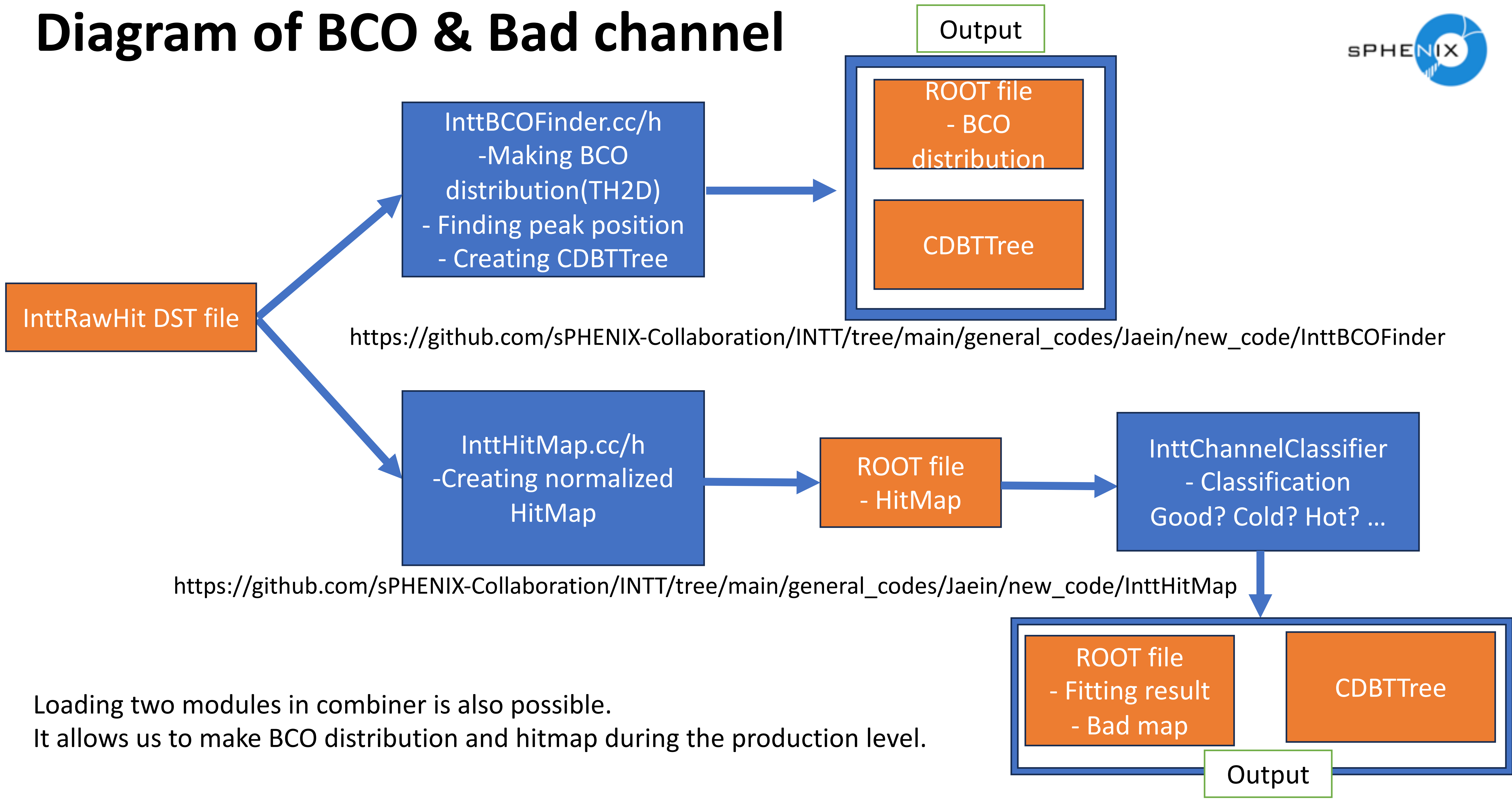
Modules for calibration parameters InttBCOFinder & InttHitMap

Jaein Hwang

Diagram of BCO & Bad channel


Output
ROOT file - BCO distribution
CDBTTree
InttBCOFinder.cc/h -Making BCO distribution(TH2D) - Finding peak position - Creating CDBTTree
InttRawHit DST file
InttHitMap.cc/h -Creating normalized HitMap
ROOT file - HitMap
InttChannelClassifier - Classification Good? Cold? Hot? …
ROOT file - Fitting result - Bad map
CDBTTree
Output

https://github.com/sPHENIX-Collaboration/INTT/tree/main/general_codes/Jaein/new_code/InttBCOFinder

https://github.com/sPHENIX-Collaboration/INTT/tree/main/general_codes/Jaein/new_code/InttHitMap

Loading two modules in combiner is also possible.
It allows us to make BCO distribution and hitmap during the production level.

Structure of the code(InttBCOFinder.h)

```cpp
class InttBCOFinder : public SubsysReco
{
  public:
  InttBCOFinder(const std::string &name = "InttBCOFinder", const std::string &fname = "outputfile.root",const std::string &fname2 = "cdbfile.root",int nevent = 10000);

  virtual ~InttBCOFinder();

  int Init(PHCompositeNode *);

  int InitRun(PHCompositeNode *);

  /// SubsysReco event processing method
  int process_event(PHCompositeNode *);

  /// SubsysReco end processing method
  int End(PHCompositeNode *);
  bool IsADCcutON_ = false;
  bool WriteCDBTTree_ = false;
  bool WriteQAFile_ = false;
  void FindBCOPeak();
  void ADCCut(const bool flag) { IsADCcutON_ = flag; }
  void WriteCDBTTree(const bool flag) { WriteCDBTTree_=flag; }
  void WriteQAFile(const bool flag) { WriteQAFile_ = flag; }
```

Main part for making BCO distribution

You may want to apply ADC0(DAC0) cut. I will keep it just in case.

Part for finding BCO peak position and making CDBTTree/root file

Structure of the code(InttHitMap.h)

```cpp
class InttHitMap : public SubsysReco
{
  public:
  InttHitMap(const std::string &name = "InttHitMap", const std::string &fname = "outfile.root",int nevent = 10000);

  virtual ~InttHitMap();

  int Init(PHCompositeNode *);

  int InitRun(PHCompositeNode *);

  /// SubsysReco event processing method
  int process_event(PHCompositeNode *);
```

Main part for making HitMap

```cpp
  /// SubsysReco end processing method
  int End(PHCompositeNode *);

  bool isBCOcutON_ = false;
  bool isBCOPeak(int felix,int ladder, int bco, uint64_t bcofull);
```

You may want to apply BCO cut before making hitmap.
During streaming readout pp run, we may not use it.

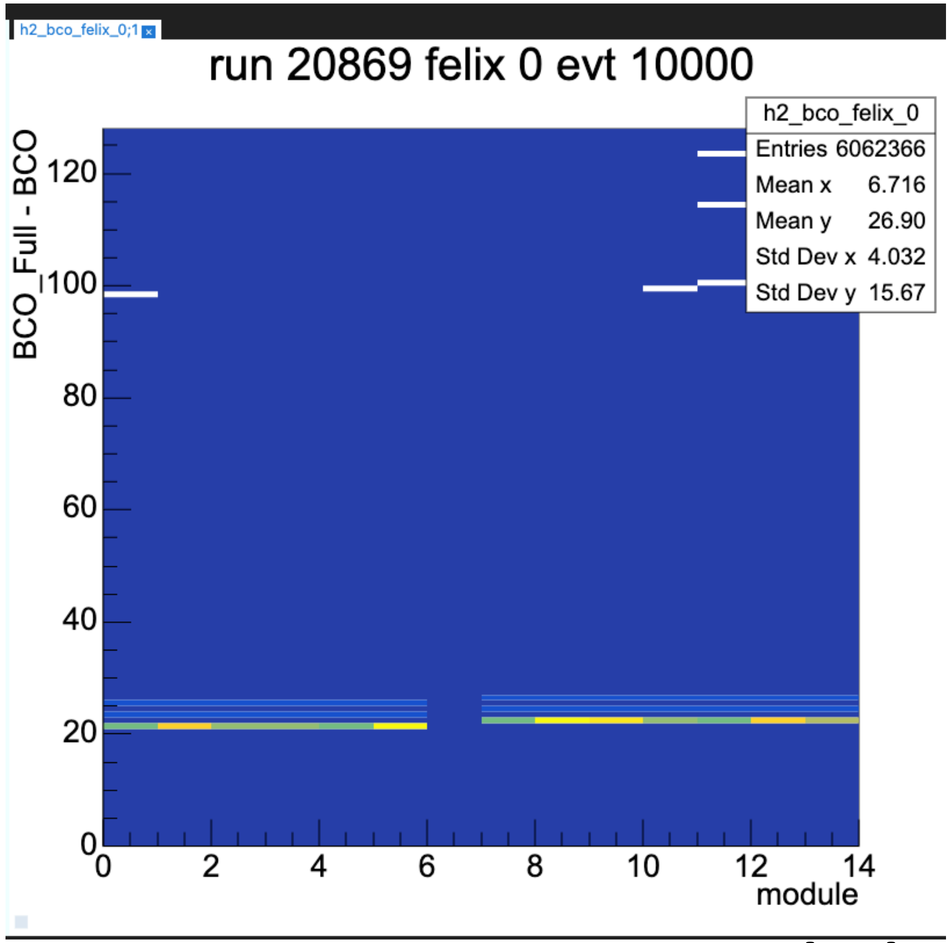
```cpp
  void SetBCOcut(const bool flag){ isBCOcutON_ = flag; }
  int SetBCOFile(const char* bcofile);
  int SetFeeMapFile(const char* feemapfile);
  InttFeeMapv1 fee_map;
  bool FillHitMap(int felix, int moudle, int barrel, int chip, int chan);
```

Part to Fill Histograms(HitMap)

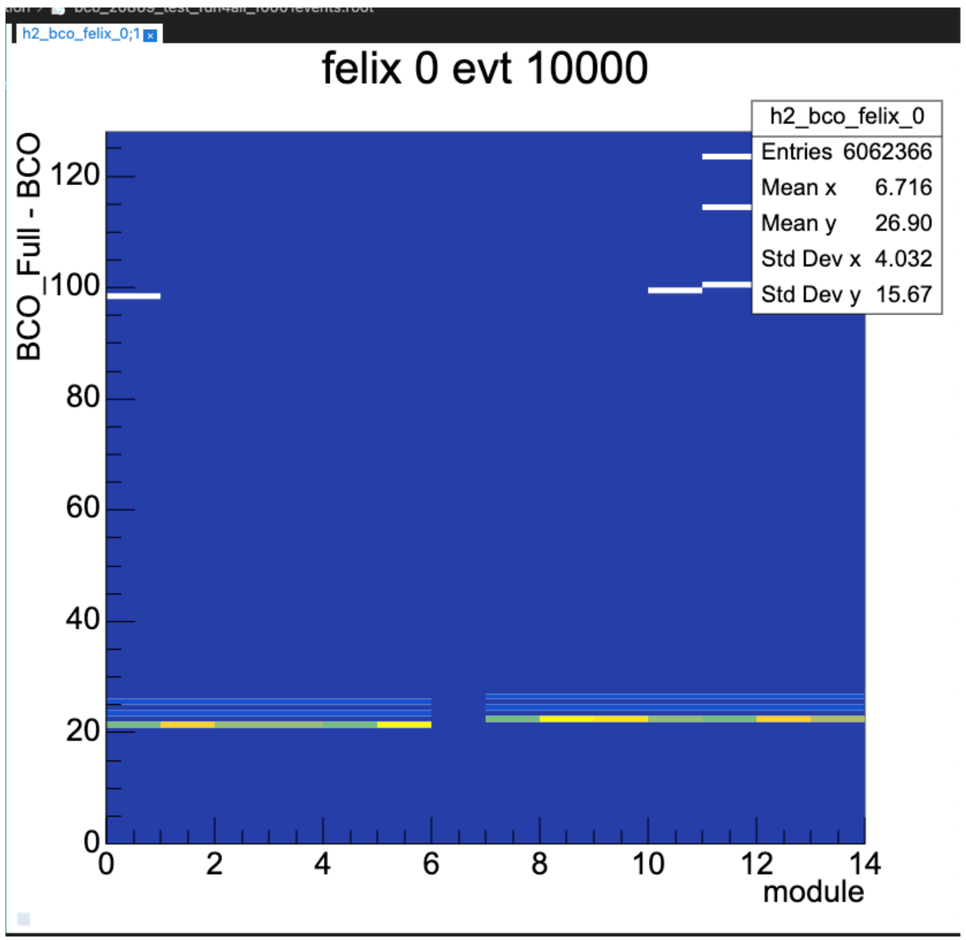
Consistency check

Comparing BCO distribution from old version code(using event based TTree) to the one from new code(using InttRawHit)

From event based TTree(old version code)



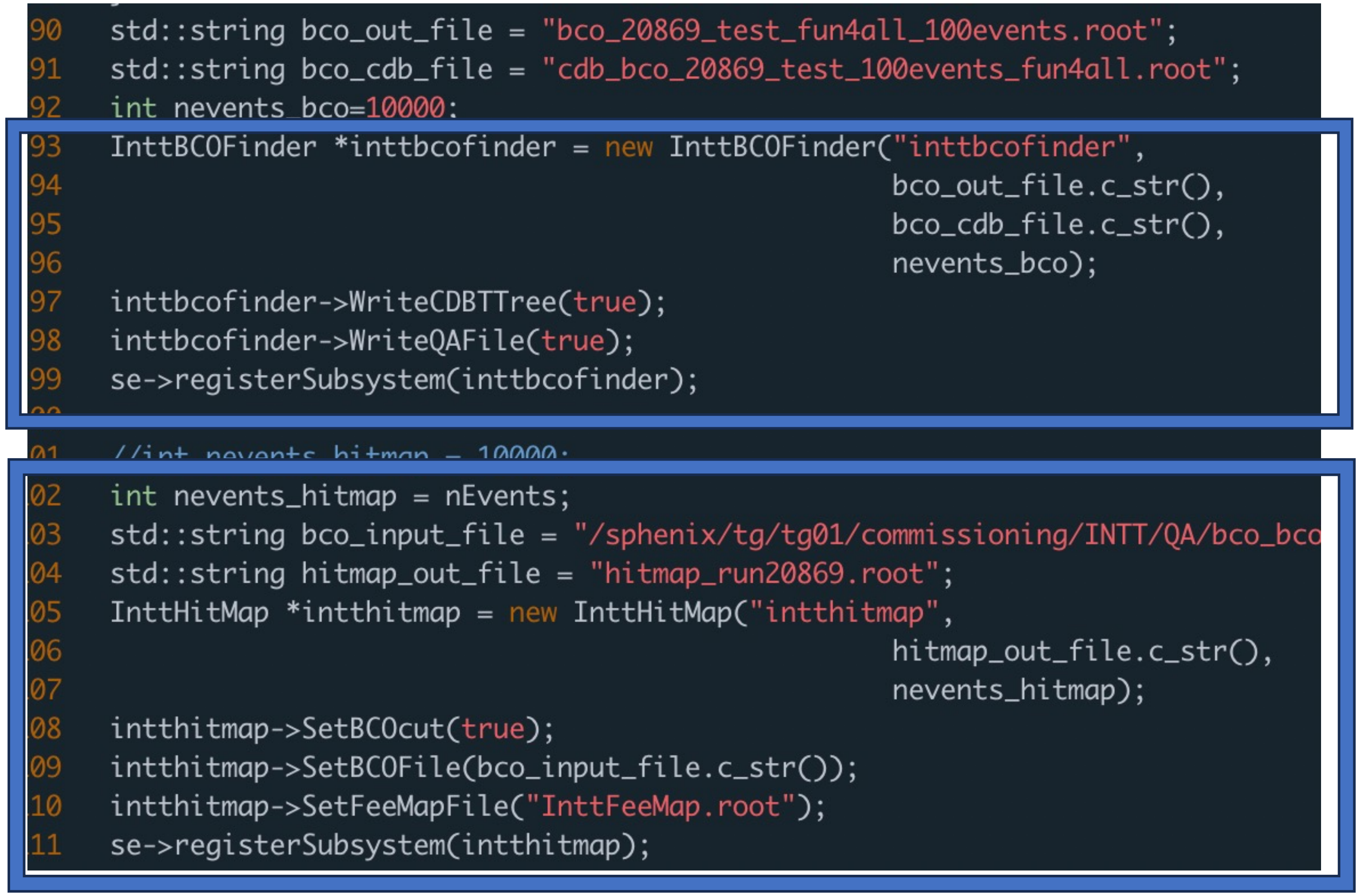
From InttRawHit(new version)



Result shows 100% consistency

Example code to use new modules in combiner

Test code is existed in : (SDCC machine)
/sphenix/tg/tg01/commissioning/INTT/work/jaein/sphenix_macro/macros/InttProduction/Fun4All_Intt_Combiner.C



```
  std::string bco_out_file = "bco_20869_test_fun4all_100events.root";
  std::string bco_cdb_file = "cdb_bco_20869_test_100events_fun4all.root";
  int nevents_bco=10000;
  InttBCOFinder *inttbcofinder = new InttBCOFinder("inttbcofinder",
                                                  bco_out_file.c_str(),
                                                  bco_cdb_file.c_str(),
                                                  nevents_bco);
  inttbcofinder->WriteCDBTTree(true);
  inttbcofinder->WriteQAFile(true);
  se->registerSubsystem(inttbcofinder);

  //int nevents_hitmap = 10000;
  int nevents_hitmap = nEvents;
  std::string bco_input_file = "/sphenix/tg/tg01/commissioning/INTT/QA/bco_bco
  std::string hitmap_out_file = "hitmap_run20869.root";
  IntttHitMap *intthitmap = new InttHitMap("intthitmap",
                                                  hitmap_out_file.c_str(),
                                                  nevents_hitmap);
  intthitmap->SetBCOcut(true);
  intthitmap->SetBCOFile(bco_input_file.c_str());
  intthitmap->SetFeeMapFile("InttFeeMap.root");
  se->registerSubsystem(intthitmap);
```

Registering InttBCOFinder module

Registering InttHitMap module

Summary & Plan

InttBCOFinder and InittHitMap have been published and tested.
New modules are working well and show 100% consistency comparing to old version.

New modules have been pushed into coresoftware (as we discussed in last INTT meeting).
Now under Pull Request review. The directory in GitHub would be

/coresoftware/tree/master/calibrations/intt

Determination of hot/cold/bead/good channel requires enough statistics.
pp data will be available soon!
Detail of algorithm to classify channel has to be tested with pp data.