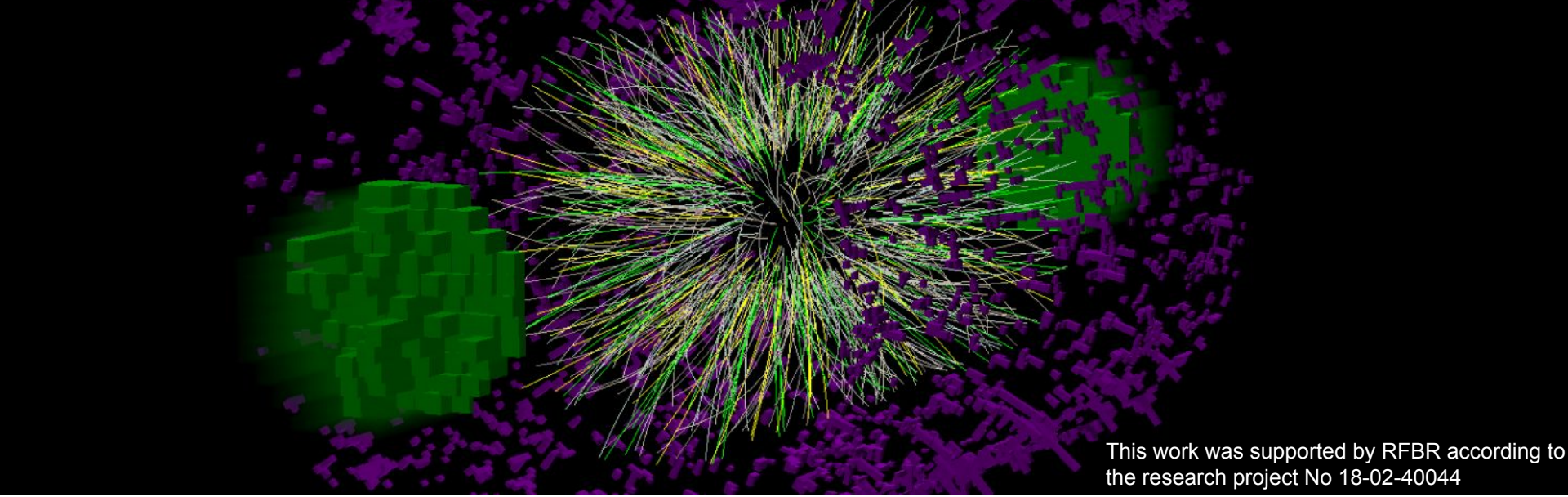


This work was supported by RFBR according to the research project No 18-02-40044

The MpdMiniDst data format (Part 1)

Grigory Nigmatkulov[1] and Pavel Batyuk[2]

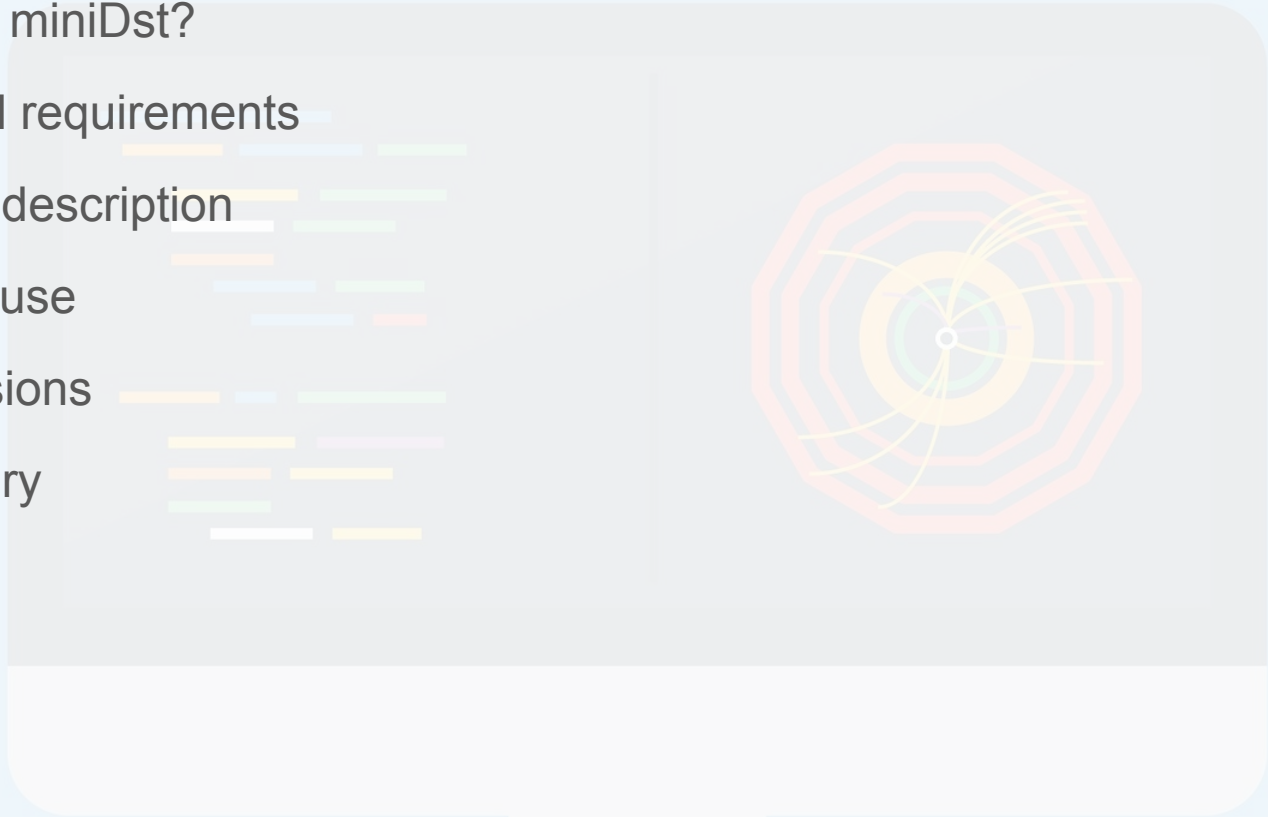
1. National Research Nuclear University MEPhI
2. Joint Institute for Nuclear Research

E-mail: nigmatkulov@gmail.com , ganigmatkulov@mephi.ru

July 16, 2020

Outline

- What is miniDst?
- General requirements
- Format description
- How to use
- Discussions
- Summary



What is miniDst?

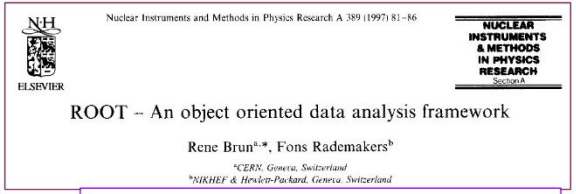
Simple answer: A data format for the MPD experiment

More detailed answer: A data format for MPD that stores an essential information from the trigger, detector, event, track, … levels.

Some general information about formation of the data formats is illustrated on the right or can be found here

DST *for reconstructed data*

The data formats may vary from an experiment to experiment, but a general (oversimplified) scheme of the experimental data flow:


Nuclear Instruments and Methods in Physics Research A 389 (1997) 81–86

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

ELSEVIER

ROOT – An object oriented data analysis framework

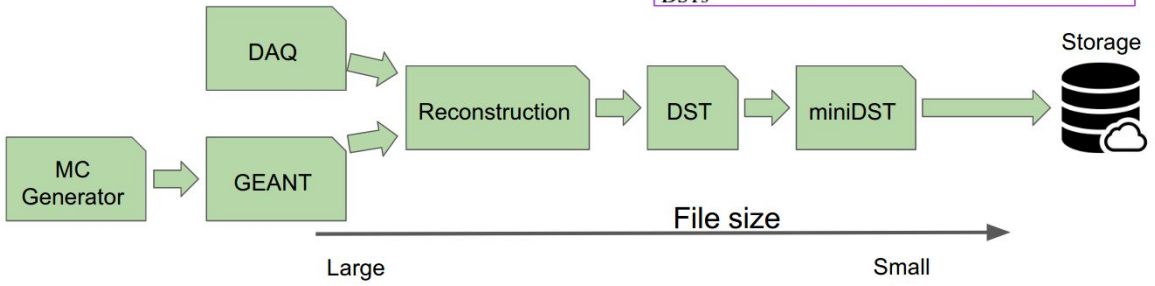
Rene Brun[a,*], Fons Rademakers[b]

[a]CERN, Geneva, Switzerland

[b]NIKHEF & Hewlett-Packard, Geneva, Switzerland

5. The ROOT Trees

For many years, the data flow model in HEP has been: Raw Data Tapes → Data Summary Tapes → Mini/Micro DSTs



The file size of the current MpdDst is ~1.3 MB/event

- ★ As small as possible
  - Small file size
  - Only vital variables for all analyses (discussions with PWGs are important)
- ★ MUST be implemented in MpdRoot
- ★ (Should also work) Independent on the MPD software
  - Must work on any computer farm or laptop with vanilla ROOT 5 or 6
  - Works on various OSs (Linux, MacOS, Windows)
  - Only simple (native) data types (int, float, ...)
  - Should be easily compiled with Makefile or CMake
- ★ Easy/fast to produce
  - During the whole production chain (DAQ->(DST)->miniDst)
  - Or reproduced for the larger format (DST->miniDst)
- ★ Any modification **must be** done via one person to avoid clashes

Format description

List of currently implemented classes (MpdMini**ClassName**):

Event - information about general event properties
Track - reconstructed track parameters
TrackCovMatrix - covariance matrix of the *global* track
BTofHit - barrel Time-Of-Flight hit information
BTofPidTraits - information about TOF-matched track
BECalCluster - information about ECal clusters
FHCalHit - information about FHCal hit
McEvent - Monte Carlo event properties
McTrack - Monte Carlo track information
DstReader - does all routing job and allows one to read miniDst
Helix and **PhysicalHelix** - to work with helix trajectories (for V^0, Kink, Xi reconstruction)
Makefile - to compile in a standalone mode
minidst_env.sh - shell script to setup the environment for standalone mode
PhysicalConstants.h and **SystemOfUnits.h** - helper headers

All classes documented in doxygen style

MpdMiniEvent

```
/// Run number (or runId)
Int_t    fRunId;
/// Event ID
Int_t    fEventId;
/// Fill number
UShort_t fFillId;
/// Number of bunch crossing
UChar_t  fBunchCrossId;
/// Magnetic field strength (kG)
Float_t  fBField;
/// GMT time
Int_t    fTime;
/// Primary vertex position X
Float_t fPrimaryVertexX;
/// Primary vertex position Y
Float_t fPrimaryVertexY;
/// Primary vertex position Z
Float_t fPrimaryVertexZ;
/// Primary vertex position error X
Float_t fPrimaryVertexErrorX;
/// Primary vertex position error Y
Float_t fPrimaryVertexErrorY;
/// Primary vertex position error Z
Float_t fPrimaryVertexErrorZ;
```

```
/// Number of ECal-matched tracks
UShort_t fNBECalMatch;
/// Number of TOF-matched tracks
UShort_t fNBTOFMatch;
/// List of triggers that were fired in the current event
std::vector<unsigned int> fTriggerIds;
/// Reference multiplicity estimated by global tracks
UShort_t fGRefMult;
/// Total number of global tracks
UShort_t fNumberOfGlobalTracks;
/// Total hit multiplicity in TOF trays
UShort_t fBTofTrayMultiplicity;
/// FHCal coincidence rate
UInt_t   fFHCalX;
/// Background rate
Float_t  fBackgroundRate;
/// East FHCal rate
Float_t  fFHCalEastRate;
/// West FHCal rate
Float_t  fFHCalWestRate;
/// Energy deposit in each of 90 towers
Float16_t fFHCalEnergyDeposit[90];
```

```
/// Reference multiplicity (-0.5<eta<0.5)
UShort_t fRefMultNeg;
/// Reference multiplicity (-0.5<eta<0.5)
UShort_t fRefMultPos;
/// Reference multiplicity (refMult2)
UShort_t fRefMult2NegEast;
/// Reference multiplicity (refMult2)
UShort_t fRefMult2PosEast;
/// Reference multiplicity (refMult2)
UShort_t fRefMult2NegWest;
/// Reference multiplicity (refMult2)
UShort_t fRefMult2PosWest;
/// TPC refMultHalf (eta<0)
UShort_t fRefMultHalfNegEast;
/// TPC refMultHalf (eta<0)
UShort_t fRefMultHalfPosEast;
/// TPC refMultHalf (eta>0)
UShort_t fRefMultHalfNegWest;
/// TPC refMultHalf (eta>0)
UShort_t fRefMultHalfPosWest;
```

And some more ...

MpdMiniTrack

```
/// Unique track ID
UShort_t fId;
/// Chi2 of the track
UShort_t fChi2;
/// Px momentum (GeV/c) of the primary track
Float_t fPMomentumX;
/// Py momentum (GeV/c) of the primary track
Float_t fPMomentumY;
/// Pz momentum (GeV/c) of the primary track
Float_t fPMomentumZ;
/// Px component of the momentum (GeV/c)
Float_t fGMomentumX;
/// Py component of the momentum (GeV/c)
Float_t fGMomentumY;
/// Pz component of the momentum (GeV/c)
Float_t fGMomentumZ;
/// Track origin x in cm (at DCAx to pvtx)
Float_t fOriginX;
/// Track origin y in cm (at DCAy to pvtx)
Float_t fOriginY;
/// Track origin z in cm (DCAy to pvtx)
Float_t fOriginZ;
/// dE/dx in arbitrary units
Float16_t fDedx;
```

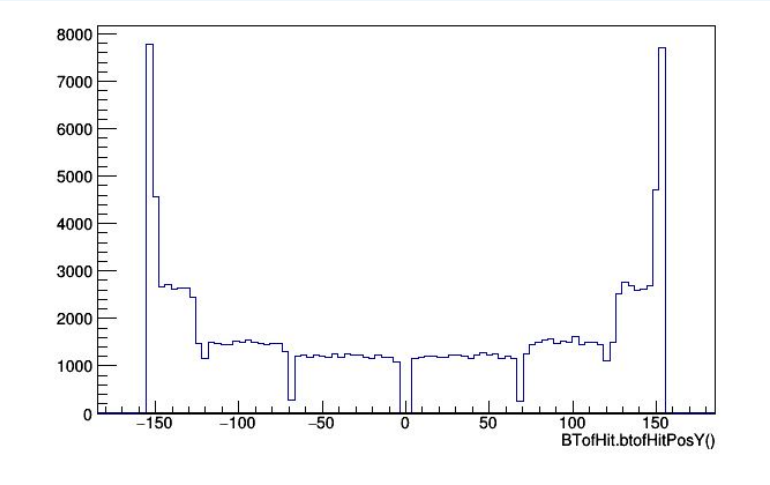
```
/// Charge * nHits
Char_t   fNHits;
/// nSigma(pion)
Short_t  fNSigmaPion;
/// nSigma(kaon)
Short_t  fNSigmaKaon;
/// nSigma(proton)
Short_t  fNSigmaProton;
/// nSigma(electron)
Short_t  fNSigmaElectron;
/// Track hit map
ULong64_t fHitMap;
///Index of the barrel ECal cluster in the
///event (-1 if no match)
Short_t  fBECalClusterIndex;
///Index of the BTOF pidTratis in the event
///(-1 if no match)
Short_t  fBTofPidTraitsIndex;
///Index of miniMcTrack that corresponds to
///miniTrack (aka IdTruth). (-1 if no match)
Short_t fMcTrackId;
```

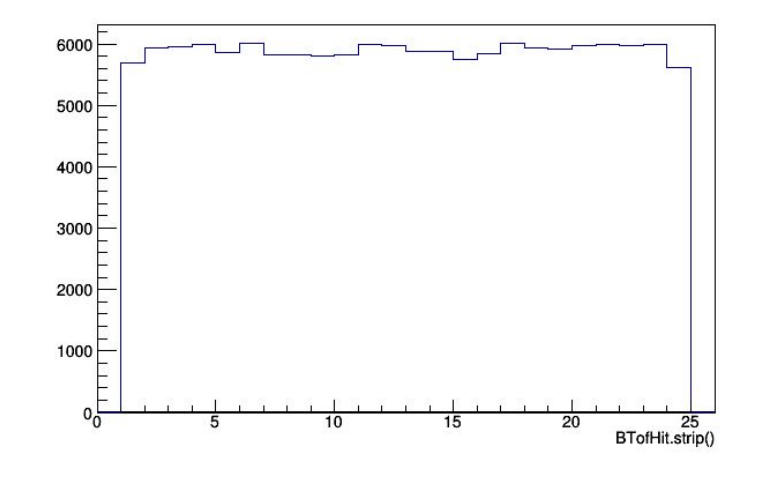
MpdMiniTrackCovMatrix

```
/// Signed impact parameter; Signed in such a way that:
/// x =  -impact*sin(Psi), y = impact*cos(Psi)
Float16_t mImp;
/// Z position of the track fitted to (0,0,z)
Float16_t mZ;
/// Psi angle of the track
Float16_t mPsi;
/// Pti of the track ( 1/p_T )
Float16_t mPti;
/// Tangent of the track momentum dip angle
Float16_t mTan;
/// Curvature
Float16_t mCurv;
/// Diagonal elements
Float16_t mSigma[5];
/// Off-diagonal elements
Float16_t mCorr[10];
```

MpdMiniBTofHit

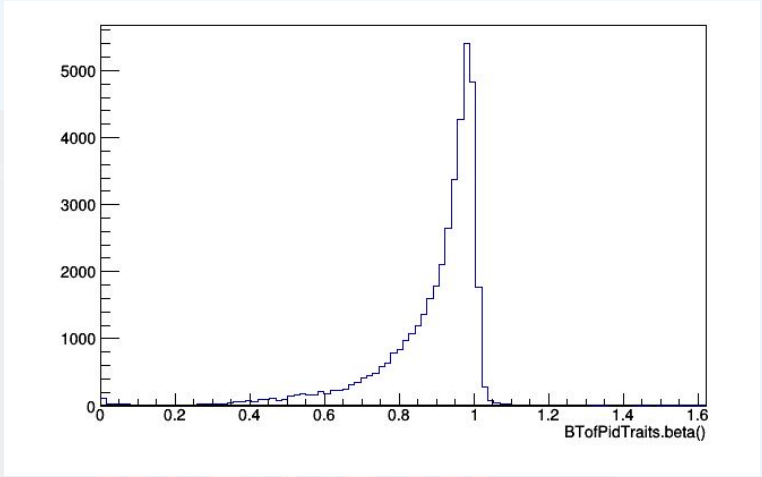
```
/// Det ID (see above how to get det. elements)
Int_t fDetectorID;
/// Hit position projected on X plane
Short_t fBTofHitPosX;
/// Hit position projected on Y plane
Short_t fBTofHitPosY;
/// Hit position projected on Z plane
Short_t fBTofHitPosZ;
/// Time since the event start [ns]
Float_t fTime;
```

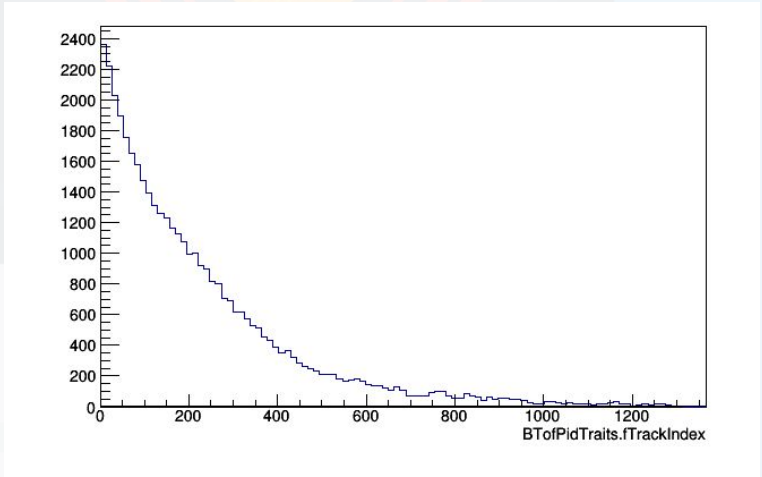




MpdMiniBTofPidTraits

```
/// Index to the associated track in the event (-1 if no matching)
Short_t   fTrackIndex;
/// Index to the associated hit in the event (-1 if no matching)
Short_t   fHitIndex;
/// Beta
UShort_t  fBTofBeta;
/// Px of the track (GeV/c)
Float16_t fPx;
/// Py of the track (GeV/c)
Float16_t fPy;
/// Pz of the track (GeV/c)
Float16_t fPz;
```

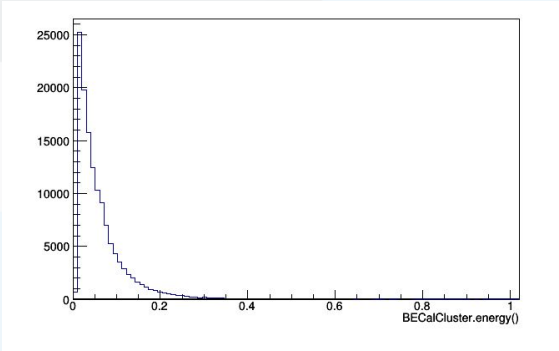




MpdMiniBECalCluster

```
/// Digit ID (cellId)
std::vector<UShort_t> fDigitIds;
/// Deposited energy per digit
std::vector<Float16_t> fDigitEDeps;
/// Index of the MC track
std::vector<UShort_t> fMcTrackIds;
/// Energy deposited by MC track
std::vector<Float16_t> fMcTrackEDeps;
 /// Energy deposited in the cluster (GeV)
Float16_t fEnergy;
/// Cluster energy core (from MpdEmcClusterKI)
Float16_t fEcore;
/// Cluster energy core (from MpdEmcClusterKI)
Float16_t fEcore1p;
/// Cluster energy core (from MpdEmcClusterKI)
Float16_t fEcore2p;
 /// Cluster average time (ns)
Float16_t fTime;
/// X coordinate of the cluster in global system
Float16_t fX;
/// Y coordinate of the cluster in global system
Float16_t fY;
/// Z coordinate of the cluster in global system
Float16_t fZ;
```

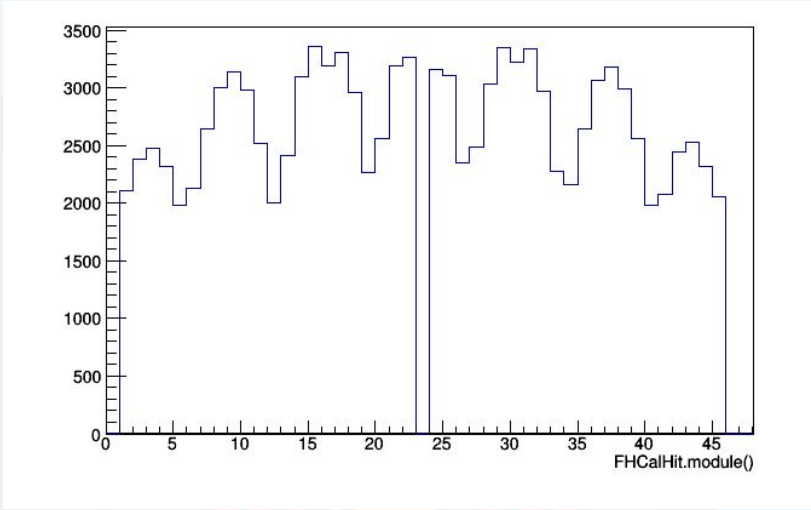
```
/// Distance to closest track in phi
Float16_t fdPhi;
/// Distance to closest track in z
Float16_t fdZ;
/// Index of closest matched track if any
/// (-1 means no match)
Short_t fTrackId;
/// Smaller dispersion axis
Float16_t fLambda1;
/// Larger dispersion axis
Float16_t fLambda2;
/// Chi2 of a fit with EM shape (chi2 * 100)
UShort_t fChi2;
/// Number of local maxima in parent
/// cluster before unfolding
UChar_t fNExLM;
```

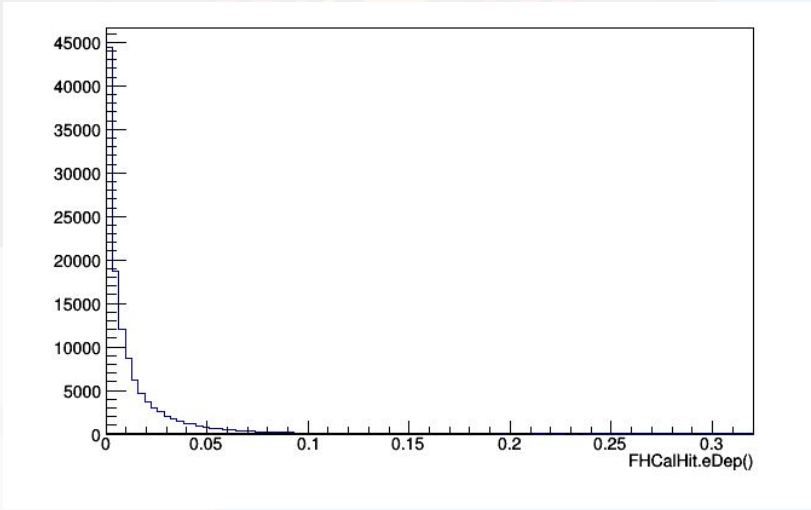


MpdMiniFHCalHit

```
/// Hit id
UShort_t fId;

/// Energy deposition
Float16_t fEDep;
```

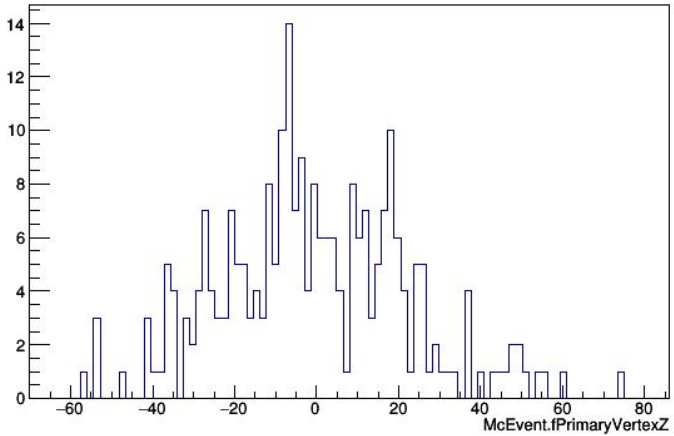


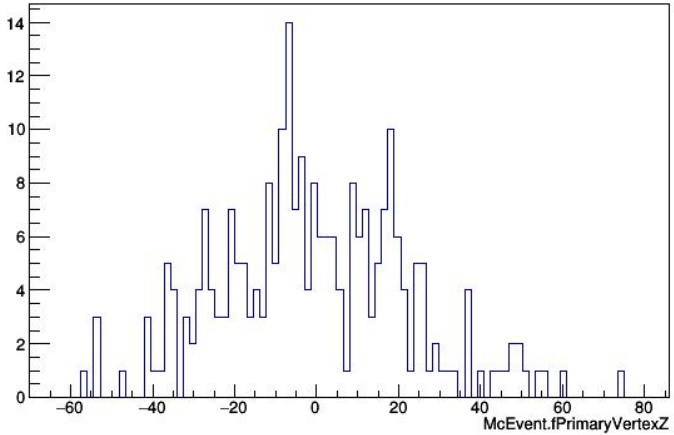


MpdMiniMcEvent

```
/// Unique run ID
UInt_t          fRunId;
/// Unique event ID
UInt_t          fEventId;
/// Reaction plane angle (rad)
Float_t   fReactionPlaneAngle;
/// Impact parameter (fm)
Float16_t fImpactParameter;
/// Number of participants
Short_t  fNpart;
/// Number of binary collisions
Short_t  fNcoll;

/// Primary vertex x position (cm)
Float_t  fPrimaryVertexX;
/// Primary vertex y position (cm)
Float_t  fPrimaryVertexY;
/// Primary vertex z position (cm)
Float_t  fPrimaryVertexZ;
/// Event time (ns)
Float_t   fTime;
```





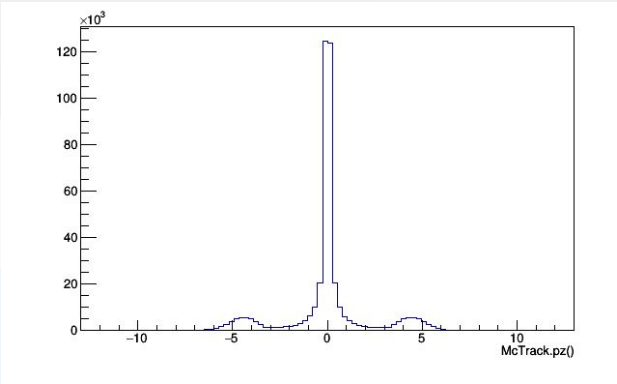
MpdMiniMcTrack

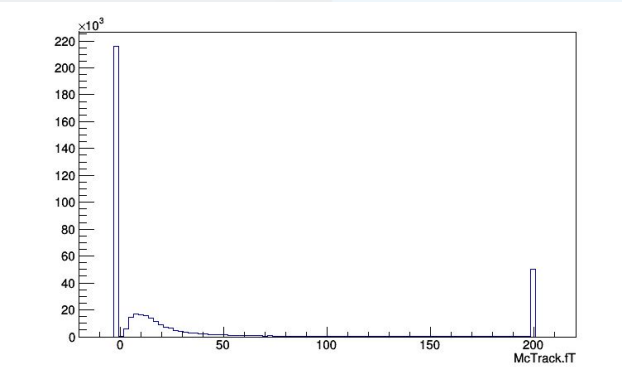
```
/// Unique track ID
UShort_t fId;
/// PDG code
Int_t  fPdgId;
/// The first and the second daughter particle
/// indices (-1 if not decayed)
UShort_t fChild[2];
/// Px (GeV/c)
Float_t  fPx;
/// Py (GeV/c)
Float_t  fPy;
/// Pz (GeV/c)
Float_t  fPz;
/// Energy from the generator (GeV/c)
Float_t  fEnergy;
/// Freeze-out x coordinate (fm)
Float_t  fX;
/// Freeze-out y coordinate (fm)
Float_t  fY;
/// Freeze-out z coordinate (fm)
Float_t  fZ;
/// Freeze-out t coordinate (fm/c)
Float_t  fT;
```

```
/// Status of the track
Char_t   fStatus;

/// Indices of the reconstructed global tracks
/// that were/ reconstructed from the current
/// Monte Carlo track
/// Empty when no tracks were reconstructed
/// from the MC track.
std::vector< UShort_t > fRecoTrackIds;

/// Is from generator
Bool_t   fIsFromGen;
```





Usage example

Path to the complete example in the MpdRoot:
macro/physical_analysis/miniDst/miniDstProcExample.C

Path to the Bi+Bi at $\sqrt{s_{NN}}$=9 GeV
(on lxpub.jinr.ru or hydra.jinr.ru):

lxpub: /eos/nica/mpd/dirac/mpd.nica.jinr/vo/mpd/data/MiniDST/dst-BiBi-09GeV-mp07-20-pwg3-250ev/BiBi/09.0GeV-0-14fm/UrQMD/BiBi-09GeV-mp07-20-pwg3-250ev-1/

hydra: /eos/eos.jinr.ru/nica/mpd/dirac/mpd.nica.jinr/vo/mpd/data/MiniDST/dst-BiBi-09GeV-mp07-20-pwg3-250ev/BiBi/09.0GeV-0-14fm/UrQMD/BiBi-09GeV-mp07-20-pwg3-250ev-1/

- It is possible to read either one file or a list of files
- One can read only specific branches
- Simple access to data

Very simple example

```
#include "Rtypes.h"
#include "TChain.h"
#include "TFile.h"
R__ADD_INCLUDE_PATH($VMCWORKDIR)
#include "macro/mpd/mpdloadlibs.C"

//_____
void miniExample(const Char_t* inFileName) {
  MpdMiniDstReader* miniDstReader = new MpdMiniDstReader(inFileName);
  // Initialize reader
  miniDstReader->Init();
  // One can specify branches to read
  miniDstReader->SetStatus("*", 0);                // Turn off all branches
  miniDstReader->SetStatus("Event*", 1);           // Turn on specific branch
  miniDstReader->SetStatus("Track*", 1);
  miniDstReader->SetStatus("BTofHit*", 1);
  miniDstReader->SetStatus("BTofPidTraits*", 1);
  miniDstReader->SetStatus("BECalCluster*", 1);
  miniDstReader->SetStatus("FHCalHit*", 1);
  miniDstReader->SetStatus("TrackCovMatrix*", 0); // Turn off specific branch
  miniDstReader->SetStatus("McEvent*", 1);
  miniDstReader->SetStatus("McTrack*", 1);
  Long64_t events2read = miniDstReader->chain()->GetEntries();
  for (Long64_t i = 0; i < events2read; i++) {
    Bool_t  isOk = miniDstReader->readMiniEvent( i );
    MpdMiniDst *dst = miniDstReader->miniDst();
    MpdMiniEvent *event = dst->event();
    Float_t z = event->primaryVertex().Z();
    // Retrieve number of reconstructed tracks
    Int_t nGTracks = dst->numberOfTracks();
    // Track loop
    for (Int_t j = 0; j < nGTracks; j++) {
      MpdMiniTrack *miniTrack = dst->track(j);
      Float_t ptot = miniTrack->gMom().Mag();
    }
    // Loop over barrel TOF hits
    for (Int_t j = 0; j < dst->numberOfBTofHits(); j++ ) {
      // Retrieve j-th hit information
      MpdMiniBTofHit *btofHit = dst->btofHit( j );
      // x position of hit
      Float_t xPosition = btofHit->btofHitPosX();
    }
  }
  // Finalize miniDst reader
  miniDstReader->Finish();
}
```

Summary for Part1

Takeaway messages

- The format that may satisfy most physics needs is ready to use and implemented in MpdRoot
- Contains most of the detector subsystems
- Everyone is welcome to test
- Feedback is important

Outlook

- Discussions on information that is missing
- Clarification of some parameter estimations
- How to process data in a standalone mode (without MpdRoot)