BigBite analyzer for the programmer

Olivier Gayou

January 15, 2004

Chapter 1

BigBite analyzer

At present the BigBite analyzer consists of seven (7) new classes incorporated in the Hall A C++ analyzer, as shown in Fig. 1.1.

1.1 THaBigBite class

This class creates the Bigbite spectrometer, derived from the THaSpectrometer class. It declares what detectors are installed on BigBite. The current list is:

- Trigger plane (THaTriggerPlane object)
- Auxiliary plane (THaAuxPlane object)
- Neutron detector (THaNeutronDetector object)

1.2 The “intermediate classes” (set of scintillator planes)

1.2.1 THaTriggerPlane class

This class creates the Trigger Plane, and is derived from the THaNonTrackingDetector class.

Constructor The constructor for the THaTriggerPlane class declares the 2 planes of scintillators (the E and the δE planes), as THaScintPlane objects.

Setup method The Setup method defines the names and descriptions of the E and the δE planes.

Init method The Init method for this class calls the Init method of the THaScintPlane class for each plane of scintillators.

Decode method The Decode method for this class calls the Decode method of the THaScintPlane class for each plane of scintillators.

1.2.2 THaNeutronDetector class

This class, derived from the THaNonTrackingDetector class, creates the neutron detector.

Constructor The constructor for the THaNeutronDetector class declares the 4 planes of scintillators (planes 1-4), and the veto plane, as THaScintPlane objects.

Other methods The Setup, Init and Decode methods are the same as for the THaTriggerPlane class.

(Does this mean that the THaNeutronDetector and the THaTriggerPlane classes could be combined into one THaSetOfScintPlanes class ? Possible)

1.3 The scintillator plane classes

1.3.1 THaScintPlane class

It is called by both THaTriggerPlane and THaNeutronDetector. This class is derived from the THaScintillator class. It creates the individual scintillators, reads out the database, reads out the data files and defines the internal output variables (listed in the header file).

Constructor This just creates the plane of scintillators.

ReadDatabase method Obviously this method reads the database. A database needs to exist for every plane of scintillator (examples of database names are db_BB.tp.e.dat, db_BB.nd.p1.dat, ...).

Note that the first line of the database is the number of scintillators (e.g 24 for the E plane). Therefore the array dimensions for the output variables are set in this method. The individual detectors are also created here, as THaTriggerPlaneScint objects (read by 2 PMs).

SetupDetector method This method establishes the correspondance between the internal output variable (e.g fLT) and the user output variable (lt, accessible for example by the user, through BB.tp.e.lt).

DeleteArrays method This method, which is necessary in C++ for memory purposes, is called by the destructor.

ClearEvent method This method initializes the output variables before each event is decoded.

Decode method This method reads out the data file, and fills the internal output variables with the raw data (ADC and TDC values).

1.3.2 THaAuxPlane class

This class is almost identical to the THaScintPlane class, except that its individual detectors are THaAuxPlaneScint objects, read by only 1 PM. Therefore its list of output variables is twice as short, and the database is read in a slightly different format.

(Possibility to merge ?)

1.4 The subdetector classes

1.4.1 THaTriggerPlaneScint class

This class handles the individual scintillator in the trigger planes and the neutron detector planes (scintillators read by 1 PM on each side). It is responsible for all operations on the individual ADC and TDC channels: set the offsets, pedestal and gains, as read in the database (SetTDCOffsets method, for example), correct for offsets, pedestals and gains (Decode method), returns the individual raw and corrected values (GetTDCLeft and GetTDCRightCor methods, for example).

1.4.2 THaAuxPlaneScint class

This class is identical to the THaTriggerPlaneScint, except that the scintillators have only one side read (no left/right distinction).

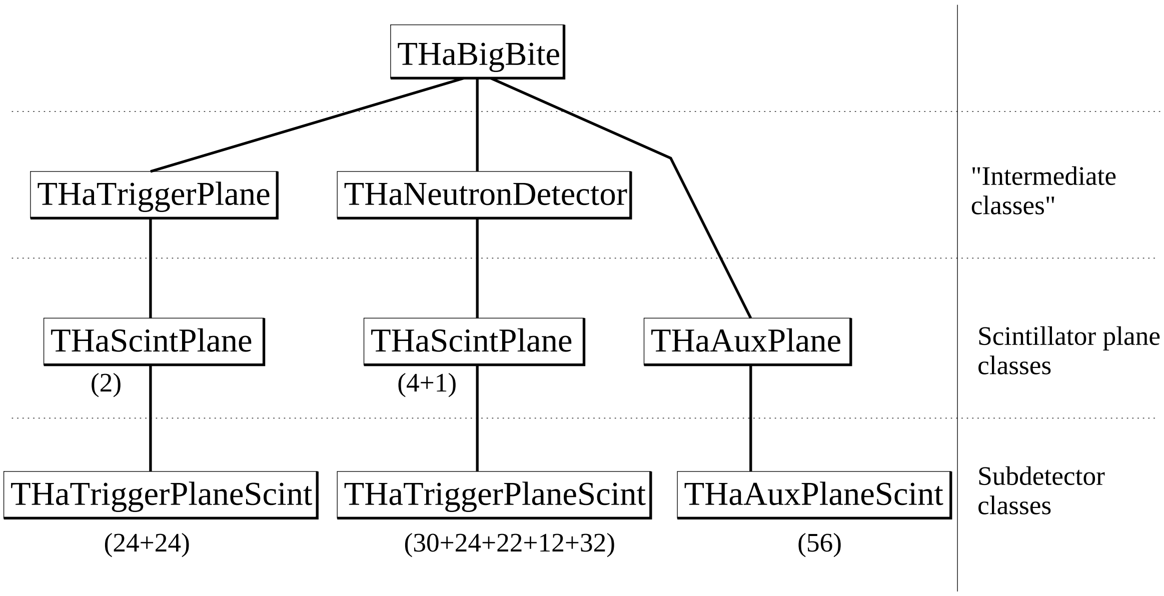

Figure 1.1: BigBite classes.

Chapter 2

To do list

2.1 BigBite inside the Hall A Analyzer

The analyzer is currently running with version 0.8 of C++ analyzer. We need to upgrade to latest version. We should also setup a CVS repository type system to be able to have several programmers working in parallel. We need to include more physical quantities.

2.2 Output

At the moment, the only available output of the analyzer is raw data: raw ADC and TDC, corrected ADC and TDC for offset, pedestal and gain, number of ADC and TDC hits, time difference between TDCs left and right.

2.2.1 Output for proton detector

"Tracking" From scintillator pattern, reconstruct dispersive angle, and momentum. We might need a way to treat multitrack events.

Time of Flight Reconstruct momentum from TOF between HRS and BigBite, and internal to BigBite (Aux-Trigger planes).

Non dispersive Get the transverse position from left-right time difference in trigger plane.

PID In the trigger plane, we can use the ADC signal from E and δE planes for PID. We can set PID flag internally, or use external cuts later in the analysis. A normalized energy deposition calculation, corrected for angle, could be usefull.

2.2.2 Output for the neutron detector

Cluster We need a cluster reconstruction algorithm.

PID We need to treat the events that fired the veto and those who didn't.

TOF We reconstruct momentum from TOF HRS-ND.

2.3 Calibration

2.4 Coincidence

We need to reconstruct all variables that require data from HRS (missing momentum,...)