ROC readout in RCDAQ

Of course that was my personal main goal that I hoped to get out of Norbert's visit

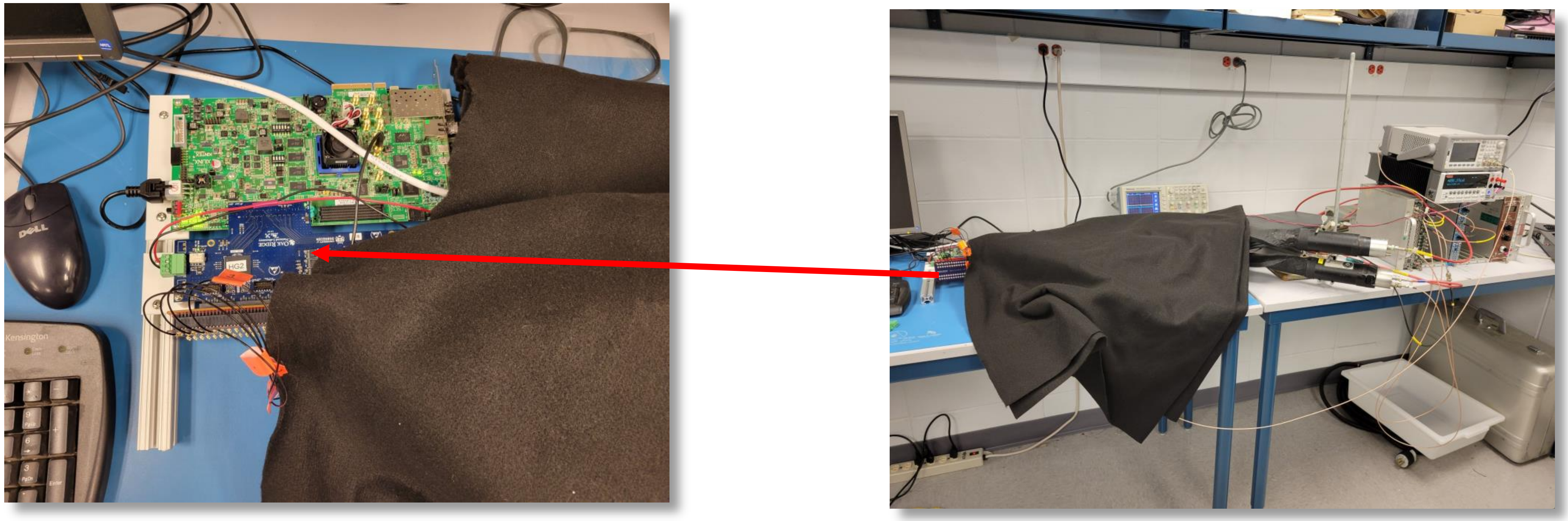
And thanks to Norbert's help, done

RCDAQ plugin is in place

I made an "ePIC" fork of what is known as the Event Libraries that give you access to the standard APIs to access the data

In there is a decoder for the "H2GCROC3" hitformat (typically, each supported device has a dedicated decoder that presents the data in a small number of standard APIs)

Yesterday I took cosmics data with RCDAQ for about an hour



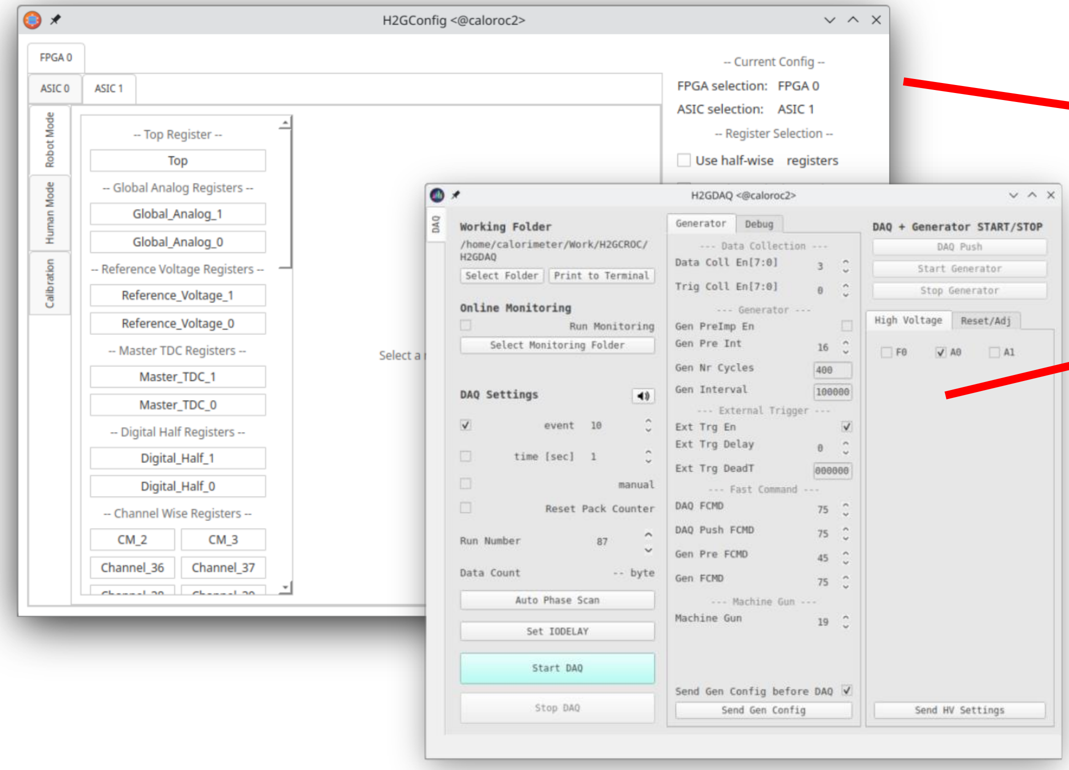
Front-end configuration vs RCDAQ configuration

Like many other devices with a vast configuration space (here: some 3000 parameters, most with reasonable defaults), the configuration of the front-end and the readout are two distinct tasks

The front-end is getting configured with two GUI-based processes (H2GConfig for the ASICs, H2GDAQ for the FPGA portion), and set up just the way we want it

RCDAQ is almost fully oblivious to the particulars of the front-end configuration, it reads whatever comes its way

It really only needs to know the IP address from which the data are coming; it also issues “start” and “stop” commands to the front-end to start or stop the flow of data – that’s all.


H2GConfig <@caloroc2>

-- Current Config --
FPGA selection: FPGA 0
ASIC selection: ASIC 1
-- Register Selection --
□ Use half-wise registers

H2GDAQ <@caloroc2>

Working Folder
/home/calorimeter/Work/H2GCROC/H2GDAQ

Online Monitoring
□ Run Monitoring

DAQ Settings
☑ event 10
□ time [sec] 1
□ manual
□ Reset Pack Counter
Run Number 87
Data Count -- byte

--- Data Collection ---
Data Coll En[7:0] 3
Trig Coll En[7:0] 0
--- Generator ---
Gen PreImp En □
Gen Pre Int 16
Gen Nr Cycles 400
Gen Interval 100000
--- External Trigger ---
Ext Trg En ☑
Ext Trg Delay 0
Ext Trg DeadT 000000
--- Fast Command ---
DAQ FCMD 75
DAQ Push FCMD 75
Gen Pre FCMD 45
Gen FCMD 75
--- Machine Gun ---
Machine Gun 19
Send Gen Config before DAQ ☑

DAQ + Generator START/STOP
□ F0 ☑ A0 □ A1

The standalone configuration process

The plan is to support command based “canned” configs – GUIs are nice but can also be tedious.

The RCDAQ Plugin and configuration

I made a standard script like the one that I presented the other week, not much new for y'all here

(and yes, I left out a “c” in “librcdaqplugin_h2gcro3.so” library name ☺   - will be fixed

```
$ more rcdaq_roc.sh
#! /bin/sh

# we need the $0 as absolute path b/c we pass it on to a "file" device
further down
MYSELF=$(readlink -f $0)

# we figure out if a server is already running
if ! rcdaq_client daq_status > /dev/null 2>&1 ; then

    echo "No rcdaq_server running, starting... log goes to $HOME/rcdaq.log"
    rcdaq_server > $HOME/rcdaq.log 2>&1 &
    sleep 2
fi
```

```
# we add this file to the begin-run event
rcdaq_client create_device device_file 9 900 "$MYSELF"

#rcdaq_client load $ONLINE_MAIN/lib/librcdaqplugin_h2gcro3.so
rcdaq_client load librcdaqplugin_h2gcro3.so

rcdaq_client create_device device_h2gcroc3 1 12001 10.1.2.208 1 128
```

This is all that the plugin needs to know

The RCDAQ Plugin at work

This is how the configured RCDAQ looks like

```
$ daq_status -ll
caloroc2 -  Stopped
  Filerule:     /home/phnxrc/data/junk/junk_ROC-%08d-%04d.evt
  Logging enabled
  Number of buffers/write threads: 2 Buffersize: 64MB
  compression disabled level: 0
  MD5 calculation enabled
  have a trigger object
  Buffer Sizes:     65536 KB adaptive buffering: 15 s
  Web control Port:  8899
  Elog: not defined
  -- defined Run Types:
          cosmics - /home/phnxrc/data/cosmics/cosmics_ROC-%08d-%04d.evt
             junk - /home/phnxrc/data/junk/junk_ROC-%08d-%04d.evt
  List of loaded Plugins:
   - H2GCROC3 Plugin, provides -
 -     device_h2gcroc3 (evttype, subid, IP addr, trigger, nr_packets) - readout an H2GCROC3
```

```
$ daq_list_readlist
File Device  Event Type: 9 Subevent id: 900 reading from /home/phnxrc/ROC/new ROC
firmware/rcdaq_roc.sh
H2GCROC3 Device  Event Type: 1 Subevent id: 12001 IP: 10.1.2.208 nr_packets: 128 trigger
```

Analysis with pmonitor

The device generates a packet with hitformat 301 (mnemonic “IDH2GCROC3”)

Remember these numbers are a simple enumeration, and numbers are plentiful

```
$ dlist /home/phnxrc/data/cosmics/cosmics_ROC-00000012-0000.evt
Packet 12001  7304 -1 (sPHENIX Packet) 301  (IDH2GCROC3)
```

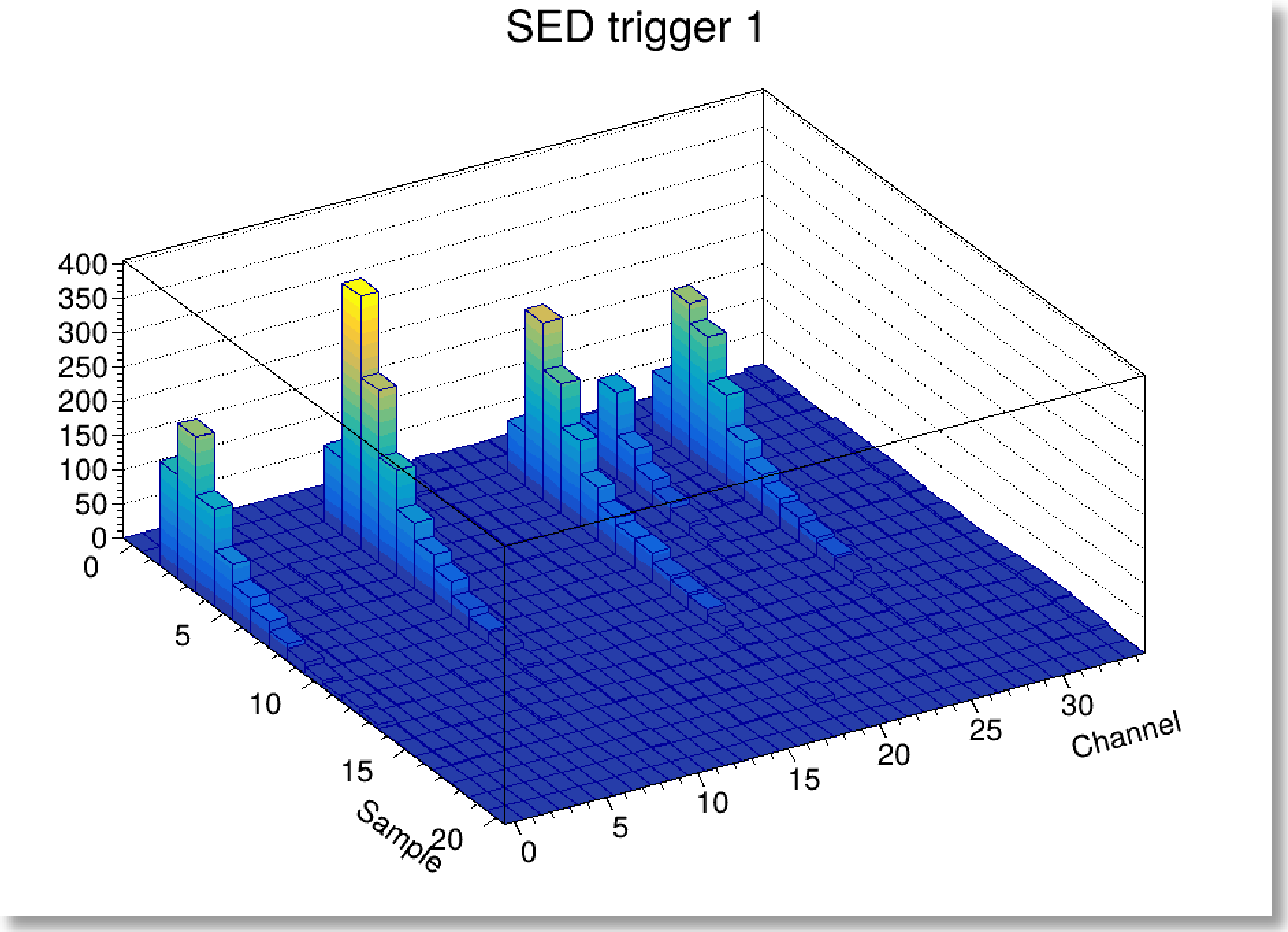
The data access APIs are exactly like those for any waveform sampler we support (CAEN V7142, DRS4, the sPHENIX digitizers, the (sPHENIX) Sampa (== ePIC’s Salsa) chips…).

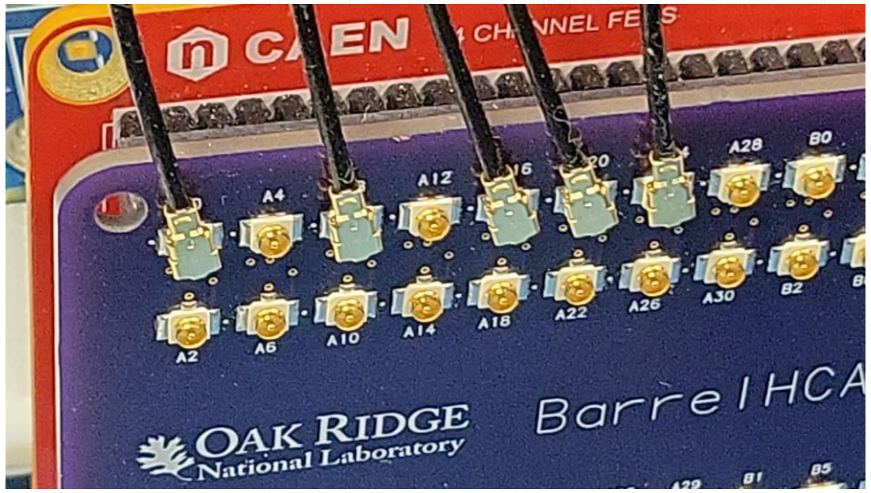
```
p->iValue(0,"SAMPLES")  tells you how many samples are present

p->iValue(0,”CHANNELS") tells you how many channels that device has  (here:144)

p->iValue(c,s)          gives you sample s from channel c  (there are faster interfaces available)
```

There are many more APIs to ask the packet all kinds of other questions, but those are the most-often used ones here

The “mlp standard plot”

Whenever I have data from a waveform sampler (that is, a device with a number of channels, and each channel provides a waveform) I make a Lego plot with Channel vs Sample, with z the sample value:





Connected channels

We see what we expect: Signals from 5 tiles, and with the right channel numbers

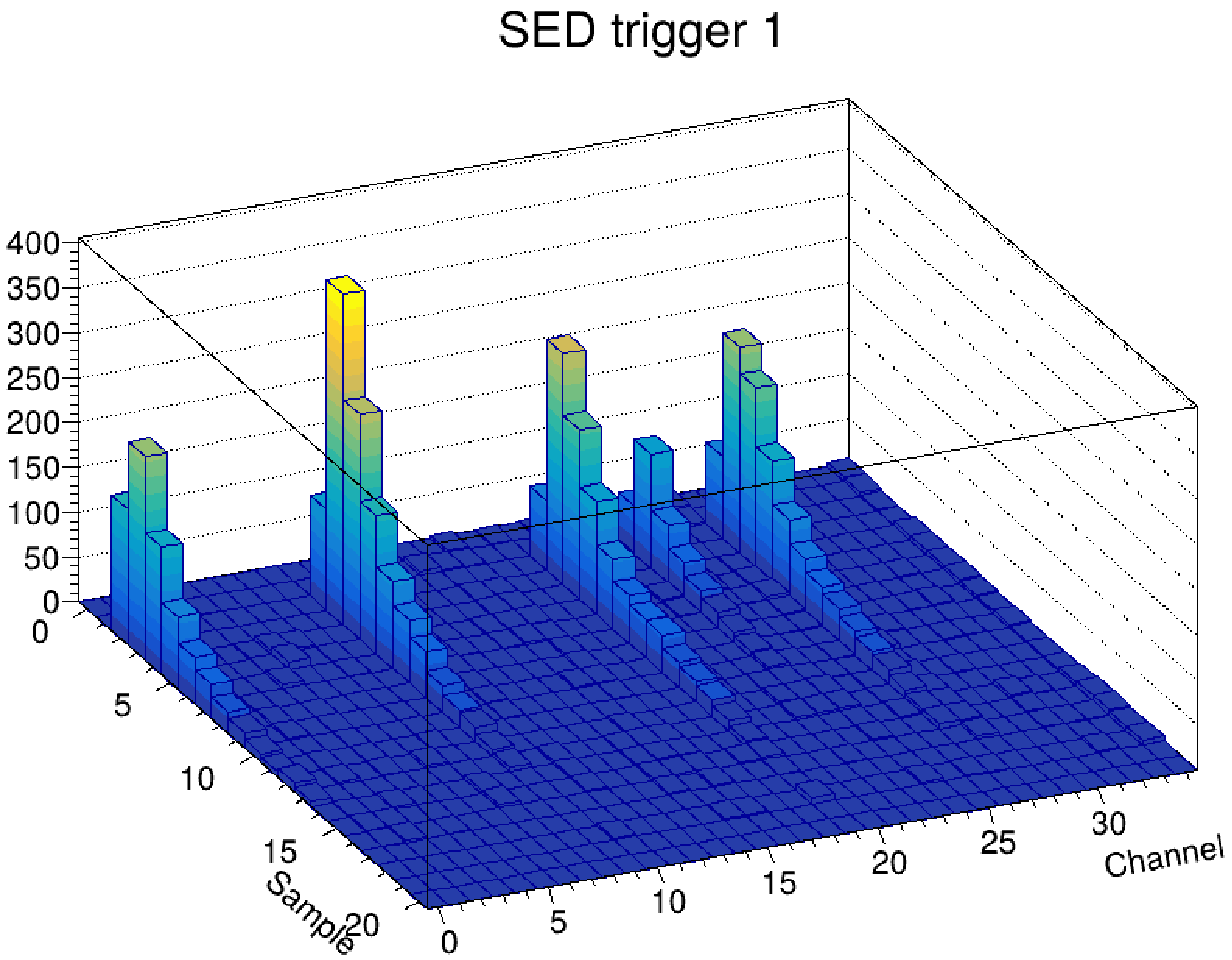
The waveforms are baseline-corrected. And we saw the first of those plots from online monitoring yesterday!

Some eye-candy…

I made a quick movie of the response from ~150 triggers.

We have some empty events, and (b/c the trigger paddles are larger than the tiles), a number of events where the triggered-on cosmic misses the tiles.

But still nice to watch…


SED trigger 1
Sample
Channel