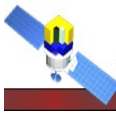
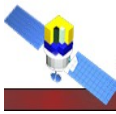
Status of Acd reconstruction



- No calibrated information, digis (raw data) only
  - Proposed to add calibrated "AcdHit" class to AcdRecon
  - Placeholders are ready, some work to access calibrations
    - Looks like AcdCalibSvc is on the horizon
- Track extrapolation to ACD is in good shape
  - Useful for detector studies
    - See talks from last week (Francesco, Stefano)
- Associating track extrapolation to hit tiles
    - The real physics use case
  - Some things exist, but want to review them

Calibrated ACD Hits



- Just want to keep track of the pulses in terms of MIPs, rather than ADC counts

```
Event::AcdHit {

  AcdId m_tileId;          // which tile was hit

  ushort m_pha[2];          // Digi level data for both PMT
  ushort m_flags[2]         // Veto, Accept bits, error flags for both PMT
  float m_mips[2];          // calibrated values for both PMT
}
```

ACD hit flags

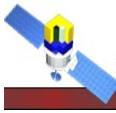
- These flags are defined so far:
  - PMT_ACCEPT // pmt is above zero suppression threshold
  - PMT_VETO // pmt fired veto discriminator
  - PMT_RANGE // pmt was read out in high range
  - PMT_ODD_PARITY_ERROR // pmt has parity error
  - PMT_HEADER_PARITY_ERROR // parity error in header
  - PMT_DEAD // pmt was dead or masked off
  - PMT_HOT // pmt was hot
- Maybe others such as:
  - PMT_IN_ROI // pmt was used in making an ROI coincidence
- Some of these require non-acd data
  - PMT_DEAD, PMT_HOT could require offline tables
    - Leave it for now

Various Distance variables

- Doca (distance of closest approach)
  - 3D distance from track to tile center
    - Always positive
- ActiveDistance (distance inside active area)
  - 2D distance from edge of tile to track intersection w/ tile plane
    - Signed: Positive is inside active area, Negative is outside
- ~~ActiveDistance3D~~
  - Same as active distance inside tile
  - 3D distance from track to tile edge or corner outside tile
    - NOTE: calculation changes as we cross tile edge
- HitRibbonDistance
  - Same as 2D active distance, w/ simplified ribbon geometry
- cornerDoca
  - 3D distance to the gaps along the corner edges of the ACD

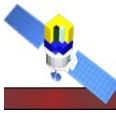
Out of fashion

Some concerns about distance variables



- 2D v 3D
  - Potential source much confusion
  - HitRibbonDistance should probably be 3D
  - ActiveDistance3D changes "meaning" inside tile (2D)
- Not all tracks are equal
  - We should be keeping track of the errors on the distances
- Not all DOCA are being stored
  - Only the best one for the top & each side row (up to 5)
  - Not distinguishing between Acd hits above or below thresholds
- The interface with the geometry is fragile
  - Assumptions that tiles are in X,Y or Z planes in 2D variables
  - The same information is retrieved several times from the geometry service.

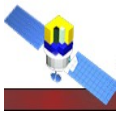
Geometry as used by Acd reconstruction



- Tiles
  - Defined as rectangular solids
  - A center & four corners. Perfectly flat, no thickness.
- Ribbons
  - Defined as three line segments
  - Top and two sides.
  - Use nominal with to decide if track hits ribbon
- Gaps
  - Defined as lines running down the sides? of the ACD

- Current test version of code encapsulates the first two of these into simple data structure that can be cached and passed around. AcdTileDim and AcdRibbonDim.

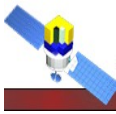
Associations Between Tracks and ACD hits



- We want to keep track of which track come close to which hit ACD element
- For each track with an arbitrary distance of a hit ACD element we can store

```
Event::AcdTkrPoca {

  AcdId m_tileId;          // which tile was hit
  int     m_trkId;         // which track did the hitting

  Point   m_location;      // 3D global postion of POCA

  TkrTrackParams m_params;   // track params at the POCA
  double m_arcLengthToISect;  // distance from last hit to POCA
  double m_arcLengthToPlane;   // distance from POCA to tile plane

  int m_region;           // where does the POCA occur
  double m_dist;          // the active distance
  double m_distErr;     // the error on the active distance
}
```

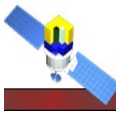
Keeping track of many associations



- Sorting the track-tile coincidences by active distance
  - Largest comes first
- Provide functions to access them in that order

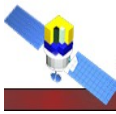
```
Event::AcdPocaMap {

   // these get only the best coincidence
   // they return null pointer if there is none
   Event::AcdTkrPoca* bestPoca(Event::TkrTrack&);
   Event::AcdTkrPoca* bestPoca(AcdId&);

   // these return all the relevent coicidences
   // of course the set could be empty
   set<Event::AcdTkrPoca*> pocas(Event::TkrTrack&);
   set<Event::AcdTkrPoca*> pocas(AcdId&);
}
```

Summary of new Data



- AcdHit
  - Hit Based calibrated data -> data structure is ready
    - Still need to do calibration code
- AcdTkrIntersection
  - Track based, extrapolation to ACD, independent of if ACD hit
    - In release, variable pulled into SVAC tuple
- AcdTkrPoca & AcdPocaMap
  - Track-Hit correlations
    - First version of code is ready
    - Might want to revist exactly what is being stored in AcdTkrPoca

Summary of work still to do



- Get data structures into releases
  - Affects Event, reconRootData, RootIo packages
- Pull some variables up into SVAC tuple
- Get Acd MIP calibration code ready
  - Pieces are all there, need to get them working together
- Test changes in distance reconstruction
  - Verify that existing variable are identical
  - Check that redundant information in new variables matches perfectly with existing variables
  - Check that other information in new variables is reasonable