SPD Calibration: Status and Plans

A) Dedicated Calibration Runs

(Foreseen to run ~ once a week.)

Data optionally through DAQ *or* DCS

--> Offline CDB

--> DCS Config. DB

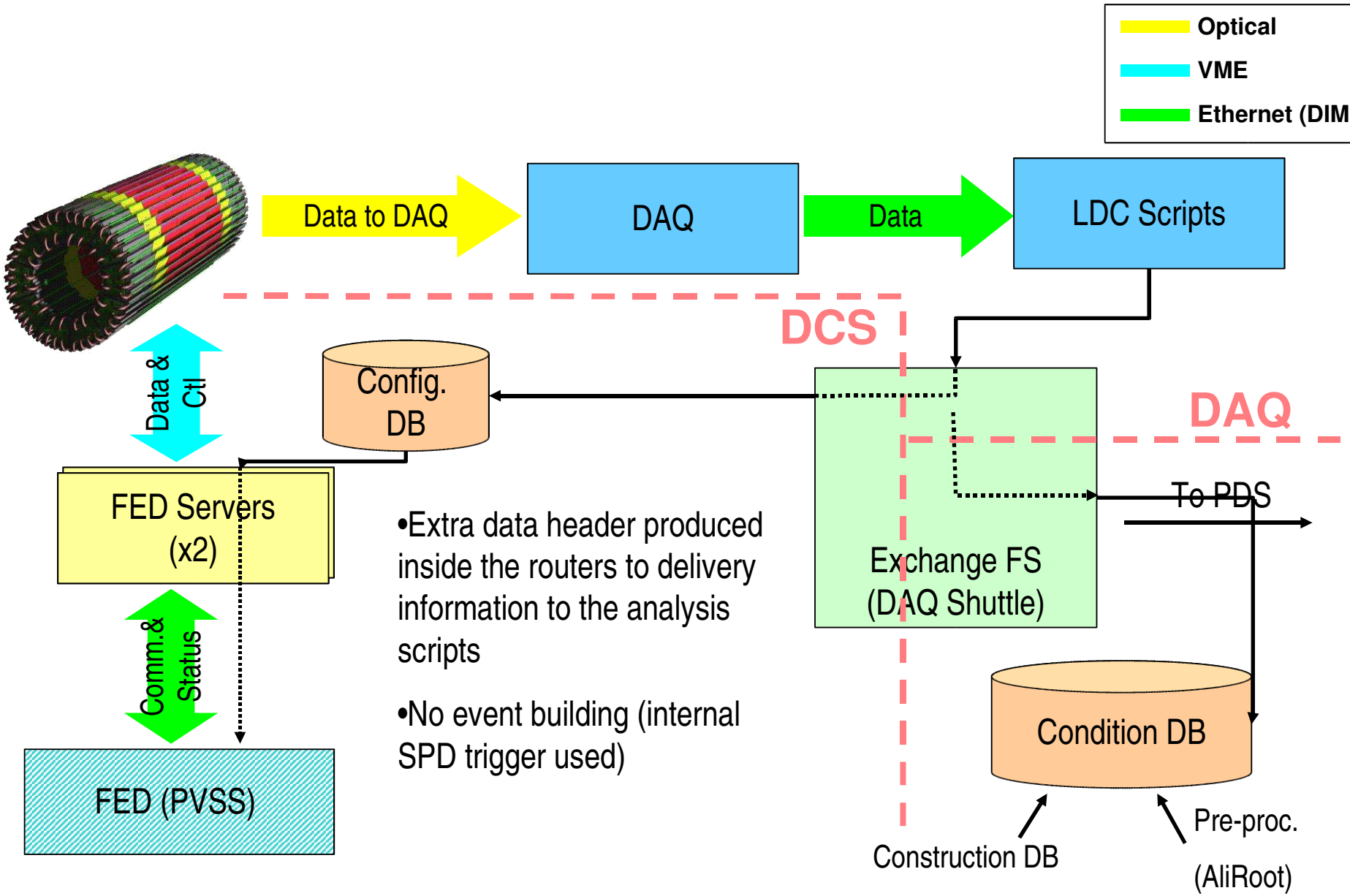
B) Physics Runs

(dead/noisy pixels only)

--> Offline CDB

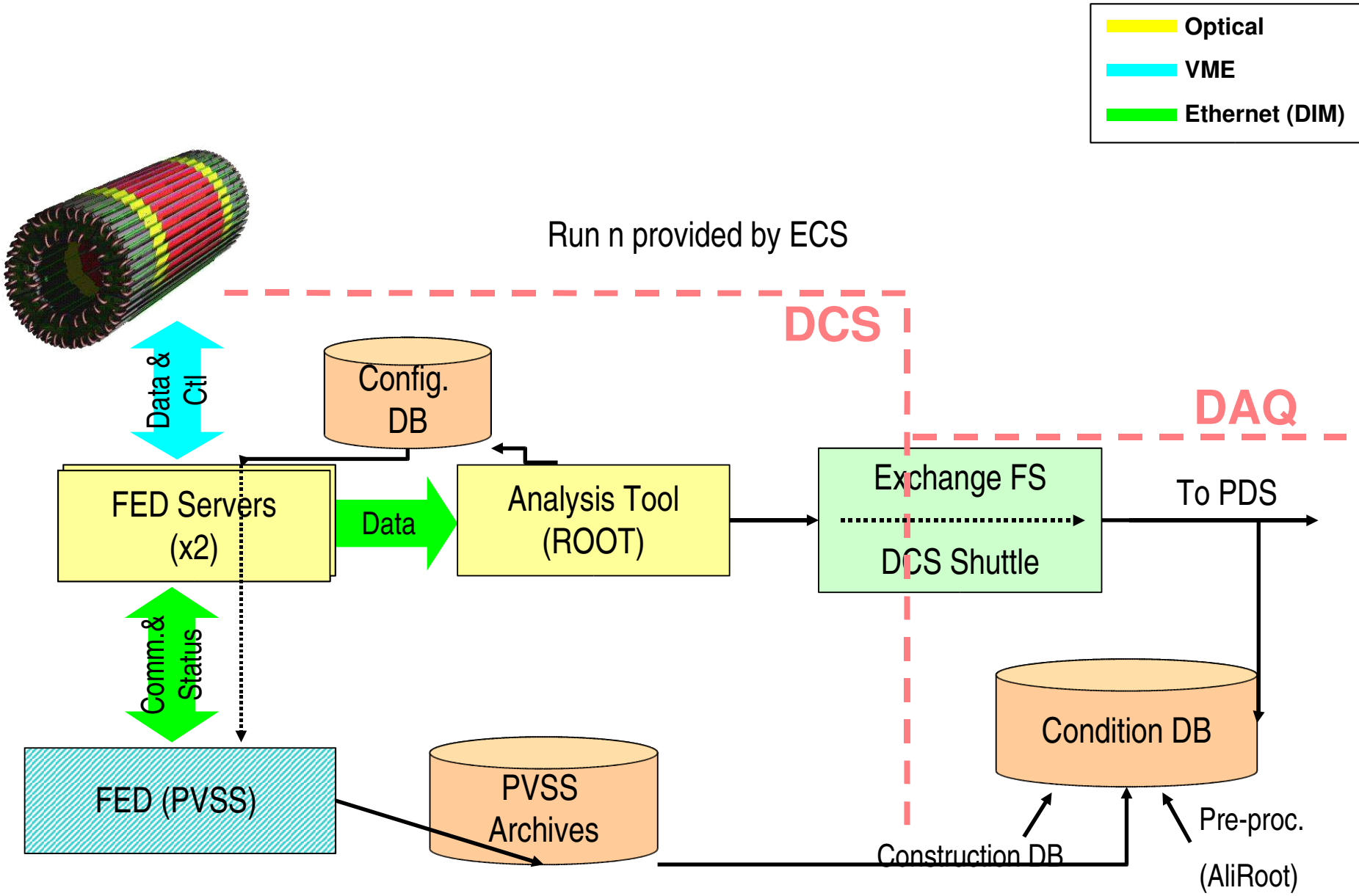
Henrik Tydesjö, 04 Oct 2006

A) Dedicated Calibration Runs (DAQ)



Ivan Amos Cali – Alice DAQ meeting 20/06/2006

A) Dedicated Calibration Runs (DCS)



Ivan Amos Cali – Alice DAQ meeting 20/06/2006

A) Dedicated Calibration Runs (DAQ)

1. ECS starts the SPD specific scan. (there are 5 types of scans)
   Detector configuration information is put in the raw data stream, as an extra calibration header.

2. A calibration script runs after the end-of-run on each LDC.
   a) c++/Root compiled code processes the raw data to produce Root container files (hitmaps and scan info).
   b) c++/Root compiled code processes the container files to produce temporary result files, their content depending on the type of scan.* Right now, lists of noisy and dead pixels are automatically produced, whereas the processing of other scan data needs to be implemented.
   The container data can be analyzed “by eye”, using a displayer GUI (see slide 9).

*) It may here be important to have the possibility to compare with previous calibrations, stored in local files on the LDCs.

A) Dedicated Calibration Runs (DAQ)

3. The temporary result files and the container files are put in the File Exchange Server. The existing code will be adjusted to the new online detector algorithm framework that S. Chapeland is providing. The relevant updates for DCS are sent through the FES to DCS.

4. The Shuttle picks up the files and the Preprocessor algorithm runs. (See next slide for a first version of this algorithm.)
   a) The information on dead/noisy pixels from the temporary result files is merged and a CDB object is created. The Offline CDB is updated.
   b) The container files are stored as “reference data”. (Estimated size of ~500 MB for a complete set of scans.)

A) Dedicated Calibration Runs (DAQ)

AliSPDPreprocessor

Merges the lists of dead and noisy pixels from all modules. (One Root file for each module from the LDCs.) Stores the information in the OCDB.

Will store the container files as reference data. (One Root file for each DDL from the LDCs.) To be implemented.

Existing code running in the shuttle test suite.

AliSPDPreprocessor.cxx

File Edit Options Buffers Tools C++ Help

```
//_____________________________________________________________________
UInt_t AliSPDPreprocessor::Process(TMap* dcsAliasMap)
{
  // Create the cdb entry filled with empty cal objects for each module
  TObjArray *SPDentry = new TObjArray(240);
  for(Int_t module=0; module<240; module++){
    AliITSCalibrationSPD* cal = new AliITSCalibrationSPD();
    SPDentry->Add(cal);
  }
  SPDentry->SetOwner(kTRUE);

  // Get all the files corresponding to the different modules and fill the dead/noisy lists
  OnlineCalibrationSPDhandler* handler = new OnlineCalibrationSPDhandler();
  for (Int_t module=0; module<240; module++) {
    Char_t id[20];
    sprintf(id,"SPD_%d",module);
    TList* list = GetFileSources(kDAQ,id); // (the id is actually unique, so always 1 file)
    if (list) {
      TObjString* fileNameEntry = (TObjString*) list->First();
      Char_t* fileName = (Char_t*) GetFile(kDAQ, id, fileNameEntry->GetString().Data());
      handler->SetModuleNr(module);
      handler->ReadFromFile(fileName);
      ((AliITSCalibrationSPD*) SPDentry->At(module)) -> SetNrDead( handler->GetNrDead() );
      ((AliITSCalibrationSPD*) SPDentry->At(module)) -> SetDeadList( handler->GetDeadArray() );
      ((AliITSCalibrationSPD*) SPDentry->At(module)) -> SetNrNoisy( handler->GetNrNoisy() );
      ((AliITSCalibrationSPD*) SPDentry->At(module)) -> SetNoisyList( handler->GetNoisyArray() );
    }
  }
  delete handler;

  // Store the cdb entry
  AliCDBMetaData metaData;
  metaData.SetBeamPeriod(0);
  metaData.SetResponsible("Henrik Tydesjo");
  metaData.SetComment("This preprocessor is for SPD.");

  UInt_t result = Store("SHUTTLE", "Calib", SPDentry, &metaData, 0, 0);
  delete SPDentry;

  // Store the container files as reference data
  // To be implemented

  return result;
}
```

-:-- AliSPDPreprocessor.cxx (C++ Abbrev)--L84--Bot

A) Dedicated Calibration Runs (DCS)

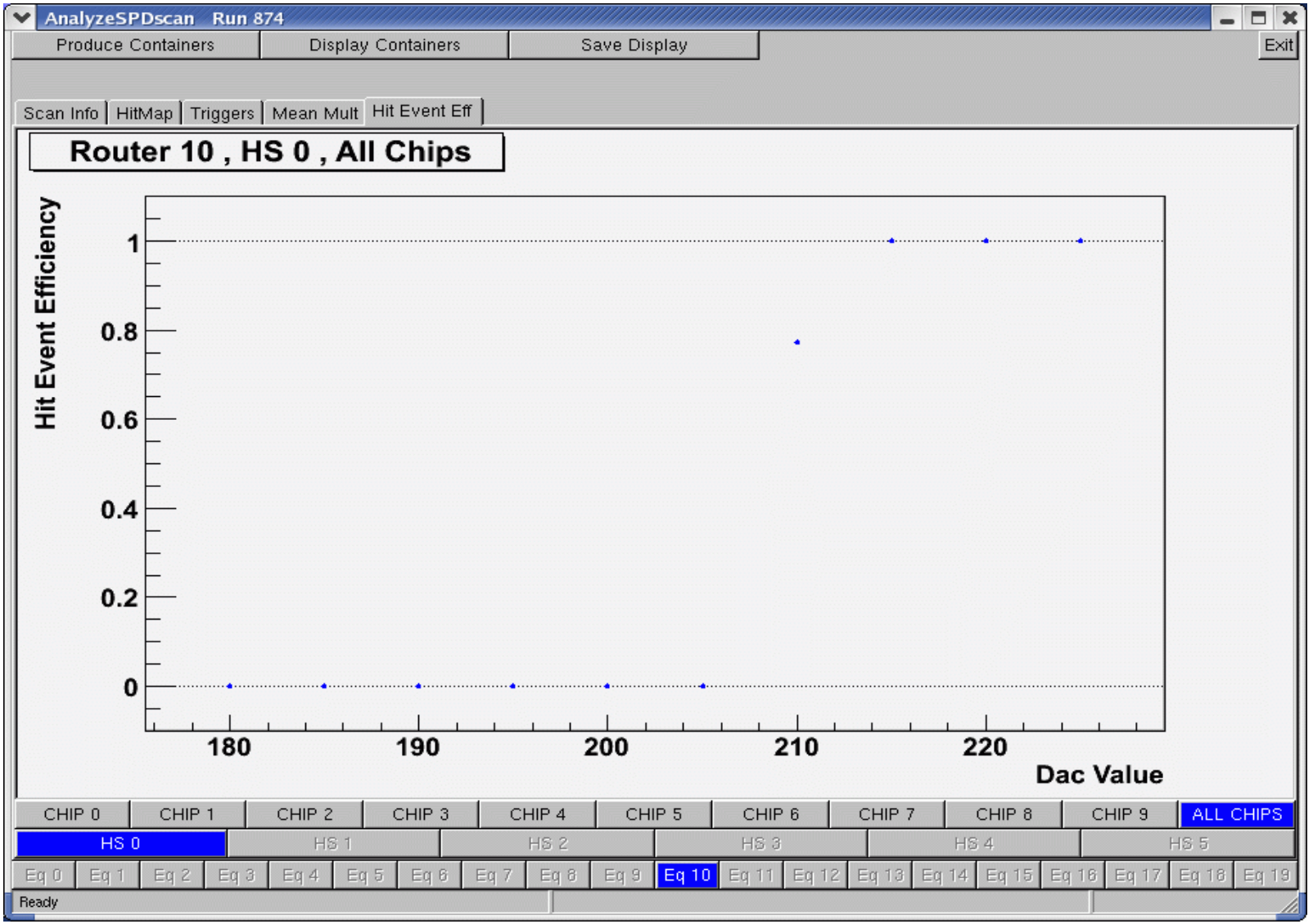
The procedures are similar to the DAQ case.
Data are now extracted directly from the FED server.
An analysis tool is running, producing the container files described in the DAQ case. In the same way as in the DAQ case, they will be processed and results sent offline via the File Exchange System. The exact same Preprocessor is then used.
DCS updates will here be straightforward.

B) Physics Runs

A monitoring program runs during physics data taking, filling histograms (hitmaps). When enough statistics have been collected, the database is updated with a set of new dead/noisy pixels in a similar way as described in case A.

The container displayer GUI

This program will be running on container files kept temporarily on the file exchange system. It can be used to check the quality of the calibration.



Future plans

The implementation for the dedicated calibration runs will be addressed first. Some of the tasks that need to be done:

Improvement of the container classes. (one week)

Further development of algorithms for analysis of calibration scans. (several months)

Integration with the online detector algorithm framework. (timescale to be determined)

Communication with DCS (for the DAQ case). (timescale to be determined)

Some additional information can be found in the SPD calibration requirements document, which was recently updated.