# Steps to run data analysis

1) Get copy of root12fms.tgz and unpack it somewhere.

2) Go to new directory "./root12fms". Edit file "SetFMSEnv" shown below,

```
#!/bin/tcsh
setenv SETFMSENV  "SETFMSENV"
setenv FMSROOT "/home/heppel/ABatch/root12fms/"
setenv FMSSRC "${FMSROOT}/FpdRoot"
setenv FMSTXT  "${FMSROOT}/fmstxt"
setenv FMSGAIN "fpdgain.txt"
setenv FMSCORR "FMScorr2.txt"
setenv QTMAP "qtmap2009V1.txt"
setenv QTMAP2PP "qtmap2pp.txt"
setenv TRIG_ROOT_DATA_SRC /home/heppel/MyBook/rootfiles
```

Change some environment variables.
FMSROOT is the local directory just created.
TRIG_ROOT_DATA_SRC is the location of new trigger data files that have been converted to root files (run*.root)

3) Make sure that you are running tcsh. If in doubt just type "tcsh". From tcsh, source the SetFMSEnv file.

```
>tcsh
>source SetFMSEnv
```

4) Figure out what runs from TRIG_ROOT_DATA_SRC you want to copy for this analysis. Also note the fill number for these runs. Use a script like **"MkRun"** to copy over and extract adc distributions. The following example does the copying and can analyze. The **Mkfiles** script takes 2 arguments, run number and fill number.

**MkRun**

```
#!/bin/tcsh
source SetFMSEnv
setenv Do_ADC_Only  "true"
setenv Do_Script_and_ADC  "false"
setenv Keep_adcTr "false"

./Mkfiles 10175012 10973
./Mkfiles 10175013 10973
./Mkfiles 10175014 10973
```

The above example creates “fake” spin files (for now) and creates a directory tree for each of the 3 runs indicated. The only analysis selected was extraction of raw adc distributions for each cell.

The result is to create a directory tree for each of the 3 chosen runs. Each tree contains ~100 smaller data segment directories.

A full reanalysis of the three runs above is accomplished by running the script **MkRedoList.**

After analysis, each segment directory contains several root files.

```
ls -l  files10175012/10175012_122/
total 9484
-rw-r--r-- 1 heppel heppel   53924 2011-01-10 02:08 adc.log
-rw-r--r-- 1 heppel heppel  486396 2011-01-10 02:08 adcout.root
-rw-r--r-- 1 heppel heppel   44609 2011-01-10 02:08 FakeLumi.txt
-rw-r--r-- 1 heppel heppel   43068 2011-01-10 02:08 log.log
-rw-r--r-- 1 heppel heppel  750020 2011-01-10 02:08 OFile.root
-rwxr-xr-x 1 heppel heppel 8319442 2011-01-10 01:15 run10175012.122.root
```

- run10175012.122.root is the root version of the trigger file.
- OFile.root is a secondary file reflecting analysis.
- adcout.root is a small file with adc distributions.

After the script **MkRedoList** has completed all the OFiles are collected in one **OF4.root** and all the adcout files into a single **adc.root**.

The result of **MkRedoList** is to redo all analysis of the 3 runs indicated. Analysis includes calibration files specified in **SetFMSEnv**.

Next we create a “**Cell**” class instance for each FMS cell. We do this as follows.

1. Run the following script
   >./OneReadM  OF4.root  92  ./

   This creates a file **Output.root** that contains all the Cell instances.
2. Move the **Output.root** file into **CodeSmall**.

3. In the **CodeSmall** directry: Now we extract the "**Cell**"s into separate root files in a directory called data2. We do this with the following sequence:

```
>mv Output.root CodeSmall/
>cd CodeSmall
>root –b –q  'Extract.C("small")'  >& log1.log
>root –b –q  'SetRank.C("small")' >& log2.log
>root –b –q  'Extract.C("large")'  >& log3.log
>root –b –q  'SetRank.C("large")' >& log4.log
>ls data2/*
```

4. The long history of gain calibration files are kept in a file **FpdFile.root.** That history is seen in **smallPlots/history.pdf** or **largePlots/history.pdf** . However, everytime a single reanalysis of a single cell occurs, a line is appended to **ChangeFpd.txt.** This appended change can occur asynchronously if multiple processes are running. Each reanalysis job finds the most recent entry from file **FpdFile.root** and then makes single channel changes as required by the **ChangeFpd.txt** file.
5. After some set of reanalysis iteration has occurred, there is a sequence required that updates **FpdFile.root** and makes lots of pdf files.

```
>root –b –q  'FileFpd.C("small")'
>root –b –q  'FileFpd.C("large")'
>root –b –q  'GcUpdate.C("small")' >& loga.log
>root –b  -b  'GcUpdate.C("large")' >& logb.log
>rm ChangeFpd.txt
>root –b –q  'PrintCorr.C("small")'
>root –b –q  'PrintCorr.C("large")'
>root –b –q  'Print_mvse.C("small")'
>root –b –q  'Print_mvse.C("large")'
```

6. The iteration process itself is by default based on getting a consistent pion mass. To reanalyze a particular cell, the root command is:

```
>cd CodeSmall
>root –b –q 'RunOne.C("Cellr3_c4_0")'
```

   The above example uses all the current two photon pairs that are associated with the indicated "Cell". They are re-reconstructed with the current values of gain corrections. Based on the mass distributions, a new gain correction factor is calculated and appended to **ChangeFpd.txt**.
7. The scripts **RunRoundL** and **RunRoundS** execute the above command many times looping over each cell in the FMS. This could be done in parallel jobs, each taking a fraction of a minute. When completed, we should go back to Step 5 and make new plots.
8. If we want to extract the new FMSCORR .txt file it can be found at **SetFpdCorr.txt_V1. T**his file can be renamed and included in **FMSTXT directory** with corresponding change in the **SetFMSEnv** file.

9. Finally, it is possible to restart the whole thing with the new set of corrections. This allows for the possibility that different pairs of clusters can be selected. This might be important if the changes in gain are too big.

# About Cell Class Iteration

For each FMS cell, we create an instance of a “Cell” class which summarizes the information associated with that particular FMS cell.

In “Output.root” we have a TObjArray called Mycells which contains all of these Cell instances. Cells have names that reflect the row0, col0 and nstb0 of a cell. For row0=3, col0=0 and nstb0=0 we describe a North large cell with the name “**Cellr3_c0_0**”.

In data2/ we split off these into separate root files the files have names like “**data2/_Cellr3_c0_0.root**”.

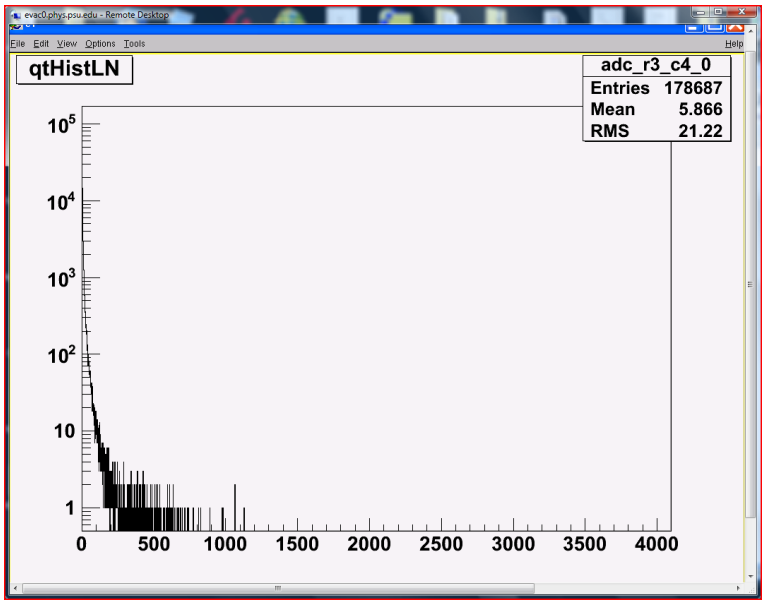
Each of these files can be referenced directly with root:

We can type (for example)

```
>root data2/_Cellr3_c4_0.root
  root [0]   .x  ../start.C
  root [1] .ls
  root [2] Cell* c= Cellr3_c4_0
```

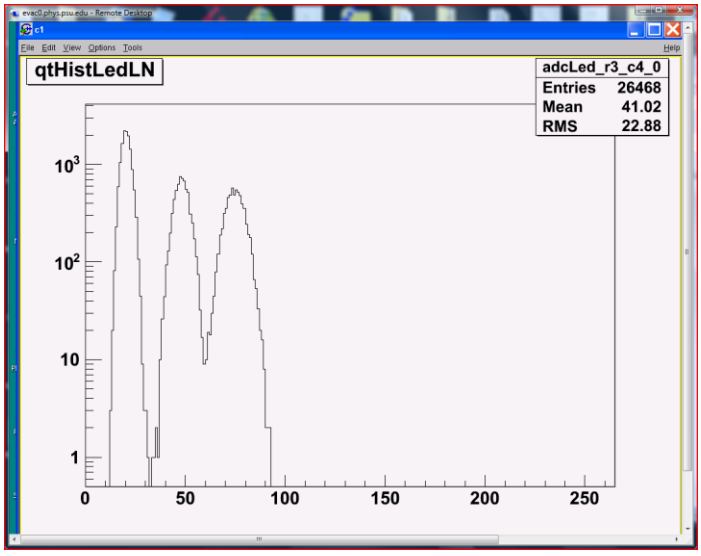
Now to look at the raw ADC distribution (non-pulser)

```
  root [3]  c->p_adc->Draw()
  root [4]  c1->SetLogy()
```



Now to look at the raw ADC distribution (pulser)

root [5] c->p_adcLed->Draw()



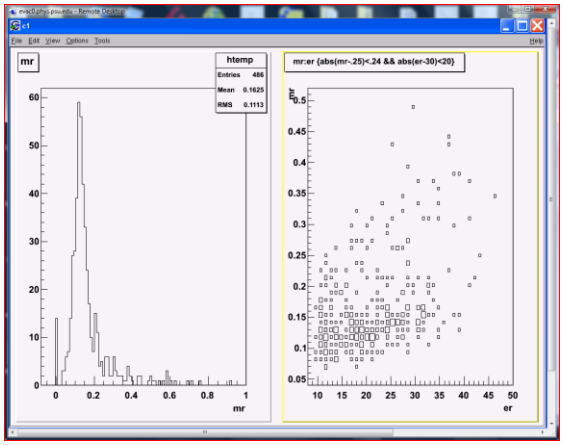

After ReReconstruction (RunOne.C) , information is stored in a TTree. That TTree is returned to the data2/ file.

root [6] c1->Divide(2,1); c1->cd(1);c1->SetLogY(0)

root [7] RecTr->Draw("mr")

root [8] c1->cd(2)

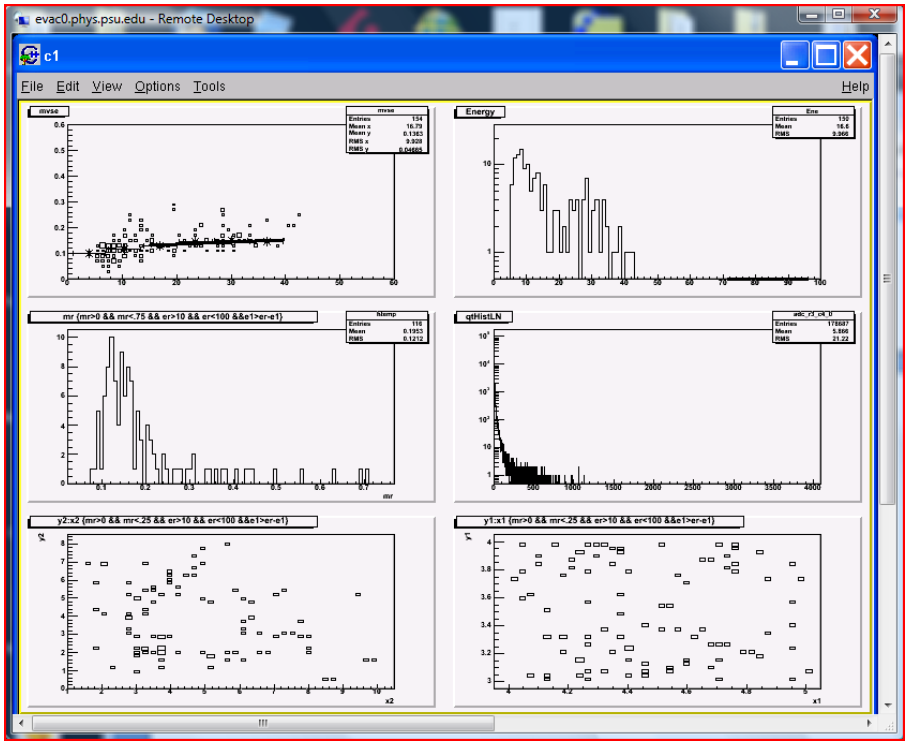
root [9] RecTr->Draw("mr:er","abs(mr-.25)<.24 && abs(er-30)<20","box")




Each time we re-reconstruct pi0s, we update this root file.

A summary plot from the last “RunOne.C” is called plot

root [10] plots->Draw()




More to come……