PHAST

Generated by Doxygen 1.6.2

Mon Aug 27 15:36:46 2018
Contents

1  "automatically-generated" PHAST web page. ........................................ 1

2  Deprecated List ................................................................................. 3

3  Directory Hierarchy ........................................................................... 5
   3.1  Directories .................................................................................. 5

4  Class Index ......................................................................................... 7
   4.1  Class List ................................................................................... 7

5  Directory Documentation ................................................................... 9
   5.1  lib/ Directory Reference ............................................................. 9
   5.2  src/ Directory Reference ............................................................ 12

6  Class Documentation ......................................................................... 13
   6.1  PaAlgo Class Reference ............................................................. 13
       6.1.1  Detailed Description .......................................................... 15
       6.1.2  Member Function Documentation ....................................... 16
           6.1.2.1  Boost ......................................................................... 16
           6.1.2.2  Boost ......................................................................... 16
           6.1.2.3  Boost ......................................................................... 16
           6.1.2.4  CombMolN .................................................................... 17
           6.1.2.5  CombNloops ................................................................ 17
           6.1.2.6  CrossCells ................................................................... 17
           6.1.2.7  DoMW1ID .................................................................... 17
           6.1.2.8  GetBeamPol .................................................................. 18
           6.1.2.9  GetDepolarizationFactor .............................................. 18
           6.1.2.10 GetDepolarizationFactor .............................................. 18
           6.1.2.11 GetDilutionFactor ......................................................... 19
           6.1.2.12 GetDYdilutionFactor ..................................................... 19
6.1.2.13 GetDYtargetPolarization .................................................. 20
6.1.2.14 GetMW1ScatMuon .............................................................. 20
6.1.2.15 GetR .................................................................................. 20
6.1.2.16 GetRadiativeWeight ............................................................ 21
6.1.2.17 GetRadiativeWeight ............................................................ 21
6.1.2.18 GetTargetLocation .............................................................. 22
6.1.2.19 InTarget ............................................................................. 22
6.1.2.20 Q2 ...................................................................................... 23
6.1.2.21 Q2 ...................................................................................... 23
6.1.2.22 Q2 ...................................................................................... 23
6.1.2.23 RKutta ................................................................................ 24
6.1.2.24 W2 ...................................................................................... 25
6.1.2.25 W2 ...................................................................................... 25
6.1.2.26 W2 ...................................................................................... 25
6.1.2.27 xbj ...................................................................................... 26
6.1.2.28 xbj ...................................................................................... 26
6.1.2.29 xbj ...................................................................................... 26
6.1.2.30 Xf ....................................................................................... 27
6.1.2.31 Xf ....................................................................................... 27

6.2 PaCaloCellType Class Reference ................................................. 28
6.2.1 Detailed Description ............................................................... 28

6.3 PaCaloClus Class Reference ....................................................... 29
6.3.1 Detailed Description ............................................................... 30
6.3.2 Member Function Documentation .......................................... 31
6.3.2.1 iCell ............................................................................... 31
6.3.2.2 iCellMaxE ....................................................................... 31

6.4 PaCalorimCell Class Reference .................................................. 32
6.4.1 Detailed Description ............................................................... 32
6.4.2 Member Function Documentation .......................................... 33
6.4.2.1 CalibConst ................................................................. 33
6.4.2.2 GetEcort ................................................................. 33
6.4.2.3 mEdcor ................................................................. 33
6.4.2.4 mTdcor ................................................................. 33

6.5 PaCalorimeter Class Reference .................................................. 34
6.5.1 Detailed Description .............................................................. 34
6.5.2 Member Function Documentation .......................................... 34
6.5.2.1 iCell .................................................. 34
6.6 PaDetect Class Reference ........................................ 36
  6.6.1 Detailed Description ........................................ 38
  6.6.2 Member Function Documentation .............................. 38
    6.6.2.1 GetWireNumber ........................................ 38
    6.6.2.2 InActive .............................................. 39
6.7 PaDigit Class Reference ......................................... 40
  6.7.1 Detailed Description ........................................ 40
  6.7.2 Member Function Documentation .............................. 41
    6.7.2.1 IWire ................................................ 41
    6.7.2.2 sMReference .......................................... 41
    6.7.2.3 vDigInfo ............................................. 41
6.8 PaEvent Class Reference ......................................... 42
  6.8.1 Detailed Description ........................................ 45
  6.8.2 Member Function Documentation .............................. 45
    6.8.2.1 DaqDecoErrors ........................................ 45
    6.8.2.2 MasterTriggerMask .................................... 46
    6.8.2.3 MCgen ............................................... 46
    6.8.2.4 MCgen ............................................... 46
    6.8.2.5 MCgen ............................................... 46
    6.8.2.6 MCgen ............................................... 46
    6.8.2.7 MCgen ............................................... 47
    6.8.2.8 SubTriggerMask ....................................... 47
    6.8.2.9 vMiscScalers ......................................... 47
6.9 PaField Class Reference ......................................... 48
  6.9.1 Detailed Description ........................................ 48
  6.9.2 Member Function Documentation .............................. 48
    6.9.2.1 getField ............................................ 48
    6.9.2.2 getField ............................................. 49
6.10 PaHit Class Reference ......................................... 50
  6.10.1 Detailed Description ...................................... 51
  6.10.2 Member Function Documentation ............................ 51
    6.10.2.1 iDet ............................................... 51
6.11 PaMaterialMaps Class Reference ................................. 52
  6.11.1 Detailed Description ...................................... 52
  6.11.2 Member Function Documentation ............................ 53
6.21.2.3 Xlow .................................................. 79
6.22 PaRichMrr Class Reference ............................... 80
  6.22.1 Detailed Description .................................. 80
6.23 PaSetup Class Reference ................................. 81
  6.23.1 Detailed Description .................................. 82
  6.23.2 Member Function Documentation ................. 82
    6.23.2.1 Ref ........................................ 82
    6.23.2.2 TargetCenterZ ................................. 83
    6.23.2.3 TargetPolarizations ........................... 83
    6.23.2.4 vNminusOne ................................ 83
6.24 PaTPar Class Reference ................................. 84
  6.24.1 Detailed Description ................................ 86
  6.24.2 Member Function Documentation ................. 87
    6.24.2.1 AddNoise .................................. 87
    6.24.2.2 Azi ........................................ 87
    6.24.2.3 CovMatrix ................................ 87
    6.24.2.4 Dip ........................................ 87
    6.24.2.5 ExtrapolNeutral .............................. 87
    6.24.2.6 Extrapolate ................................ 87
    6.24.2.7 FindCDA .................................. 88
    6.24.2.8 HitChi2 ................................... 88
    6.24.2.9 operator() ................................ 89
    6.24.2.10 Phi ....................................... 89
    6.24.2.11 Print ..................................... 89
    6.24.2.12 Rotate .................................... 89
    6.24.2.13 Theta ..................................... 90
    6.24.2.14 Update ................................... 90
    6.24.2.15 Update ................................... 90
6.25 PaTrack Class Reference ............................... 91
  6.25.1 Detailed Description ................................ 94
  6.25.2 Member Function Documentation ................. 94
    6.25.2.1 CanBeMuon ................................ 94
    6.25.2.2 Chi2tot .................................. 94
    6.25.2.3 CrossYokeSM2 ................................ 94
    6.25.2.4 Extrapolate ................................ 95
    6.25.2.5 FieldInteg ................................ 95

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
6.25.2.6 FullKF

6.25.2.7 GetSmoothed

6.25.2.8 GetSmoothed

6.25.2.9 GetSmoothed

6.25.2.10 KeepOnly

6.25.2.11 Nd1

6.25.2.12 PassedDetectorNames

6.25.2.13 PointsHodoscopes

6.25.2.14 Q

6.25.2.15 QuickKF

6.25.2.16 vTPar

6.25.2.17 ZFirst

6.25.2.18 ZLast

6.26 PaTrigger Class Reference

6.26.1 Detailed Description

6.26.2 Member Function Documentation

6.26.2.1 BeamKiller2of3

6.26.2.2 HCal2

6.26.2.3 Instance

6.26.2.4 MultiplicityCounter

6.26.2.5 OnlineFilter

6.27 PaVertex Class Reference

6.27.1 Detailed Description

6.27.2 Member Function Documentation

6.27.2.1 Cov

6.27.2.2 iMuPrim

6.27.2.3 iMuPrimCoral

6.27.2.4 iOutMuPrim

6.27.2.5 iOutMuPrimCoral

6.27.2.6 nMuPrim

6.28 Phast Class Reference

6.28.1 Detailed Description

6.28.2 Member Function Documentation

6.28.2.1 IntegratedBeamFlux

6.28.2.2 Ref

6.28.2.3 TotalBeamFlux
Chapter 1

"automatically-generated" PHAST web page.

This documentation corresponds to PhastVersion = 7.165;
Look on "human-generated" PHAST web page for more details.
Chapter 2

Deprecated List
Member `PaAlgo::GetRadiativeWeight` (p. 21)(float xBj, float y, int flag, float &rc)

This function will exit PHAST with an error message. Please use instead the function:

static float `PaAlgo::GetRadiativeWeight` (float xBj, float y, int flag) (p. 21)
Chapter 3

Directory Hierarchy

3.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

lib ................................................................. 9
src ................................................................. 12
Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

- **PaAlgo** (Miscellaneous functions) .............................................. 13
- **PaCaloCellType** (Calorimeter cell properties) .......................... 28
- **PaCaloClus** (Calorimeter cluster) ............................................. 29
- **PaCalorimCell** (Calorimeter cell type, position, calibrations) ....... 32
- **PaCalorimeter** (Calorimeter Detector geometry) ...................... 34
- **PaDetect** (Tracking detector) .................................................. 36
- **PaDigit** (Raw information) ..................................................... 40
- **PaEvent** (COMPASS event) ..................................................... 42
- **PaField** (Magnetic fields) ..................................................... 48
- **PaHit** (Detector’s measurement) ............................................. 50
- **PaMaterialMaps** (Material Maps) ........................................... 52
- **PaMCgen** (MC generator information) ..................................... 55
- **PaMChit** (MonteCarlo hit) .................................................... 56
- **PaMChit** (MonteCarlo track) ................................................ 58
- **PaMCvertex** (MonteCarlo vertex) ........................................... 62
- **PaMetaDB** (Calibrations, coefficients etc) ................................ 64
- **PaMtx** (Scalar / Vector / Matrix) .......................................... 66
- **PaParticle** (Reconstructed particle) ....................................... 68
- **PaPid** (Functions for RICH particle identification) .................... 72
- **PaRich** (Rich Detector geometry) ........................................... 77
- **PaRichDet** (Rich photon detector element (photocathode)) ............ 78
- **PaRichMirr** (Rich mirror element) .......................................... 80
- **PaSetup** (COMPASS setup (geometry, magnetic fields etc.)) ............ 81
- **PaTPar** (Trajectory parameters) ........................................... 84
- **PaTrack** (Track) ............................................................... 91
- **PaTrigger** (Trigger and Online filter related algorithms and methods) 99
- **PaVertex** (Vertex) ............................................................ 102
- **Phast** (General purpose interface) ......................................... 106
Chapter 5

Directory Documentation

5.1 lib/ Directory Reference

Files

- file G3part.h
- file Pa3Par.cc
- file Pa3Par.h
- file PaAlgo.cc
- file PaAlgo.h
- file PaAlgoComb.cc
- file PaAlgoFindVtx.cc
- file PaAlgoGetDYdilutionFactor.cc
- file PaAlgoGetDYtargetPolarization.cc
- file PaAlgoInv5.cc
- file PaAlgoRKutta.cc
- file PaCaloCellType.cc
- file PaCaloCellType.h
- file PaCaloClus.cc
- file PaCaloClus.h
- file PaCalorimCell.cc
- file PaCalorimCell.h
- file PaCalorimeter.cc
- file PaCalorimeter.h
- file PaDetect.cc
- file PaDetect.h
- file PaDigit.cc
- file PaDigit.h
- file PaEvent.cc
- file PaEvent.h
- file PaEvent_BestPrimaryVertex.cc
- file PaEvent_Discard.cc
- file PaEvent_Prep.cc
- file PaField.cc
- file PaField.h
• file PaHit.cc
• file PaHit.h
• file PaMagInfo.cc
• file PaMagInfo.h
• file PaMaterialMaps.cc
• file PaMaterialMaps.h
• file PaMatMap.cc
• file PaMatMap.h
• file PaMCgen.cc
• file PaMCgen.h
• file PaMChit.cc
• file PaMChit.h
• file PaMCtrack.cc
• file PaMCtrack.h
• file PaMCvertex.cc
• file PaMCvertex.h
• file PaMetaDB.cc
• file PaMetaDB.h
• file PaMtx.cc
• file PaMtx.h
• file PaParticle.cc
• file PaParticle.h
• file PaPid.cc
• file PaPid.h
• file PaRich.cc
• file PaRich.h
• file PaRichDet.cc
• file PaRichDet.h
• file PaRichMrr.cc
• file PaRichMrr.h
• file PaRichPrepare.cc
• file PaSetup.cc
• file PaSetup.h
• file PaSetupPrepare.cc
• file PaTPar.cc
• file PaTPar.h
• file PaTParAddNoise.cc
• file PaTParExtrap.cc
• file PaTParExtrapolate.cc
• file PaTParFindCDA.cc
• file PaTParHitChi2.cc
• file PaTParUpdate.cc
• file PaTrack.cc
• file PaTrack.h
• file PaTrackFullKF.cc
• file PaTrackGetSmoothed.cc
• file PaTrackPointsHodoscopes.cc
• file PaTrackQuickKF.cc
• file PaTrigger.cc
• file PaTrigger.h
• file PaUtils.cc
• file PaUtils.h
• file PaUtilsSorting.cc
• file PaVertex.cc
• file PaVertex.h
• file PaVertexMuPrime.cc
• file Phast.cc
• file Phast.h
• file ProtoLinkDef.h
5.2  src/ Directory Reference

Files

- file CallUserFcn.cc
- file GetSpecialPaSetup.cc
- file main.cc
- file PaPanel.cc
- file PaPanel.h
- file UserEvent.cc
- file UserJobEnd.cc
- file UserRunEnd.cc
Chapter 6

Class Documentation

6.1 PaAlgo Class Reference

Miscellaneous functions.

#include <PaAlgo.h>

Static Public Member Functions

- static bool GetTargetLocation (int run, double &xU, double &yU, double &zU_1, double &zU_2, double &xD, double &yD, double &zD_1, double &zD_2, double &R, double &yCUT)
  
  Gives the target location in space: shift and tilting. Returns false if no information for the given year.

- static bool CrossCells (PaTPar par, int run, double R_U=-9999, double yCUT_U=-9999)

  The requirement that the muon beam trajectory crosses entirely two target cells. It is used in order to equalize fluxes through both cells.

- static bool InTarget (PaTPar par, char Cell, int run, double R_U=-9999, double yCUT_U=-9999)

  The check for the primary vertex to be in one of the target cells.

- static double GetBeamPol (float mom, int year)

  Returns the average muon beam polarization.

- static void GetDepolarizationFactor (double q2, double xBj, double y, double R, double dR, double &D, double &dD, bool do_err=true)

  Returns the depolarization factor.

- static void GetDepolarizationFactor (double q2, double xBj, double y, double &D, double &dD, bool do_err=true)

  Returns the depolarization factor.

- static void GetDilutionFactor (float xBj, float y, char Cell, int run, int flag, float &f, float &df)
Returns the dilution factor and its error.

- static void GetR (float q2, float xBj, float &R, float &dR, bool do_err=true)  
  Returns the R1990 value.

- static void GetRadiativeWeight (float xBj, float y, int flag, float &w)  
  Returns the radiative correction weight (DEPRECATED).

- static float GetRadiativeWeight (float xBj, float y, int flag)  
  Returns the radiative correction weight.

- static bool DoMW1ID (const PaTrack &t, int *nmal=0, int *nma2=0)  
  return "true" is track "t" is identified in MW1

- static int GetMW1ScatMuon (const PaEvent &e)  
  returns the index of the candidate scattered muon identified in MW1

- static double Q2 (double pmu0, double pmu, double Cos, double mnu=0.105658357)  
  \(Q^2\) via momenta and Cos.

- static double Q2 (const TVector3 &pmu0, const TVector3 &pmu, double mnu=0.105658357)  
  \(Q^2\) via 3-vectors.

- static double Q2 (const TLorentzVector &pmu0, const TLorentzVector &pmu)  
  \(Q^2\) via Lorentz vectors.

- static double xBj (double pmu0, double pmu, double Cos, double m_P=0.93827231, double mnu=0.105658357)  
  \(x_{\text{Bjorken}}\) via momenta and Cos

- static double xBj (const TVector3 &pmu0, const TVector3 &pmu, double m_P=0.93827231, double mnu=0.105658357)  
  \(x_{\text{Bjorken}}\) via 3-vectors

- static double xBj (const TLorentzVector &pmu0, const TLorentzVector &pmu, double m_P=0.93827231)  
  \(x_{\text{Bjorken}}\) via Lorentz vectors

- static double W2 (double pmu0, double pmu, double Cos, double m_P=0.93827231, double mnu=0.105658357)  
  \(W^2\) via momenta and Cos.

- static double W2 (const TVector3 &pmu0, const TVector3 &pmu, double m_P=0.93827231, double mnu=0.105658357)  
  \(W^2\) via 3-vectors.

- static double W2 (const TLorentzVector &pmu0, const TLorentzVector &pmu, double m_P=0.93827231)  
  \(W^2\) via Lorentz vectors.
• static void Boost (const TVector3 &v0, double m0, const TVector3 &v1, double m1, TVector3 &v10)  
  
  \[ \text{boost via 3-vectors} \]

• static void Boost (const TLorentzVector &v0, const TLorentzVector &v1, TVector3 &v10)  
  
  \[ \text{boost via Lorentz vectors} \]

• static void Boost (const TLorentzVector &v0, const TLorentzVector &v1, TLorentzVector &v10)  
  
  \[ \text{boost via Lorentz vectors} \]

• static double Xf (const TVector3 &pmu0, const TVector3 &pmu, const TVector3 &ph, double mh=0.13957018, double m_P=0.93827231, double mnu=0.105658357)  
  
  \[ \text{x}_F \text{ via 3-vectors} \]

• static double Xf (const TLorentzVector &pmu0, const TLorentzVector &pmu, const TLorentzVector &ph, double m_P=0.93827231)  
  
  \[ \text{x}_F \text{ via Lorentz vectors} \]

• static double E (double m, const TVector3 &p)  
  
  \[ \text{Energy via mass and 3-vector.} \]

• static double M (double e, const TVector3 &p)  
  
  \[ \text{Mass via energy and 3-vector.} \]

• static bool RK2 (double &SU, double &VO, double &Path)  
  
  \[ \text{Trajectory propagation in magnetic field by Runge-Kutta method with Jacobian calculations.} \]

• static void RedefMass (TLorentzVector &lv, double mass)  
  
  \[ \text{Change the energy part of the Lorentz vector "lv" to corresponds exactly to mass "mass".} \]

• static bool CombMoN (int N, int M, int i[])  
  
  \[ \text{"M out of N" combinator} \]

• static bool CombNLoops (int Nloops, int Niter, int ind[])  
  
  \[ \text{"N nested loops" combinator} \]

• static double GetDYTargetPolarization (const PaEvent &ev, double z)  
  
  \[ \text{return interpolated pol.} \]

• static bool GetDYdilutionFactor (const PaEvent &ev, double x2, double qT, double Q2, char cell, double &dilution, double &edilution)  
  
  \[ \text{return a boolean and the dilution factor as well as its associated error by reference} \]

### 6.1.1 Detailed Description

6.1.2 Member Function Documentation

6.1.2.1 void PaAlgo::Boost (const TLorentzVector & v0, const TLorentzVector & v1, TLorentzVector & v10) [inline, static]

boost via Lorentz vectors The Lorentz transformation of the particle \( v1 \) to the rest frame of the particle \( v0 \).

Parameters:

\( v0 \) the four-vector of particle 0 in laboratory frame
\( v1 \) the four-vector of particle 1 in laboratory frame
\( v10 \) the four-vector of the particle 1 in the rest frame of 0

Author:

Alexandre.Korzenev@cern.ch

References Boost().

6.1.2.2 void PaAlgo::Boost (const TLorentzVector & v0, const TLorentzVector & v1, TVector3 & v10) [inline, static]

boost via Lorentz vectors The Lorentz transformation of the particle \( v1 \) to the rest frame of the particle \( v0 \).

Parameters:

\( v0 \) the four-vector of particle 0 in laboratory frame
\( v1 \) the four-vector of particle 1 in laboratory frame
\( v10 \) the momentum vector of the particle 1 in the rest frame of 0

Author:

Alexandre.Korzenev@cern.ch

References Boost().

6.1.2.3 void PaAlgo::Boost (const TVector3 & v0, double m0, const TVector3 & v1, double m1, const TVector3 & v10) [inline, static]

boost via 3-vectors The Lorentz transformation of the particle \( \{v1, m1\} \) to the rest frame of the particle \( \{v0, m0\} \).

Parameters:

\( v0 \) the momentum vector of particle 0 in laboratory frame
\( m0 \) the mass of the particle 0
\( v1 \) the momentum vector of particle 1 in laboratory frame
\( m1 \) the mass of the particle 1
\( v10 \) the momentum vector of the particle 1 in the rest frame of 0
6.1 PaAlgo Class Reference

Author:

Alexandre.Korzenev@cern.ch

References E().

Referenced by Boost().

6.1.2.4 bool PaAlgo::CombMofN (int N, int M, int i[]) [static]

"M out of N" combinator If to call in the loop (while it returns "true"), on every call this function
put in array "i[M]" indexes [0-N] for "M out of N" combinations. Could be used in combination
of tracks for vertex candidates, in combination of tracks or calorimeter clusters for invariant mass
calculation etc.

6.1.2.5 bool PaAlgo::CombNloops (int Nloops, int Niter, int ind[]) [static]

"N nested loops" combinator If to call in the loop (while it returns "true"), on every call this
function put in array "ind[Nloops] running indexes of N nested loops where every loop do Niter
cycles.

6.1.2.6 bool PaAlgo::CrossCells (PaTPar par, int run, double R_U = -9999,
double yCUT_U = -9999) [static]

The requirement that the muon beam trajectory crosses entirely two target cells. It is used in
order to equalize fluxes through both cells.

Parameters:

par the beam track parameters in the primary vertex
run the run number ("-2" = 2 cell MC, "-3" = 3 cell MC LiD, "+4" = 3 cell MC NH3)
R_U the user defined radial cut (R<1.4cm)
yCUT_U the user defined vertical cut (y<1cm)

Author:

Alexandre.Korzenev@cern.ch

References PaTPar::Extrapolate(), and GetTargetLocation().

6.1.2.7 bool PaAlgo::DoMW1ID (const PaTrack & t, int * nma1 = 0, int * nma2
= 0) [static]

return "true" is track "t" is identified in MW1. The number of clusters associated to the track
before and after the absorber is stored in nma1 and nma2 respectively if they are not null pointers.

Author:

Andrea Ferrero <aferrero@to.infn.it>

References PaTrack::NHitsFoundInDetect().

Referenced by GetMW1ScatMuon().
6.1.2.8  double PaAlgo::GetBeamPol (float mom, int year)  [static]

Returns the average muon beam polarization.

Parameters:
   
   `mom`  the momentum of the beam muon track  
   `year`  the year of data taking (2002, 2003 or 2004).

6.1.2.9  void PaAlgo::GetDepolarizationFactor (double q2, double xBj, double y,  
    double & D, double & dD, bool do_err = true)  [static]

Returns the depolarization factor.

Parameters:
   
   `q2`  squared invariant mass of the virtual photon  
   `xBj`  Bjorken x  
   `y`  energy fraction carried by the virtual photon  
    `D` resulting depolarization factor  
   `dD` calculated error on D (if do_err==true)  
    `do_err`  flag which controls whether the error calculation should be done (defaults to 'yes')

This function is not available when Phast (p. 106) is compiled with NO_FORTRAN-1

This function is a convenience wrapper for the first variant which takes the value of R±dR from  
PaAlgo::GetR() (p. 20).

Author: 

Roland.Kuhn@cern.ch

References GetDepolarizationFactor(), and GetR().

6.1.2.10 void PaAlgo::GetDepolarizationFactor (double q2, double xBj, double y,  
    double R, double dR, double & D, double & dD, bool do_err = true)  
    [static]

Returns the depolarization factor.

Parameters:
   
   `q2`  squared invariant mass of the virtual photon  
   `xBj`  Bjorken x  
   `y`  energy fraction carried by the virtual photon  
    `R`  R value (e.g. from r1990.F')  
   `dR`  error on R  
    `D` resulting depolarization factor  
   `dD` calculated error on D (if do_err==true)  
    `do_err`  flag which controls whether the error calculation should be done (defaults to 'yes')
6.1 PaAlgo Class Reference

The code in this function is taken from the A1 analysis code (thanks to A. Korzenev). This function does not invoke any external FORTRAN code, which means that if you have your own source for the R values you can use it even when compiling Phast (p. 106) with NO_FORTRAN-1.

Author:

Roland.Kuhn@cern.ch

References M().

References by GetDepolarizationFactor().

6.1.2.11 void PaAlgo::GetDilutionFactor (float \( xBj \), float \( y \), char \( \text{Cell} \), int \( \text{run} \),
int \( \text{flag} \), float & \( f \), float & \( df \) ) [static]

Returns the dilution factor and its error.

Parameters:

\( xBj \) Bjorken x

\( y \) energy fraction carried by the virtual photon

\( \text{Cell} \) target cell (U=upstream, C=central, D=downstream)

\( \text{run} \) run number.

\( \text{flag} \) processing flag (1=inclusive, 2=hadron_tagged)

\( f \) dilution factor

\( df \) error for the dilution factor

This function is not available when Phast (p. 106) is compiled with NO_FORTRAN-1

This function calls the FORTRAN routine dfdeut from ./fortran/dilut_main.F to get the dilution factor for the various regions of the target.

Beware: The FORTRAN routine gives only float values!

Author:

Alexandre.Korzenev@cern.ch

6.1.2.12 bool PaAlgo::GetDYdilutionFactor (const PaEvent & \( ev \), double \( x2 \),
double \( qT \), double \( Q2 \), char \( \text{cell} \), double & \( \text{dilution} \), double & \( \text{edilution} \) ) [static]

return a boolean and the dilution factor as well as its associated error by reference

Parameters:

\( x2 \) Proton Bjorken x

\( qT \) Transverse momentum of the virtual photon

\( Q2 \) Di-muon invariant mass Squared

\( \text{cell} \) Target cell: \'U', \'D' for upstream, downstream respectively

\( \text{dilution} \) Dilution factor

\( \text{edilution} \) Error on the dilution factor

Author:

vincent.andrieux@cern.ch

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
6.1.2.13 static double PaAlgo::GetDYtargetPolarization (const PaEvent & ev, 
    double z) [static]

return interpolated pol.

Parameters:
    ev PaEvent (p. 42) object
    z z coordinate where you want to calculate pol. at

only pol. in 2015 is available

Author:
    Nukazuka, Genki [genki@quark.kj.yamagata-u.ac.jp]

6.1.2.14 int PaAlgo::GetMW1ScatMuon (const PaEvent & e) [static]

returns the index of the candidate scattered muon identified in MW1 Returns the best scattered muon candidate as identified by MW1. The track is selected among those coming out of the "best primary vertex", as given by iBestPrimaryVertex() function. The identification is performed only in the case of calorimetric trigger events.

Author:
    Andrea Ferrero <aferrero@to.infn.it>

References DoMW1ID(), PaEvent::iBestPrimaryVertex(), PaVertex::iOutParticle(), PaParticle::iTrack(), PaVertex::NOutParticles(), PaEvent::TrigMask(), PaEvent::vParticle(), PaEvent::vTrack(), and PaEvent::vVertex().

6.1.2.15 void PaAlgo::GetR (float q2, float xBj, float & R, float & dR, bool 
    do_err = true) [static]

Returns the R1990 value.

Parameters:
    q2 squared invariant mass of the virtual photon
    xBj Bjorken x
    R result (float!)
    dR error on R (if do_err = true)
    do_err flag which controls whether the error calculation should be done (defaults to 'yes')

This function is not available when Phast (p. 106) is compiled with NO_FORTAN=1
This function calls the FORTRAN routing r1990 from ./fortran/r1990.F to get the R parameterization.

Beware: The FORTRAN routine gives only float values!

Author:
    Roland Kuhn <roland.kuhn@cern.ch>

Referenced by GetDepolarizationFactor().
6.1.2.16  float PaAlgo::GetRadiativeWeight (float \( xB_j \), float \( y \), int \( \text{flag} \)) [static]

Returns the radiative correction weight.

Parameters:

- \( xB_j \) Bjorken x
- \( y \) energy fraction carried by the virtual photon
- \( \text{flag} \) processing flag (1=inclusive(deuter), \( 2= \text{hadron} \_\text{tagged}(\text{deuter}) \), \( 11= \text{inclusive}(\text{hydrogen}) \), \( 12= \text{hadron}\_\text{tagged}(\text{hydrogen}) \))

Returns:

The radiative correction factor defined as:

\[
\eta(x, y) = \frac{\sigma_{1\gamma}}{\sigma_{\text{measured}}}
\]

This factor can be both smaller and larger than 1. For the reference in case of related questions, please look

---


---

This function is not available when Phast (p. 106) is compiled with NO\_FORTRAN-1

This function calls the FORTRAN routine reclin or rechld from ./fortran/dfdeut.F to get the radiative correction weights from precalculated tables. Tables that are used were generated for dilution factor calculations.

Beware: The FORTRAN routine gives only float values!

Author:

Konrad.Klimaszewski@cern.ch

References Phast::Ref(), and PaEvent::Year().

6.1.2.17  void PaAlgo::GetRadiativeWeight (float \( xB_j \), float \( y \), int \( \text{flag} \), float & \( \text{rc} \)) [static]

Returns the radiative correction weight (DEPRECATED).

Deprecation

This function will exit PHAST with an error message. Please use instead the function: static float PaAlgo::GetRadiativeWeight(float \( xB_j \), float \( y \), int \( \text{flag} \)) (p. 21)

Parameters:

- \( xB_j \) Bjorken x
- \( y \) energy fraction carried by the virtual photon
- \( \text{flag} \) processing flag (1=inclusive(deuter), \( 2= \text{hadron} \_\text{tagged}(\text{deuter}) \), \( 11= \text{inclusive}(\text{hydrogen}) \), \( 12= \text{hadron}\_\text{tagged}(\text{hydrogen}) \))
rc resulting radiative correction weight it is defined as:

\[ rc = \frac{1}{\eta(x, y)} = \frac{\sigma_{\text{measured}}}{\sigma_{1\gamma}} \]

This function is not available when Phast (p. 106) is compiled with NO\_FORTRAN-1

This function calls the FORTRAN routine rcinlx or rcinld from ./fortran/dfdeut.F to get the radiative correction weights from precalculated tables. Tables that are used were generated for dilution factor calculations.

**Beware:** The FORTRAN routine gives only float values!

**Author:**

Konrad.Klimaszewski@cern.ch

6.1.2.18 `bool PaAlgo::GetTargetLocation (int run, double & xU, double & yU, double & zU_1, double & zU_2, double & xD, double & yD, double & zD_1, double & zD_2, double & R, double & yCUT) [static]`

Gives the target location in space: shift and tilting. Returns false if no information for the given year.

**Parameters:**

- `run` the run number ("-2" = 2 cell MC)
- `xU` the shift of the target most upstream part along x
- `yU` the shift of the target most upstream part along y
- `zU_1` z position of the upstream cell most upstream part
- `zU_2` z position of the upstream cell most downstream part
- `xD` the shift of the target most downstream part along x
- `yD` the shift of the target most downstream part along y
- `zD_1` z position of the downstream cell most upstream part
- `zD_2` z position of the downstream cell most downstream part
- `R` the radial cut
- `yCUT` the cut for upper part of the cell

**Author:**

Alexandre.Korzenev@cern.ch

Referenced by CrossCells(), and InTarget().

6.1.2.19 `bool PaAlgo::InTarget (PaTPar par, char Cell, int run, double R_ U = -9999, double yCUT_ U = -9999) [static]`

The check for the primary vertex to be in one of the target cells.

**Parameters:**

- `par` The beam track parameters in the primary vertex
6.1 PaAlgo Class Reference

**Cell** The one of cells (if 'U' - upstream cell, if 'D' - downstream, if 'C' - central)

**run** The run number ("-2" = 2 cell MC, "-3" = 3 cell MC LiD, ",4" = 3 cell MC NH3)

**R U** the user defined radial cut (R<1.4cm)

**yCUT U** the user defined vertical cut (y<1cm)

Author:
Alexandre.Korzenev@cern.ch

References GetTargetLocation().

6.1.2.20 **double PaAlgo::Q2 (const TLorentzVector & pmu0, const TLorentzVector & pmu) [inline, static]**

Q^2 via Lorentz vectors.

Parameters:

- **pmu0** the beam Lorentz vector
- **pmu** the scattered lepton Lorentz vector

Author:
Alexandre.Korzenev@cern.ch

References Q2().

6.1.2.21 **double PaAlgo::Q2 (const TVector3 & pmu0, const TVector3 & pmu, double mnu = 0.105658357) [inline, static]**

Q^2 via 3-vectors.

Parameters:

- **pmu0** the beam vector
- **pmu** the scattered lepton vector
- **mnu** the mass of the lepton (default is muon)

Author:
Alexandre.Korzenev@cern.ch

References E().

6.1.2.22 **double PaAlgo::Q2 (double pmu0, double pmu, double Cos, double mnu = 0.105658357) [inline, static]**

Q^2 via momenta and Cos.

Parameters:

- **pmu0** the absolute value of the beam momentum
**pmu** the absolute value of the scattered lepton momentum

**Cos** the cos of angle between beam and scattered lepton

**mmu** the mass of the lepton (default is muon)

**Author:**

Alexandre.Korzennev@cern.ch

References E().

Referenced by Q2(), W2(), and xbj().

### 6.1.2.23 bool PaAlgo::RKutta (double * SU, double * VO, double & Path) [static]

Trajectory propagation in magnetic field by Runge-Kutta method with Jacobian calculations.

```cpp
// Runge-Kutta method for tracking particles through a magnetic field.
// Uses Nystroem algorithm (See Handbook Net. Bur. of Standards, procedure 25.5.20)

// Input parameters:
//  SU - plane parameters
//  SU[0] - direction cosines normal to surface Ex
//  SU[1] - Ey
//  SU[2] - Ex, Ex*Ex+Ey+Ez+Ez=1
//  SU[3] - distance to surface from (0,0,0) > 0 cm
//  VO - initial parameters (coordinates(cm), direction cosines,
//       charge/momentum (GeV-1) and derivatives this parameters
//       Az*Az+Ay*Ay+Az*Az-1
//  dX/dp  dY/dp  dZ/dp  dAX/dp  dAY/dp  dAZ/dp  d(pq/p)/dp*VO[6]
//  dX/dp  dY/dp  dZ/dp  dAX/dp  dAY/dp  dAZ/dp  d(pq/p)/dp*VO[6]

// Output parameters:
//  VO - output parameters and derivatives after propagation in magnetic field
//        defined by Mfield (K Gauss)
//        Where a Mfield(R,H) - is interface to magnetic field information
//        input R[0],R[1],R[2] - X, Y and Z of the track
//        output H[0],H[1],H[2] - Hx, Hy and Hz of the magnetic field
//        H[3],H[4],H[5] - dHx/dx, dHy/dy and dHz/dz
//        H[6],H[7],H[8] - dHx/dy, dHy/dy and dHz/dz
//        H[ 9],H[10],H[11] - dHx/dz, dHy/dz and dHz/dz
//  Path - trajectory length

// Author: R.Brun, M.Hamon, V.Pereshvatzhikov (Geant3)
// had been done by Igor.Gavrilenko@cern.ch
// Major modification by Sergei.Gerassimov@cern.ch for use in Traffic and Phast
// Here is assumed, that field is smooth enough i.e. field gradients are negligible
```

References PaSetup::Ref().
6.1.2.24  double PaAlgo::W2 (const TLorentzVector & pmu0, const TLorentzVector & pmu, double m_ P = 0.93827231) [inline, static]

W^2 via Lorentz vectors.

Parameters:

    * pmu0 the beam Lorentz vector
    * pmu the scattered particle Lorentz vector
    * m_ P the mass of the target particle (default is proton)

Author:

    Alexandre.Korzenev@cern.ch

References W2().

6.1.2.25  double PaAlgo::W2 (const TVector3 & pmu0, const TVector3 & pmu, double m_ P = 0.93827231, double mnu = 0.105658357) [inline, static]

W^2 via 3-vectors.

Parameters:

    * pmu0 the beam vector
    * pmu the scattered particle vector
    * m_ P the mass of the target particle (default is proton)
    * mnu the mass of the scattering particle (default is muon)

Author:

    Alexandre.Korzenev@cern.ch

References E(), and Q2().

6.1.2.26  double PaAlgo::W2 (double pmu0, double pmu, double Cos, double m_ P = 0.93827231, double mnu = 0.105658357) [inline, static]

W^2 via momenta and Cos.

Parameters:

    * pmu0 the absolute value of the beam momentum
    * pmu the absolute value of the scattered lepton momentum
    * Cos the cos of angle between beam and scattered lepton
    * m_ P the mass of the target particle (default is proton)
    * mnu the mass of the scattering particle (default is muon)

Author:

    Alexandre.Korzenev@cern.ch

References E(), and Q2().

Referenced by W2().
6.1.2.27  double PaAlgo::xbj (const TLorentzVector & pmu0, const TLorentzVector & pmu, double m_\_P = 0.93827231) [inline, static]

\text{x}Bjorken via Lorentz vectors

Parameters:

- \textit{pmu0} the beam Lorentz vector
- \textit{pmu} the scattered particle Lorentz vector
- \textit{m_\_P} the mass of the target particle (default is proton)

Author:

Alexandre.Korzenew@cern.ch

References xbj().

6.1.2.28  double PaAlgo::xbj (const TVector3 & pmu0, const TVector3 & pmu, double m_\_P = 0.93827231, double mmu = 0.105658357) [inline, static]

\text{x}Bjorken via 3-vectors

Parameters:

- \textit{pmu0} the beam vector
- \textit{pmu} the scattered particle vector
- \textit{m_\_P} the mass of the target particle (default is proton)
- \textit{mmu} the mass of the scattering particle (default is muon)

Author:

Alexandre.Korzenew@cern.ch

References E(), and Q2().

6.1.2.29  double PaAlgo::xbj (double pmu0, double pmu, double Cos, double m_\_P = 0.93827231, double mmu = 0.105658357) [inline, static]

\text{x}Bjorken via momenta and Cos

Parameters:

- \textit{pmu0} the absolute value of the beam momentum
- \textit{pmu} the absolute value of the scattered lepton momentum
- \textit{Cos} the \textit{cos} of angle between beam and scattered lepton
- \textit{m_\_P} the mass of the target particle (default is proton)
- \textit{mmu} the mass of the scattering particle (default is muon)

Author:

Alexandre.Korzenew@cern.ch

References E(), and Q2().

Referenced by xbj().
6.1.2.30  double PaAlgo::Xf (const TLorentzVector & pmu0, const TLorentzVector & pmu, const TLorentzVector & ph, double m_P = 0.93827231)  
[inline, static]

xf via Lorentz vectors

Parameters:

- **pmu0** the beam four-vector
- **pmu** the scattered lepton four-vector
- **ph** the hadron four-vector
- **m_P** the target particle mass (default is proton)

Author:

Alexandre.Korzenev@cern.ch

References Xf().

6.1.2.31  double PaAlgo::Xf (const TVector3 & pmu0, const TVector3 & pmu, const TVector3 & ph, double mh = 0.13957018, double m_P = 0.93827231, double mmu = 0.105658357)  
[inline, static]

xf via 3-vectors

Parameters:

- **pmu0** the beam vector
- **pmu** the scattered lepton vector
- **ph** the hadron vector
- **mh** the hadron mass
- **m_P** the target particle mass (default is proton)
- **mmu** the lepton mass

Author:

Alexandre.Korzenev@cern.ch

References E(), and M().

Referenced by Xf().

The documentation for this class was generated from the following files:

- lib/PaAlgo.h
- lib/PaAlgo.cc
- lib/PaAlgoComb.cc
- lib/PaAlgoFindVtx.cc
- lib/PaAlgoGetDYdilutionFactor.cc
- lib/PaAlgoInv5.cc
- lib/PaAlgoRKutta.cc

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
6.2 PaCaloCellType Class Reference

Calorimeter cell properties.

`#include <PaCaloCellType.h>`

**Public Member Functions**

- `string Name () const`
  
  *Cell type name.*

- `const TVector3 & Size () const`
  
  *Size.*

- `float RadiationLength () const`
  
  *Radiation length.*

- `float NuclearLength () const`
  
  *Nuclear length.*

- `float ConstantTerm () const`
  
  *Constant term.*

- `float StochasticTerm () const`
  
  *Stochastic term.*

- `float ReadoutTerm () const`
  
  *Readout term.*

- `float ActiveMaterial () const`
  
  *Active material.*

### 6.2.1 Detailed Description

Calorimeter cell properties.

The documentation for this class was generated from the following files:

- `lib/PaCaloCellType.h`
- `lib/PaCaloCellType.cc`
6.3 PaCaloClus Class Reference

Calorimeter cluster.

\#include <PaCaloClus.h>

**Public Member Functions**

- const string & **CalorimName** () const  
  *Calorimeter name.*
- const int & **iCalorim** () const  
  *Calorimeter number (index in PaSetup::vecCalorim).*
- const float & **E** () const  
  *Cluster’s energy.*
- const float & **Err** () const  
  *Cluster’s energy error.*
- const float & **X** () const  
  *X position of cluster’s center.*
- const float & **Y** () const  
  *Y position of cluster’s center.*
- const float & **Z** () const  
  *Z position of cluster (≈ calorimeter Z position).*
- float **Xerr** () const  
  *error of X position*
- float **Yerr** () const  
  *error of Y position*
- float **Zerr** () const  
  *error of Z position*
- const float * **Cov** () const  
  *Full covariance matrix of the cluster coordinates. Only lower triangle (6 elements).*
- const float & **Chi2** () const  
  *\( \text{Chi}^2 \) of cluster fit.*
- const int & **Ndf** () const  
  *\( \text{NDF} \) of cluster fit. When \( \text{NDF} = 0 \), the fit has not been performed for this cluster.*
- float **ECellSum** () const  
  *Sum of cells’ energies.*

- int **Size ()** const
  
  > Cluster's size.

- bool **HasTime ()** const
  
  > "true" if cluster has measured time

- float **Time ()** const
  
  > cluster time [ns]

- float **SigmaT ()** const
  
  > cluster time resolution [ns]. Returns -1 if not available.

- int **NParticles ()** const
  
  > number of particles (**PaParticle** (p. 68)) with the reference to this cluster.

- int **iParticle (int i)** const
  
  > index of particle ≠ i (i≠0, **NParticles()** (p. 30))

- int **iTrack ()** const
  
  > index of charged track associated with this cluster. Returns -1 in not associated.

- const vector< **Int_t** > & **vCellNumber ()** const
  
  > Cell numbers in the cluster.

- const vector< **Float_t** > & **vCellEnergy ()** const
  
  > Energy depoisitions per cell in the cluster.

- const float & **RWprob ()** const
  
  > Probability to be charge particle cluster as it is given by RichWall code. -1 if not defined.

- const int & **ADCshapeFlag ()** const
  
  > ADC pulse shape analysis flag (set in coral/PaEventImportRD.cc).

- int **iCell (double &xc, double &yc, double margin=0)** const
  
  > Return corresponding cell number for this cluster's X,Y. Tolerance "margin" for the borders of cell could be specified (in [cm]).

- int **iCellMaxE (double &frac, double frac_cut)** const
  
  > Return cell number of this cluster which contains more than "frac_cut" fraction of total energy.

- **PaCaloClus & operator=(** (const PaCaloClus &c)
  
  > "-" operator

### 6.3.1 Detailed Description

Calorimeter cluster. Class to store result of cluster finding and fit in COMPASS calorimeters.

**Author:**

Sergei.Gerassimov@cern.ch
6.3.2 Member Function Documentation

6.3.2.1 int PaCaloClus::iCell (double & xc, double & yc, double margin = 0) const

Return corresponding cell number for this cluster's X,Y
Tolerance "margin" for the borders of cell could be specified (in [cm]). If cell not found, function returns "-1"
If cell is found, function also returns cell center coordinates (xc,yc)
References PaSetup::Calorimeter(), iCalorim(), PaCalorimCell::iType(), PaCalorimCell::Position(), PaCalorimeter::Position(), PaSetup::Ref(), PaCaloCellType::Size(), PaCalorimeter::vCaloCellType(), PaCalorimeter::vCalorimCell(), vCellNumber(), X(), and Y().

6.3.2.2 int PaCaloClus::iCellMaxE (double & frac, double frac_cut) const

Return cell number of this cluster which contains more than "frac_cut" fraction of total energy.
If such cell not found, function returns "-1"
Function also returns "Max Cell Energy / Sum of cell energies" in "frac"
References vCellEnergy(), and vCellNumber().
The documentation for this class was generated from the following files:

- lib/PaCaloClus.h
- lib/PaCaloClus.cc
6.4 PaCalorimCell Class Reference

Calorimeter cell type, position, calibrations.
#include <PaCalorimCell.h>

Public Member Functions

- const int & iType () const
  
  Cell type.

- const string & Name () const
  
  Unique cell name (as defined in detectors.dat).

- const TVector3 & Position () const
  
  Position (in detector’s reference system).

- const int & iColumn () const
  
  Column number where this cell is located.

- const int & iRow () const
  
  Row number where this cell is located.

- bool HasCalibConst () const
  
  returns "true" if calibration for this cell is stored

- bool HasEcor () const
  
  returns "true" if E-dependent corrections for this cell is stored

- double GetEcor (double E, bool interp) const
  
  returns correction factor which corresponds to cell energy E

- bool HasTcor () const
  
  returns "true" if T-dependent corrections for this cell is stored

- float CalibConst () const
  
  Calibration constant.

- const map< double, double > & mEdcor () const
  
  Energy dependent correction factors.

- const map< double, double > & mTdcor () const
  
  Time-in-spill dependent correction factors.

6.4.1 Detailed Description

Calorimeter cell type, position, calibrations.
6.4.2 Member Function Documentation

6.4.2.1 float PaCalorimCell::CalibConst () const [inline]

Calibration constant. Returns calibration constant used to transform ADC -> Energy

6.4.2.2 double PaCalorimCell::GetEcor (double E, bool interp = true) const

returns correction factor which corresponds to cell energy E If "interp" is true - interpolation is used. Otherwise coefficient from the first E bin with E > E cell is used.

If corrections for this cell are not available, function returns 1.0

References mEdcor(), and Name().

6.4.2.3 const map<double, double>& PaCalorimCell::mEdcor () const [inline]

Energy dependent correction factors. Returns map "cell energy -> correction factor" obtained in pi0 based calibration procedure.

Note:

In the case of "1-d" calibrations (1 correction factor per cell) this map contains one <0, correction factor> pair.

Referenced by GetEcor().

6.4.2.4 const map<double, double>& PaCalorimCell::mTdcor () const [inline]

Time-in-spill dependent correction factors. Returns map "Event time-in-spill -> correction factor" obtained in pi0 based calibration procedure. (to compensate rate-dependent variation of gain)

The documentation for this class was generated from the following files:

- lib/PaCalorimCell.h
- lib/PaCalorimCell.cc
6.5 PaCalorimeter Class Reference

Calorimeter Detector geometry.
#include <PaCalorimeter.h>

Public Member Functions

- const string & Name () const
  detector name

- const TVector3 Position () const
  Position (in the main reference system).

- const vector< PaCaloCellType > & vCaloCellType () const
  Vector of cell types.

- const vector< PaCalorimCell > & vCalorimCell () const
  Vector of cells.

- bool IsMyCluster (const PaCaloClus &cc) const
  Test if cluster "cc" belongs to "this" calorimeter using its Z position (by
  Vladimir.Kolesov@cern.ch).

- int iCell (double x, double y, double &xc, double &yc, double margin=0) const
  Return corresponding cell number for X,Y (in MRS) on this calorimeter front surface.
  Tolerance "margin" for the borders of cell could be specified (in [cm]).

6.5.1 Detailed Description

Calorimeter Detector geometry. Object of this class contains static (event-independent) part of
information about one calorimeter.

6.5.2 Member Function Documentation

6.5.2.1 int PaCalorimeter::iCell (double x, double y, double &xc, double &yc,
double margin = 0) const

Return corresponding cell number for X,Y (in MRS) on this calorimeter front surface.
Tolerance "margin" for the borders of cell could be specified (in [cm]). If cell not found, function
returns "-1"
If cell is found, function also returns cell center coordinates (xc,yc)
fill if it's not yet done
References PaCalorimCell::iType(), PaCalorimCell::Position(), Position(), PaCaloCell-
Type::Size(), vCaloCellType(), and vCalorimCell().
The documentation for this class was generated from the following files:
• lib/PaCalorimeter.h
• lib/PaCalorimeter.cc
6.6  PaDetect Class Reference

Tracking detector.

#include <PaDetect.h>

Public Member Functions

- const string & Name () const
detector name

- int IDet () const
unique detector ID

- short int Kind () const
detector’s kind (0 - wire/strip; 1 - drift)

- double RadLen () const
radiation length of detector materials

- double Pitch () const
Pitch.

- unsigned int Nwires () const
Number of wires.

- const vector<double> & W2Pos () const
vector of wire positions (for variable pitch detectors only)

- double Uorig () const
coordinate of the first wire.

- double Resol () const
detector resolution

- double Ca () const
Cos of measurement angle.

- double Sa () const
Sin of measurement angle.

- double Range () const
length of active area in measurement direction

- double Cframe () const
Cos of detector frame rotation angle.

- double Sframe () const
Sin of detector frame rotation angle.
• double **Efficiency** () const
  
  *Detector efficiency used in MC.*

• const double & **X** () const
  
  *X coordinate of center of detector in MRS.*

• const double & **Y** () const
  
  *Y coordinate of center of detector in MRS.*

• const double & **Z** () const
  
  *Z coordinate of center of detector in MRS.*

• const double & **XR** () const
  
  *X coordinate of center of detector in WRS.*

• const double & **YR** () const
  
  *Y coordinate of center of detector in WRS.*

• const double & **ZR** () const
  
  *Z coordinate of center of detector in WRS.*

• const double & **XSiz** () const
  
  *X detector size.*

• const double & **YSiz** () const
  
  *Y detector size.*

• const double & **ZSiz** () const
  
  *Z detector size.*

• const **TDZType** & **DZ_type** () const
  
  *Dead zone: shape (NO, RECTANGULAR, CIRCULAR).*

• const **Double_t** & **DZ_ydim** () const
  
  *Dead zone: X 1/2 size in case rectangular shape or radius squared (circular).*

• const **Double_t** & **DZ_zdim** () const
  
  *Dead zone: Y 1/2 size in case rectangular shape.*

• const **Double_t** & **DZ_Cads** () const
  
  *Cos of dead zone rotation angle in DRS.*

• const **Double_t** & **DZ_Sads** () const
  
  *Sin of dead zone rotation angle in DRS.*

• const **Double_t** & **DZ_ydrd** () const
  
  *X position of dead zone in DRS.*

• const **Double_t** & **DZ_zdrd** () const
  
  *Y position of dead zone in DRS.*
• const Double_t & DZ_ymrs () const 
  *X position of dead zone in WRS.*

• const Double_t & DZ_zmrs () const 
  *Y position of dead zone in WRS.*

• const vector< Int_t > & vHitRef () const 
  *Vector of references to hits on this detector (if hits was saved).*

• void Print (int level=0) const 
  *Print detectors info.*

• void ShiftHits (double dU, double dV) const 
  *introduce artificial shift of detector in DRS*

• void ExcludeFromFit () 
  *Exclude all hits of this detector from Kalman fit.*

• bool InActive (double x, double y) const 
  *Check if point (x,y) is in active area.*

• int GetWireNumber (double x, double y) const 
  *Find the wire number for a given point (x,y) Returns i-wire if point (x,y) is in the active area. -1 if not.*

6.6.1 Detailed Description

Tracking detector. Object of this class contains static (event-independent) part of information about one tracking detector’s plane.

Following notation is used:

**MRS** - main spectrometer reference system:
  *right-hand system with Z - along the beam, Y - vertical with the origin in the center of PT*

**DRS** - detector reference system
  System shifted to center of detectors parallelepiped and rotated 2 times: 
  by Cframe/Fframe and then by wire angle → U - perpendicular to wires, V - along wires, Z = 0

**WRS** - Wire reference system: origin like in MRS, but rotated around Z 
  to make X perpendicular to wires)

Author:

Sergei.Gerassimov@cern.ch

6.6.2 Member Function Documentation

6.6.2.1 int PaDetect::GetWireNumber (double x, double y) const

Find the wire number for a given point (x,y) Returns i-wire if point (x,y) is in the active area. -1 if not.
6.6 PaDetect Class Reference

Author:

marco.meyer@cern.ch

References InActive(), and Pitch().

6.6.2.2 bool PaDetect::InActive (double x, double y) const

Check if point (x, y) is in active area. Returns "true" if point (x, y) on detector's surface is not in dead zone and not out of detector.

Author:

Yann.Bedfer@cea.fr

References Pitch().

Referenced by PaTrack::CanBeMuon(), PaTrack::FullKF(), and GetWireNumber().

The documentation for this class was generated from the following files:

- lib/PaDetect.h
- lib/PaDetect.cc
6.7 PaDigit Class Reference

Raw information.

```
#include <PaDigit.h>
```

Public Member Functions

- **const int & IWire () const**
  
  *wire|strip|fiber|straw number or "electronic channel" number*

- **const vector< float > & vDigInfo () const**
  
  *vector of raw information (ADC, TDC, etc.)*

- **int NDigInfo () const**
  
  *Number of digit information words (size of the vector, returned by vDigInfo() (p. 41)).*

- **const float & DigInfo (int i) const**
  
  *Digit's word ≠ i (i-th element of the vector, returned by vDigInfo() (p. 41)).*

- **set<int > sMReference () const**
  
  *Reference(s) to "mother" MC track.*

- **void Print (int level=0) const**
  
  *Print contents of PaHit's digit (copy of CsDigit).*

- **void PrintRaw (int level=0) const**
  
  *Print contents of raw digit (copy of DAQ Digit).*

- **string DecodeMapName () const**
  
  *Returns "detector" name (as it was encoded by StorName() function).*

6.7.1 Detailed Description

Raw information. Class to store raw information of tracking detectors (e.g. amplitudes, times) and of other sources of information (e.g. scalers, RICH etc.)

This class has **double use** in PHAST:

- digits of **PaHit** (p. 50) (accessed by **PaHit::vDigits** (p. 50) or **PaEvent::Digits()** (p. 44)).
  
  In this case it contains information as CORAL’s **CsDigit** has (named 'standard' below).

- digits of **PaEvent** (p. 42) (accessed by **PaEvent::RawDigits()** (p. 44)).
  
  In this case it contains information as **CS::Chip::Digit** of DaqDecoding library has (named 'DAQ digit' below)

**Warning:**

meaning of information in vector returned by **PaDigit::vDigInfo()** (p. 41) is different in above mentioned cases.
6.7.2 Member Function Documentation

6.7.2.1 `const int & PaDigit::iWire () const [inline]`

wire|strip|fiber|straw number or "electronic channel" number Sequential number of tracking detector's sensitive element

In the case of DAQ digits, it is -1*("electronic channel"+1). This information is redundant and here it's just a flag. Retrieve "electronic channel" number by `vDigInfo()` (p. 41) function. (usually it's the first elements in the vector)

Referenced by Print().

6.7.2.2 `set< int > PaDigit::sMCReference () const`

Reference(s) to "mother" MC track. In the case of MC data, the function returns set of indexes of:

- **PaMCtrack** (p. 58) objects (for tracking detector digits) or
- **PaMChit** (p. 56) objects (for non-tracking detectors: RICH ...)

6.7.2.3 `const vector< float > & PaDigit::vDigInfo () const [inline]`

vector of raw information (ADC, TDC, etc.) In the case of 'standard' digits this vector contains the same information CORAL's function `CsDigit::getData()` returns.

In case of DAQ digits, this vector contains exactly what function `CS::Chip::Digit::GetNtupleData()` of DaqDecoding library returns for your particular type of digit (see docs of classes, derived from `CS::Chip::Digit`)

The documentation for this class was generated from the following files:

- `lib/PaDigit.h`
- `lib/PaDigit.cc`
6.8 PaEvent Class Reference

COMPASS event.

#include <PaEvent.h>

Public Member Functions

- void Print (int level=0) const
  
  Print event. Set level > 1 for more details.

- int NParticle () const
  
  number of particles

- const PaParticle & vParticle (int i) const
  
  reference to particle ≠ i

- int NTrack () const
  
  number of tracks

- const PaTrack & vTrack (int i) const
  
  reference to track ≠ i

- int NVertex () const
  
  number of vertices

- const PaVertex & vVertex (int i) const
  
  reference to vertex ≠ i

- int iBestPrimaryVertex () const
  
  best primary vertex index (if there are more than 1). Returns -1 if there is no primary vertexes in this event.

- int iBestCoralPrimaryVertex () const
  
  best primary (as determined in CORAL) vertex index. Returns -1 if there is no primary vertexes in this event.

- int NMCvertex () const
  
  number of MC vertices

- const PaMCvertex & vMCvertex (int i) const
  
  reference to MC vertex ≠ i

- int NMCtrack () const
  
  number of MC tracks

- const PaMCtrack & vMCtrack (int i) const
  
  reference to MC track ≠ i

- int NCaloClus () const
number of Calo. clusters

• const PaCaloClus & vCaloClus (int i) const
  reference to calorimetric cluster ≠ i

• bool IsMC () const
  "true" for MonteCarlo event

• int RunNum () const
  run number

• int TrigMask () const
  trigger mask

• int SpillNum () const
  burst number

• int EvInSpill () const
  event number in spill

• long long UniqueEvNum () const
  Unique event numbers, constructed from run number, spill number, event in spill.

• double TimeInSpill () const
  event time in spill (in seconds)

• int UnixSeconds () const
  Astronomic time when this event had been received by DAQ (in seconds of Unix epoch).

• int ChunkNumber () const
  Returns raw data file "chunk" number.

• string PeriodName () const
  Returns year and data taking period for this event (e.g. 02P2D, 03P1A etc).

• int Year () const
  Returns the year when this event had been received.

• int Date () const
  Returns year, month and day when event had been received in format YYYYMMDD (GMT).

• void Date (int &year, int &month, int &day, int &hour, int &min, int &sec) const
  Returns year, month, day, hour, min and sec when event had been received (GMT).

• float TCSphase () const
  returns TCS clock phase (in nanoseconds)

• int Type () const
  Event type (e.g. 1 - SOR,...5 - SOB, 6 - EOB, 7 - physics, 8 - calibration,... etc.).
• double InstFlux () const
  
  *instant flux in spill (particles per second). Negative value means "can't compute".*

• bool MCgen (LUDATA &ld) const
  
  *MonteCarlo LUND generator information (if stored) (COMGEANT vers. < - 6.9).*

• bool MCgen (NLUDATA &ld) const
  
  *MonteCarlo LUND generator information (if stored).*

• bool MCgen (int &Nparticles, vector< LIJET &vlij) const
  
  *Particle-dependent generator information (if stored).*

• bool MCgen (PYSUBS &ps) const
  
  *MonteCarlo PYTHIA (v > 6.2) generator information (if stored).*

• bool MCgen (PYPARS &pp) const
  
  *MonteCarlo PYTHIA (v > 6.2) generator information (if stored).*

• void PrintMCgen (int level)
  
  *Print content of PaMCgen (p. 55) object (if stored), level = 0, 1, 2.*

• bool OnlFltAccepted () const
  
  *"true" if Online Filter had accepted this event*

• bool OnlFltMonitor () const
  
  *"true" if it’s "monitor event"*

• const vector< UInt_t > & vMiscScalers () const
  
  *returns vector of miscellaneous scalers*

• const vector< PaHit > & Hits () const
  
  *returns vector of hits (clusters) of tracking detectors. Every hit also contains corresponding digits.*

• const vector< PaMChit > & MChits () const
  
  *returns vector of MC hits*

• const vector< PaDigit > & RawDigits () const
  
  *returns vector of "raw" (DAQ) digits if they had been stored. Function returns empty vector in the case of MC data.*

• const vector< PaDigit > & Digits () const
  
  *returns vector of "non-tracking" detector's digits (CsDigits of RICH, Calorimeters etc.)*

• int SubTriggerMask () const
  
  *bitmask of extra trigger information*

• int MasterTriggerMask () const
  
  *Master trigger mask.*
vector< pair< unsigned int, unsigned int > > DaqDecoErrors () const

   DAQ decoding error IDs with its counters.

void TagToSave ()

   tag this event to be written to output

void DoNotSave ()

   do not write this event to the output

void SaveToStream (int i)

   tag event to be written to output stream # i (only for multiple stream output mode)

void AddParticle (PaParticle &p)

   add new particle to the event

void AddVertex (PaVertex &v)

   add new vertex to the event

bool ToSave () const

   "true" is event is tagged to be saved

set< UInt_t > & sMisSrcID ()

   Set with "missing SourceIDs" returned by decoding library.

6.8.1 Detailed Description

COMPASS event. Main class for analysis. PaEvent (p.42) contains all event dependent information.

It may also contains MonteCarlo information, hits, digits.

Author:

Sergei.Gerassimov@cern.ch

6.8.2 Member Function Documentation

6.8.2.1 vector< pair< unsigned int, unsigned int > > PaEvent::DaqDecoErrors () const

   DAQ decoding error IDs with its counters. Returns vector of pairs where:

      * first number is: DAQ decoding error types which had happen at decoding of this event
      * second number is: now many times this error had happen

   Names (short description) of errors could be retrieved by means of const string& PaEvent::DaqDecoErrorName(unsigned int id) function

   Referenced by Print().

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
6.8.2.2 int PaEvent::MasterTriggerMask () const [inline]

Master trigger mask. If the event had more than 1 trigger bit set, master trigger mask shows which trigger was used to calculate trigger time in DaqDecoding library. Referenced by Print().

6.8.2.3 bool PaEvent::MCgen (PYPARS & pp) const

MonteCarlo PYTHIA (v >= 6.2) generator information (if stored).

Returns:
"false" if not found.

Parameters:

pp - PYPARS structure (see PaMCgen (p. 55) header)

6.8.2.4 bool PaEvent::MCgen (PYSUBS & ps) const

MonteCarlo PYTHIA (v >= 6.2) generator information (if stored). get PYTHIA event-independent structure

Returns:
"false" if not found.

Parameters:

ps - event independent PYSUBS structure (see PaMCgen (p. 55) header)

6.8.2.5 bool PaEvent::MCgen (int & Nparticles, vector< LUJET > & vlj) const

Particle-dependent generator information (if stored). get LUND particles

Returns:
"false" if not found.

Parameters:

Nparticles - number of particles (i.e. number of elements in vlj vector)
vlj - vector of LUJET structures: particle dependent information (see PaMCgen (p. 55) header)

6.8.2.6 bool PaEvent::MCgen (NLUDATA & ld) const

MonteCarlo LUND generator information (if stored). get LUND data

Returns:
"false" if not found.
Parameters:

\( ld \) - NLUDATA structure of event dependent generator information (see PaMCgen (p. 55) header)

6.8.2.7 bool PaEvent::MCgen (LUDATA & \( ld \)) const

MonteCarlo LUND generator information (if stored) (COMGEANT vers. <= 6.9). get LUND data (old format)

Returns:

"false" if not found.

Parameters:

\( ld \) - LUDATA structure of event dependent generator information (see PaMCgen (p. 55) header)

6.8.2.8 int PaEvent::SubTriggerMask () const [inline]

 bitmask of extra trigger information

- bit \#0: semi-inclusive HCAL1 (threshold 1)
- bit \#1: semi-inclusive HCAL2 (threshold 1)
- bit \#2: semi-inclusive ECAL1 (threshold 1)
- bit \#3: pure-calo HCAL1 (threshold 2)
- bit \#4: pure-calo HCAL2 (threshold 2)
- bit \#5: pure-calo ECAL1 (threshold 2)
- bit \#6: HCAL1 threshold 1, layer 1
- bit \#7: HCAL1 threshold 1, layer 2
- bit \#8: HCAL1 threshold 1, layer 3
- bit \#9: HCAL1 threshold 1, layer 4

(Ask trigger experts for more details on conditions of those triggers)

Referenced by Print().

6.8.2.9 const vector<UInt_t*>& PaEvent::vMiscScalers () const [inline]

returns vector of miscellaneous scalers See ./coral/PaEventImportScalers.cc to know what is stored in elements of this vector

The documentation for this class was generated from the following files:

- lib/PaEvent.h
- lib/PaEvent.cc
- lib/PaEventBestPrimaryVertex.cc
- lib/PaEventDiscard.cc
- lib/PaEventPrepare.cc
6.9 PaField Class Reference

Magnetic fields.

#include <PaField.h>

Public Member Functions

- `Int_t getNumOfMags ()`
  
  Returns the number of magnets.

- `PaMagInfo * getMagInfo ()`
  
  Returns array of size 3 with magnets’ information.

- `bool getField (Float_t pos_x, Float_t pos_y, Float_t pos_z, Float_t &field_x, Float_t &field_y, Float_t &field_z, TMatrix &grad) const`
  
  Returns the magnetic field and gradient of that field in a particular point in space.

- `bool getField (Float_t pos_x, Float_t pos_y, Float_t pos_z, Float_t &field_x, Float_t &field_y, Float_t &field_z) const`
  
  Returns the magnetic field in a particular point in space.

6.9.1 Detailed Description

Magnetic fields. Provides access to magnetic field maps of all COMPASS magnets.

Author:

Alexandre.KORZEN@cern.ch

6.9.2 Member Function Documentation

6.9.2.1 bool PaField::getField (Float_t pos_x, Float_t pos_y, Float_t pos_z, Float_t &field_x, Float_t &field_y, Float_t &field_z) const

Returns the magnetic field in a particular point in space. To find the value of field between nodes of grid linear interpolation is used.

Parameters:

- `pos_x` X position in space.
- `pos_y` Y position in space.
- `pos_z` Z position in space.
- `field_x` Return value: X component of magnetic field vector in pos.
- `field_y` Return value: Y component of magnetic field vector in pos.
- `field_z` Return value: Z component of magnetic field vector in pos.

References Phast::Ref(), PaSetup::Ref(), and Phast::UserFieldScale.
6.9.2.2 bool PaField::getField (Float t pos x, Float t pos y, Float t pos z, 
Float t & field_x, Float t & field_y, Float t & field_z, TMatrix & 
grad) const

Returns the magnetic field and gradient of that field in a particular point in space. To find the 
value of field between nodes of grid linear interpolation is used.

For calculation of derivative the following formula is used:
\[
\frac{\partial B(x, y, z)}{\partial x} = \frac{B(x + h, y, z) - B(x, y, z)}{h}
\]

Parameters:
- **pos_x** X position in space.
- **pos_y** Y position in space.
- **pos_z** Z position in space.
- **field_x** Return value: X component of magnetic field vector in pos.
- **field_y** Return value: Y component of magnetic field vector in pos.
- **field_z** Return value: Z component of magnetic field vector in pos.
- **grad** Return value: the gradient of magnetic field in pos.

\[
\operatorname{grad} = \begin{pmatrix}
\frac{\partial B_x}{\partial x} & \frac{\partial B_x}{\partial y} & \frac{\partial B_x}{\partial z} \\
\frac{\partial B_y}{\partial x} & \frac{\partial B_y}{\partial y} & \frac{\partial B_y}{\partial z} \\
\frac{\partial B_z}{\partial x} & \frac{\partial B_z}{\partial y} & \frac{\partial B_z}{\partial z}
\end{pmatrix}
\]

Referenced by PaSetup::MagField().

The documentation for this class was generated from the following files:

- lib/PaField.h
- lib/PaField.cc
6.10 PaHit Class Reference

Detector’s measurement.

#include <PaHit.h>

Public Member Functions

- int iDet () const
  detector’s index

- double U () const
  measurement (perpendicular to "wires")

- double V () const
  measurement (along "wires")

- double SigU () const
  measurement error (perpendicular to "wires")

- double SigV () const
  measurement error (along "wires")

- double SigT () const
  Time measurement error (-1 if detector do not measure time).

- double Time () const
  cluster time (main time of all digits in cluster)

- double DeltaR () const
  distance form the wire, calculated from the drift time. (= 0 for non-drift detectors)

- const vector< PaDigit > & vDigits () const
  Digits of this hit.

- const set< unsigned int > & sTracks () const
  references to reconstructed tracks.

- const vector< float > & vExtraData () const
  Associated extm information (e.g. cluster amplitude).

- const vector< float > & vExtraDataErr () const
  Errors of associated extm information.

- const vector< float > & vMisc () const
  Other associated information.

- const PaDetect & DetRef () const
  get reference to detector
• bool IsSpecialHit () const
  
  *true in case of special 'hit-container' of DAQ digits*

• void Print (int level=0) const

  *Print hit info.*

• PaHit & operator= (const PaHit &h)

  *assignment operator*

### 6.10.1 Detailed Description

Detector’s measurement. Class to store tracking detector’s measurements (clusters) and associated raw information (digits).

For MC runs, the clusters from simulated detector response are stored, but digits are not available. It also may serve as ‘container’ for non-tracking detector’s digits (e.g. scalers, RICH etc.)

**Author:**

Sergei.Gerassimov@cern.ch

### 6.10.2 Member Function Documentation

#### 6.10.2.1 int PaHit::iDet () const [inline]

detector’s index Detector’s position in vecDetect of PaSetup (p. 81).

To simplify access to detector’s object, DetRef() (p. 50) function is provided.

In such special case when both hits and DAQ digits had been requested to store, user should skip ‘hit’ with iDet == -777 and iDet == -888 (or better use function IsSpecialHit() (p. 51)), as those ‘special’ hits are not tracking detector measurement but just a ‘home’ for DAQ digits or CORAL digits (CsDigits) of non-tracking detectors.

The documentation for this class was generated from the following files:

• lib/PaHit.h
• lib/PaHit.cc
6.11 PaMaterialMaps Class Reference

Material Maps.
#include <PaMaterialMaps.h>

Public Member Functions

- void **getRadLength** (const **PaTPar** &hel, bool dir, float &RadLen, float &StepZ, bool limit_step=true) const
  
  Returns the radiative length RadLen in the point defined by hel. Also returns recommended step size along Z axis from hel to next medium.

- float **getdE** (const **PaTPar** &hel, double Len) const
  
  Returns the coefficient \( \frac{\Delta E}{\Delta x} \) for most probable energy loss at the point defined by hel for a step of length \( \text{Len} \) (in cm).

- float **getdEStraggling** (const **PaTPar** &hel, float Len) const
  
  Returns the sigma of energy loss distribution \( \frac{\Delta E}{\Delta x} \) for momentum losses during a step of length \( \text{Len} \) at the point defined by hel in target.

- bool **usingROOTGeometry** () const
  
  Return true if the ROOTGeometry is in use, false otherwise.

- bool **usingROOTGeometryTGEANT** () const
  
  Return true if the ROOTGeometry (in the form of GDML) is in use, false otherwise.

- const **TMacro &** **getROOTGeometry** () const
  
  Returns a reference to the macro containing the currently loaded ROOT Geometry.

- double **getMassDefault** () const
  
  Returns the default mass used for energy loss calculations in the case if ROOT geometry is used.

- bool **InMaterialMap** (**PaTPar** &hel) const
  
  Returns true if "hel" is inside material map.

- int **GDMILSize** ()
  
  returns the size of the GDML-Cache

- void **loadGDML** ()
  
  writes and loads the GDML-Cache

6.11.1 Detailed Description

Material Maps. The class **PaMaterialMaps** (p. 52) (taken from CsMaterialMap) contains the table of radiative lengths and the table of dE/dX. The volume is split into boxes and an average radiative lengths or dE/dX is assigned to every box. The basic idea of introducing material maps is to provide a track extrapolation code with a way to take into account effects of multiple scattering and energy losses.
Author:
Alexandre Korzenev
Jan P. Nassalski
Tobias.Schlueter@physik.uni-muenchen.de (implementation of ROOT geometry in CORAL)
Sergei.Gerasimov@cern.ch (ROOT geometry in mDST and Phast (p. 106))
Tobias.Szameitat@cern.ch (patches to use GDML-based ROOT geometry)
Christopher.Ralph.Regali@cern.ch (patches to use GDML-based ROOT geometry)

6.11.2 Member Function Documentation

6.11.2.1 float PaMaterialMaps::getE (const PaTPar & hel, double Len) const

Returns the coefficient (dE = GeV) for most probable energy loss at the point defined by hel for a step of length len (in cm).

Parameters:

hel The position in space (must be filled according to TRAFFIC).
Len The length of the step to take.

References PaTPar::DirCos(), usingROOTGeometry(), and usingROOTGeometryTGEANT().
Referenced by PaTPar::Extrapolate().

6.11.2.2 float PaMaterialMaps::getEStraggling (const PaTPar & hel, float Len) const

Returns the sigma of energy loss distribution (dE = GeV) for momentum losses during a step of length Len at the point defined by hel in target.

Parameters:

hel The position in space (must be filled according to TRAFFIC).
Len length of the step.

References PaTPar::DirCos(), usingROOTGeometry(), and usingROOTGeometryTGEANT().
Referenced by PaTPar::Extrapolate().

6.11.2.3 void PaMaterialMaps::getRadLength (const PaTPar & hel, bool direc, float & RadLen, float & StepZ, bool limit_step = true) const

Returns the radiative length RadLen in the point defined by hel. Also returns recommended step size along Z axis from hel to next medium.

Parameters:

hel reference to PaTPar (p. 84) object. I.e. position in space and directions.
direc Direction along beam axis:
  * direc = true : extrapolation is performed along beam.
  * direc = false : extrapolation is performed in opposite to beam direction.
**RadLen** The radiative length (in cm) at XYZ defined by *hel*.

**StepZ** Recommended step (in cm) along Z axis from *hel* position to the next medium in the direction defined by *hel*.

**limit_step** not used

Track is assumed to be extrapolated as straight line. If extrapolation is in backward direction than StepZ < 0.

References PaTPar::DirCos(), usingROOTGeometry(), and usingROOTGeometryTGEANT(). Referenced by PaTPar::Extrapolate().

6.11.2.4 bool PaMaterialMaps::InMaterialMap (PaTPar & hel) const

Returns true if "hel" is inside material map.

**Warning:**

As material map (if defined) are assumed to include also all detectors’ materials in sufficiently wide XY ranges, only Z coordinate is checked.

Referenced by PaTrack::FullKF().

The documentation for this class was generated from the following files:

- lib/PaMaterialMaps.h
- lib/PaMaterialMaps.cc
6.12 PaMgen Class Reference

MC generator information.

#include <PaMgen.h>

Public Member Functions

- void Print (int level=0) const
  
  Print MCgen information.

6.12.1 Detailed Description

MC generator information. It’s just a "storage place" for original MC generator information, available together with other MC data. Exact content and meaning of "packed" information depends on used generator.

For access to this information use set of PaEvent::MCgen() (p. 47) functions.

Author:

Sergsi.Gerassimov@cern.ch

The documentation for this class was generated from the following files:

- lib/PaMgen.h
- lib/PaMgen.cc
6.13 PaMChit Class Reference

MonteCarlo hit.
#include <PaMChit.h>

Public Member Functions

- double X () const
  
  $X$ coordinate.

- double Y () const
  
  $Y$ coordinate.

- double Z () const
  
  $Z$ coordinate.

- double Px () const
  
  $Px$ momentum component.

- double Py () const
  
  $Py$ momentum component.

- double Pz () const
  
  $Pz$ momentum component.

- double ELos () const
  
  Total energy lost in detector volume [GeV].

- double Time () const
  
  Time in [ns] (with respect to GEANT's time "zero").

- int iMCtrack () const
  
  index of MC track this hit belongs to

- int iDet () const
  
  Tracking detector's position in the vector vecDet of PaSetup (p. 81).

- int iOrig () const
  
  GEANT's "origin" flag: !0 for "secondary" MC hits (showers, delta-mys etc.).

- const PaDetect & DetRef () const
  
  returns reference to detector this MC hit belongs to.

- void Print (int level=-0) const
  
  Print MC hit info.
6.13.1 Detailed Description

MonteCarlo hit. Information on MC trajectory parameters at crossing with GEANT sensitive volumes.

Author:

Sergei.Gerassimov@cern.ch

6.13.2 Member Function Documentation

6.13.2.1 const PaDetect& PaMChit::DetRef () const [inline]

returns reference to detector this MC hit belongs to.

Warning:

valid for tracking detectors only. Check iDet() (p. 57) first (must be >= 0)

References PaSetup::Detector(), and PaSetup::Ref().
Referenced by Print().

6.13.2.2 int PaMChit::iDet () const [inline]

Tracking detector’s position in the vector vecDetect of PaSetup (p. 81). Returns -1 if MC hit comes from non-tracking detector such as RICH, HCAL etc. (In this case detector of origin could be easily recognized just by Z coordinate)

Referenced by Print().

6.13.2.3 double PaMChit::Time () const [inline]

Time in [ns] (with respect to GEANT’s time "zero").

Warning:

c*L is subtracted, where c - speed of light, L is minimal path length of trajectory from vertex of origin to position of this hit.

Referenced by Print().

The documentation for this class was generated from the following files:

- lib/PaMChit.h
- lib/PaMChit.cc
6.14 PaMCtrack Class Reference

MonteCarlo track

#include <PaMCtrack.h>

Public Member Functions

- int Pid () const
  
  Particle GEANT3 id.

- const string Name () const
  
  Particle name (as in GEANT3).

- double E () const
  
  Energy.

- double P (int i) const
  
  Px, Py, Pz (i=0,1,2).

- int IVertexOfOrigin () const
  
  Vertex of origin index.

- int NMCHits () const
  
  Number of MC hits.

- int NHits () const
  
  Number of hits, made of MC hits.

- bool IsPileup () const
  
  "true" for pileup track

- bool IsBeam () const
  
  "true" for beam track (always 1-st track in primary vertex with Pz < 0)

- bool IsMuPrim () const
  
  "true" for scattered muon (always 2-d track in primary vertex)

- TVector3 Mom3 () const
  
  3-vector of momentum components.

- TLorentzVector LzVec () const
  
  4-vector of the particle

- PaTPar ParInVtx () const
  
  Trajectory parameters in the vertex of origin.

- const vector<int> & vMCvertex () const
  
  vector of references to MC vertexes, produces by this MC track.
• const set<int>& sTrkRef() const
  set of references to associated reconstructed tracks (indexes in PaEvent::vTrack())

• const set<int>& sMChrRef() const
  MC hit references.

• const set<int>& sAllMChrRef() const
  MC hit references including "not-original" MC hits.

• int iTrack() const
  associated reconstructed track index

• TBits HitMap() const
  return hit map as TBits object

• const unsigned int* HitMapArray() const
  return hit map as is (as int[HIT_MAP_SIZE] array)

• int NHitsInDetect (string str) const
  Number of hits in detector(s).

• void Print (int level=0) const
  Print MCtrack info.

• int Q() const
  charge

• double Pinv() const
  q/P

• double MCmass() const
  calculated mass

• int iGenParticle() const
  generator particle index

• int iBestTrkRef() const
  Best associated reconstructed track index.

• void SetBestTrkRef(int itrk)
  Set the best associated reconstructed track index.

6.14.1 Detailed Description

MonteCarlo track. Simulated track (contains GEANTs KINE bank information and much more)

Author:

Sergei.Gerassimov@cern.ch
6.14.2 Member Function Documentation

6.14.2.1 int PaMCtrack::iBestTrkRef () const [inline]

Best associated reconstructed track index.

Warning:

No "best association" is done by default (index == -1). It's for "user-defined" association. (could be set it by SetBestTrkRef() (p. 61) function)

6.14.2.2 int PaMCtrack::iGenParticle () const [inline]

generator particle index Returns element number of corresponding LUJET structure in the vector<LUJET> which could be retrieved by function PaEvent::MCgen(int& Nparticles, vector<LUJET>& vj) or "-1" if not exists

Referenced by Print().

6.14.2.3 int PaMCtrack::iTrack () const [inline]

associated reconstructed track index It is just first element of sTrkRef() (p. 59).

-1 means "no associated track"

Warning:

If there are few associations of this MC track to reconstructed tracks (i.e. sTrkRef() (p. 59).size() > 1),

first reference in sTrkRef() (p. 59) doesn't mean "the best"

References sTrkRef().

6.14.2.4 TVector3 PaMCtrack::Mom3 () const [inline]

3-vector of momentum components. 3-momentum at the point of origin of the simulated particle trajectory. This is "native" (no transformations on the way) information as it's coming from GEANT.

Warning:

If this representation of trajectory parameters are used in analysis, user has "to invert" signs of beam's momentum components, as in simulation beam goes "upward".

6.14.2.5 int PaMCtrack::NHitsInDetect (string str) const

Number of hits in detector(s). Returns total number of hits (clusters) in detectors with name begins on substring "str" (e.g. NHitsInDetect("FI") gives number of hits in SciFi, NHitsInDetect("Si01") gives N hits in first silicon station)

References PaSetup::Detector(), PaDetect::Name(), PaSetup::NDetectors(), and PaSetup::Ref().
6.14.2.6 const set<
t>& PaMCTrack::sAllMChitRef () const [inline]

MC hit references including "not-original" MC hits. Similar to "set<
t>& sMChitRef()" function but returns also references to MC hits from showers and delta rays produced by this MC track. So one may observe more than 1 MC hit per detector.

6.14.2.7 void PaMCTrack::SetBestTrkRef (int itrk) [inline]

Set the best associated reconstructed track index. It's the function for users who wants to have more "strict" "MC track" -> "reconstructed track" association than out of sTrkRef() (p. 59).

6.14.2.8 const set<
t>& PaMCTrack::sMChitRef () const [inline]

MC hit references. Returns set of references to available MC hits (could be accessed by PaEvent::MChits() (p. 44)).

- As only MC hits from some particular detectors could be saved (controlled by "mDST MChits" option), sMChitRef() (p. 61).size() \leq \text{NMCHits}() (p. 58)
- Only references to "original" MC hits had been saved (\Rightarrow only 1 MC hits per MC track per sensitive volume is possible)

The documentation for this class was generated from the following files:

- lib/PaMCTrack.h
- lib/PaMCTrack.cc
6.15 PaMCvertex Class Reference

MonteCarlo vertex.

#include <PaMCvertex.h>

Public Member Functions

- `bool IsPrimary () const
  "true" for primary vertex.

- `int iBeam () const
  index of beam particle in PaEvent::vecMCtrk. ...

- `int iTrackOfOrigin () const
  index of mother particle (incoming trajectory) in PaEvent::vecMCtrk. ...

- `bool IsPileup () const
  "true" for pseudo vertex, emitting pileup track.

- `double Pos (int i) const
  X,Y,Z (i=0,1,2).

- `double TimeOfFlight () const
  Time of flight.

- `int NMCtrack () const
  Number of outgoing MC tracks in the vertex.

- `int iMCtrack (int i) const
  index of outgoing MC track # i

- `void Print (int level=0) const
  Print MCvertex info.

6.15.1 Detailed Description

MonteCarlo vertex. Simulated vertex - source of MC tracks (contains GEANTs VTX bank information and much more)

Author:

Sergei.Gerassimov@cern.ch

6.15.2 Member Function Documentation

6.15.2.1 `int PaMCvertex::iBeam () const [inline]

index of beam particle in PaEvent::vecMCtrk. ... Valid only for primary vertex. For all other vertices returns -1
6.15.2.2  bool PaMCvertex::IsPileup () const  [inline]
"true" for pseudo vertex, emitting pileup track. "Pileup vertex" is the vertex with only 1 track, without mother track, and with Z far upstream ( < -800 cm)

6.15.2.3  bool PaMCvertex::IsPrimary () const  [inline]
"true" for primary vertex. By COMGEANT convention, "primary" vertex is the first in the list of MC vertices.
Primary vertex also do not have "mother" track.
Referenced by PaTrigger::MultiplicityCounter().

6.15.2.4  int PaMCvertex::iTrackOfOrigin () const  [inline]
index of mother particle (incomming trajectory) in PaEvent::vecMTrk. ...

Warning:
Primary vertex and "pileup" vertices do not have mother particle (i.e.function returns -1).

Referenced by Print().

6.15.2.5  int PaMCvertex::NMCTrack () const  [inline]
Number of outgoing MC tracks in the vertex.

Warning:
In the primary vertex beam track is also counted as "outgoing" (see FAQ for more details)

Referenced by iMTrack(), and PaTrigger::MultiplicityCounter().
The documentation for this class was generated from the following files:

- lib/PaMCvertex.h
- lib/PaMCvertex.cc
6.16 PaMetaDB Class Reference

Calibrations, coefficients etc.

\#include <PaMetaDB.h>

Public Member Functions

- void Print ()
  
  *print brief content*

- vector< float > vNminusOne (int irun) const
  
  *Refractive index(es) for run "irun".*

- bool TargetSpinZproj (vector< float > &polar) const
  
  *function returns deuteron polarizations in target cells with the sign of spin projection on Z direction.*

- void PrintPolarDBentry (int irun) const
  
  *Print "offline" target DB entry for run \# irun.*

Static Public Member Functions

- static PaMetaDB & Ref ()
  
  *Return const reference to this object.*

6.16.1 Detailed Description

Calibrations, coefficients etc. It’s a class for various kind of calibrations, constants, measurements which are needed for analysis on miniDST level but was not accessible or was not ready at the time of data production.

6.16.2 Member Function Documentation

6.16.2.1 static PaMetaDB& PaMetaDB::Ref () [inline, static]

Return const reference to this object. Could be used for access to PaMetaDB (p. 64) from any place in the code.

Referenced by PrintPolarDBentry(), and vNminusOne().

6.16.2.2 bool PaMetaDB::TargetSpinZproj (vector< float > & polar) const

function returns deuteron polarizations in target cells with the sign of spin projection on Z direction.
Parameters:

polar - vector of polarizations. Size of the vector corresponds to number of target cells (2 or 3)

Data are taken from "offline" target database
Function returns "false" if polarizations was not found in the database for the current run number.
References PaSetup::Ref(), and PaEvent::RunNum().

6.16.2.3 vector< float > PaMetaDB::vNminusOne (int irun = -777) const

Refractive index(es) for run "irun". Returns (n-1) in [ppm] (1.e-6). Source of information: "snapshot" of DB (see time stamp in ./src/RICH1_ref_index.db or in ./src/RICH1_2ref_index.db)
If the function returns empty vector, it means "no data for this run"
Refractive index(es), used at production could be accessed by PaSetup::vNminusOne() (p. 83) function.
References Ref().
The documentation for this class was generated from the following files:

- lib/PaMetaDB.h
- lib/PaMetaDB.cc
6.17 PaMtx Class Reference

Scalar / Vector / Matrix
#include <PaMtx.h>

Public Member Functions

- **PaMtx ()**
  
  1x1 matrix constructor (scalar)

- **PaMtx (int m)**
  
  Mx1 matrix (vector) constructor.

- **PaMtx (int m, int n)**
  
  MxN matrix constructor.

- **PaMtx (const PaMtx &)**
  
  copy constructor

- **~PaMtx ()**
  
  destructor

- **void Print (string comment) const**
  
  Print matrix.

- **PaMtx t ()**
  
  Transpose matrix.

- **PaMtx inv (int &ierr)**
  
  Invert symmetric matrix.

- **PaMtx i5 (int &ierr)**
  
  Invert 5x5 matrix.

- **PaMtx i (int &ierr)**
  
  Invert matrix.

- **PaMtx & operator= (const PaMtx &)**
  
  = operator

- **PaMtx operator* (const PaMtx &)**
  
  * operator

- **PaMtx operator+ (const PaMtx &)**
  
  + operator

- **PaMtx operator- (const PaMtx &)**
  
  - operator
• PaMtx & operator += (const PaMtx &)
  "+=" operation

• PaMtx & operator *= (const double &)
  "*=" operation

• double & operator() (const int i, const int j=1) const
  accessor to i,j element

• operator double ()
  conversion 1x1 matrix to double

6.17.1 Detailed Description

Scalar / Vector / Matrix. Matrix class. Valid also for scalars (like 1x1 matrix) and vectors (like Mx1 matrix). (mostly is used in Kalman fit)

Note:
  Matrix inversion is based on CERNLIB function

Author:
  Sergei.Gerassimov@cern.ch

6.17.2 Member Function Documentation

6.17.2.1 void PaMtx::Print (string str) const

Print matrix.

Parameters:
  str comment to be printed

References i().
Referenced by PaTPar::Update().
The documentation for this class was generated from the following files:

• lib/PaMtx.h
• lib/PaMtx.cc
6.18 PaParticle Class Reference

Reconstructed particle.

#include <PaParticle.h>

Public Member Functions

- int iTrack () const
  
  PaTrack’s index (-1 in not exists).

- int PID () const
  Particle ID (GEANT3 notation).

- Byte_t BitFlags () const
  get bit flags

- bool IsBeam () const
  is it beam?

- int Q () const
  particle’s charge (-?? if charge is not known)

- float RMass () const
  Reconstructed mass of the particle.

- bool IsMuPrim () const
  is it a scattered muon? (based on PaVertex::iMuPrim (p.104))

- bool IsMuPrimFast () const
  is it a scattered muon? Fast version. (based on PaVertex::iMuPrim (p.104))

- bool IsMuPrimCoral () const
  is it scattered muon? (based on CORAL)

- bool Chi2CutFlag () const
  returns flag according to recommended cut on probability of chi2_BMS (0 if not set)

- double BackPropLH () const
  In case of BMS reconstruction introduced in 2011 the function returns probability of chi2_BMS calculated for time and spatial compatibility between given beam track and BMS hits associated to it. In case of old BMS reconstruction only spatial compatibility is considered here. The function returns 0 if value is not set.

- const PaTPar & ParInVtx (int iv) const
  parameters of this particle at vertex with index iv

- int NCalorim () const
  number of calorimeter cluster references
- int iCalorim (int i) const
  
  *colorimeter cluster index ≠ i*

- int NVertex () const
  
  *number of vertices this particle associated with.*

- int iVertex (int i) const
  
  *provides access to vertex indexes (in PaEvent::vVertex()) this particle associated with.* (0 ≤ i < NVertex() (p. 70))

- int NOutVertex () const
  
  *number of outgoing (daughter) vertices this particle produce*

- int iOutVertex (int i) const
  
  *outgoing vertex indexes.*

- void SetPID (const Short_t &id)
  
  *Set particle ID.*

### 6.18.1 Detailed Description

Reconstructed particle. Reconstructed particle (charged or neutral). Main input for analyses. PaPartice contains:

- references to all available information:
  
  - reconstructed charged track
  
  - reconstructed calorimeters’ clusters
  
  - reconstructed vertices

- trajectory parameters in vertices (if associated)

- particle ID (if identified)

- misc. flags *(beam, mu’ etc.)*

Here and in all other classes, *reference, index* of an object means its sequential number in corresponding vector of PaEvent (p. 42) class.

*index == -1 means "information is missing"

**Author:**

*Sergei.Gerassimov@cern.ch*

### 6.18.2 Member Function Documentation

#### 6.18.2.1 bool PaParticle::IsMuPrim () const

is it a scattered muon? (based on PaVertex::iMuPrim (p. 104))
Warning:

if the mu’ identification was not executed it is checked if the particle is associated to the Best Primary Vertex. If so the `PaVertex::iMuPrim` (p. 104) with default options is executed for the Best PV. Then the result is returned. If a user is interested in particles that belong to NOT Best PV the `PaVertex::iMuPrim` (p. 104) has to be called explicitly.

Author:

Konrad.Klimaszewski@cern.ch

References PaEvent::iBestPrimaryVertex(), PaVertex::iMuPrim(), PaVertex::MuPrimCalled(), Phast::Ref(), and PaEvent::vVertex().

6.18.2.2 bool PaParticle::IsMuPrimFast () const [inline]

is it a scattered muon? Fast version. (based on PaVertex::iMuPrim (p. 104))

Warning:

to return a meaningful result the PaVertex::iMuPrim (p. 104) must have been called explicitly (can be checked with PaVertex::MuPrimCalled() (p. 103)). This is not checked by PaParticle::IsMuPrimFast() (p. 70) and no automatic execution of the Mu’ ID is performed.

Author:

Konrad.Klimaszewski@cern.ch

6.18.2.3 int PaParticle::NVertex () const [inline]

number of vertices this particle associated with. Please note that this NVertex() (p. 70) is not equal to PaEvent::NVertex() (p. 42).

Referenced by iVertex(), and ParInVtx().

6.18.2.4 const PaTPar& PaParticle::ParInVtx (int iv) const [inline]

parameters of this particle at vertex with index iv Function returns this particle parameters at vertex ≠ iv (result of the vertex fit).

Warning:

Please note, that iv is a vertex index in the list of all event vertexes. So, not all iv are allowed as particle is associated with subset of event vertexes. If this particle is not associated with specified iv error message will be printed and job will be stopped. It is user’s responsibility to check that this particle is really associated with vertex iv.

Features:

- X,Y,Z of fitted PaTPar (p. 84) are taken from X,Y,Z of corresponding PaVertex (p. 102).
- X’,Y’,q/P are result of vertex Klman fit.
• Not all covariances are filled ("."")

\[
\begin{array}{cccccc}
 \times & \gamma & \times' & \gamma' & q/p \\
\times & + & + & - & - & - \\
\gamma & + & + & - & - & - \\
\times' & - & - & + & + & + \\
\gamma' & - & - & + & + & + \\
q/p & - & - & + & + & + \\
\end{array}
\]

Full covariance matrix of \(X, \gamma, Z\) could be taken from corresponding vertex.

References iVertex(), and NVertex().

6.18.2.5 int PaParticle::PID () const [inline]

Particle ID (GEANT3 notation).

Warning:

Currently only trivial muon identification based on radiation lengths passed by the particle is done.

6.18.2.6 int PaParticle::Q () const [inline]

particle’s charge (-777 if charge is not known) This charge is taken from momentum sign of associated track in the first measured point.

Warning:

\(Q == -777\) means that this particle has a link to reconstructed track with unknown curvature (e.g. it is a track piece reconstructed only in field free region)

6.18.2.7 float PaParticle::RMass () const [inline]

Reconstructed mass of the particle. Returns -1 if not defined.

The documentation for this class was generated from the following files:

• lib/PaParticle.h
• lib/PaParticle.cc
6.19 PaPid Class Reference

functions for RICH particle identification.

```cpp
#include <PaPid.h>
```

**Public Member Functions**

- `bool CheckRichInfo (PaTrack track)`
  
  *Check the RICH buffer in mDST; false if no RICH information is found.*

- `bool CheckCedarInfo (const PaTrack &track)`
  
  *Check the RICH buffer in mDST; false if no RICH information is found.*

- `float GetLike (int tag, PaTrack track)`
  
  *Gives Likelihood values.*

- `float GetLikeToBackground (int tag, PaTrack track)`
  
  *Gives Likelihood values divided to the background likelihood value.*

- `bool IsKaon (PaTrack track, float qnom, float polar_rich)`
  
  *True if particle is identified as K, otherwise false.*

- `int LikePid (PaTrack track)`
  
  *Function to identify the particles with Likelihood; suggested as a starting point for identification.*

- `float SecondLike (PaTrack track)`
  
  *Gives the ratio of the likelihood of the identified particle (from function LikePid) to the second maximum likelihood.*

- `double Mass (PaTrack track, double n, double mom)`
  
  *Compute hadron mass starting from n (refractive index value) and momentum values.*

- `double Theta_Ch (PaTrack track)`
  
  *Gives Cherenkov angle (in rad) from ring fit.*

- `double PMTRingTime (PaTrack track)`
  
  *Gives the mean ring time in ns.*

- `double NphoPMT (PaTrack track)`
  
  *Gives the number of photons in the PMT part of the reconstructed ring, used to evaluate the mean ring time.*

- `double Npho (PaTrack track)`
  
  *Gives the number of photons in the reconstructed ring.*

- `int CedarPid (const PaTrack &track, const std::vector< double > limits=std::vector< double >(8, 0))`
  
  *Function to identify the particles with CEDAR Likelihood.*
6.19 PaPid Class Reference

6.19.1 Detailed Description

functions for RICH particle identification. This class can be used to identify particles using the RICH informations (likelihoods - LH).

A starting point for the identification is provided in the function "LikePid" in which the hadrons are identified choosing the mass hypothesis giving the maximum likelihood.

To have a cleaner identification, afterwards cuts on the ratios LH/LH_{background} and LH/LH_{2nd max} are usually done: the two ratios are provided from the class members "GetLikeToBackground" and "SecondLike"; the cuts have to be tuned on data from the user.

The function IsKaon is given just as an example of cuts tuned as a function of the phase space of the particle.

NB: In the function explanation, the sentence "old versions of RICH code" indicates the RICH code used before 2004, in which the muon and electron LH were not computed.

Author:
Federica Sozzi <fsozzi@mail.cern.ch>

6.19.2 Member Function Documentation

6.19.2.1 int PaPid::CedarPid (const PaTrack & track, const std::vector< double > & limits = std::vector< double > (8, 0))

Function to identify the particles with CEDAR Likelihood.

Returns:
- 0 pion
- 1 kaon
- 2 unidentified

References CheckCedarInfo(), and PaTrack::RichInf().

6.19.2.2 float PaPid::GetLike (int tag, PaTrack track)

Gives Likelihood values.

Parameters:
- tag
  - 0 - pion
  - 1 - kaon
  - 2 - proton
  - 3 - electron (not available in old versions of RICH code)
  - 4 - muon (not available in old versions of RICH code)
  - 5 - background (equal to 1 in old versions of RICH code)

- track PaTrack (p. 91) object

Returns:
- Likelihood value (-1 if no RICH info available)
Warning:

in the old versions of RICH code, the likelihood values are divided for likelihood background.

References CheckRichInfo(), PaTrack::NRichInf(), and PaTrack::RichInf().
Referenced by LikePid(), and SecondLike().

6.19.2.3 float PaPid::GetLikeToBackground (int tag, PaTrack track)

Gives Likelihood values divided to the background likelihood value:

Parameters:

  tag
  • 0 - pion
  • 1 - kaon
  • 2 - proton
  • 3 - electron (not available in old versions of RICH code)
  • 4 - muon (not available in old versions of RICH code)
  • 5 - background

track PaTrack (p. 91) object

Returns:

Likelihood value divided to the background likelihood value (-1 if no RICH info available)

Warning:

in the old code of the RICH, this function is equivalent to the GetLike, since likelihood values are already divided for likelihood background in the buffer.

References CheckRichInfo(), PaTrack::NRichInf(), and PaTrack::RichInf().
Referenced by IsKaon().

6.19.2.4 bool PaPid::IsKaon (PaTrack track, float qnom, float polar_rich)

True if particle is identified as K, otherwise false. The identification is made asking the kaon to have the maximum LH value (function LikePid) and then performing a cut on the 2 variables: LH/LH_{background} and LH/LH_{2nd max}

The cuts depend on the particle charge, on the momentum and on the polar angle (in rad) at the RICH entrance. The parameters have been tuned on phi samples in data.

Warning:

This function is given only as a possible example of identification procedure; it should be better to tune the cuts of identification independently in every analysis.

References CheckRichInfo(), GetLikeToBackground(), LikePid(), PaTrack::NRichInf(), and SecondLike().
6.19.2.5 int PaPid::LikePId (PaTrack track)

Function to identify the particles with Likelihood; suggested as a starting point for identification. Select the mass hypothesis corresponding to the max likelihood value

Returns:

0 pion hypothesis
1 kaon
2 proton
3 electron (not available in old versions of RICH code - in this function excluded from identification if p > 8 GeV/c)
4 muon (not available in old versions of RICH code - in this function excluded from identification)
5 background (not available in old versions of RICH code)
-1 if no RICH info available or no maximum found or momentum > 50 GeV/c.

Warning:

RICH is not expected to distinguish between pions and muons, and between pions and electron above ~ 8 GeV/c

References CheckRichInfo(), GetLike(), PaTPar::Mom(), PaTrack::NRichInf(), and PaTrack::vPar().

Referenced by IsKaoon(), and SecondLike().

6.19.2.6 double PaPid::Mass (PaTrack track, double n, double mom)

Compute hadron mass starting from n (refractive index value) and momentum values.

Returns:

-1 if no RICH info available or cherenkov angle from the ring fit is zero.

References CheckRichInfo(), and Theta_Ch().

6.19.2.7 double PaPid::Npho (PaTrack track)

Gives the number of photons in the reconstructed ring.

Returns:

-1 if no RICH info available.

References CheckRichInfo(), and PaTrack::RichInf().

6.19.2.8 double PaPid::NphoPMT (PaTrack track)

Gives the number of photons in the PMT part of the reconstructed ring, used to evaluate the mean ring time.

Returns:

-1 if no RICH info available.

References PaTrack::NRichInf(), and PaTrack::RichInf().
6.19.2.9  double PaPid::PMTRingTime (PaTrack track)

Gives the mean ring time in ns.

Returns:

1000 if no RICH info available or there are no photons in the PMT part

References PaTrack::NRichInf(), and PaTrack::RichInf().

6.19.2.10  float PaPid::SecondLike (PaTrack track)

Gives the ratio of the likelihood of the identified particle (from function LikePid) to the second maximum likelihood.

Returns:

-1 if no RICH info available or no max LH found

References GetLike(), LikePid(), PaTPar::Mom(), PaTrack::NRichInf(), and PaTrack::vTPar(). Referenced by IsKion().

6.19.2.11  double PaPid::Theta_Ch (PaTrack track)

Gives Cherenkov angle (in rad) from ring fit.

Returns:

-1 if no RICH info available

References CheckRichInfo(), and PaTrack::RichInf().
Referenced by Mass().

The documentation for this class was generated from the following files:

- lib/PaPid.h
- lib/PaPid.cc
6.20 PaRich Class Reference

Rich Detector geometry.

```c
#include <PaRich.h>
```

**Public Member Functions**

- **string Name () const**
  
  
  *detector name*

- **const TVector3 & MirrC (int i) const**

  *Mirror center of curvature position in MRS ("top" (i=0) or "bottom" (i=1)).*

- **const float & MirrR (int i) const**

  *Mirror radius ("top" (i=0) or "bottom" (i=1)).*

- **const TVector3 & DetPos (int i) const**

  *Photon detector center position in MRS ("top" (i=0) or "bottom" (i=1)).*

- **const TRotation & DetRot (int i) const**

  *Photon detector rotation with respect to MRS ("top" (i=0) or "bottom" (i=1)).*

- **const vector< PaRichMirr > & vMirrElem (int i) const**

  *Mirror elements vector ("top" (i=0) or "bottom" (i=1)).*

- **const vector< PaRichDet > & vRichDet (int i) const**

  *Photon detector elements vector ("top" (i=0) or "bottom" (i=1)).*

### 6.20.1 Detailed Description

Rich Detector geometry. Object of this class contains static (event-independent) information about RICH detector.

**Author:**

Sergei.Gerassimov@cern.ch

The documentation for this class was generated from the following files:

- lib/PaRich.h
- lib/PaRich.cc
- lib/PaRichPrepare.cc
6.21 PaRichDet Class Reference

Rich photon detector element (photocathode).

```
#include <PaRichDet.h>
```

**Public Member Functions**

- `string Name () const`
  
  *detector name*

- `int CathNum () const`
  
  *cathod number (as in DAQ digits)*

- `float Xpad00 () const`
  
  *Pad 0,0 X position in DRS.*

- `float Ypad00 () const`
  
  *Pad 0,0 Y position in DRS.*

- `float Xcenter () const`
  
  *Photocathode center X position in DRS.*

- `float Ycenter () const`
  
  *Photocathode center Y position in DRS.*

- `int NpadX () const`
  
  *Number of pads in X direction.*

- `int NpadY () const`
  
  *Number of pads in X direction.*

- `float PitchX () const`
  
  *pitch in X*

- `float Xlow () const`
  
  *pitch in X*

- `float Xhigh () const`
  
  *X high (in DRS) of active area.*

- `float Ylow () const`
  
  *Y low (in DRS) of active area.*

- `float Yhigh () const`
  
  *Y high (in DRS) of active area.*
6.21.1 Detailed Description

Rich photon detector element (photocathode).

6.21.2 Member Function Documentation

6.21.2.1 int PaRichDet::NpadX () const

Number of pads in X direction.
< Number of pads in X direction
References PaSetup::Ref(), and PaSetup::RunNum().
Referenced by NpadY(), Xhigh(), and Xlow().

6.21.2.2 float PaRichDet::PitchX () const

< Pads’ pitch in X direction
References PaSetup::Ref(), and PaSetup::RunNum().
Referenced by PitchY(), Xhigh(), and Xlow().

6.21.2.3 float PaRichDet::Xlow () const

pitch in X X low (in DRS) of active area
References NpadX(), PitchX(), and Xcenter().
The documentation for this class was generated from the following files:

- lib/PaRichDet.h
- lib/PaRichDet.cc
6.22 PaRichMIRR Class Reference

Rich mirror element.
#include <PaRichMIRR.h>

Public Member Functions

- string Name () const
  
  Mirror element name.

- const TVector3 & MirrPos () const
  
  Mirror element center position (in MRS).

- const TVector3 & MirrCC () const
  
  Mirror element center of curvature position (in MRS).

- const float & MirrR () const
  
  Radius of curvature.

6.22.1 Detailed Description

Rich mirror element.

The documentation for this class was generated from the following files:

- lib/PaRichMIRR.h
- lib/PaRichMIRR.cc
6.23 PaSetup Class Reference

COMPASS setup (geometry, magnetic fields etc.).

#include <PaSetup.h>

Public Member Functions

- **int RunNum () const**
  
  *Run number.*

- **int NDetectors () const**
  
  *Number of detectors in the setup.*

- **const PaDetect & Detector (int i) const**
  
  *Reference to detector ≠ i*

- **int iDetector (string name) const**
  
  *Index of detector "name" (returns -1 if "name" not found)*

- **int iDetFirst (string str) const**
  
  *Index of the first (in Z) detector with name starting on "str" (-1 means "not found")*

- **void ClearHitRefs ()**
  
  *Clear references to hits on all detectors (if hits was saved).*

- **void AddHitRef (int idet, int ihit)**
  
  *Add to detector "idet" hit reference "ihit"*

- **int NCalorimeter () const**
  
  *Number of calorimeters in the setup.*

- **const PaCalorimeter & Calorimeter (int i) const**
  
  *Reference to calorimeter ≠ i*

- **const PaMaterialMaps & MaterialMaps () const**
  
  *Reference to material map*

- **float SolenoidCurrent () const**
  
  *Solenoid current from CORAL CsMagInfo class (in Amps with sign).*

- **const vector<float> & TargetPolarizations () const**
  
  *Targets polarizations (up /down)*

- **void MagField (float x, float y, float z, float &Fx, float &Fy, float &Fz) const**
  
  *Returns 3 components of magnetic field Fx,Fy,Fz [Tesla] in space point x,y,z.*

- **TVector3 MagField (TVector3 &v) const**
  
  *Returns magnetic field vector [Tesla] in space point "v".*
• const vector< Float_t > & vNminusOne () const
  Refractive index(es) (n-1) for current run in [ppm] (1.e-6) as it was used in production.

• const string & DaqDecoErrorName (unsigned int id) const
  For DAQ decoding error ID returns error name.

• int iCalorim (const string &calo_name) const
  For calorimeter name "calo_name" returns it's number (index in vecCalorim). Returns -1 if "calo_name" is not known.

• const PaRich & Rich () const
  Return const reference to RICH geometry object.

• float TargetCenterZ () const
  Return center of target Z position.

• void Print (int level=0) const
  Print setup info.

Static Public Member Functions

• static const PaSetup & Ref ()
  Return const reference to current PaSetup (p. 81) object.

6.23.1 Detailed Description

COMPASS setup (geometry, magnetic fields etc.). Class for "slowly changing" (run-dependent) information and for "static" (not-changing) information.

Author:
Sergei.Gerassimov@cern.ch

6.23.2 Member Function Documentation

6.23.2.1 static const PaSetup& PaSetup::Ref () [inline, static]

Return const reference to current PaSetup (p. 81) object. Could be used for access to PaSetup (p. 81) from any place in the code. Example: PaSetup::Ref() (p. 82). Print() (p. 82);

Referenced by PaCaloClus::CalorimName(), PaTrack::CanBeMuon(), PaTrack::CrossYokeSM2(), PaMChit::DetRef(), PaHit::DetRef(), PaTPar::Extrapolate(), PaTrack::FullKF(), PaField::getField(), PaTrack::GetSmoothed(), PaTrigger::HCal2(), PaCaloClus::iCell(), PaTrack::iHit(), PaVertex::iMuPrim(), PaCaloClus::iTrack(), PaTrack::KeepOnly(), PaTrack::NHitsFoundInDetect(), PaTrack::NHitsFoundInRange(), PaMTrack::NHitsInDetect(), PaRichDet::NpadX(), PaTrack::PassedDetectorNames(), PaDetect::Pitch(), PaRichDet::PitchX(), PaTrack::PointsHodoscopes(), PaEvent::Print(), PaTrack::QuickKF(), PaAlgo::RKutta(), PaDetect::ShiftHits(), PaCaloClus::SigmaT(), PaMetaDB::TargetSpinZproj(), PaTrack::Zmax(), and PaTrack::Zmin().

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
6.23.2.2 float PaSetup::TargetCenterZ () const [inline]

Return center of target Z position.

Warning:

if the values is not stored (mDST written with older PHAST versions), default value -35.0 cm is returned.

6.23.2.3 const vector<float> & PaSetup::TargetPolarizations () const

targets polarizations (up /down) Returns vector of target polarizations as it was known at time of production (what CORAL function CsMagInfo::getPolarization(run, pol) had returned)

Warning:

Please note that all values are already multiplied by solenoid current sign.

References RunNum(), and SolenoidCurrent().

Referenced by Print().

6.23.2.4 const vector<Float_t>& PaSetup::vNminusOne () const [inline]

Refractive index(es) (n-1) for current run in [ppm] (1.e-6) as it was used in production. Returns empty vector if not known.

More up-to-date (post-production) refractive index(es) for this run could be accessed by PaMetaDB::vNminusOne(int irun) function.

Referenced by Print().

The documentation for this class was generated from the following files:

- lib/PaSetup.h
- lib/PaSetup.cc
- lib/PaSetupPrepare.cc
6.24 PaTPar Class Reference

Trajectory parameters.

#include <PaTPar.h>

Public Member Functions

- float & operator() (const int i, const int j=0)
  
  Access to individual trajectory parameters and its covariance matrix by "()" operator.

- double Dip (bool in_deg=false) const
  
  Dip angle.

- double Azi (bool in_deg=false) const
  
  Azi angle.

- double Phi (bool in_deg=false) const
  
  $\phi$ angle.

- double Theta (bool in_deg=false) const
  
  $\Theta$ angle.

- double DirCos (char axis) const
  
  return direction cosine (axis = 'x', 'y', 'z' for COSx, COSy, COSz)

- void DirVector (TVector3 &vec) const
  
  return track direction unit vector "vec"

- double Mom () const
  
  return $|P|$.

- int Q () const
  
  return charge.

- double qP () const
  
  return $P\cdot\text{charge}$.

- double Pt () const
  
  return Pt.

- double Pos (int i) const
  
  returns X if i == 0, Y if i == 1 and Z if i == 2.

- TVector3 Pos () const
  
  returns X,Y,Z as TVector3.

- double X () const
  
  X of the trajectory.
- double Y () const
  
  *Y of the trajectory.*

- double Z () const
  
  *Z (fixed) where trajectory parameters are defined.*

- double dXdZ () const
  
  *dX/dZ (horizontal slope) of the trajectory*

- double dYdZ () const
  
  *dY/dZ (vertical slope) of the trajectory*

- TVector3 Mom3 () const
  
  *returns 3-vector of momentum components*

- TVector LzVec (double mass) const
  
  *returns Lorentz vector for specified mass*

- TVector ParVect () const
  
  *returns parameters as TVector*

- TMatrix CovMatrix () const
  
  *returns covariance matrix as TMatrix*

- bool isEmpty () const
  
  *"true" if it's empty (not defined) trajectory params*

- bool HasEmptyCov () const
  
  *true" if covariance matrix is not defined*

- bool HasMom () const
  
  *"true" if momentum had been defined*

- void Print (int level=0, string comment="") const
  
  *Print.*

- void Rotate (double ca, double sa, PaTPar &Pout) const
  
  *Rotate to "wire reference system".*

- double Dist (PaTPar &P) const
  
  *Straight line distance from "this" PaTPar (p. 84) (X,Y,Z) to PaTPar (p. 84) P.*

- void AddNoise (float x, float RadLen, bool reverse_ms=false)
  
  *Add multiple scattering "noise" to covariance matrix after propagation through a material.*

- bool Extrapolate (double Z, PaTPar &Hout, bool useMatMap=true, bool reverse_ms=false) const
  
  *Propagate charged particle trajectory parameters.*

- bool ExtrapNeutral (double Z, PaTPar &Hout) const
Propagate neutral particle trajectory parameters.

- double Path () const
  
  Returns trajectory length of previous extrapolation.

- double RadLenFr () const
  
  Returns fractions of radiation length passed in previous extrapolation.

- bool FindCDA (PaTPar &H, float Z0=0, float Zmin=-1000, float Zmax=+1000, bool neutral1=false, bool neutral2=false)

  Find closest distance of approach for "this" PaTPar (p. 84) and PaTPar (p. 84) "H".

- bool Update (PaTPar &H1, PaTPar &Hout, double &Chi2)
  
  do Kalman update of "this" state vector by "H1"

- bool Update (const PaHit &h, PaTPar &Hout)
  
  do Kalman update of "this" state vector by measurement (hit) "h"

- double HitChi2 (PaHit &h)

  calculate $\chi^2$ increment for "this" state vector by measurement "h" (object of PaHit (p. 50) class)

- void Set (double &x, PaMtx &V, PaMtx &M)

  Pack vector and matrix into helix.

- void Get (double &x, PaMtx &V, PaMtx &M) const

  Unpack helix to vector and matrix.

### 6.24.1 Detailed Description

Trajectory parameters. Class for charged particle trajectory parameters $X$, $Y$, $dX/dZ$, $dY/dZ$, $q/P$ at fixed $Z$ position with corresponding covariance matrix.

Read and write access to parameters is provided by means of overloaded "operator ()".

E.g. if $p$ is an object of this class, $p(i)$ are references to parameters, $p(i,j)$ - are references to covariance matrix elements.

$i, j=1, 2, 3, 4, 5$ and corresponds to $X, Y, X', Y'$, $q/P$.

For example: $p(0)$ is "fixed parameter" (i.e. $Z$ position where trajectory parameters are defined), $p(1)$ is $X$, $p(3)$ is $dX/dY$, $p(5)$ is $q/P$ etc.

This class also could be used to store neutral particles parameters, assuming that 5-th parameter is just $1/P$.

(Warning: Do not use Extrapolate () (p. 87) in this case. Use ExtrapNeutral() (p. 87) instead.)

**Author:**

Sergei.Gerassimov@cern.ch
6.24.2 Member Function Documentation

6.24.2.1 void PaTPar::AddNoise (float x, float RadLen, bool reverse_ms = false)

Add multiple scattering "noise" to covariance matrix after propagation through a material.

Parameters:

- $x$ - thickness of material traversed
- $RadLen$ - radiation length of material traversed
- $reverse\_ms$ - if set to true", "noise" will be _subtracted_ (for very special purposes)

Referenced by Extrapolate(), and PaTrack::FullKF().

6.24.2.2 double PaTPar::Azi (bool in_deg = false) const

Azi angle. Angle between projection of $\vec{P}$ on plane X0Z and Z (rad/deg depending on parameter)

6.24.2.3 TMatrix PaTPar::CovMatrix () const

returns covariance matrix as TMatrix

Warning:

Returns matrix "by value". Avoid to use in CPU time consuming code

6.24.2.4 double PaTPar::Dip (bool in_deg = false) const

Dip angle. Angle between $\vec{P}$ and it's projection on plane X0Z (rad/deg depending on parameter)

6.24.2.5 bool PaTPar::ExtrapNeutral (double Z, PaTPar & Hout) const

Propagate neutral particle trajectory parameters. Perform straight line extrapolation of neutral particle parameters to specified "Z"

Result is stored in "Hout". References Dist().

6.24.2.6 bool PaTPar::Extrapolate (double Z, PaTPar & Hout, bool useMatMap = true, bool reverse_ms = false) const

Propagate charged particle trajectory parameters.

Parameters:

- $Z$ - z-coordinate of desired destination point of the propagation
- $Hout$ - resulting trajectory parameters at $Z$ (if $Z$ had been reached)
- $useMatMap$ - set to "false" to ignore multiple scattering and energy losses effects
- $reverse\_ms$ - special purpose mode. Please always leave default (false).
Function returns "false" is Z had not been reached.

Warning:

Do not use it for neutral particles extrapolation. Use `ExtrapNeutral()` (p. 87) instead.

References AddNoise(), Dist(), PaMaterialMaps::getdE(), PaMaterialMaps::getdEStraggling(), PaMaterialMaps::getRadLength(), HasMom(), PaSetup::MaterialMaps(), Mom(), Path(), and PaSetup::Ref().

Referenced by PaTrack::CanBeMuon(), PaAlgo::CrossCells(), PaTrack::CrossYokeSM2(), PaTrack::Extrapolate(), PaTrack::FullKF(), PaTrack::GetSmoothed(), and PaTrack::PointsHodoscopes().

6.24.2.7 `bool PaTPar::FindCDA (PaTPar & H, float Z0 = 0, float Zmin = -1000, float Zmax = +1000, bool neutral1 = false, bool neutral2 = false)`

Find closest distance of approach for "this" `PaTPar` (p. 84) and `PaTPar` (p. 84) "H". Function find such Z (within specified range) that distance between 2 trajectories propagated to this Z is minimal.

At the exit, "this" `PaTPar` (p. 84) and `PaTPar` (p. 84) "H" has it’s values at the Z where they have minimal distance.

Parameters:

- `H` - 2-d input `PaTPar` (p. 84) (first one is "this")
- `Z0` - starting value for minimization
- `Zmin` - lower limit
- `Zmax` - upper limit
- `neutral1` - to be set to "true" if `PaTPar` (p. 84) "this" is neutral particle
- `neutral2` - to be set to "true" if `PaTPar` (p. 84) "H" is neutral particle

Returns:

"false" if minimization did not converged or had reached one of 2 limits

Warning:

- Function changes both "this" object and H!
- What is minimized is not the distance in space but the distance at the plane Z=const.
- Extrapolations of charged particles are used by default. Use "neutral1" and "neutral2" flags to force straight line extrapolations.

6.24.2.8 `double PaTPar::HitChi2 (PaHit & h)`

calculate $\chi^2$ increment for "this" state vector by measurement "h" (object of `PaHit` (p. 50) class)

Returns:

$\Delta \chi^2$

Referenced by PaTrack::FullKF(), and PaTrack::QuickKF().
6.24.2.9 float & PaTPar::operator() (const int i, const int j = 0)

Access to individual trajectory parameters and it’s covariance matrix by "()" operator. If "p" is an object of class PaTPar (p. 84), then

- p(0) is Z position where trajectory parameters are defined
- p(1) is X
- p(2) is Y
- p(3) is dX/dZ
- p(4) is dY/dZ
- p(5) is q/|P|
- p(i,j) is COV_{i,j}, where i,j = 1,2,3,4,5

6.24.2.10 double PaTPar::Phi (bool in_deg = false) const

φ angle. Angle between projection of \( \vec{P} \) on plane X0Y and X (rad/deg depending on parameter)

References DirCos().

6.24.2.11 void PaTPar::Print (int level = 0, string str = "") const

Print. Print trajectory parameters

Parameters:

- \textit{level} if > 0 full cov. matrix is printed
- \textit{str} comment to be printed

References Mom3().

Referenced by PaTrack::FullKF(), PaTrack::GetSmoothed(), PaMTrack::Print(), PaTrack::QuickKF(), and Update().

6.24.2.12 void PaTPar::Rotate (double \textit{ca}, double \textit{sa}, PaTPar & \textit{Pout}) const

Rotate to "wire reference system". Rotate trajectory parameters (together with cov. matrix) to reference system rotation of which is defined by matrix:

\[
\begin{bmatrix}
\cos(a) & \sin(a) & 0 & 0 & 0 \\
-\sin(a) & \cos(a) & 0 & 0 & 0 \\
0 & 0 & \cos(a) & \sin(a) & 0 \\
0 & 0 & -\sin(a) & \cos(a) & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Parameters:

- \textit{ca} - \cos(a)
- \textit{sa} - \sin(a)
- \textit{Pout} - resulting trajectory parameters.

Referenced by Update().
6.24.2.13  double PaTPar::Theta (bool \texttt{in\_deg} = \texttt{false}) const

\( \Theta \) angle. Angle between \( \vec{P} \) and \( Z \) (rad/deg depending on parameter)

6.24.2.14  bool PaTPar::Update (const PaHit & \textit{h},  \textit{PaTPar} & \textit{Hout})

do Kalman update of "this" state vector by measurement \textit{hit} "h" Result is stored to "Hout".
References PaDetect::Ca(), PaHit::DetRef(), Print(), Rotate(), PaDetect::Sa(), PaHit::U(), and PaHit::V().

6.24.2.15  bool PaTPar::Update (\textit{PaTPar} & \textit{H1},  \textit{PaTPar} & \textit{Hout},  \textit{double} & \textit{Chi2})

do Kalman update of "this" state vector by "H1" Result is stored to "Hout". At the same time \( \chi^2 \) increment is returned in "Chi2"
References Get(), PaMtx::i(), PaMtx::Print(), and Set().
Referenced by PaTrack::FullKF(), PaTrack::GetSmoothed(), and PaTrack::QuickKF().
The documentation for this class was generated from the following files:

- lib/PaTPar.h
- lib/PaTPar.cc
- lib/PaTParAddNoise.cc
- lib/PaTParExtrapolate.cc
- lib/PaTParExtrapolate.cc
- lib/PaTParFindCDA.cc
- lib/PaTParHitChi2.cc
- lib/PaTParUpdate.cc
6.25 PaTrack Class Reference

Track.

#include <PaTrack.h>

Public Member Functions

- float Zmax () const
  
  Z coordinate of the most downstream detector of the track.

- float Zmin () const
  
  Z coordinate of the most upstream detector of the track.

- float ZLast () const
  
  Z of the last measured point.

- float ZFirst () const
  
  Z of the first measured point.

- int CountNHits ()
  
  Count NHits in hit pattern and store it in "nhits" data member.

- int NHits () const
  
  Number of hits.

- const float & Chi2tot () const
  
  Total \( \chi^2 \).

- int Ndf () const
  
  Number degrees of freedom.

- const float & MeanTime () const
  
  Track time (with respect to trigger time) in [ns].

- const float & SigmaTime () const
  
  Error of track time, excluding BMS (0 if track time is unknown).

- const float & Chi2Time () const
  
  Time \( \chi^2 \): excluding BMS (0 if track time is unknown).

- const float & XX0 () const
  
  Track pass in units of radiation lengths \((X/X_0)\).

- int NTPar () const
  
  Number of trajectory parameters.

- const PaTPar & vTPar (int i) const
  
  Trajectory parameter \( \neq i \).
- bool `Extrapolate` (double Z, `PaTPar` &Hout, bool useMatMap=true) const
  
  Extrapolate track to position Z.

- bool `Extrap` (double Z, `PaTPar` &Hout) const
  
  The same as `Extrapolate()` (p. 95) but without use of material maps (i.e. without multiple scattering and energy losses).

- int `Q` () const
  
  charge

- bool `HasMom` () const
  
  "true" if momentum had been defined

- float `qP` () const
  
  track momentum with sign (charge*|P|)

- int `NRichInf` () const
  
  Rich information array size.

- const float & `RichInf` (int i) const
  
  Rich information array element ≠ i.

- TBits `HitsExpected` () const
  
  "Expected" hit pattern

- TBits `HitsFound` () const
  
  "Found" hit pattern

- int `NHitsFoundInDetect` (string str) const
  
  returns number of found hits in detectors with name begins on substring "str" (e.g. `NHitsFoundInDetect("FI")` gives number of hits in SciFi)

- int `NHitsFoundInRange` (double Zmin, double Zmax) const
  
  returns number of found hits in detectors with Z positions within specified range [Zmin; Zmax]

- vector<string> `PassedDetectorNames` () const

- const vector<Int_t> & `vHitRef` () const
  
  References to PaHits of this track (not stored in "standard" mDST).

- float `FieldInteg` (float &r) const
  
  Estimation of the field integral [T*m] "accumulated" by this track.

- bool `QuickKF` (int dir, int mode)
  
  Quick (no multi. scattering corrections) Kalman fit through already found hits.

- bool `FullKF` (int dir)
  
  Kalman fit through already found hits (no hits re-assignment) with use of material maps (if switched ON).

- double `GetSmoothed` (`PaTPar` &Hout, string detname, bool flg)
Calculate "smoothed" track parameters on the detector with name "detname" and returns Chi2/n df. (function returns "-1" if this track do not cross detector "detname").

- **double GetSmoothed** (PaTPar &Hout, int ipl, bool flg)
  Calculate "smoothed" track parameters on the plane ≠ ipl and returns Chi2/n df. (function returns "-1" value in the case of problem).

- **double GetSmoothed** (PaTPar &Hout, double Z)
  Calculate "smoothed" track parameters at given Z and returns Chi2/n df. If track do not cross plane X, returns "-1".

- **bool KFdone ()**
  return "true" if Kalman fit had been done in both directions for this track.

- **int iHit (string detname)**
  return hit reference to the track's hit on the detector "detname" or "-1" if there is no hit associated to track on this detector.

- **int KeepOnly (const list< string >&detectors)**
  Remove all hits from the track except hits of detectors with names specified in "detectors" (only few first letters of detector names are allowed).

- **bool CanBeMuon () const**
  Check if the track is a possibly misidentified muon.

- **bool CrossYokeSM2 (bool strict=true) const**
  Check if the track passes through the SM2 yoke.

- **bool HasBMSMom () const**
  "true" if BMS momentum had been defined

- **bool PointsHodoscopes () const**
  Check if the track points to a trigger that fired. It is checked that a track goes through the active area of one of the trigger hodoscope pairs that gave the trigger.

- **int IMCtrack () const**
  corresponding MC track index (or -1 in not associated)

- **int iParticle () const**
  corresponding Particle index

- **const UInt_t* FoundHitsBitMap () const**
  return "Found hits" bitmap as plane array (size = HIT_MAP_SIZE defined in PaSetup.h (p. ??))

- **void SetMCtrack (int i)**
  Set corresponding MC track index.
6.25.1 Detailed Description

Track. Reconstructed charged track.
This class contains information, that normally is **not** needed for physical analysis but for various kinds of performance studies only. For physics, use **PaParticle** (p. 68) instead.

Author:

Sergei.Gerassimov@cern.ch

6.25.2 Member Function Documentation

6.25.2.1 bool PaTrack::CanBeMuon (void) const

Check if the track is a possibly misidentified muon. It is checked if the track goes through the hole of COMPASS absorber system.

Author:

Konrad.Klimaszewski@cern.ch

References PaSetup::Detector(), PaTPar::Extrapolate(), PaSetup::iDetector(), PaDetect::InActive(), PaTPar::Mom(), NTPar(), Q(), PaSetup::Ref(), Phast::Ref(), PaEvent::RunNum(), vTPar(), PaDetect::Z(), and ZLast().

Referenced by PaVertex::iMuPrim().

6.25.2.2 const float & PaTrack::Chi2tot () const [inline]

Total $\chi^2$. Estimation of total $\chi^2$ as $\Sigma \delta \chi_i^2$ of every update step on backward Kalman fit of the track

Referenced by FullKF(), and QuickKF().

6.25.2.3 bool PaTrack::CrossYokeSM2 (bool strict = true) const

Check if the track passes through the SM2 yoke.

Parameters:

*strict* - take into account the extrapolation error

Author:

Konrad.Klimaszewski@cern.ch

References PaTPar::Extrapolate(), PaField::getMagInfo(), NTPar(), PaSetup::Ref(), vTPar(), and ZLast().

Referenced by PaVertex::iMuPrim().
6.25.2.4   bool PaTrack::Extrapolate (double Z, PaTPar & Hout, bool useMatMap = true) const

Extrapolate track to position Z. Takes track parameters (PaTPar (p. 84)) closest to desired Z position (if more than 1 is stored) and extrapolate it. If track has it’s parameters only in the first measured point, then this PaTPar (p. 84) is used.

Extrapolated track parameters are returned via "Hout".
Function returns "false" if extrapolation had failed.
References PaTPar::Extrapolate(), NTPar(), and vTPar().

6.25.2.5   float PaTrack::FieldInteg (float & r) const

Estimation of the field integral [T*m] "accumulated" by this track.

Parameters:

r - ratio of field integral in Z range [1-st track point; 1000cm] to full field integral

References NTPar(), vTPar(), and ZLast().

6.25.2.6   bool PaTrack::FullKF (int dir)

Kalman fit through already found hits (no hits re-assignment) with use of material maps (if switched ON). Accumulated fraction of rad. lengths is stored into radLenFr datamember

Parameters:

dir - fit direction

Input:

<table>
<thead>
<tr>
<th>dir</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fit in forward direction (resulting helix is last element of vecTPar)</td>
</tr>
<tr>
<td>-1</td>
<td>fit in backward direction (resulting helix is first element of vecTPar)</td>
</tr>
</tbody>
</table>

Kalman fit through already found hits (no hits re-assignment) with use of material maps (if switched ON) Accumulated fraction of rad. lengths is stored into radLenFr datamember

Parameters:

dir - fit direction

Input:

<table>
<thead>
<tr>
<th>dir</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fit in forward direction (resulting helix is last element of vecTPar)</td>
</tr>
<tr>
<td>-1</td>
<td>fit in backward direction (resulting helix is first element of vecTPar)</td>
</tr>
</tbody>
</table>

References PaTPar::AddNoise(), Chi2tot(), PaSetup::Detector(), PaTPar::DirCos(), PaTPar::Extrapolate(), PaTPar::HasMom(), PaTPar::HitChi2(), PaDetect::InActive(), PaMaterialMaps::InMaterialMap(), PaSetup::MaterialMaps(), PaDetect::Name(), PaTPar::Path(), PaEvent::Print(), PaHit::Print(), PaTPar::Print(), QuickKF(), PaDetect::RadLen(), PaTPar::RadLenFr(), Phast::Ref(), PaSetup::Ref(), PaTPar::Update(), and PaDetect::Z().

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
6.25.2.7 double PaTrack::GetSmoothed (PaTPar & Hout, double Z)

Calculate "smoothed" track parameters at given Z and returns Chi2/ndf. If track do not cross
plane X, returns "-1".

Parameters:

  Hout - output smoothed helix
  Z - input: coordinate along the beam, where smoothed track parameters has to be calculated

References PaTPar::Extrapolate(), PaTPar::Print(), and PaTPar::Update().

6.25.2.8 double PaTrack::GetSmoothed (PaTPar & Hout, int ipl, bool flg)

Calculate "smoothed" track parameters on the plane # ipl and returns Chi2/ndf. (function returns
"-1" value in the case of problem).

Parameters:

  Hout - output smoothed helix
  ipl - input: plane number where smoothed track parameters has to be calculated
  flg - input: if == false, measurement on the current plane is excluded. If there is no hit on
  the plane ipl, this flag has no effect.

References KFdone(), PaEvent::Print(), PaTPar::Print(), and PaTPar::Update().

6.25.2.9 double PaTrack::GetSmoothed (PaTPar & Hout, string detname, bool flg)

Calculate "smoothed" track parameters on the detector with name "detname" and returns
Chi2/ndf. (function returns "-1" if this track do not cross detector "detname").

Parameters:

  Hout - output smoothed helix
  detname - input: detector name where smoothed track parameters has to be calculated
  flg - input: if == false, measurement on the current plane is excluded. If there is no hit on
  the detector detname, this flag has no effect.

References PaSetup::iDetector(), and PaSetup::Ref().

6.25.2.10 int PaTrack::KeepOnly (const list<string> & detectors)

Remove all hits from the track excluding hits of detectors with names specified in "detectors"
(only few first letters of detector names are allowed). Returns number of left over hits.

Example: If list contains 2 elements "FI" and "SI" all hits excluding hits in silicon detectors and
SciFi will be removed.

References CountNHits(), PaSetup::Detector(), PaDetect::Name(), PaSetup::NDetectors(), and
PaSetup::Ref().

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
6.25.2.11  int PaTrack::Ndf () const  [inline]

Number degrees of freedom. Number degrees of freedom of the track fit.
References HasMom(), and NHits().

6.25.2.12  vector< string > PaTrack::PassedDetectorNames () const

Returns vector of detector names this track had passed through
References PaSetup::Detector(), PaDetect::Name(), PaSetup::NDetectors(), and PaSetup::Ref().

6.25.2.13  bool PaTrack::PointsHodoscopes () const

Check if the track points to a trigger that fired. It is checked that a track goes through the active
area of one of the trigger hodoscope pairs that gave the trigger.

Author:

Konrad.Klimaszewski@cern.ch

References PaSetup::Detector(), PaTPar::Extrapolate(), PaField::getMagInfo(),
PaSetup::iDetector(), PaEvent::IsMC(), NTPar(), PaSetup::Ref(), Phast::Ref(),
PaEvent::RunNum(), PaEvent::TrigMask(), vTPar(), PaTPar::X(), PaTPar::Y(), and
PaEvent::Year().

Referenced by PaVertex::iMuPrim().

6.25.2.14  int PaTrack::Q () const  [inline]

change

Warning:

If the momentum of track is unknown, this function prints a message and returns 999

Referenced by CanBeMuon(), and PaVertex::iMuPrim().

6.25.2.15  bool PaTrack::QuickKF (int dir, int mode)

Quick (no mult. scattering corrections) Kalman fit through already found hits.

Parameters:

  *dir*  - fit direction

  *mode*  - defines how to fit

Input:

dir  =  1  - fit in downstream direction (resulting helix is last element of vecTPar)
dir  =  -1  - fit in upstream direction  (resulting helix is first element of vecTPar)

mode  =  0  - straight line fit
mode  =  1  - starts from known (in the first approximation) helix
(first or last, depending on direction).

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
Scale up covariance matrix and do the fit
mode = 2 - start from fully uncertain (except of momentum) helix (for debug only)

References Chi2tot(), PaSetup::Detector(), PaTPar::HasMom(), PaTPar::HitChi2(), PaDetec-
t::Name(), PaEvent::Print(), PaHit::Print(), PaTPar::Print(), Phast::Ref(), PaSetup::Ref(),
PaTPar::Update(), and PaDetect::Z().

Referenced by FullKF().

6.25.2.16 const PaTPar& PaTrack::vTPar (int i) const [inline]

Trajectory parameter # i. In "standard" mDST only parameters 2 exists (first and last measured point).
(In productions of 2006 and earlier only PaTPar (p. 84) in the first measured point was stored)

Warning:

For beam tracks (tracks ending before the target) order or track parameters is reversed, so
vTPar(0) are parameters at the point closest to target.

References NTPar().

Referenced by CanBeMuon(), CrossYokeSM2(), Extrapolate(), FieldInteg(), Pa-
Pid::LikePid(), PointsHodoscopes(), and PaPid::SecondLike().

6.25.2.17 float PaTrack::ZFirst () const

Z of the first measured point. This function returns Z of the first track parameter: vTPar(0)

Warning:

For beam tracks (tracks ending before the target) order of parameters is reversed, i.e. first
track parameters is closest to the target.

6.25.2.18 float PaTrack::ZLast () const [inline]

Z of the last measured point. This function return Z coordinate of the last hitted detectors (cor-
responding to last bit of HitsFound() (p. 92)).

References Zmax().

Referenced by CanBeMuon(), CrossYokeSM2(), FieldInteg(), and PaVertex::iMuPrim().

The documentation for this class was generated from the following files:

- lib/PaTrack.h
- lib/PaTrack.cc
- lib/PaTrackFullKF.cc
- lib/PaTrackGetSmoothed.cc
- lib/PaTrackPointsHodoscopes.cc
- lib/PaTrackQuickKF.cc
6.26 PaTrigger Class Reference

Trigger and Online filter related algorithms and methods.
#include <PaTrigger.h>

Public Member Functions

- bool OnlineFilter (const PaEvent &e, int tmask=-1, bool force=false) const
  implements the online filter for RD and MC of 2004 hadron run, returns true if event is accepted

- bool MultiplicityCounter (const PaEvent &e) const
  implements the 2004 Multiplicity counter for Monte Carlo data, returns true if event is accepted

- bool BeamKiller2of3 (const PaEvent &e) const
  implements the 2004 Beamkiller-Veto for Monte Carlo data, returns true if event is accepted

- bool HCal2 (const PaEvent &e) const
  implements the 2004 HCal2 Trigger, returns true if event is accepted

Static Public Member Functions

- static PaTrigger * Instance ()
  returns pointer to global PaTrigger (p. 99) object, which is automatically spawned on access of this function

6.26.1 Detailed Description

Trigger and Online filter related algorithms and methods.

Author:

thiemo.nagel@ph.tum.de

6.26.2 Member Function Documentation

6.26.2.1 bool PaTrigger::BeamKiller2of3 (const PaEvent & e) const

implements the 2004 Beamkiller-Veto for Monte Carlo data, returns true if event is accepted.
Uses MCHits.

Note: Positions (6.0,0,2268.0)cm and radius r=2.5cm hardcoded.

Note: Positions (13.0,0,2934.0)cm and radius r=2.5cm hardcoded.

Note: Positions (16.5,0,3274.0)cm and radius r=2.5cm hardcoded.

Author:

smbubert@e18.physik.tu-muenchen.de

References PaHit::DetRef(), PaHit::IsSpecialHit(), and PaDetect::Name().
6.26.2.2 bool PaTrigger::HCal2 (const PaEvent & e) const

implements the 2004 HCal2 Trigger, returns true if event is accepted **Note**: Threshold 6GeV hardcoded

References PaSetup::Calorimeter(), PaCaloClus::E(), PaCalorimeter::IsMyCluster(), PaEvent::NCaloClus(), PaSetup::Ref(), and PaEvent::vCaloClus().

6.26.2.3 static PaTrigger* PaTrigger::Instance () [inline, static]

returns pointer to global PaTrigger (p. 99) object, which is automatically spawned on access of this function BTW: This doesn’t mean that PaTrigger (p. 99) is a singleton: Any number of PaTrigger (p. 99) objects may be created and used independently.

6.26.2.4 bool PaTrigger::MultiplicityCounter (const PaEvent & e) const

implements the 2004 Multiplicity counter for Monte Carlo data, returns true if event is accepted

**Note**: Position (0,0,-247.0)cm and radius r=2.5cm hardcoded

!! z-coordinate hardcoded here!

!! radius hardcoded here!

References PaMCvertex::iMTrack(), PaMCvertex::IsPrimary(), PaMCvertex::NMTrack(), PaEvent::NMVertex(), PaMTrack::ParInVtx(), PaEvent::vMTrack(), and PaEvent::vMCvertex().

6.26.2.5 bool PaTrigger::OnlineFilter (const PaEvent & e, int tmask = -1, bool force = false) const

implements the online filter for RD and MC of 2004 hadron run, returns true if event is accepted

**Real Data**: For runs which already have been filtered during data taking, the monitoring sample (3.3% of data are written to tape even if the filter would reject it) is removed to homogenise the data. For runs which were taken in "mark-only" mode, the original filter decision (extracted from the event data) is applied. For all other runs, the filter decision is approximated using a similar algorithm as in the online filter.

**Monte Carlo**: The filter decision is approximated using a similar algorithm as in the online filter.

**Rationale**: All 2004 hadron analysis of Diff1 trigger data should use this function both for RD and MC to achieve best conformance between RD and MC.

The implementation is based on a function kindly provided by Quirin Weitzel.

**Parameters**:

- **e** the event

  - **tmask** provides a trigger mask that determines whether online filter cut is applied or not.
    This is mainly useful for MC data (which per se doesn’t have a trigger mask). The default (-1) is to use the trigger mask from the event data.

  - **force** enforce simulation, even if the run contains original online filter information that could be used, and even if the run does not belong to the 2004 hadron pilot run (both conditions apply to RD only)
References: PaM::Track::HitMap(), PaTrack::HitsFound(), PaEvent::IsMC(), PaEvent::NM::Track(), PaEvent::NTrack(), PaEvent::OnlFlt::Accepted(), PaEvent::RunNum(), PaEvent::TrigMask(), PaEvent::vM::Track(), and PaEvent::vTrack().

The documentation for this class was generated from the following files:

- lib/PaTrigger.h
- lib/PaTrigger.cc
6.27 PaVertex Class Reference

Vertex.
#include <PaVertex.h>

Public Member Functions

- float X () const
  Vertex X position.

- float Y () const
  Vertex Y position.

- float Z () const
  Vertex Z position.

- float Pos (int i) const
  Vertex position X,Y,Z (i=0,1,2).

- float Cov (int i) const
  Vertex position covariance matrix (3x3).

- float Chi2 () const
  \( \chi^2 \) of vertex fit

- int Ndf () const
  Number of degrees of freedom in vertex fit.

- int InParticle () const
  index (in the PaEvent::nParticle()) of incoming particle

- int NOutParticles () const
  N outgoing particles.

- int iOutParticle (int i) const
  index (in the PaEvent::nParticle()) of outgoing particle \# i (out of NOutParticles() (p 102))

- bool IsPrimary () const
  True if the vertex is primary.

- bool IsBestPrimary () const
  True if the vertex was tagged in CORAL as "best primary" (CsVertex::isBestVertex() function).

- int iMuPrimCoral () const
  index of mu' (in the PaEvent::nParticle()) or -1 if mu' is not found (based on CORAL).

- int iOutMuPrimCoral () const
  mu' position in outgoing particles list or -1 if mu' is not found (based on CORAL). For description see presentation on a Analysis Meeting by Yann Balter from May 2009.
• int iMuPrim (bool checkYokeSM2=true, bool reject2MuEvents=true, bool checkCanBeMuon=true, bool checkHodos=true, double XX0=15.0) const

  index of mu' (in the PaEvent::nParticle()) or -1 if mu' is not found.

• int iOutMuPrim (bool checkYokeSM2=true, bool reject2MuEvents=true, bool checkCanBeMuon=true, bool checkHodos=true, double XX0=15.0) const

  mu' position in outgoing particles list or -1 if mu' is not found.

• int nMuPrim () const

  number of identified mu' in the vertex

• bool MuPrimCalled () const

  has the mu' identification been executed?

6.27.1 Detailed Description

Vertex. Output of CORAL vertex package.

If the vertex contains beam particle (as incoming), such vertex is called primary (other vertexes are called secondary).

Warning:

• event may contain no primary vertex or more then 1 primary vertex.
• there were no cuts applied on "primary"-"secondary" space separation. I.e. position of secondary vertex may coincide with primary or even be more upstream.
• there were no cuts applied on number of shared particles. E.g. all particles of secondary vertex may also belongs to primary vertex.
• only 2-particle secondary vertexes are reconstructed.

Author:

Sergei.Gerassimov@cern.ch

6.27.2 Member Function Documentation

6.27.2.1 float PaVertex::Cov (int i) const [inline]

Vertex position covariance matrix (3x3).

• i=0,1,..,5 (elements of lower triangle of the matrix):

  \[
  \begin{pmatrix}
  X & Y & Z \\
  X & 0 & - & - \\
  Y & 1 & 2 & - \\
  Z & 3 & 4 & 5 \\
  \end{pmatrix}
  \]
6.27.2.2  int PaVertex::iMuPrim (bool checkYokeSM2 = true,  bool reject2MuEvents = true,  bool checkCanBeMuon = true,  bool checkHodos = true, double XX0 = 15.0) const

index of mu' (in the PaEvent::vParticle()) or -1 if mu' is not found. Mu' candidates are searched only in Primary Vertices. A mu' candidate is a track that:

- Has passed >30XX0
- Has Zlast after the Muon Filter 1
- Has the same sign as the beam particle (checked based on SM2 polarisation)
- Belongs to a PV that does not contain other outgoing muon tracks (mu track passes 3 previous requirements)
- It’s extrapolation goes through the active area of a pair of hodoscopes that gave the trigger
- It does not cross the yoke of SM2
- It belongs to an event that does not contain a track that goes through the hole in the absorber system Part of the checks can be turned off via options.

For detailed description see presentation on Analysis Meeting by Konrad Klimaszewski from November 2009.

Parameters:

checkYokeSM2 should mu’ candidates crossing SM2 yoke be rejected
reject2MuEvents should events with >=2 outgoing muon tracks
checkCanBeMuon should events with tracks pointing to the absorbers hole be rejected
checkHodos should mu’ candidates be required to cross active area of trigger hodoscope pair corresponding to one of the fired triggers

Warning:

In the case of few mu’ in the vertex (if reject2MuEvents==false), it returns index of muon with bigger momentum.

Author:

Konrad.Klimaszewski@cern.ch

References PaTrack::CanBeMuon(), PaTrack::CrossYokeSM2(), PaSetup::Detector(), PaField::getMagInfo(), PaTrack::HasMom(), PaSetup::iDetFirst(), iOutParticle(), IsPrimary(), PaParticle::iTrack(), NOutParticles(), PaTrack::PointsHodoscopes(), PaTrack::Q(), PaTrack::qP(), Phast::Ref(), PaSetup::Ref(), PaEvent::RunNum(), PaEvent::vParticle(), PaEvent::vTrack(), PaTrack::XX0(), PaDetect::Z(), and PaTrack::ZLast().

Referenced by iOutMuPrim(), PaParticle::IsMuPrim(), and nMuPrim().

6.27.2.3  int PaVertex::iMuPrimCoral () const [inline]

index of mu’ (in the PaEvent::vParticle()) or -1 if mu’ is not found (based on CORAL). For description see presentation on a Analysis Meeting by Yann Bedfer from May 2009.

Warning:

In the case of few mu’ in the vertex, it returns index of muon with bigger momentum.
6.27.2.4 int PaVertex::iOutMuPrim (bool checkYokeSM2 = true, bool reject2MuEvents = true, bool checkCanBeMuon = true, bool checkHodos = true, double XX0 = 15.0) const

mu’ position in outgoing particles list or -1 if mu’ is not found. For description see presentation on Analysis Meeting by Konrad Klimaszewski from November 2009.

Parameters:
- checkYokeSM2 should mu’ candidates crossing SM2 yoke be rejected
- reject2MuEvents should events with >=2 outgoing muon tracks
- checkCanBeMuon should events with tracks pointing to the absorbers hole be rejected
- checkHodos should mu’ candidates be required to cross active area of trigger hodoscope pair corresponding to one of the fired triggers

Warning:
In the case of few mu’ in the vertex (if reject2MuEvents==false), it returns index of muon with bigger momentum.

Author:
Konrad.Klimaszewski@cern.ch

References iMuPrim().

6.27.2.5 int PaVertex::iOutMuPrimCoral () const  [inline]

mu’ position in outgoing particles list or -1 if mu’ is not found (based on CORAL). For description see presentation on Analysis Meeting by Yann Bedfer from May 2009.

Warning:
In the case of few mu’ in the vertex, it returns position of muon with bigger momentum.

6.27.2.6 int PaVertex::nMuPrim () const  [inline]

number of identified mu’ in the vertex

Warning:
if mu’ identification was not executed the PaVertex::iMuPrim() (p. 104) method will be called with default options.

References iMuPrim(), and MuPrimCalled().

The documentation for this class was generated from the following files:
- lib/PaVertex.h
- lib/PaVertex.cc
- lib/PaVertexMuPrime.cc
6.28 Phast Class Reference

General purpose interface.

#include <Phast.h>

Public Member Functions

- float IntegratedBeamFlux ()
  calculate estimation of total beam flux.

- float TotalBeamFlux ()
  calculate estimation of total beam flux.

- void PrintMinMaxTimeInSpill ()
  Print Min/Max time of events in spills.

- int UserFlag (int i) const
  flag for user’s code set my -U options ("i" corresponds to -U option sequential number: 0,1,2...)

- int NUserFlag () const
  number of user flags given by the command line option(s) -U

- string TextUserFlag (int i) const
  text flags for user’s code set my -T options ("i" corresponds to -T option sequential number: 0,1,2...)

- int NTextUserFlag () const
  number of text user flags given by the command line option(s) -T

- int NProcessedSpills ()
  Number of processed spills.

Static Public Member Functions

- static Phast & Ref ()
  Return const reference to this object.

Public Attributes

- vector<TTree *> mso_setup_trees
  pointers to output setup trees in the case of multiple streams output

- vector<TTree *> mso_event_trees
  pointers to output event trees in the case of multiple streams output

- TFile * in_file
pointer to input ROOT file

- TFile * out_file
  pointer to output ROOT file

- string out_file_name
  output ROOT file name

- TFile * h_file
  pointer to opened histogram file

- vector< string > mso_file_names
  output file names in the case of multiple streams output

- vector< TFile * > mso_files
  pointers to output files in the case of multiple streams output

- int NevTot
  total number of events in all opened files

- int NevProc
  number of processed events

- int NevOut
  number of events saved in output file(s)

- float UserFieldScale [3]
  User's field scaling factor for solenoid, SM1 and SM2 (used in PaField (p. 48)).

- bool next_file
  set to "true" to jump to next input file

- bool next_event
  set to "true" to jump to next event (skipping other UserEvent functions)

6.28.1 Detailed Description

General purpose interface. It’s place for various kind of counters, pointers etc. accessible everywhere in the code.

All data-members are public.

6.28.2 Member Function Documentation

6.28.2.1 float Phast::IntegratedBeamFlux ()
calculate estimation of total beam flux. Return number of beam particles, counted by SciFi scaler over sum of time intervals between 2 consecutive events if this time interval is not more than 0.1 sec. (see histogram "Phast_h1_02" in hist.root/PhastHist)
Double counting in case of group of events with overlapped spill-time intervals is avoided.
This function gives better estimation of flux for original (not filtered out) samples of mDST events
if events distributed over spill non-uniformly (grouped in time)

6.28.2.2 static Phast& Phast::Ref () [inline, static]

Return const reference to this object. Could be used for access to Phast (p. 106) from any place
in the code.

Example: Phast::Ref() (p. 108).NevProc - is number of processed events

Referenced by PaTrack:CanBeMuon(), PaTrack::FullKF(), PaField::getField(),
PaAlgo::GetRadiativeWeight(), PaVertex::iMuPrime(), PaParticle::IsMuPrime(), PaMaterialMaps::loadGDML(), PaTrack::PointsHodoscopes(), PaSetup::Print(), PaEvent::Print(),
PaDigit::Print(), and PaTrack::QuickKF().

6.28.2.3 float Phast::TotalBeamFlux ()

calculate estimation of total beam flux. Return number of beam particles, counted by SiFi scaler
as sum (over processed data spills) of differences between scaler counts from last-in-spill event and
first-in-spill.

Please note, this is only estimation, as completeness of input sample is not guaranteed.

E.g. if you have in your sample all events from 1 spill, this function will returns you N -
Scaler{last event} - Scaler{first event}.

Even if fraction of events inside spill will be skipped, nevertheless you will get the same result.

It is better to use this function for flux estimation in the case of reduced (filtered out) samples of
mDST events.

The documentation for this class was generated from the following files:

- lib/Phast.h
- lib/Phast.cc
Index

AddNoise
  PaTPar, 87
Azi
  PaTPar, 87
BeamKiller2of3
  PaTrigger, 99
Bocet
  PaAlgo, 16
CalibConst
  PaCalorimCell, 33
CanBeMuon
  PaTrack, 94
CedarPid
  PaPid, 73
Chi2tot
  PaTrack, 94
CombinomN
  PaAlgo, 17
Combinoops
  PaAlgo, 17
Cov
  PaVertex, 103
CovMatrix
  PaTPar, 87
CrossCells
  PaAlgo, 17
CrossYokeSM2
  PaTrack, 94
DaqDecoErrors
  PaEvent, 45
DetRef
  PaMChit, 57
Dip
  PaTPar, 87
DoMW1ID
  PaAlgo, 17
ExtrapNeutral
  PaTPar, 87
Extrapolate
  PaTPar, 87
  PaTrack, 94
FieldInteg
  PaTrack, 95
FindCDA
  PaTPar, 88
FullKF
  PaTrack, 95
GetBeamPol
  PaAlgo, 17
getdE
  PaMaterialMaps, 53
GetDepolarizationFactor
  PaAlgo, 18
getdEStraggling
  PaMaterialMaps, 53
GetDilutionFactor
  PaAlgo, 19
GetDYdilutionFactor
  PaAlgo, 19
GetDYtargetPolarization
  PaAlgo, 19
GetEcor
  PaCalorimCell, 33
getField
  PaField, 48
GetLike
  PaPid, 73
GetLikeToBackground
  PaPid, 74
GetMW1ScatMuon
  PaAlgo, 20
GetR
  PaAlgo, 20
GetRadiativeWeight
  PaAlgo, 20, 21
getRadLength
  PaMaterialMaps, 53
GetSmoothed
  PaTrack, 95, 96
GetTargetLocation
  PaAlgo, 22
GetWireNumber
  PaDetect, 38
HCal2
PaTrigger, 99
HitChi2
PaTPar, 88

iBeam
PaMCvertex, 62
iBestTrkRef
PaMCtrack, 60
iCell
PaCaloClus, 31
PaCalorimeter, 34
iCellMaxE
PaCaloClus, 31
iDet
PaHit, 51
PaMChit, 57
iGenParticle
PaMCtrack, 60
iMuPrim
PaVertex, 103
iMuPrimCoral
PaVertex, 104
InActive
PaDetect, 39
InMaterialMap
PaMaterialMaps, 54
Instance
PaTrigger, 100
InTarget
PaAlgo, 22
IntegratedBeamFlux
Phast, 107
iOutMuPrim
PaVertex, 104
iOutMuPrimCoral
PaVertex, 105
IsKoan
PaPid, 74
IsMuPrim
PaParticle, 69
IsMuPrimFast
PaParticle, 70
IsPileup
PaMCvertex, 63
IsPrimary
PaMCvertex, 63
iTrack
PaMCtrack, 60
iTrackOfOrigin
PaMCvertex, 63
IWire
PaDigit, 41
KeepOnly

PaTrack, 96
lib/ Directory Reference, 9
LikePid
PaPid, 74

Mass
PaPid, 75
MasterTriggerMask
PaEvent, 45
MCgen
PaEvent, 46, 47
mEdcor
PaCalorimCell, 33
Mom3
PaMCtrack, 60
mTdcor
PaCalorimCell, 33
MultiplicityCounter
PaTrigger, 100

Ndf
PaTrack, 96
NHitsInDetect
PaMCtrack, 60
NMChit
PaMCvertex, 63
nMuPrim
PaVertex, 105
NpadX
PaRichDet, 79
Npho
PaPid, 75
NphoPMT
PaPid, 75
NVertex
PaParticle, 70

OnlineFilter
PaTrigger, 100
operator()
PaTPar, 88

PaAlgo, 13
Boost, 16
CombMofN, 17
CombNloops, 17
CrossCells, 17
DoMW1ID, 17
GetBeamPol, 17
GetDepolarizationFactor, 18
GetDilutionFactor, 19
GetDYdilutionFactor, 19
GetDYtargetPolarization, 19
GetMW1ScatMuon, 20

---

Generated on Mon Aug 27 15:36:46 2018 for PHAST by Doxygen
GetR, 20
GetRadiativeWeight, 20, 21
GetTargetLocation, 22
InTarget, 22
Q2, 23
RKutta, 24
W2, 24, 25
xbj, 25, 26
Xf, 26, 27
PaCaloCellType, 28
PaCaloChls, 29
iCell, 31
iCellMaxE, 31
PaCalorimCell, 32
CalibConst, 33
GetEcor, 33
mEdcor, 33
mTdcor, 33
PaCalorimeter, 34
iCell, 34
PaDetect, 36
GetWireNumber, 38
InActive, 39
PaDigit, 40
IWire, 41
sMCreference, 41
vDigitInfo, 41
PaEvent, 42
DaqDecoErrors, 45
MasterTriggerMask, 45
MCgen, 46, 47
SubTriggerMask, 47
vMiscScalars, 47
PaField, 48
getField, 48
PaHit, 50
iDet, 51
PaMaterialMaps, 52
getdE, 53
getdEStraggling, 53
getRadLength, 53
InMaterialMap, 54
PaMCgen, 55
PaMChit, 56
DetRef, 57
iDet, 57
Time, 57
PaMTrack, 58
iBestTrkRef, 60
iGenParticle, 60
iTrack, 60
Mom3, 60
NHitsInDetect, 60
sAllMCChitRef, 60
SetBestTrkRef, 61
sMCChitRef, 61
PaMCvertex, 62
iBeam, 62
IsPileup, 63
IsPrimary, 63
iTrackOfOrigin, 63
NMCTrack, 63
PaMetaDB, 64
Ref, 64
TargetSpinZproj, 64
vNminusOne, 65
PaMtx, 66
Print, 67
PaParticle, 68
IsMuPrim, 69
IsMuPrimFast, 70
NVertex, 70
ParInVtx, 70
PID, 71
Q, 71
RMass, 71
PaPid, 72
CedarPid, 73
GetLike, 73
GetLikeToBackground, 74
IsKaon, 74
LikePid, 74
Mass, 75
Npho, 75
NphoPMT, 75
PMTRingTime, 75
SecondLike, 76
ThetaCh, 76
PaRich, 77
PaRichDet, 78
NpadX, 79
PitchX, 79
Xlow, 79
PaRichMirr, 80
ParInVtx
PaParticle, 70
PaSetup, 81
Ref, 82
TargetCenterZ, 82
TargetPolarizations, 83
vNminusOne, 83
PassedDetectorNames
PaTrack, 97
PaTPar, 84
AddNoise, 87
Azi, 87
CovMatrix, 87
Dip, 87
ExtrapNeutral, 87
Extrapolate, 87
FindCDA, 88
HitChi2, 88
operator(), 88
Phi, 89
Print, 89
Rotate, 89
Theta, 89
Update, 90
PaTrack, 91
CanBeMuon, 94
Chi2tot, 94
CrossYokeSM2, 94
Extrapolate, 94
FieldInteg, 95
FullKF, 95
GetSmoothed, 95, 96
KeepOnly, 96
Ndf, 96
PassedDetectorNames, 97
PointsHodoscopes, 97
Q, 97
QuickKF, 97
vTPar, 98
ZFirst, 98
ZLast, 98
PaTrigger, 99
BeamKiller2of3, 99
HCal2, 99
Instance, 100
MultiplicityCounter, 100
OnlineFilter, 100
PaVertex, 102
Cov, 103
iMuPrim, 103
iMuPrimCoral, 104
iOutMuPrim, 104
iOutMuPrimCoral, 105
nMuPrim, 105
Phast, 106
IntegratedBeamFlux, 107
Ref, 108
TotalBeamFlux, 108
Phi
PaTPar, 89
PID
PaParticle, 71
PitchX
PaRichDet, 79
PMTRingTime
PaPid, 75
PointsHodoscopes
PaTrack, 97
Print
PaMtx, 67
PaTPar, 89
Q
PaParticle, 71
PaTrack, 97
Q2
PaAlgo, 23
QuickKF
PaTrack, 97
Ref
PaMetaDB, 64
PaSetup, 82
Phast, 108
RKutta
PaAlgo, 24
RMass
PaParticle, 71
Rotate
PaTPar, 89
sAllMChitRef
PaMtrack, 60
SecondLike
PaPid, 76
SetBestTrkRef
PaMtrack, 61
sMChitRef
PaMtrack, 61
sMReference
PaDigit, 41
src / Directory Reference, 12
SubTriggerMask
PaEvent, 47
TargetCenterZ
PaSetup, 82
TargetPolarizations
PaSetup, 83
TargetSpinZproj
PaMetaDB, 64
Theta
PaTPar, 89
Theta_Ch
PaPid, 76
Time
PaMChit, 57
TotalBeamFlux
Phast, 108
Update
PaTPar, 90
vDigInfo
  PaDigit, 41
vMiscScalers
  PaEvent, 47
vNminusOne
  PaMetaDB, 65
  PaSetup, 83
vTPar
  PaTrack, 98

W2
  PaAlgo, 24, 25

xbj
  PaAlgo, 25, 26

Xf
  PaAlgo, 26, 27

Xlow
  PaRichDet, 79

ZFirst
  PaTrack, 98

ZLast
  PaTrack, 98