Subject: EMC mapper and other modifications
Posted by Dima Melnychuk on Fri, 26 Feb 2010 11:10:22 GMT
View Forum Message <> Reply to Message

Hi all,

I made some changes in PndEmcMapper class to make its usage more transparent.
Initial implementation of EmcMapper was a singleton, but it appeared to be not the best idea since theoretically different runs can be simulated with different geometry and correspondingly with different mappers.

So now PndEmcMapper is initialized with the Init() method and can be reinitialized if necessary with different input parameter. And mapper is accessed with static Instance() method which now does not have input parameters.

Second modification is that I removed from the mapper procedure to create neighboring elements for each crystal. Initial purpose of Mapper is to map to unique crystal ID two coordinates in (theta, phi) for barrel or (X,Y) for Endcaps, which are used in clusterisation algorithm. The creation of neighbors was extension of this functionality but probably not in proper place.

Initially neighbors for each crystal were filled in EmcMapper to check in clusterisation algorithm if one element is in neighbor list of another then they are neighbors. However it appeared to be more time consuming then the check which is used now in PndEmcTwoCoordIndex.cxx.

The difficulty with creation of neighbors was that it depended on geometry (gGeoManager) and this created difficulty with initialization of EmcMapper from RunTimeDataBase.

So here I come to another important point, i.e. initialization of EmcMapper from RunTimeDataBase. Some time ago Ola Biegun introduced functionality in PndEmc.cxx class to store mapper version in RTDB depending on chosen geometry. This will allow to avoid mismatching between geometry and mapper in emc reconstruction. However this functionality was not fully used by now.

So I removed mapper version from PndEmcDigiPar parameter container and it accessed everywhere in the code from PndEmcGeoPar which is filled in PndEmc::SetGeometryVersion(). So from now I would propose to use this method to set geometry for simulation

Emc->SetGeometryVersion(15);

instead of

Emc->SetGeometryFileNameTriple("emc_module125.dat","emc_module3new.root","emc_module4_StraightGeo24.4.root");

Important remark here is that in simulation macro the RunTimeDataBase should be initialized and especially output be set

```
FairRuntimeDb* rtdb = fRun->GetRuntimeDb();
FairParAsciiFileIo* parIo1 = new FairParAsciiFileIo();
```

```
  parIo1->open(emcDigiFile.Data(),"in");
  rtdb->setFirstInput(parIo1);
  Bool_t kParameterMerged=kTRUE;
  FairParRootFileIo* output=new   FairParRootFileIo(kParameterMerged);
  output->open(parFile);
  rtdb->setOutput(output);
```

before the call

```
Emc->SetGeometryVersion(15);
```

to store the mapper version properly.

In PndEmcGeoPar I introduced the method PndEmcGeoPar::InitEmcMapper() which is actually initialise PndEmcMapper from RTDB.

So the proper method to initialize EmcMapper in analysis macro would be (as it is implemented now in /macro/qa/emc/QAmacro_emc_3.C)

```
 FairRunAna *fRun= new FairRunAna();
 fRun->SetInputFile("sim_emc.root");
 fRun->SetOutputFile("dummy_out.root");

 FairRuntimeDb* rtdb = fRun->GetRuntimeDb();
 FairParRootFileIo* parInput1 = new FairParRootFileIo();
 parInput1->open("simparams.root");

   TString emcAsciiPar = gSystem->Getenv("VMCWORKDIR");
 emcAsciiPar += "/macro/params/";
 emcAsciiPar += "emc.par";

 FairParAsciiFileIo* parInput2 = new FairParAsciiFileIo();
 parInput2->open(emcAsciiPar.Data(),"in");

 rtdb->setFirstInput(parInput1);
 rtdb->setSecondInput(parInput2);

 PndEmcGeoPar *geoPar = (PndEmcGeoPar*) rtdb->getContainer("PndEmcGeoPar");
 fRun->Init();

 geoPar->InitEmcMapper();
```

however shortcut

```
 PndEmcMapper::Init(6);
```

will work if you know proper mapper version.

In addition I introduced small modification in EMC digitisation.
In PndEmcHitsToWaveform by now waveforms were produced in the crystals with hits and their neighbors. Now by default waveforms are not produced in neighbors of hited crystals

however I introduced a flag NoiseAllChannels in PndEmcDigiPar, which by default if off. When it is on waveforms are produced in all the channels of EMC and they will contain electronic noise. It will allow to study the effect of noise excess over the threshold. However this option is very time consuming.

I hope these modifications will not have unpredicted side effects.

I have updated macros in
/macro/emc/
/macro/qa/emc/
/macro/run/

so if there will be problems with other macros please look there for examples.

Best regards,

Dima