DAQ1000 Fee Study Notes

jml@bnl.gov 6/20/06

Just so I can rememeber…

The current sector broker code is a command line interface called 'ddl.' It runs on tpx01. I performed the tests on 6/12/06 – 6/20/06.

There are 6 fee's are currently installed in sector 18 numbered 198,199,249,251,252,253. They are powered on using Blair's FEE slow controls button labeled, "MWC sector 17&18"

Procedure for running "ddl":

```
// Check whether the board is properly configured
// 0x00000044  (factory configuration: needs config)
// 0x0AABB044  (already configured)

> get id

> set reconfig 1    // to configure if necessary

// To set and read register values
// The set functions are split by argument (press return
// on the command line to see the values)
// The get functions are not split.
// Also, the registers must be writeable or the set command
// has no effect.  For example, the following does not set
// "vpd", but does set "fpd"

// (See the altro manual for the list of register)
> set vfped 0 0
> get vfped

// Set the run type
// first argument specifies whether the trigger is emulated
// or from the TCD
// Second argument specifies that we read adc data from
// altros

> set run_type 1 1   // for emulated events
> set run_type 0 1   // for triggered events


// finally start the run

> run start -f run10.daq 10
```

Procedure for calculating pedestals:

Various reader programs are available in cvs/RTS/src/ALTRO

To calculate pedestals:

```
> altro_ex filename.daq -dump | genped ped.dat
```

To load pedestals from "ddl"

```
> write ped -f ped.dat

// Be carefull, because loading pedestals also seems to
// modify register values.  In particular, dpcfg gets set
// to 0
```

Pedestal Calculation:

Pedestals runs were emulated triggers, with the following settings:

vfped 0 0
zsthr 0 0
dpcfg 0
dpcf2 1 0 0 0

This turns off all filters, offsets, and zero suppression.

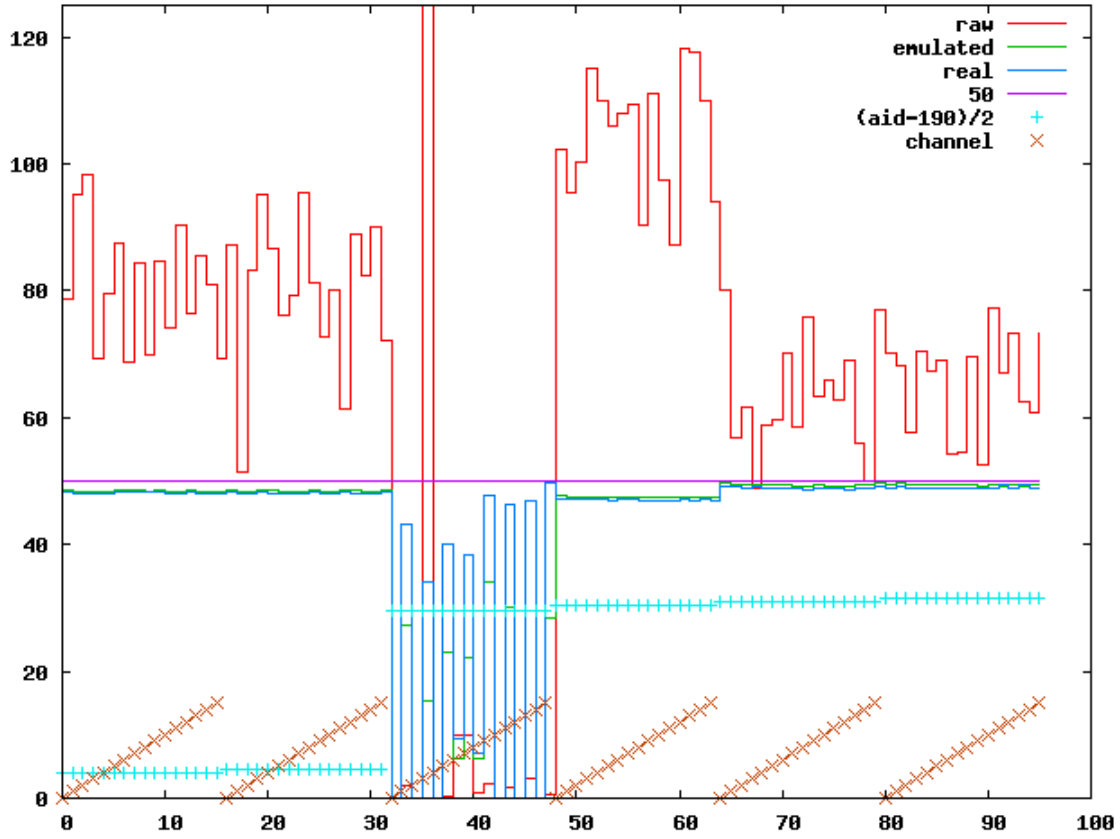
The algorithm I use to calculate pedestals by determining the median for each channel. I round the median to the nearest integer. (if x is the value for which Nabove<.5*Ntot and Nbelow<=.5*Ntot, then I use x if Nabove>Nbelow and x-1 if Nabove < Nbelow)

Physics events were taken with the following parameters:

vfped 0 0
zsthr 50 0 // apply a positive offset of 50, no zero suppression
dpcfg 1 // apply pedestal subtraction
k1, k2, k3 // 60882, 36241, 16777
l1, l2, l3 // 61931, 44236, 3644
dpcf2 1 0 0 0 // for tail filter off (or)
dpcf2 1 1 0 0 // for tail filter on

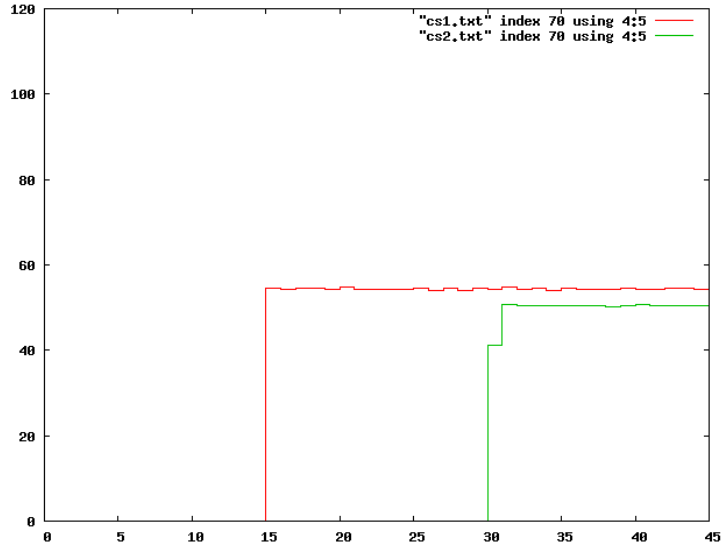
The trigger I ran for the data I will show was with a minbias trigger.

Here are the pedestals, for raw data and after subtraction:



Altro's 249 is clearly messed up, 251 also seems to have some anomaly.

The mean values should be 50. The reason they are smaller, is that the first 15 timebins of each channel are 0, even when the offset is applied:

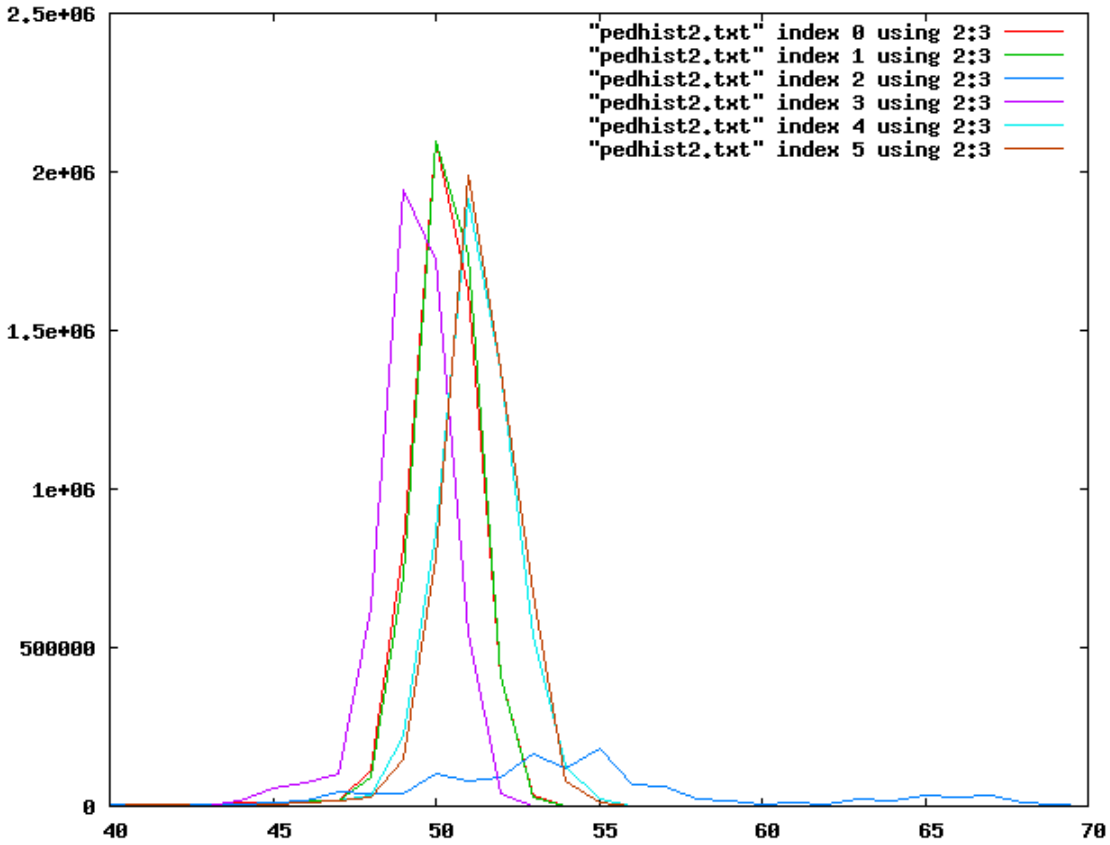


The red line is the mean adc's from a given channel and they show the 15 timebins of zeros. The green line is the same using my original pedestal subtraction algorithm. There was a fortunate bug in this algorithm, that if all the adc values were zero, the pedestal was set to 0xffff. This made the first 15 pedestal values be 0xffff.

What you can see from this is that the pedestal subtraction is actually starting from the 15th timebin. Or, in other words, the timebin the labeled in the datafile (by the FEE?) as 15 corresponds to timebin 0 in the pedestal memory.

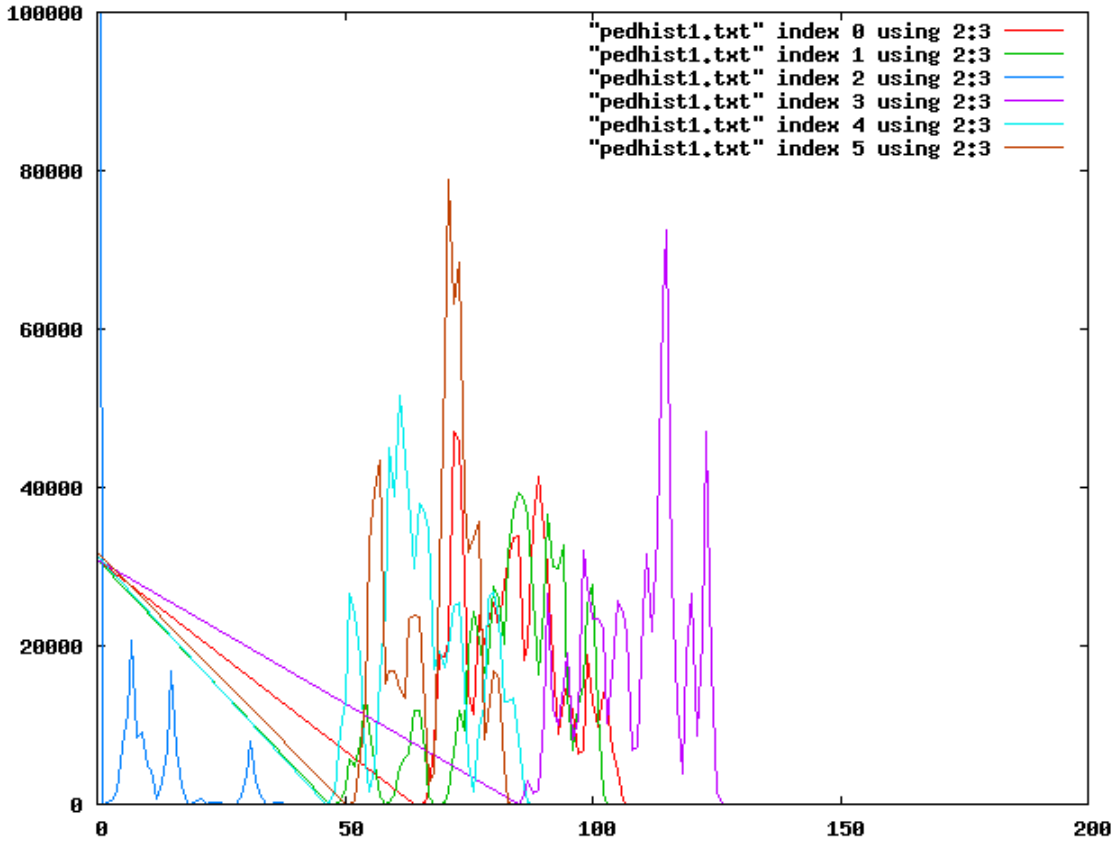
I corrected this in the pedestal calculation for all runs to be described.

Here is the histogram of ADC values for empty events, for each altro, after pedestal subtraction (zsthr.offset = 50):



With the exception of altro 249, they all have a similar shape but are shifted by 1 ADC count in 3 sets. The shifts are well correlated with the absolute values of the raw pedestal values. Altro 249 (pink) has the lowest subtracted distribution, but the highest un-subtracted mean (~110 adc's). Altro's 198 & 199 are intermediate with un-subtracted mean (~80 adc's). Altro's 252 & 253 have the highest subtracted distribution, but the lowest un-subtracted (~60 adc's).

The same code is used for the pedestal subtraction, so my only guess is that something in the un-subtracted pedestal distribution is different. However, these distributions are dominated by the channel variations and so I can't make any sense of them. Here they are just for completeness:



TPC Data Studies:

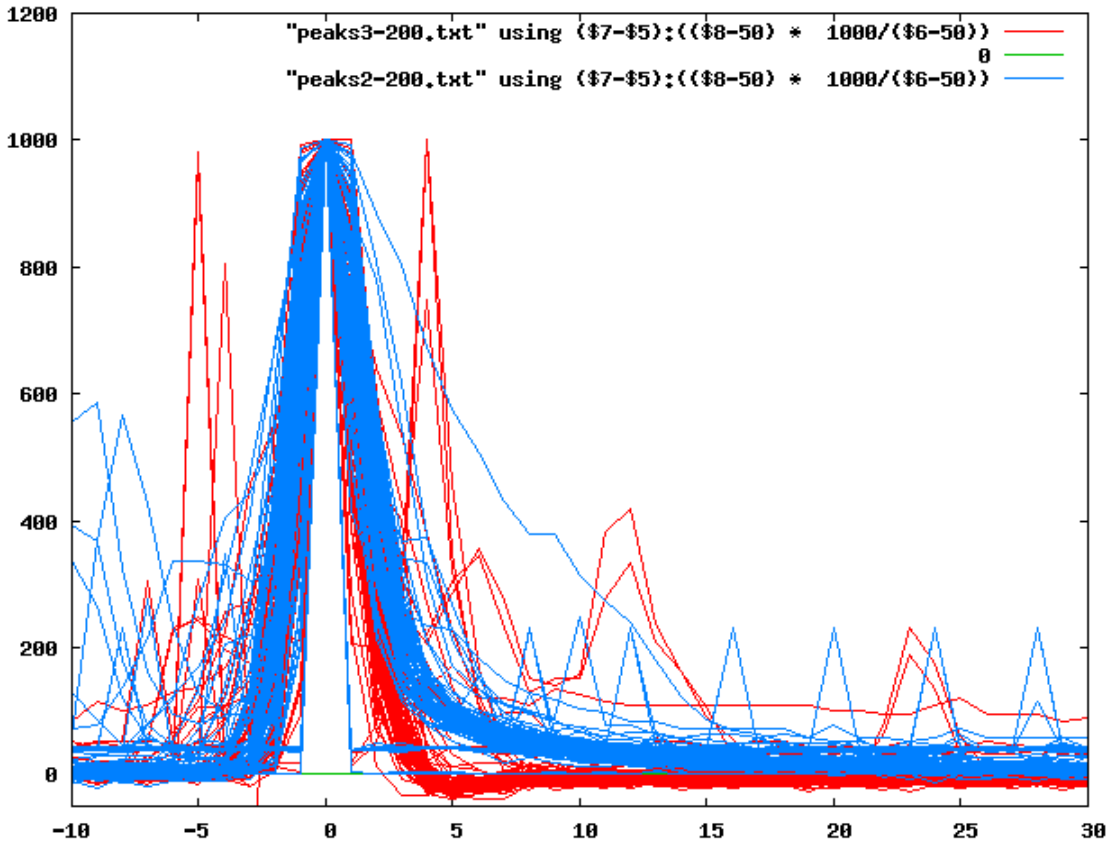
When I ran triggered data, I clearly see hits. For this data, I ran with an offset of 50, so that I could see any filter induced undershoots. The data was pedestal subtracted, but not zero-suppressed.

The peak finding algorithm, so far, is dead simple: I look for the maximum adc on a pad, and if the maximum > 200, I shift the maximum adc to be at x=0. I find that the shape of the hits is relatively independent of the size of the hit (200 < ADCmax < ~1000). The following plot shows real hits scaled to max adc=1000.

The blue lines represent raw hits, with not tail suppression filter applied. The red lines represent tail suppression using the best parameters I found under my TPC simulation last year.

The filter definitely suppresses the tails, but I do see a noticeable undershoot. I'm guessing that the reason for the undershoot is that the TPC response function I used is not perfect. I intend to take the unfiltered data, and run it back into the ALTRO simulation to recalculate another iteration of filter values – though I haven't done this yet.

I have the data to perform a more careful study of smaller hits, and also intend to do this.



To Do:

I still want to test out the zero suppression. Even though 62GeV is finishing today (and it looks like there will be no beam before the end). I can try hacking the pedestal values to fake hits. The only thing that makes sense to test is we properly understand the parameters, that they work, and that the data format is as expected.

Also, I may get a chance to run with real data if and when the 500GeV run starts.

Problems:

There are several issues I noticed when running. (I'm aware of the rushed installation, that the cabling might not be perfect, that the boards are one prototype old, and that the software is not yet finished etc…: This is for information, not a complaint…)

1. Lots of errors while running. (They cause no actual problems)

2. Lots of bad events, in the data file. (Again, the bad events are simply skipped)

3. Sometimes on boot the system seems to run, but there are channels persistently missing from the readout. Once this seemed to increase with time, until altro 251 was entirely gone from the data file. I didn't investigate the datafiles in detail to determine if the headers are there for the missing channels or if for some reason the altro's are simply suppressing the data. (I found this out when I took a pedestal run, and then cycled power, but used the same pedestals… the data wasn't pedestal subtracted for 1&1/2 altros which made me scratch my head a bit)

4. The system frequently stops when running with real triggers. After it stops, I am able to run emulated triggers, however real triggers either:

a. Do not work. (Triggers are sent to the TPC by trigger, but DAQ1000 doesn't fire)

b. Do not pay attention to the run_type. (I set the run_type as 0 1, for real data, but DAQ1000 fired despite no tokens from trigger.)

I never noted this problem when running the minbias trigger, but only when running ppProduction (which includes L2 aborts). My extremely tentative conclusion is that aborts are messing things up, although this is based on only 2-3 attempted runs.

5. Sporadically, the system goes into a mode where it doesn't listen to "get" commands. When this happens, I get the "ddlRead: timeout after 5000000 loops" message. In this mode, the gets fail frequently, however, the read will often work the second or third time I tried. I never noticed this problem on writes. (I usually check the state with a get before continuing)

6. The "read ped" command does not work. It seems to be implemented (at least nothing complains), however it always results in the "timeout after 5000000 loops" message.

7. I see bit errors in the data values. I haven't yet parametrized how frequent they are, but I clearly see them because I track the mean/rms/max values and see 1 timebin glitches of 64/256/512 adc's.