What is the matter with TriggerAlg????

Toby Burnett
A shocking discovery:

- According to triggerAlg there are almost 3 times more triggers than recorded by the GEM
  - Is there a problem with the hardware or is it a "feature" of triggerAlg

Almost factor of 3 difference in CAL_LO triggers

E. do Couto e Silva
So what happened? How does TriggerAlg work, anyway?

A little background: what is it supposed to do?

Set bits in a **trigger word** corresponding to expected trigger design

Allow actual (software) triggering of the reconstruction based on a mask.

• We have always used this to skip over incoming particles that miss the LAT.

Manage simulated livetime

What is wrong with this picture?

The “real” trigger is based on the state of digital electronics that we do not simulate

• Have to make do with readout (until recently only simulated)

Code design did not support evolution to handling real data

• Local functions with locally-defined parameters

• Management/control issue
The specific problem with CAL

Bit definitions: examine adc counts and decide:
  Low cal: any end of any log > 100 MeV
  High cal: any end of any log > 1 GeV.
Oops! How do you get from adc counts to energy?
Answer: You don’t here! The code is only valid for the old simple-minded simulation.
Solution

The natural place to simulate the CAL trigger conditions is when the energy deposits are actually turned into digitizations.

The calibration constants are all available here

What about reading in actual data (the present case)?

Plan A: just copy the GEM bits!

Plan B: Apply the same algorithm (perhaps a special algorithm) in CalDigi to examine the digis (or the reconstructed energies) and mimic the hardware behavior.
Conclusions

Thanks to Eduardo for pointing this out!
I am talking to the CAL people (currently Sasha) to implement the code to set the two trigger conditions. Then TriggerAlg will only set its trigger word.