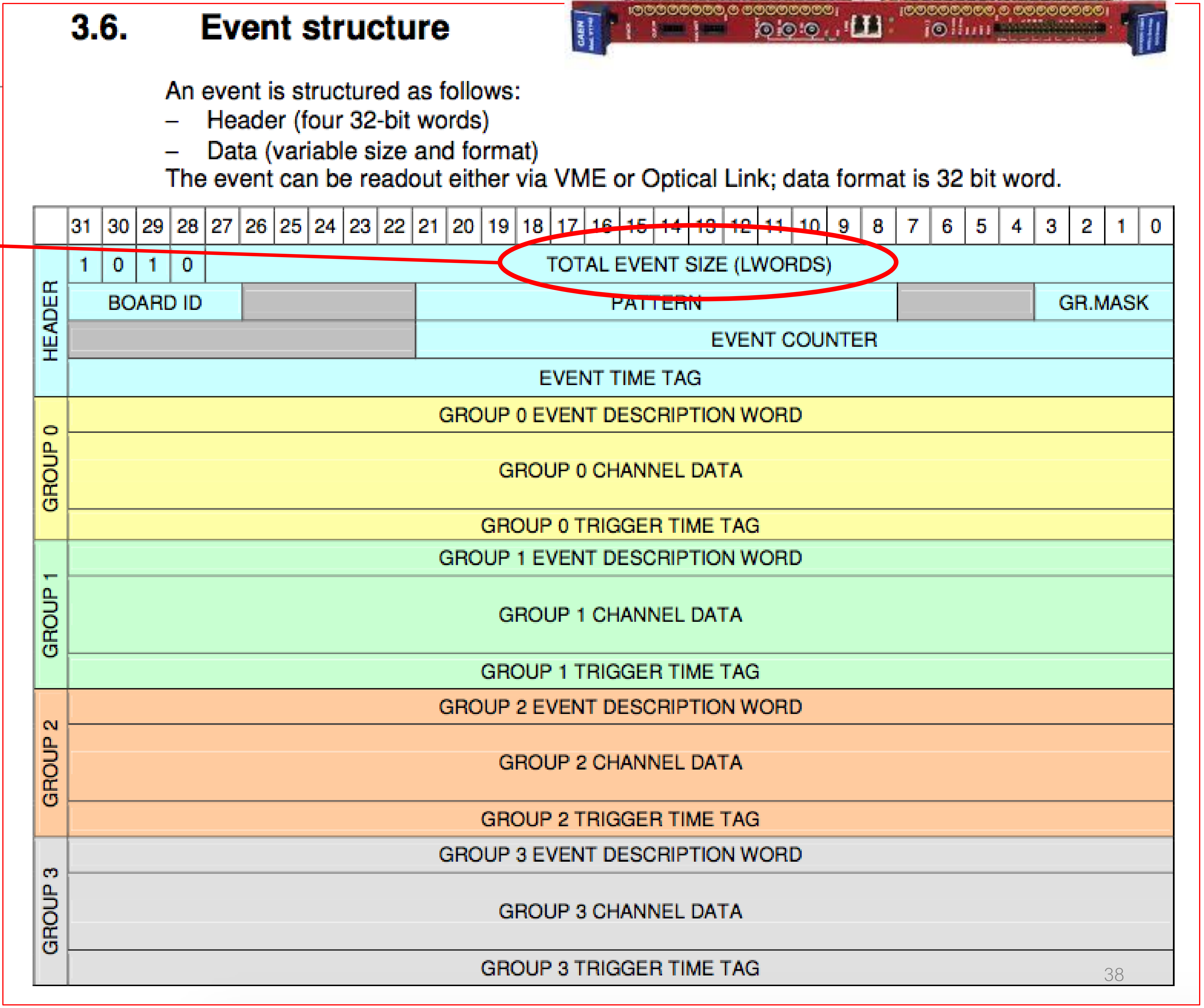
We will see a surprisingly large number of similarities with that approach in a minute

Example: CAEN's V1742 format

We just take that blob of memory, "put it in a box", done.

The analysis software takes care of the unpacking and interpretation later

Just grab it. Don't waste time here.



3.6. Event structure

An event is structured as follows:

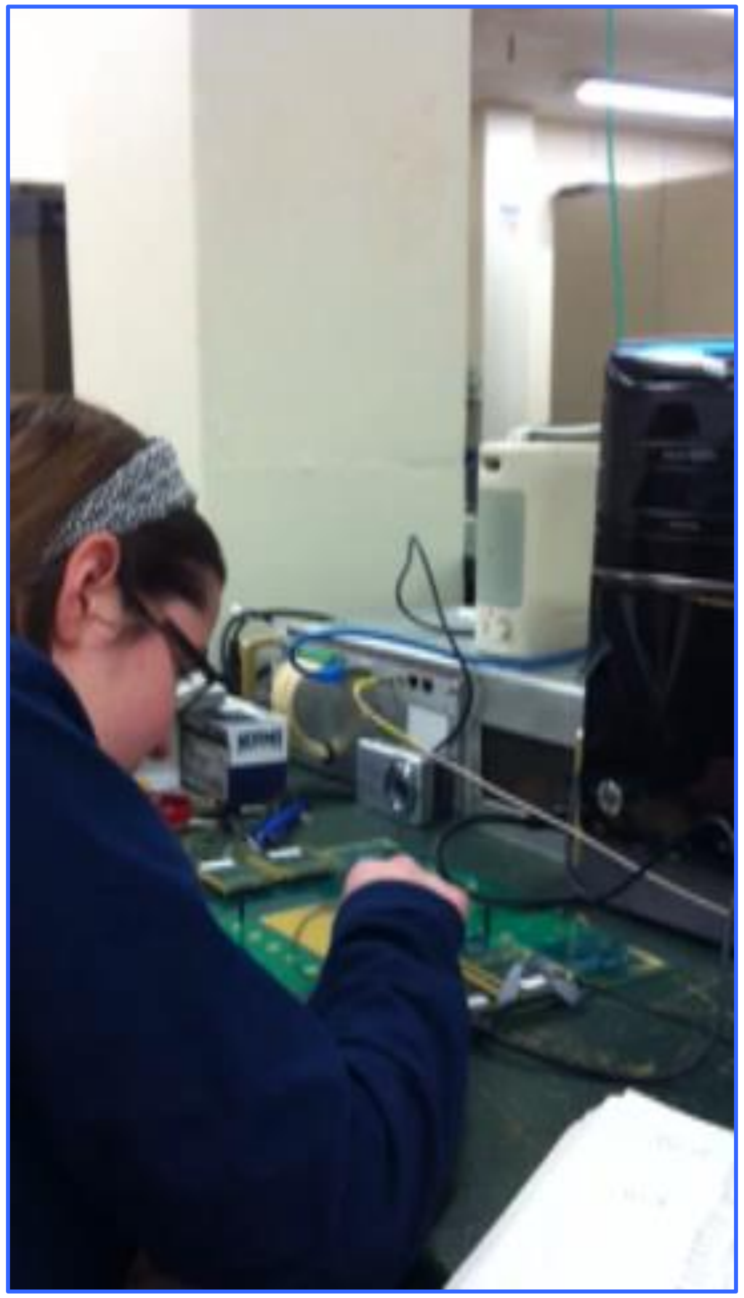
– Header (four 32-bit words)

– Data (variable size and format)

The event can be readout either via VME or Optical Link; data format is 32 bit word.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEADER	1	0	1	0	TOTAL EVENT SIZE (LWORDS)																											
	BOARD ID										PATTERN																		GR.MASK			
											EVENT COUNTER																					
	EVENT TIME TAG																															
GROUP 0	GROUP 0 EVENT DESCRIPTION WORD																															
	GROUP 0 CHANNEL DATA																															
	GROUP 0 TRIGGER TIME TAG																															
GROUP 1	GROUP 1 EVENT DESCRIPTION WORD																															
	GROUP 1 CHANNEL DATA																															
	GROUP 1 TRIGGER TIME TAG																															
GROUP 2	GROUP 2 EVENT DESCRIPTION WORD																															
	GROUP 2 CHANNEL DATA																															
	GROUP 2 TRIGGER TIME TAG																															
GROUP 3	GROUP 3 EVENT DESCRIPTION WORD																															
	GROUP 3 CHANNEL DATA																															
	GROUP 3 TRIGGER TIME TAG																															

Shell integration



THE SPEAKING DAQ

```
#! /bin/sh

rcdaq_client daq_setfilerule /home/sbeic/calibfiles/srs-%010d-%02d.evt

for column in $(seq $1 $2) ; do

   for row in $(seq 0 20) ; do
      echo "$column and row $row" | festival --tts
      sleep 2

      echo "Go" | festival --tts

      echo rcdaq_client daq_begin  ${column}555${row}
      rcdaq_client daq_begin  ${column}555${row}

      sleep 3
      echo "End" | festival --tts

      echo rcdaq_client daq_end
      rcdaq_client daq_end


   done
done

rcdaq_client daq_setfilerule /home/sbeic/datafiles/srs-%04d-%02d.evt
```

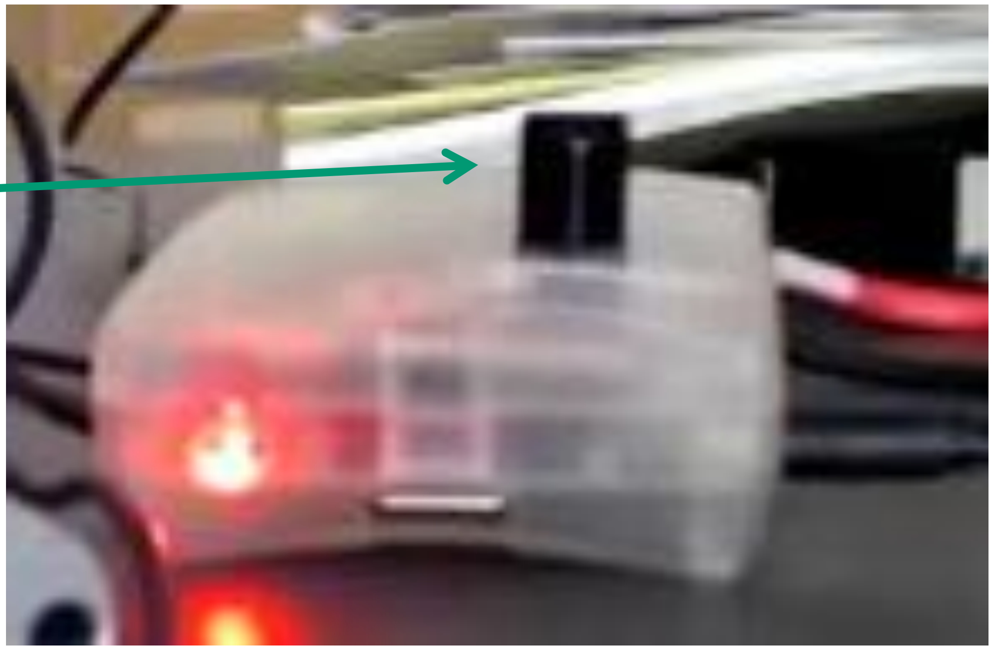
One more cool thing

Anything that's capable of issuing a shell command can control the DAQ

I have said (but not shown you yet) that the DAQ can be controlled remotely, through the network

I have a Raspberry Pi connected here that I have set up so it controls RCDAQ running on my Laptop

And you see it has developed some kind of growth on its head… That's an infrared receiver



We know we can assign arbitrary commands to buttons pressed on virtually any IR remote

I guess you see where this is going…