HIPO4 + CLAS12TOOL + ROOT

12/06/19

Clas12Tool

Data Analysis Tools for HIPO data format.

Seperate code is provided for hipo3 and hipo4 format. The corresponding directories, libraries and binaries have an additional 3 or 4 to distinguish. Users are responsible for using the correct format for their hipo files.

Examples are given for running in interactive ROOT sessions and ROOT-Jupyter notebooks.

HipoX -> Clas12BanksX -> Clas12Root

The Hipo c++ reader library can be used independent of specific banks and ROOT, but depends on Hipo.

The Clas12Banks implementation can be used independent of ROOT, although currently ROOT dictionaries are created for the classes via cmake (this could be removed). This defines the specific CLAS12 DST banks and provides an interface to the data.

For actual Clas12Banks definitions see HIPO4 DSTs

The Clas12Root package depends on both Hipo and Clas12Banks. This provides ROOT-like analysis tools for operating on clas12 hipo DSTs.

```
HipoDraw
HipoTreeMaker
HipoProof
```

To Download

git clone --recurse-submodules https://github.com/dglazier/Clas12Tool.git

git checkout dst4

cd Clas12Tool

TEMPORARY will move to master

To setup Run ROOT

for cshrc

setenv CLAS12TOOL $PWD (the actual path can be added in your bashrc or tchrc)

setenv PATH "$PATH":"$CLAS12TOOL/bin"

or for bash

export CLAS12TOOL=$PWD

export PATH="$PATH":"$CLAS12TOOL/bin"

To install (either Hipo3 or Hipo4)

installHipo3 and/or installHipo4

New clas12reader class

```
//reader
hipo::reader     _reader;
hipo::event      _event;

//DST banks
head_ptr  _bhead;
par_ptr _bparts;
mcpar_ptr _bmcparts;
covmat_ptr _bcovmat;
cal_ptr  _bcal;
scint_ptr _bscint;
trck_ptr _btrck;
traj_ptr _btraj;
cher_ptr _bcher;
ft_ptr _bft;
vtp_ptr _bvtp;
scaler_ptr _bscal;

//Detector region vectors,
//each particle in an event will have
//one associated
std::vector<region_fdet_ptr> _rfdets;
std::vector<region_cdet_ptr> _rcdets;
std::vector<region_ft_ptr> _rfts;
std::vector<region_part_ptr> _detParticles;
```

region_particles
= particle +
associated detector info

Seperate class for regions:
FT, FD, CD

Ex 0 Plotting an item from any bank (HIPO4)

This is faster than the particle draw as it only requires the reading of 1 bank.

```
clas12root4

BankHist bankDraw("/WHERE/IS/MY/HIPO/file.hipo");
bankDraw.Hist1D("REC::Particle::Pz",100,0,10,"")->Draw()
bankDraw.Hist1D("REC::Scintillator::Time",1000,0,200,"")->Draw()
```

You can group histograms together for lazy execution if they all come from the same bank.

```
bankDraw.Hist1D("REC::CovMat::C11",100,0,1,"")
bankDraw.Hist1D("REC::CovMat::C22",100,0,1,"")
bankDraw.Hist1D("REC::CovMat::C33",100,0,1,"")
bankDraw.Hist1D("REC::CovMat::C44",100,0,1,"")
bankDraw.Hist1D("REC::CovMat::C55",100,0,1,"")->Draw("(3x2)")
```

Creating your own analysis loop

```
//create the event reader
 clas12reader c12("file.hipo");

 //Add some event Pid based selections
 //////////c12.AddAtLeastPid(211,1); //at least 1 pi+
 c12.addExactPid(11,1);    //exactly 1 electron
 c12.addExactPid(211,1);    //exactly 1 pi+
 c12.addExactPid(-211,1);    //exactly 1 pi-
 c12.addExactPid(2212,1);    //exactly 1 proton
 c12.addExactPid(22,2);    //exactly 2 gamma
 //////c12.addZeroOfRestPid();  //nothing else
 //////c12.useFTBased(); //and use the Pids from RECFT

 while(c12.next()==true){
   c12.event()->getStartTime(); //hipo4
  // c12.head()->getStartTime(); //hipo3
   //Loop over all particles to see how to access detector info.

   for(auto& p : c12.getDetParticles()){
    //  get predefined selected information
    p->getTime();
    p->getDetEnergy();
    p->getDeltaEnergy();
```

i.e. for FD track you get time from FTOF1B,FTOF1A,FTOF2,PCAL In order of precedence

```
//create the clas12 event reader
clas12reader c12event(inputFile);

while(c12event.next()==true){//loop over all events
  //loop over particles
  for(auto& p : c12event.getDetParticles()){

    int   pid = p->par()->getPid();
    float  px = p->par()->getPx();
    float  py = p->par()->getPx();
    float  pz = p->par()->getPx();
  }
}
```

```
//create the clas12 event reader
clas12reader c12event(inputFile);

while(c12event.next()==true){//loop over all events
  //loop over particles
  for(auto& p : c12event.getDetParticles()){

    int   pid = p->par()->getPid();
    float  px = p->par()->getPx();
    float  py = p->par()->getPx();
    float  pz = p->par()->getPx();
  }
}
```

Only need to loop over Particle!

```
//  get predefined selected information
p->getTime();
p->getDetEnergy();
// get any detector information (if exists for this particle)
// there should be a get function for any entry in the bank
switch(p->getRegion()) {
case FD :
  p->cal(PCAL)->getEnergy();
  p->cal(ECIN)->getEnergy();
  p->cal(ECOUT)->getEnergy();
  p->sci(FTOF1A)->getEnergy();
  p->sci(FTOF1B)->getEnergy();
  p->sci(FTOF2)->getEnergy();
  p->trk(DC)->getSector();
  p->che(HTCC)->getNphe();
  p->che(LTCC)->getNphe();
  p->traj(TRAJ_HTCC)->getX();
  break;
case FT :
  p->ft(FTCAL)->getEnergy();
  p->ft(FTHODO)->getEnergy();
  break;
case CD:
  p->sci(CTOF)->getEnergy();
  p->sci(CND)->getEnergy();
  break;
}
```

Or analyse particular final states

```
// get particles by type
auto electrons=c12.getByID(11);
auto gammas=c12.getByID(22);
auto protons=c12.getByID(2212);
auto pips=c12.getByID(211);
auto pims=c12.getByID(-211);

if(electrons.size()==1 && gammas.size()==2 && protons.size()==1 &&
   pips.size()==1 &&pims.size() == 1){

  // set the particle momentum
  SetLorentzVector(el,electrons[0]);
  SetLorentzVector(pr,protons[0]);
  SetLorentzVector(g1,gammas[0]);
  SetLorentzVector(g2,gammas[1]);
  SetLorentzVector(pip,pips[0]);
  SetLorentzVector(pim,pims[0]);

  TLorentzVector miss=beam+target-el-pr-g1-g2-pip-pim;
  hmiss->Fill(miss.M2());
  TLorentzVector pi0 = g1+g2;
  hm2g->Fill(pi0.M());
  if(TMath::Abs(miss.M2())<0.5)hm2gCut->Fill(pi0.M());

  //could also get particle time etc. here too
  //Double_t eTime=electrons[0]->sci(FTOF1A)->getTime();
}
```

Ex 2 Drawing particle histograms from hipo files

```
particleDraw4 /WHERE/IS/MY/HIPO/file.hipo
```

Or chain together files with wildcard, note the ' '

```
particleDraw4 '/WHERE/IS/MY/HIPO/file_*.hipo'
```

You will get an interactive ROOT prompt where you can draw histograms:

```
ParticleHist [0] hists.Hist1D("P.P",100,0,10,"P.P")
ParticleHist [1] hists.Hist2D("P.P:P.DetEnergy",100,0,10,100,0,5,"P.P")->Draw("(2x1)")
```

Note you only have to call draw once, and then it only has to loop over the date once. The option (2x1) specifies the dimensions of the pads in the produced canvas, the parenthesis is required.

Remember at the end you can save all the histograms to file :

```
ParticleHist [0] hists.Save("HistFile.root");
```

There are predefined aliases for DST bank detector layers :

```
    ECIN. , ECOUT. , PCAL. , FTOF1A. , FTOF1B. , FTOF2. , CTOF. , CND1. , CND2. , CND3. , FTCAL. , FTHODO.
    e.g. ECIN.Energy  , HTCC.Nphe , DC.TrChi2 , CTOF.Time
```

The REC::Particle bank should be directly accessed with

```
    PBANK.
    e.g. PBANK.Pid , PBANK.Px
```

The FT based equaivalent PID varaibles can be accessed from the particle bank by

```
    e.g. PBANK.FTBPid , PBANK.FTBBeta
```

The region particle has derived quantities such as theta and phi as well as selected variabels for a particle for example time from a particular ToF layer. Note the order of precedence for the FD is TOF1B, TOF1A, TOF2, PCAL and DeltaEnergy is the corresponding timing detector energy. These should be accessed with

```
    P.
    e.g. P.Theta , P.P , P.Phi , P.Region , P.Time , P.DetEnergy , P.DeltaEnergy , P.Path , P.Pid , P.Calc
e.g. P.Region==FT,  P.Region==FD
```

For REC::EVNT use (adding FTB for RECFT::EVNT banks)

```
    e.g. EVNT.StartTime (Hipo3) EVNT4.StartTime (Hipo4) EVNT4.FTBStartTime (hipo4 from RECFT)
```

For Run::config (hipo4)

```
   e.g. RUN.Trigger
```

Ex 4 Filtering and Skimming into a ROOT ntuple (tree)

```
particleTree4  /WHERE/IS/MY/HIPO/file.hipo /OUTPUT/tree.root Ex4_TreeMaker.C
```

Or chain together files with wildcard, note the ''

```
particleTree4  '/WHERE/IS/MY/HIPO/file_*.hipo' /OUTPUT/tree.root Ex4_TreeMaker.C
```

The script $CLAS12ROOT/RunRoot/Ex4_TreeMaker.C defines which branches are to be written and what cuts to put on the event topology. You can copy and edit this file to do what you want. The branches should use the conventions above for accessing different bank items e.g.

```
treemaker.Branch("P.Time/F"); //create tree with time branch
treemaker.Branch("PBANK.Px/F"); //create tree with particle Px branch
```

You can perform some arithmetic and define a new branch e.g.

```
        treemaker.Branch("P.Time-EVNT4.StartTime/F","Time"); //branch name Time
treemaker.Branch("P.Time-EVNT4.FTBStartTime/F","FTBTime"); //branch name FTBTime


treemaker.AddExactPid(11,1); //filter events with exactly 1 e-
treemaker.AddAtLeastPid(211,1);//and at least 1 pi+
treemaker.AddZeroOfRestPid(); //and zero of any other particle type (default is any)
```

Ex 3 Using HipoSelector & PROOFLite

ONLY FOR HIPO4

This assumes you are aware of and understand the ROOT TSelector and PROOF scheme. See https://root.cern.ch/proof.

Create a HipoSelector (similar to tree->MakeSelector("mySelector");), using the executable makeHipoSelector :

```
makeHipoSelector mySelector
```

You should use some meaningful name rather than mySelector. Edit it to perform the tasks you would like. But use the ProcessEvent function instead of the Process function as you would in a TSelector. You can use the _c12 clas12reader object to access all the data as shown in Ex1_CLAS12Reader.C

e.g.

```
Bool_t HipoFileSelector::ProcessEvent(){

   _hist1->Fill(_c12->head()->getStartTime());
           return kTRUE;
}
```

To execute (note the + is important) :

```
clas12proof4 4 mySelector.C+ Ex3_ProofLite.C
```

```
Bool_t testSelector::ProcessEvent(){
  //Equivalent to TSelector Process
  //Fill in what you would like to do with
  //the data for each event....

  auto electrons=_c12->getByID(11);
  auto gammas=_c12->getByID(22);
  auto protons=_c12->getByID(2212);
  auto pips=_c12->getByID(211);
  auto pims=_c12->getByID(-211);

  if(electrons.size()==1 && gammas.size()==2 && protons.size()==1 &&
     pips.size()==1 &&pims.size() == 1){

    // set the particle momentum
    SetLorentzVector(_el,electrons[0]);
    SetLorentzVector(_pr,protons[0]);
    SetLorentzVector(_g1,gammas[0]);
    SetLorentzVector(_g2,gammas[1]);
    SetLorentzVector(_pip,pips[0]);
    SetLorentzVector(_pim,pims[0]);

    TLorentzVector miss=_beam+_target-_el-_pr-_g1-_g2-_pip-_pim;
    hmiss->Fill(miss.M2());
    TLorentzVector pi0 = _g1+_g2;
    hm2g->Fill(pi0.M());
    if(TMath::Abs(miss.M2())<0.5)hm2gCut->Fill(pi0.M());

    //could also get particle time etc. here too
    //Double_t eTime=electrons[0]->sci(FTOF1A)->getTime();
  }
  return kTRUE;
```