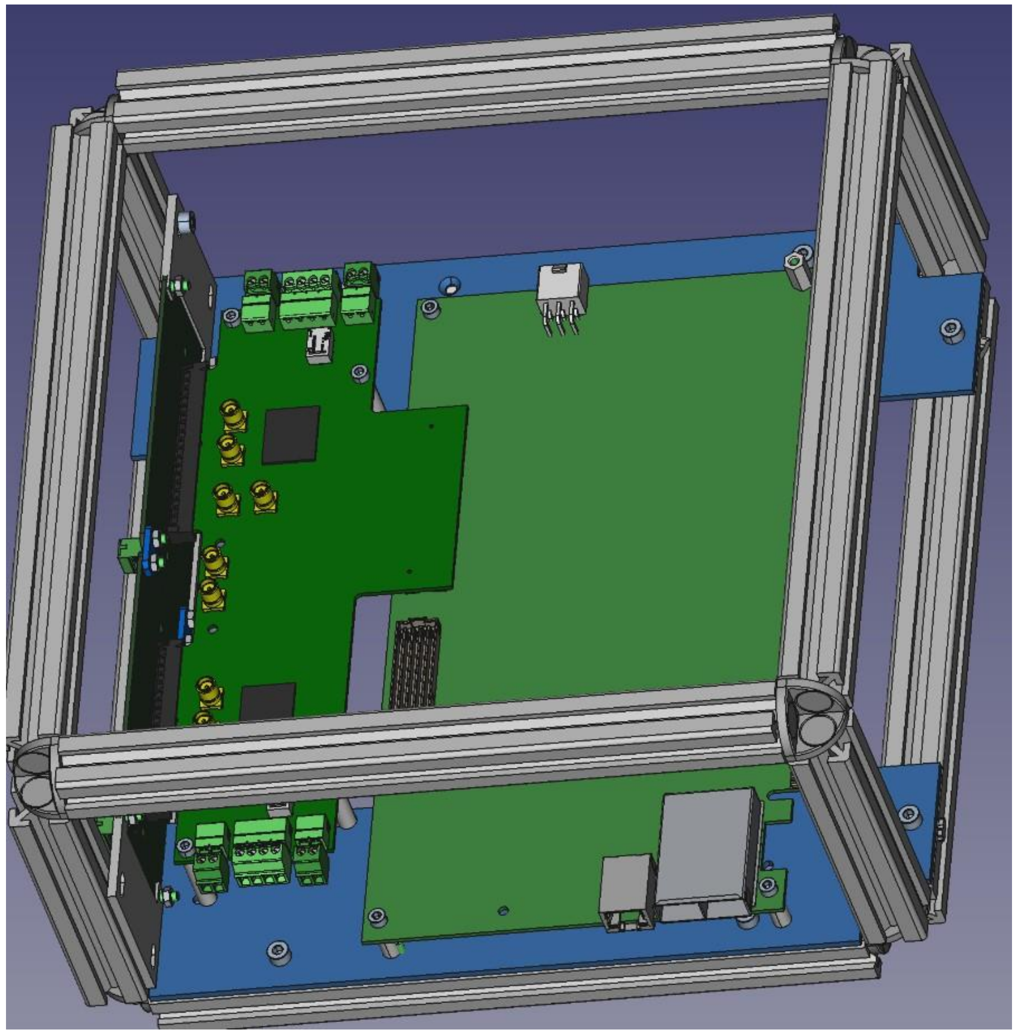
ROC Readout Status

First off, I'm working with Dan Cacace to make a proper case for the system – Nobert has shown his 80/20-based case, and I got the design files.

As it is, the setup is too fragile to be taken to a lab, or even the more factory floor-like HighBay



Where we are

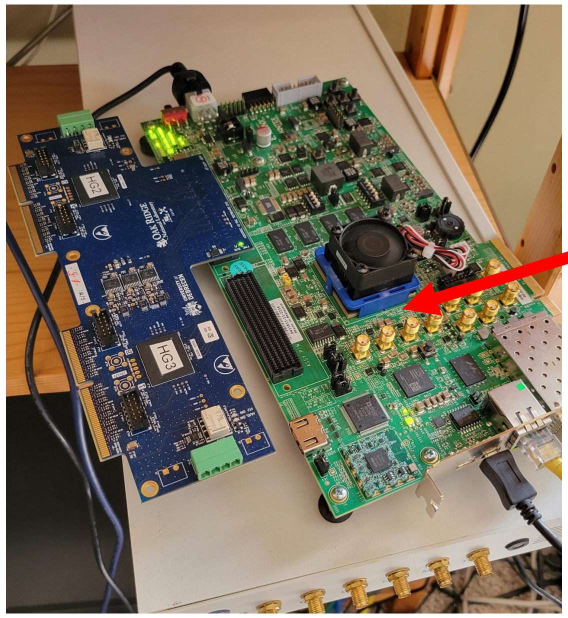
I have taken the actual readout part as far as I can take it with the current firmware

Waiting for the firmware mods that chiefly allow me to recognize the end-of event, and wrap up

I am working on storing the firmware on the KCU's SD card (or flash it) so a simple power-cycle will bring a gone-stale card back to life

As it stands, I need to keep the system close to a Vivado-equipped (or at least Hwmanager-equipped) PC

(and it does go into a weird state often enough – I basically reload the firmware each time I get back to taking some data).



this is the KCU105, btw

For those who attended my tutorial Wednesday…

I showed the equivalent of this in the tutorial yesterday…

this now means something to y'all :

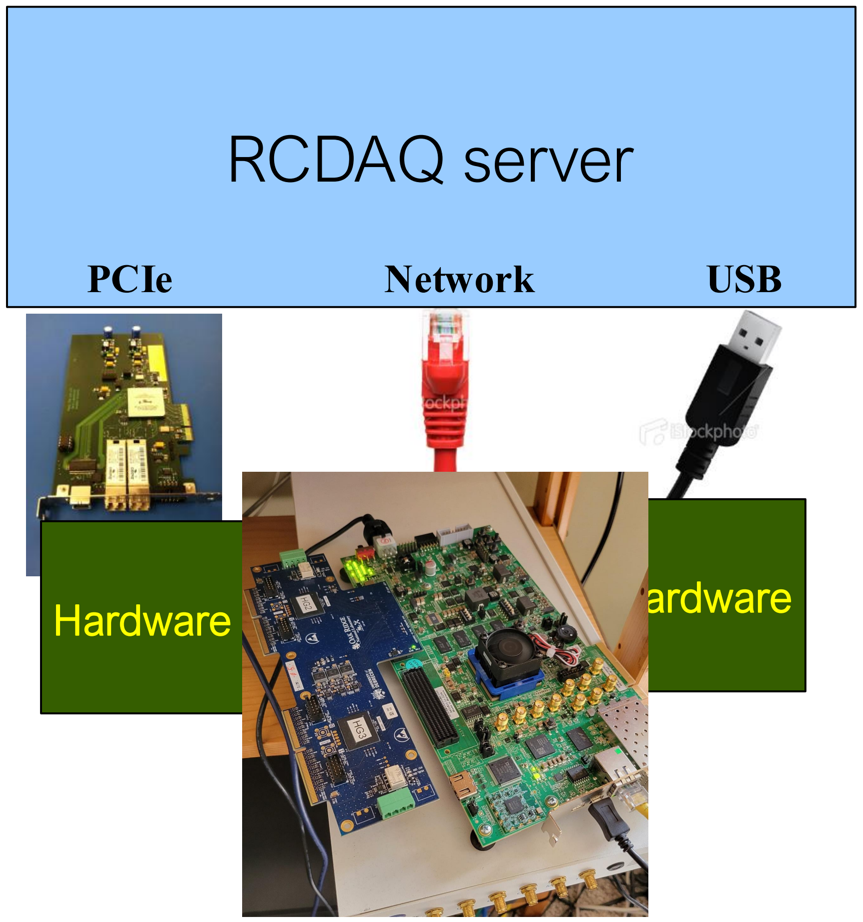
```
rcdaq_client load librcdaqplugin_h2gcro3.so

rcdaq_client create_device device_h2gcroc3 1 12001 10.1.2.208
```

Now to the issue(s)

Really only two –

- My inability to recognize the end of the event with the current firmware
- Beyond that, the mixing of different-trigger data into the same network packet

The way the system sends data is through the network (I showed this on Wednesday):



Each network packet is 1452 bytes long and contains 40byte-chunks that hold the information

We need the end-of event markers…

Not really true – yes we find 19 samples in this event, but it should really be 9

I have to make a guess when to “stop”, and I guessed wrong.

```
$ ddump /bigdata/purschke/muell/ROC-00000013-0000.evt
Packet 12001  3290 -1 (sPHENIX Packet) 301 (IDH2GCROC3)
Nr of samples: 19
33 74 115 156 197 238 279 320 361 402 100051 100092 100133 100174 100215 100256
100297 100338 100420
  0 |  146 146 147 148 147 146 146 148 147 147 148 147 147 148 147 148 148 147   0
  1 |  181 181 181 181 181 181 182 183 181 180 181 182 181 182 182 183 183 182   0
  2 |  169 169 169 169 168 168 168 169 168 168 167 169 169 168 169 168 169 169   0
  3 |  176 176 176 175 176 176 177 177 177 176 176 177 177 176 176 177 178 178   0
 <lines deleted >
 44 |  162 162 162 163 163 161 162 165 163 163 162 163 162 161 161 162 162 163 162
 45 |  165 164 165 164 165 164 165 164 164 164 163 164 164 165 163 165 164 165 164
 46 |  164 163 165 163 165 164 164 165 164 164 164 164 165 165 164 164 164 165 164
 <lines deleted >
138 |  196 196 196 197 195 196 196 196   0 196 197 196 195 196 197 195 194 196   0
139 |  164 165 164 165 165 165 166 165   0 166 164 164 164 164 165 166 165 164   0
140 |  186 185 186 186 186 186 187 187   0 186 185 185 186 186 185 186 186 186   0
141 |  185 185 186 187 186 186 185 185   0 187 186 186 186 185 185 185 185 185   0
142 |  191 194 191 191 191 191 191 191   0 191 191 193 193 191 191 191 191 191   0
143 |  188 187 189 189 188 188 188 188   0 188 186 187 187 188 186 187 188 186   0
```

“Guessing better” doesn’t help…

Ok, I didn’t say “stop” after the right number of network packets, but this wouldn’t help

I made a quick pmonitor project to show this better:

```
root [1] prun(5)
Sample    0 time 33   0   0
Sample    1 time 74   1   0
Sample    2 time 115   2   0
Sample    3 time 156   3   0
Sample    4 time 197   4   0
Sample    5 time 238   5   0
Sample    6 time 279   6   0
Sample    7 time 320   7   0
Sample    8 time 361   8   0
Sample    9 time 402   9   0
Sample   10 time 100051   2440   1
Sample   11 time 100092   2441   1
Sample   12 time 100133   2442   1
Sample   13 time 100174   2443   1
Sample   14 time 100215   2444   1
Sample   15 time 100256   2445   1
Sample   16 time 100297   2446   1
Sample   17 time 100338   2447   1
```

I’m showing here the sample numbers and assorted times

It’s clear that those samples are one “event”/trigger

The time difference is 41 ticks that I make into that 0,1,2, tick units counter (2nd-last column)

I then assign a trigger number (last column) to the group

Here starts a new group

So what’s the problem?

One group should be one event.

But…

Different groups are mixed within the same network packet

So I added “pidentify(0)” to see the event boundaries

Lets look at that “group 1” from the last slide

```
Sample    9 time 402   9   0
Sample   10 time 100051  2440   1
Sample   11 time 100092  2441   1
Sample   12 time 100133  2442   1
Sample   13 time 100174  2443   1
Sample   14 time 100215  2444   1
Sample   15 time 100256  2445   1
Sample   16 time 100297  2446   1
Sample   17 time 100338  2447   1
Sample   18 time 100420  2449   1
 -- Event      3 Run:     13 length:   3298  type:   1  (Data Event)   1745360950
Sample    0 time 100297  2446   1
Sample    1 time 100338  2447   1
Sample    2 time 100379  2448   1
Sample    3 time 100420  2449   1
Sample    4 time 200069  4879   2
```

Some of the different time ranges are mixed together in on *network* packet

That means that I have no way to “close the event” when its samples have been sent

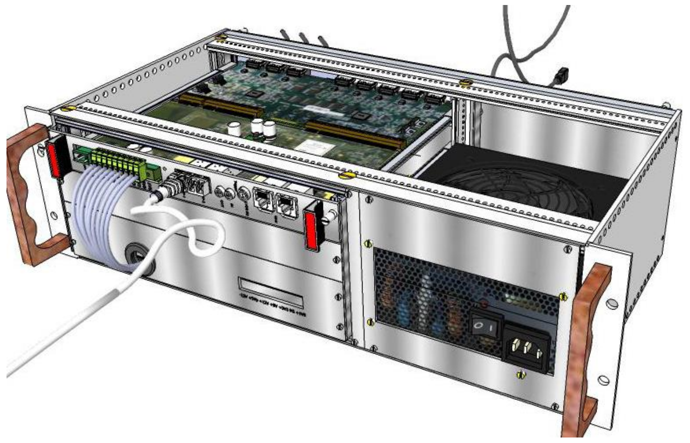
I need any given 1452 byte long network packet to **only contain the data from one trigger**.

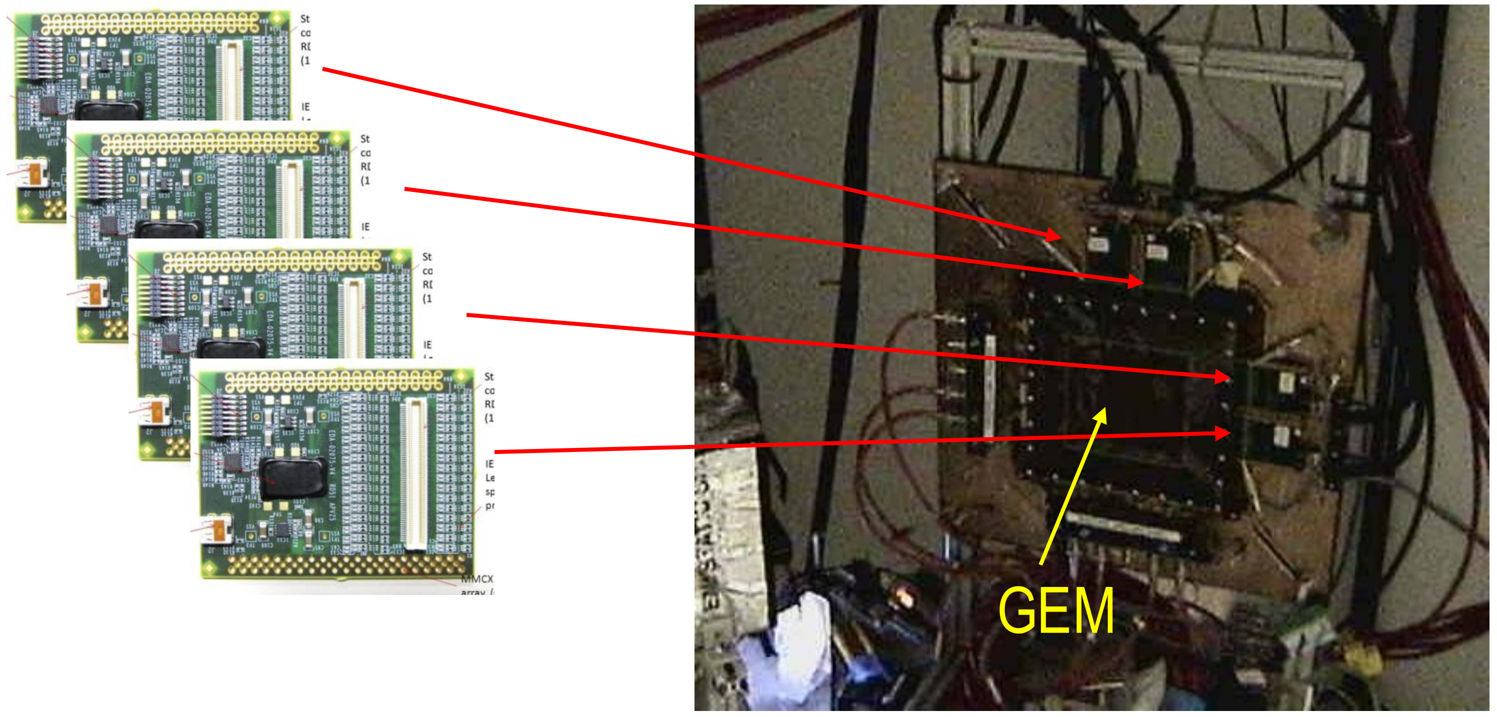
And then I need a way to recognize that “end of this trigger”

End-of-Event

I suggested to Norbert (and, by extension, to Miklos) to adopt the SRS system's protocol (CERN's "Scalable Readout System"

The SRS can do many things, we usually read out GEMs with "APV25" readout boards that mount to the detector (almost always 4 cards, 2x128 channels in x, 2x128 channels in the y direction





I showed this picture on Wednesday

Here is the RDDAQ plugin:
https://github.com/sPHENIX-Collaboration/srs

What do I get from the SRS?

For each APV25 card, the SRS sends one (jumbo) network packet per trigger/event

So with 4 APV25s I would get 4 “data” network packets

Then it sends an “End-of-Event” packet that is easily recognizable (shorter to begin with, and the payload says “EoE Marker”)

Data packet

Data packet

Data packet

Data packet

EoE packet

Data packet

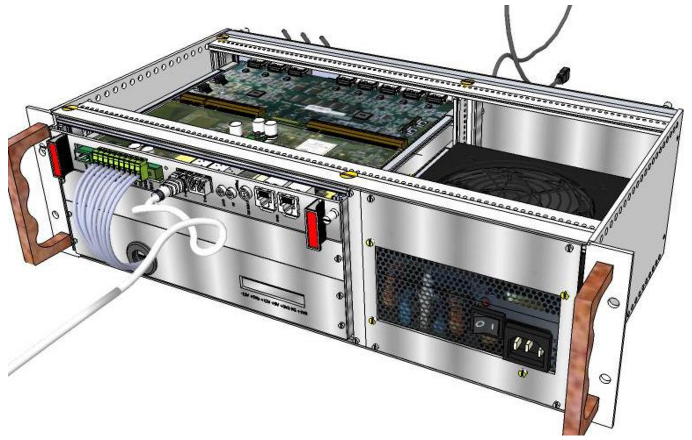
Data packet

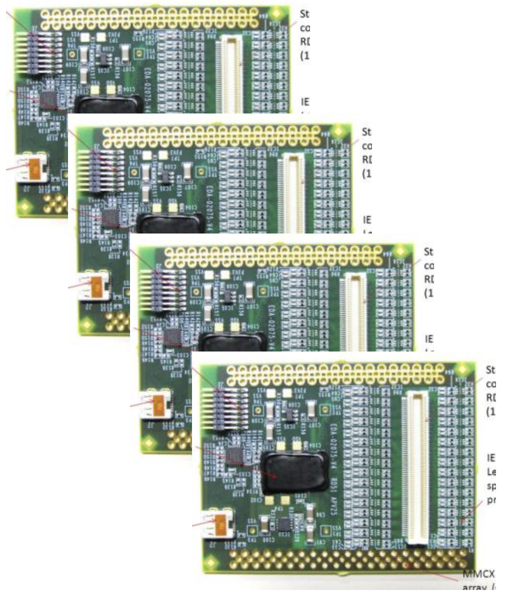
Data packet

Data packet

EoE packet

…





All data that belong to one trigger are completely contained in the network packets preceding the the EoE packet

But the real advantage is that the RCDAQ SRS plugin does not need to be aware of how the SRS is actually configured

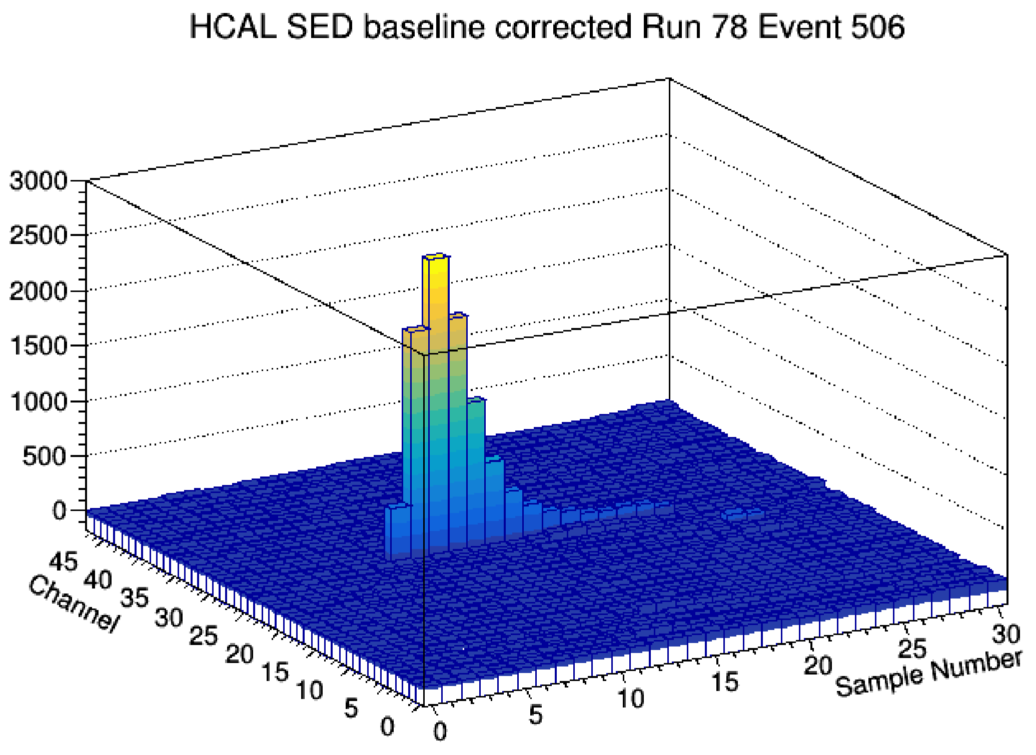
2,4,or 16 APV cards, it’s all the same – keep reading network packets until you see EoE. Super-Easy!

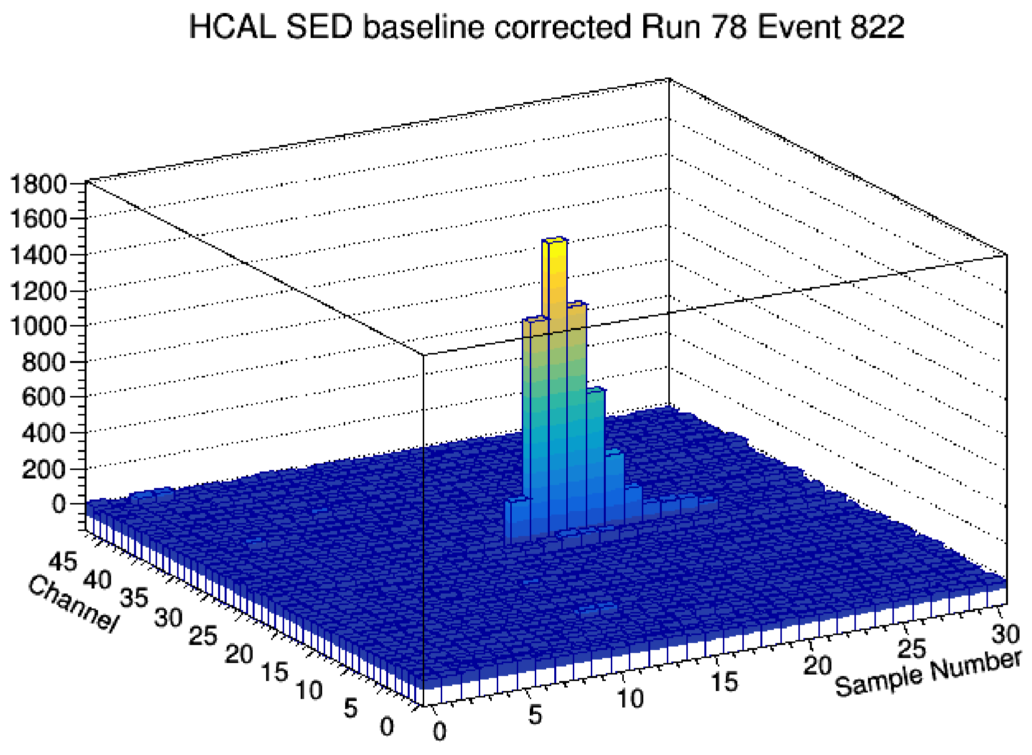
But in a super-cude way, I made “events”…

With a lot of assumptions only valid this particular data file, I broke out events

Each time I have a multi-channel waveform sampler (the sPHENIX Hcal readout, a CAEN 1742, and yes, this card) the first thing I make is a “Single Event Display” where I make a 2-dim histo with time on x, channel nr on y, pulse height on z

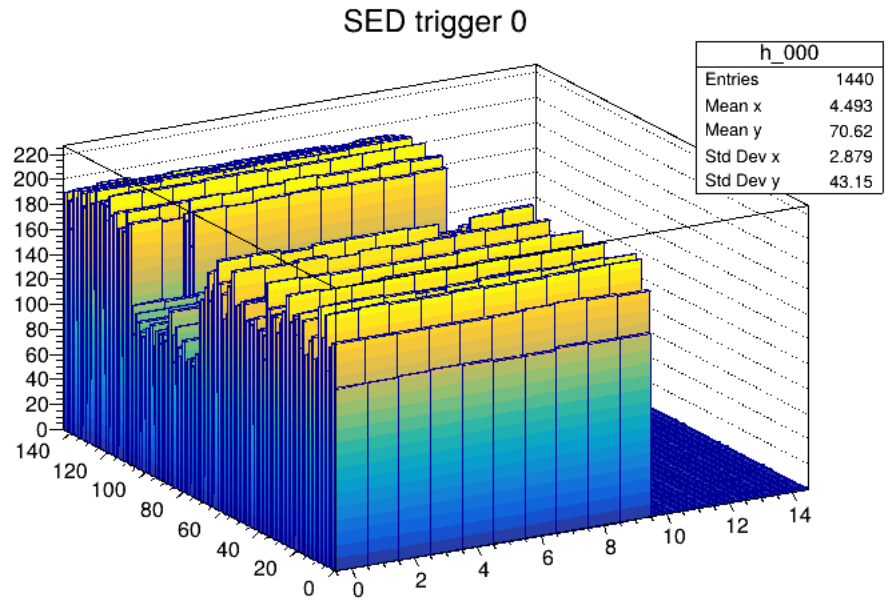
Like the ones I showed from the Hcal -

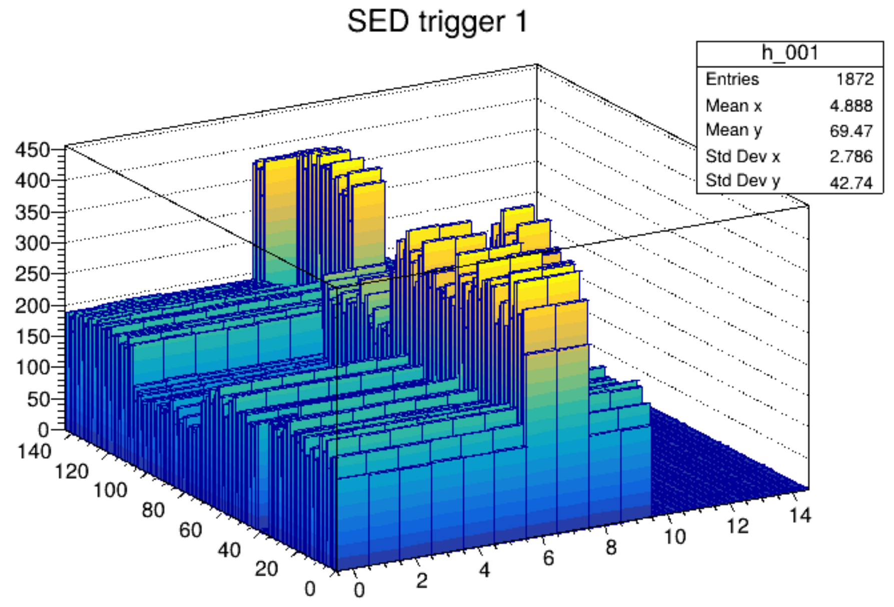


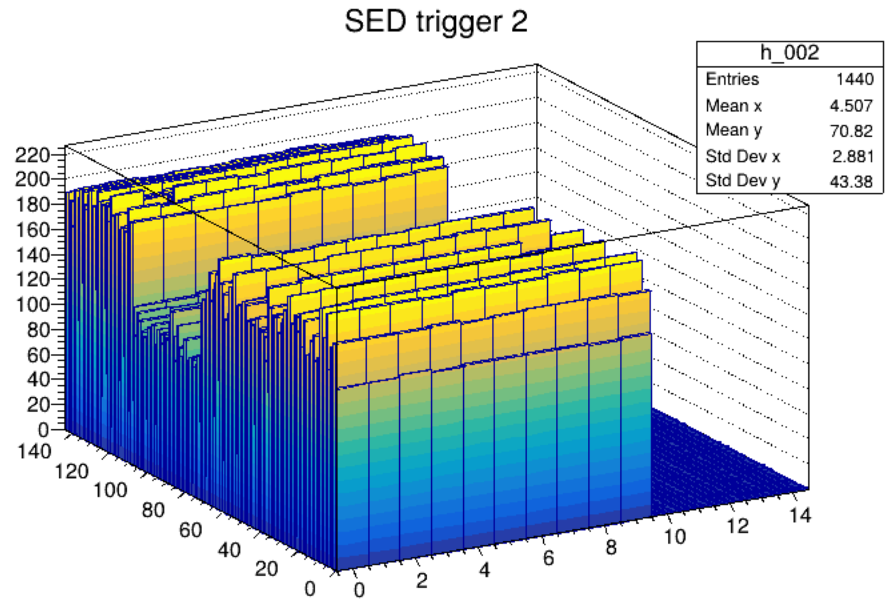


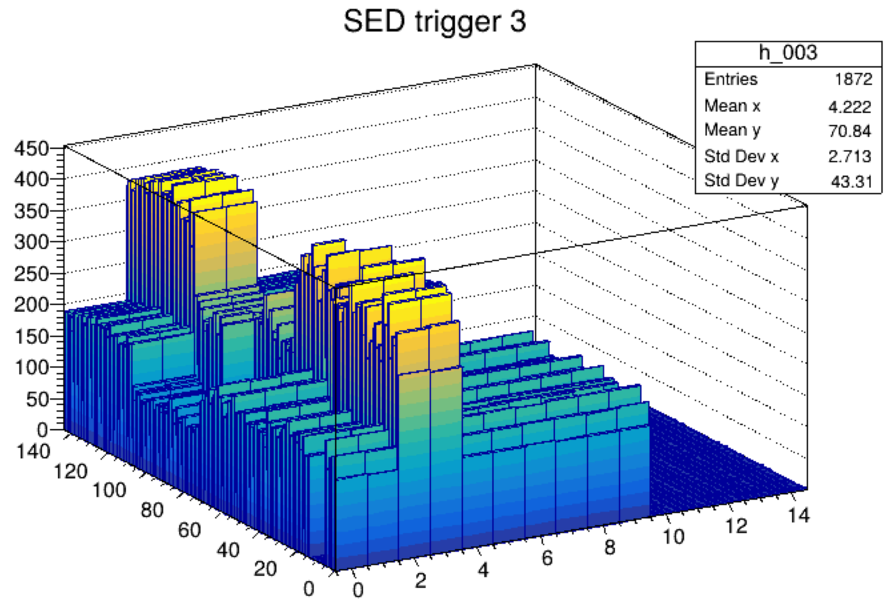
And here it is

Not too pretty, not baseline corrected, or anything, but, hey…









Summary – what I need from Norbert/Miklos

- I need network packets that only have data from one trigger
- I need an “end-of-event” marker so I know when to wrap up the event.