Kaon LT Status Update

October 22nd, 2019

Richard Trotta

First fix to hcana

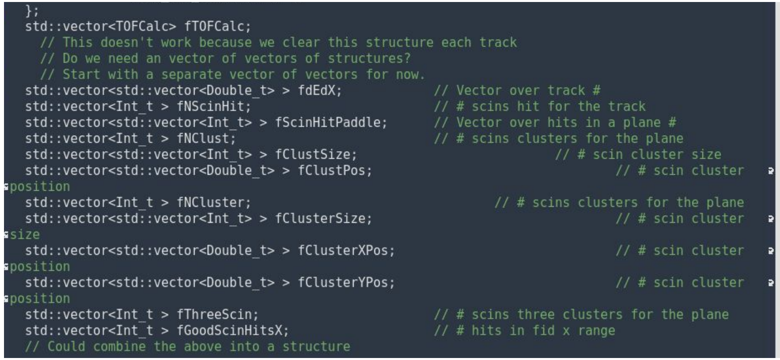
- A portion of some pulses, that are above threshold, intersect the 4-channel wide pedestal (~0.25 ns) in the PADC
  - This means all pulses in the TADC channel have a pulseAmpRaw value of null and pulseIntRaw that is too low (mostly negative)
- Existing code throws out all events
  - This creates a rate-dependent efficiency
- Peter modified this code for…
  - Scint, Aero, Cher
- He used a typical average pulse value (ideally this would be calculated on a per run basis)
- All three Cherenkov counters (HMS, NG, HG) have roughly the same typical pulse amplitude and integral so differentiation wasn't required (hence the average was used)

Second fix to hcana

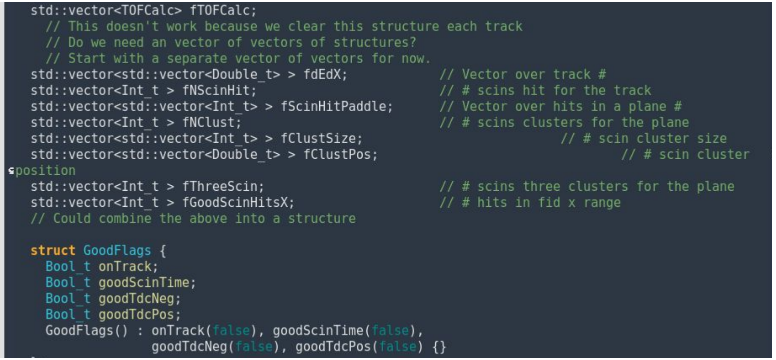
- A larger issue was that sometimes the reference time for the FADC was very far off.
- The most common problem is that it uses the time for EL-REAL which comes 130 ns after ¾
    - This was similar to my concern I brought up last meeting, an OR’d event is either ¾ or EL-REAL so for higher rates you get lots of junk if a track is found
- This issue happens ~6% of the time at high SHMS rates.
- To fix this, Peter made a routine in ScintilatorHits that finds the best value of timeoffset by means of the most TDC and ADC matches (since TDC-ADC should be a delta function)
- This is used by all the routines that use the FADC (i.e. scint, aero, cher, shower plane, and shower array)
    - Since all FADC crates use the same reference time this should be a valid solution

Outlook

- The code seems to be working as intended, but I still need to do a little debugging.
- Peter doesn’t use GitHub so I was given privileges to view c-sidis directory. He allowed me to copy his changes which can be found https://github.com/trottar/hcana/tree/peter_bosted. Make sure you’re in my branch named peter_bosted.
- The next steps after debugging would ideally be to find the pulse values on a run by run basis through the replay.



```
  };
  std::vector<TOFCalc> fTOFCalc;
    // This doesn't work because we clear this structure each track
    // Do we need an vector of vectors of structures?
    // Start with a separate vector of vectors for now.
  std::vector<std::vector<Double_t> > fdEdX;           // Vector over track #
  std::vector<Int_t > fNScinHit;                       // # scins hit for the track
  std::vector<std::vector<Int_t> > fScinHitPaddle;     // Vector over hits in a plane #
  std::vector<Int_t > fNClust;                         // # scins clusters for the plane
  std::vector<std::vector<Int_t> > fClustSize;                         // # scin cluster size
  std::vector<std::vector<Double_t> > fClustPos;                               // # scin cluster
position
  std::vector<Int_t > fNCluster;                               // # scins clusters for the plane
  std::vector<std::vector<Int_t> > fClusterSize;                               // # scin cluster
size
  std::vector<std::vector<Double_t> > fClusterXPos;                            // # scin cluster
position
  std::vector<std::vector<Double_t> > fClusterYPos;                            // # scin cluster
position
  std::vector<Int_t > fThreeScin;                      // # scins three clusters for the plane
  std::vector<Int_t > fGoodScinHitsX;                  // # hits in fid x range
  // Could combine the above into a structure
```



```
  std::vector<TOFCalc> fTOFCalc;
    // This doesn't work because we clear this structure each track
    // Do we need an vector of vectors of structures?
    // Start with a separate vector of vectors for now.
  std::vector<std::vector<Double_t> > fdEdX;           // Vector over track #
  std::vector<Int_t > fNScinHit;                       // # scins hit for the track
  std::vector<std::vector<Int_t> > fScinHitPaddle;     // Vector over hits in a plane #
  std::vector<Int_t > fNClust;                         // # scins clusters for the plane
  std::vector<std::vector<Int_t> > fClustSize;                         // # scin cluster size
  std::vector<std::vector<Double_t> > fClustPos;                               // # scin cluster
position
  std::vector<Int_t > fThreeScin;                      // # scins three clusters for the plane
  std::vector<Int_t > fGoodScinHitsX;                  // # hits in fid x range
  // Could combine the above into a structure

  struct GoodFlags {
    Bool_t onTrack;
    Bool_t goodScinTime;
    Bool_t goodTdcNeg;
    Bool_t goodTdcPos;
    GoodFlags() : onTrack(false), goodScinTime(false),
                  goodTdcNeg(false), goodTdcPos(false) {}
```