Calibration Procedures and Software for HiRA
(Si detectors and CsI detectors)

Contents

- **All programs are developed by Sergei Lobastov (Dubna)**
- **All programs are in /projects/proj4/hira/LS_program/**

Chapter 1

Calibration Principles and Procedures

1.1 Silicon Detectors

^{228}Th source is used for calibrations of silicon detectors. This source is characterized by eight well-known alpha energies and the alpha peaks are separated by at least 100keV. This allows clean identification and fitting of all energy peaks.

From the 1D raw Si histogram of each strip, we use the five strongest alpha peaks with energies at 8.787, 6.778, 6.288, 5.685 and 5.423MeV. The corresponding peak channels are found by fitting with a Gaussian function and the channel number for each peak is then found. As the energy of each peak is known, we can establish a linear relation between energy and channel by fitting those five peaks. This procedure is done individually for all strips. Software routine "ThCalib" [2.1] in Fitter [Ch.2] is written for this application.

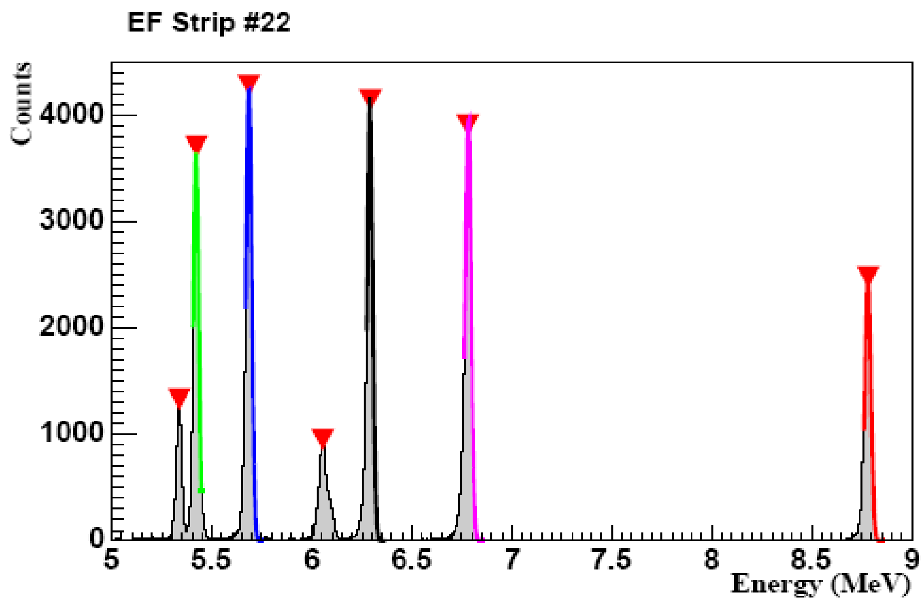


Figure 1.1 Energy spectrum of the ^{228}Th alpha source in a single strip.

If a dE silicon detector is placed in front of the double sided E silicon detector, the dE detector can be calibrated by ^{228}Th source as discussed above; while pin source is used to calibrate E detector. The pin source is activated by electroplating the tip with daughter nuclei from ^{228}Th which emits strong alpha particles at 8.785MeV, 6.050MeV and 6.089MeV. Using the same procedure, we can calibrate each strip in E detector.

The linearity of the electronics system can be determined by ramping the pulser over the ADC range, which gives information if further corrections in calibrations are needed. The software "Pulser" [2.3] in Fitter is specialized for this application. In the experiment 02018, 02019, 02023, and 05038, good linearity is observed in dE detectors, but not in EF and EB detectors.

1.2 CsI Detectors

The calibration of the CsI is accomplished through the normal data run combined with corrections obtained from proton elastic scattering.

From the simulations package Geant4 [4.1], we can get the relation between energy deposited in Si detector and CsI crystal for each particle by higher order polynomial fitting (usually 5th order). With Si calibrations, we can generate 2D histograms with calibrated Si E vs raw CsI E for each CsI crystal. Profile for each type of particle is made from the histogram and the raw CsI E is converted to calibrated CsI E by applying the parameters obtained from Geant4. A linear relationship is then established between CsI energy and channel by fitting all different types of particles inside the profiles drawn. The software routine "CsICalib" [2.4] in Fitter package has been written specifically for this application. This procedure is done individually for all four crystals in each telescope.

To get a more accurate calibrations for CsI detectors, including the correction of energies lost in target, deadlayers and two mylar foils, the elastic scatting data with calibration parameters obtained from normal data run are analyzed. Gate of each particle band is created from normal data run and applied to the calibrated elastic scattering data as shown below.

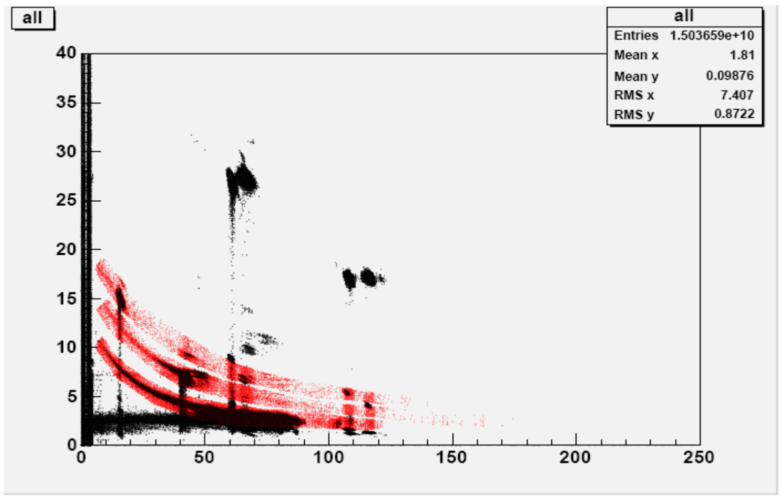


Figure 1.2 Histogram of calibrated elastic scattering data (Si vs CsI). Black points are data and red bands are the gates from normal data run.

Sharp elastic scattering peaks are observed. The peak energies are then extracted by fitting with single Gaussian function, and the energies of different particles deposited in each CsI detector can be found.

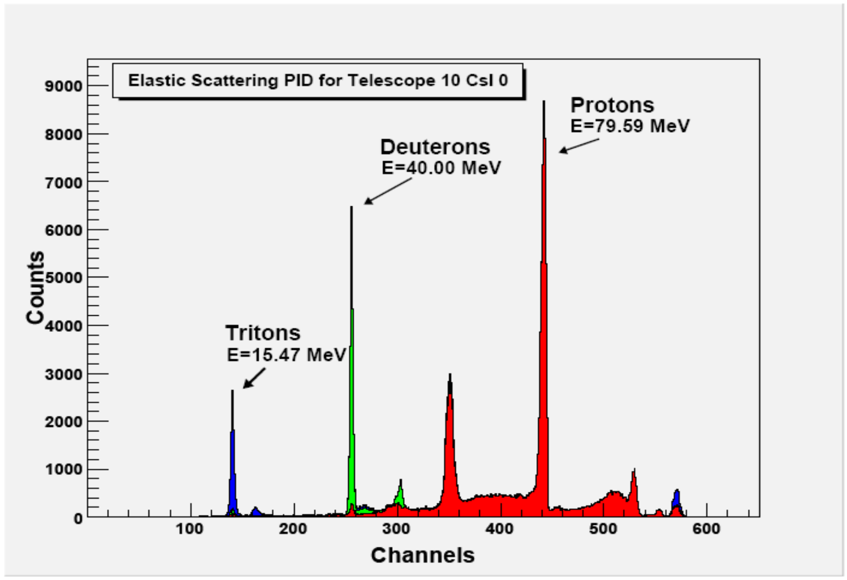


Figure 1.3 Histogram of proton, deuteron and triton peaks from elastic scattering data gated by pdt cuts from normal data run.

Since the initial energies of incident particle are known from Bp value, we can use Lise software to calculate the theoretical energies deposited in CsI crystals after taking the effect of target and other materials into account. By comparing the theoretical energy deposited and the energy loss obtained from calibrated elastic scattering data, we can establish a linear relationship for each crystal by fitting one type of isotope. The slope and intercept values are used to modify the previous calibrations and a complete calibration for CsI are obtained.

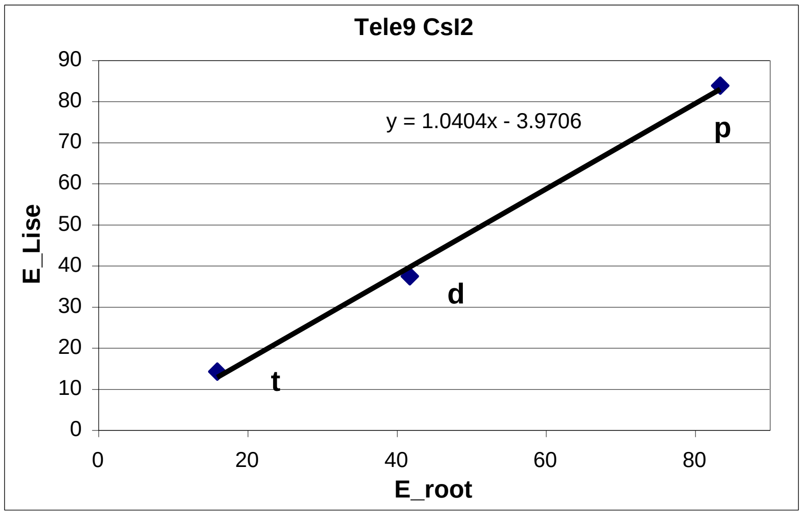


Figure 1.4 Linear fit of calibrated proton, deuteron and triton peaks to Lise calculations.

Program Etot [3.7] shows how accurate CsI calibration is by summing the energies deposited in target, deadlayers, mylar foil, Si detectors and CsI detectors. The energies loss in target, deadlayers and mylar foil are simulated by Ziegler [4.4]. Thus Etot gives us the initial energy of incident particles deduced partly from calibrated data and partly from simulations and then compares it with the energies calculated from Bρ. Further corrections in CsI calibration related to angular dependence will be included in Etot program in the future.

The linearity of the electronics system can be determined by ramping the pulser over the ADC range, which gives information if further corrections in calibrations are needed [2.3]. In the experiment 02018, 02019, 02023, and 05038, good linearity is observed in CsI detectors in the energy range we are concerned.

Chapter 2

Instructions of Calibration Programs in Fitter package

Fitter is a software combining six subroutines for calibrating Si and CsI detectors and pulser fitting as well as spectra viewer and it can read both .asc files and .root files.

Make sure the configuration file (fitter.cfg) is in the same directory where you bring up Fitter (type: /projects/proj4/hira/LS_program/Fitter/fitter). It is suggested to make a copy of the coufiguration file (fitter.cfg) and put it in your working area, so that you can specialize your own configuration file without corrupting the original one.

If you exit Fitter by Ctrl Z, you can re-activate the program by typing "fg".
You can select the detectors and telescopes you want to calibrate by going to "File" and clicking "Select tele and plane". You can also save the spectrum diplaced by using "Save the avtive canvas" under "File"

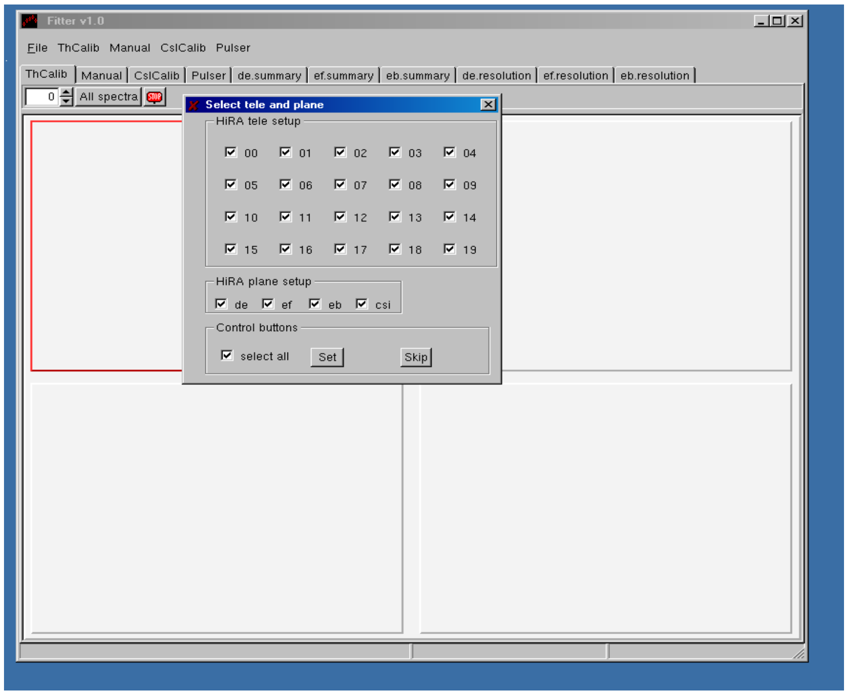
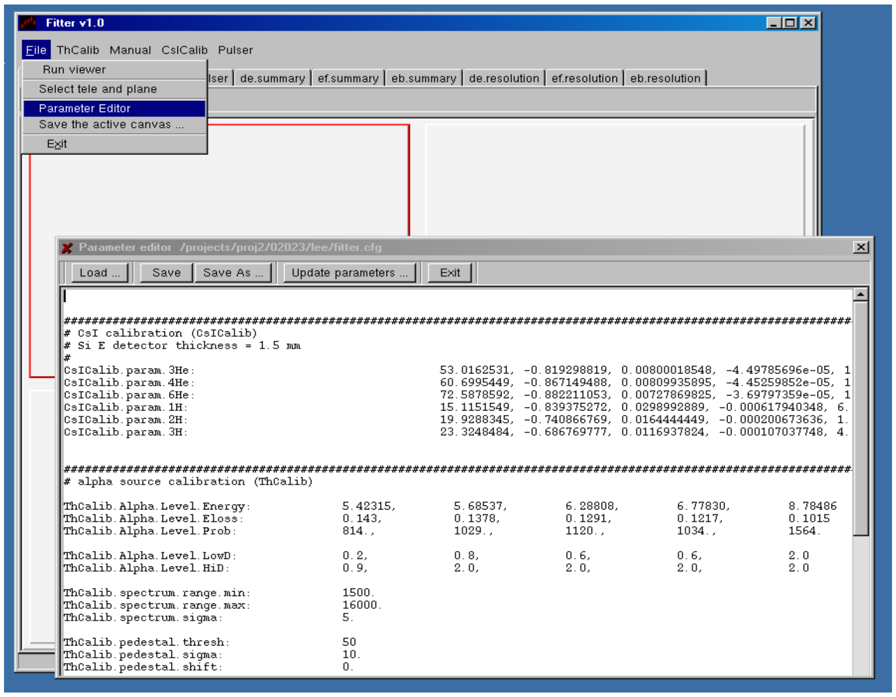


Figure 2.2 Fitter interface with "Select tele and plane" window

2.1 Configuration file for Fitter

fitter.cfg contains parameters needed for peak search and peak fitting. The fitter.cfg will be automatically loaded when Fitter is activated. To change patameters in fitter.cfg after Fitter is active, you can go to “File” and click “Parameter Editor”. A window will pop up, then you can load, edit, save the fitter.cfg file and update parameters.



```
##########################################################################################################
# CsI calibration (CsICalib)
# Si E detector thickness = 1.5 mm
#
CsICalib.param.3He:                                    53.0162531, -0.819298819, 0.00800018548, -4.49785696e-05, 1
CsICalib.param.4He:                                    60.6995449, -0.867149488, 0.00809935895, -4.45259852e-05, 1
CsICalib.param.6He:                                    72.5878592, -0.882211053, 0.00727869825, -3.69797359e-05, 1
CsICalib.param.1H:                                     15.1151549, -0.839375272, 0.0298992889, -0.000617940348, 6.
CsICalib.param.2H:                                     19.9288345, -0.740866769, 0.0164444449, -0.000200673636, 1.
CsICalib.param.3H:                                     23.3248484, -0.686769777, 0.0116937824, -0.000107037748, 4.


##########################################################################################################
# alpha source calibration (ThCalib)

ThCalib.Alpha.Level.Energy:             5.42315,        5.68537,        6.28808,        6.77830,        8.78486
ThCalib.Alpha.Level.Eloss:              0.143,          0.1378,         0.1291,         0.1217,         0.1015
ThCalib.Alpha.Level.Prob:               814.,           1029.,          1120.,          1034.,          1564.

ThCalib.Alpha.Level.LowD:               0.2,            0.8,            0.6,            0.6,            2.0
ThCalib.Alpha.Level.HiD:                0.9,            2.0,            2.0,            2.0,            2.0

ThCalib.spectrum.range.min:             1500.
ThCalib.spectrum.range.max:             16000.
ThCalib.spectrum.sigma:                 5.

ThCalib.pedestal.thresh:                50
ThCalib.pedestal.sigma:                 10.
ThCalib.pedestal.shift:                 0.
```

Figure 2.2 “Parameter Editor” interface

Meaning of parameter in fitter.cfg

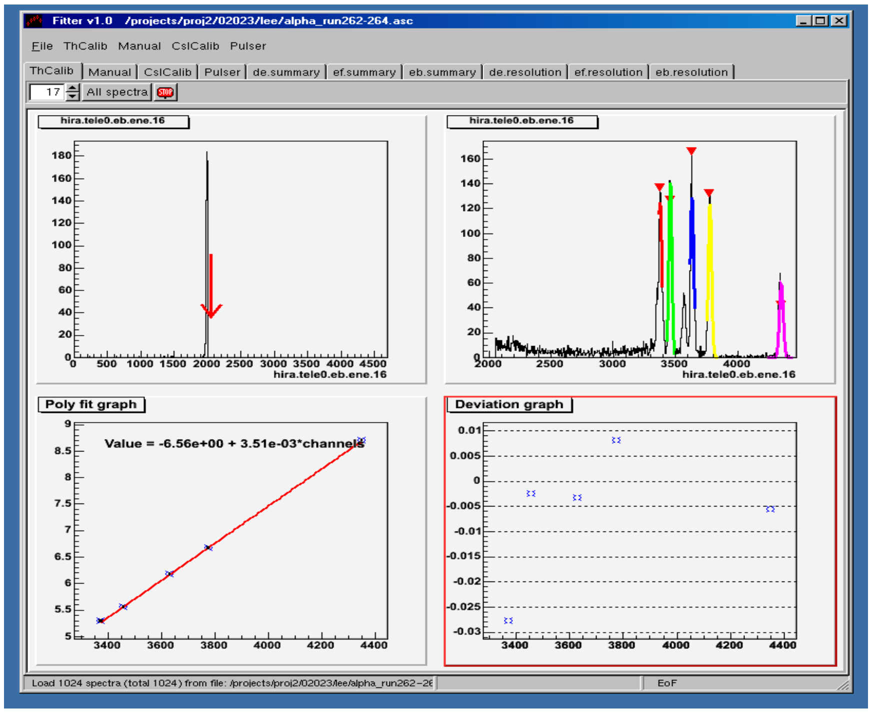
CsICalib.para.XX	Parameters for CsI calibrations for particle XX, obtained from simulations Geant4
ThCalib.Alpha.Level.Energy	Energy for each alpha peak
ThCalib.Alpha.Level.Prob	Probability of peak, help peak search and define the peak threshold [2.2]
ThCalib.Alpha.Level.LowD	Width of left side of peak used for fitting, unit: sigma from gaussina function
ThCalib.Alpha.Level.HiD	Width of right side of peak used for fitting, unit: sigma from gaussina function
ThCalib.spectrum.range.min	Mimimum channel number for alpah peak search
ThCalib.spectrum.range.max	Maximum channel number for alpah peak search
ThCalib.spectrum.sigma	Peak with sigma larger than this value will not be used (usually 5)
ThCalib.pedestal.thresh	Threshold for Pedestal
ThCalib.pedestal.sigma	Pedestal with sigma less than this value is expected if fitting for Pedestal is needed
ThCalib.pedestal.shift	Shift in channels for Pedestal; make sure to get rid of pedestal from threhold file
Thcalib.matching.pedestal	Matching pedestal (1=yes;0=no)
ThCalib.matching.deviation	Show pedestal in deviation graph (1=yes;0=no)
ThCalib.deadlayer.thickness.default	Deadlayer thickness
ThCalib.resolution.peak.id	Resolution for alpha peak (usually 4)
Pulser.basePeak.range	Range for the maximum peak
Pulser.basePeak.overHeight	Peak with height more than this multiple of Max. peak will be regarded as pedestal
Pulser.basePeak.volts	Voltage corresponds to the maximum peak
Pulser.peaks.step	Voltage difference betweem two successive peaks
Pulser.peaks.resol	Resolution fo peak
Pulser.peaks.sigma	Peak with sigma larger than this value will not be used
Pulser.peaks.thres	Fraction of the Max. peak. Only Peak with height above this fraction will be used
Pulser.peaks.used.before	Peaks before the maximum peak
Pulser.peaks.used.after	Peaks after the maximum peak
Pulser.peaks.output.range	Range of voltage being outputed

2.2 ThSource

The program can automatically calibrate all spectrum when you click “All Spectra” icon. Or you can calibrate the spectrun one by one by the button “Enter” on the keyboard.

After the calibration is finished, you have to go to “ThCalib” and click “Save offset/slope vdef”. If you want to continue the calibration that has not been finished before, you can first choose “load offset/slope vdef” and then calibrate the remaining spectra , so that your vdef values will be written in the same vdef file.

You can clear the current loaded Spectra and calibration results in database by clicking “Reset spectra list” and “Reset DB”.


Fitter v1.0 /projects/proj2/02023/lee/alpha_run262-264.asc
File ThCalib Manual CsICalib Pulser
ThCalib Manual CsICalib Pulser de.summary ef.summary eb.summary de.resolution ef.resolution eb.resolution
17 All spectra
hira.tele0.eb.ene.16
Poly fit graph
Deviation graph
EoF

Value = -6.56e+00 + 3.51e-03*channels

Load 1024 spectra (total 1024) from file: /projects/proj2/02023/lee/alpha_run262-26

Figure 2.3 Calibration of Th source. The top left panel shows the pedestal with red arrow indicating the threshold level. The top right panel shows the alpha with Gaussian fitting. The buttom left panel shows the linear fitting, and the right panel displays the deviation of each peak fitting.

To look at the summary spectra and resolution plots of dE detectors, you can click “de.summary” and “de.resolution” icons. Same as for EF and EB. Zoom function is available.

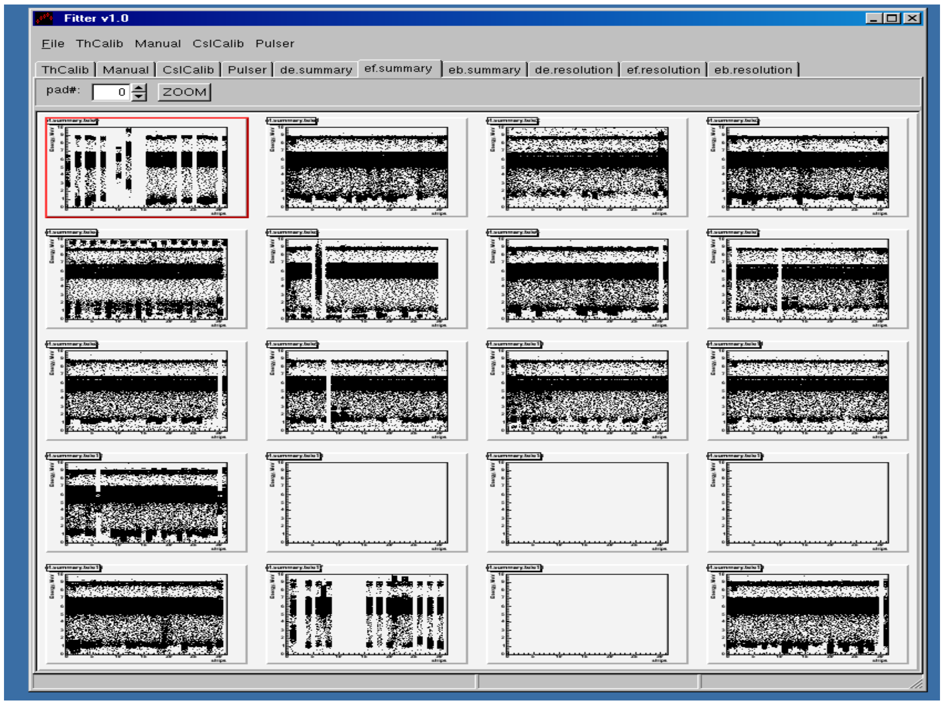

Fitter v1.0
File ThCalib Manual CsICalib Pulser
ThCalib | Manual | CsICalib | Pulser | de.summary | ef.summary | eb.summary | de.resolution | ef.resolution | eb.resolution
pad#: 0 ZOOM

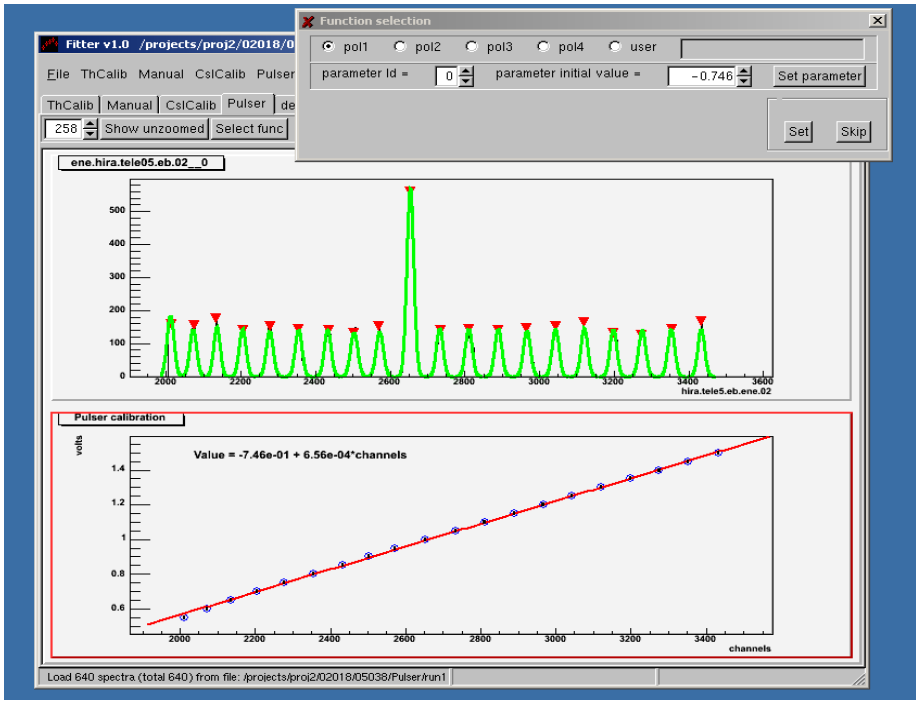
Figure 2.4 Interface for displaying summary histogram

To calibrate bad spectra with double peaks, we need to increase the ThCalib threshold. ThCalib threshold is defined as 1-(max Prob - min Prob) / max Prob. For example, in the configuration file, ThCalib.Alpha.Level.Prob : 814., 1029., 1120., 1034., 1564. Then tresh = 1-(814. - 1029.) / 1564. %. Try to increase threshold parameter by increasing the value of min Prob (eg, ThCalib.Alpha.Level.Prob: 1200., 1029., 1120., 1034.,1564).

2.3 Pulser

Select the Pulser interface. Load the spectra by clicking “Pulser” and “Add Spectra to list”. Use the “up” and “down” button on the keyboard to fit selected spectrum. You can save peak parameter (eg, mean and sigma), fit parameter (slope, offset) and zero point for each spectrum. You can also turn on the “matching log ON” under “Pulser” to save each fitted spectrum into postscript file.

Linear fitting is the default function. To fit the peaks with higher order polynomial, go to “Select func” and choose the order of polynomial function. Self-defined function is available, but the initial value for each parameter should be defined.



Function selection

(✓ pol1) pol2 pol3 pol4 user

parameter Id = 0   parameter initial value = -0.746   Set parameter

258

Value = -7.46e-01 + 6.56e-04*channels

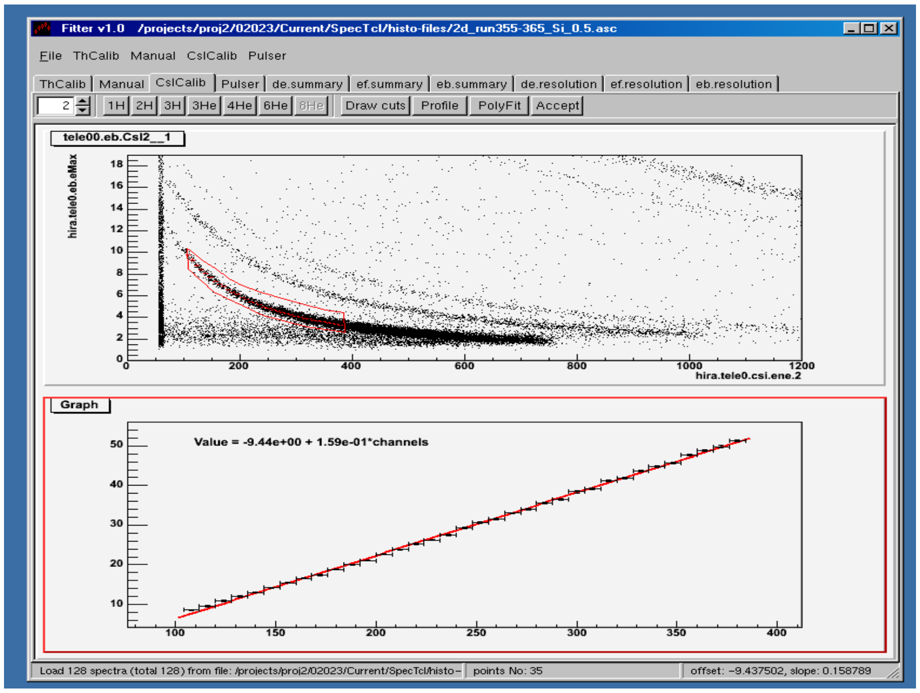
Load 640 spectra (total 640) from file: /projects/proj2/02018/05038/Pulser/run1

Figure 2.5 Interface for pulser fitting

2.4 CsI calibration

Click "CsiCalib" button to bring up the interface. Go to "CsICalib" and "Add Spectra to list". Open spectra file and use the "up" and "down" button on the keyboard to select spectra for calibration. Click the particle button and draw the contour for that particle in the spectra. Then click "Profile". The program will take the verticle cut and get the mean data point in each bin. To close the coutour, click the particle button again. Do the same steps for other particles. After that, click the button "PolyFit" and the program will establish the linear fit for all particles chosen and display it in the lower panel. If the fitting is acceptable, click "Accept" to save the calibrated parameters in database; otherwise repeat the above procedure (as shown in the figure). To use the cuts from the previous spectrum instead of drawing cuts, you can click "Draw cuts" in the new spectrum.

It is important to draw the contour with sharply verticle cuts because of the analysis strategy and not include the data which reach the saturation line (eg: proton data below 2 MeV). Also, we calibrate CsI by either EF or EB. To avoid confusion, it is suggested to select the EF or EB plane in "Select tele and plane" window before starting the calibrations.


Fitter v1.0 /projects/proj2/02023/Current/SpecTcl/histo-files/2d_run355-365_Si_0.5.asc
File ThCalib Manual CsICalib Pulser
ThCalib Manual CsICalib Pulser de.summary ef.summary eb.summary de.resolution ef.resolution eb.resolution
2 1H 2H 3H 3He 4He 6He 8He Draw cuts Profile PolyFit Accept
tele00.eb.CsI2__1
hira.tele0.eb.eMax
hira.tele0.csi.ene.2
Graph
Value = -9.44e+00 + 1.59e-01*channels

```
Load 128 spectra (total 128) from file: /projects/proj2/02023/Current/SpecTcl/histo-   points No: 35   offset: -9.437502, slope: 0.158789
```

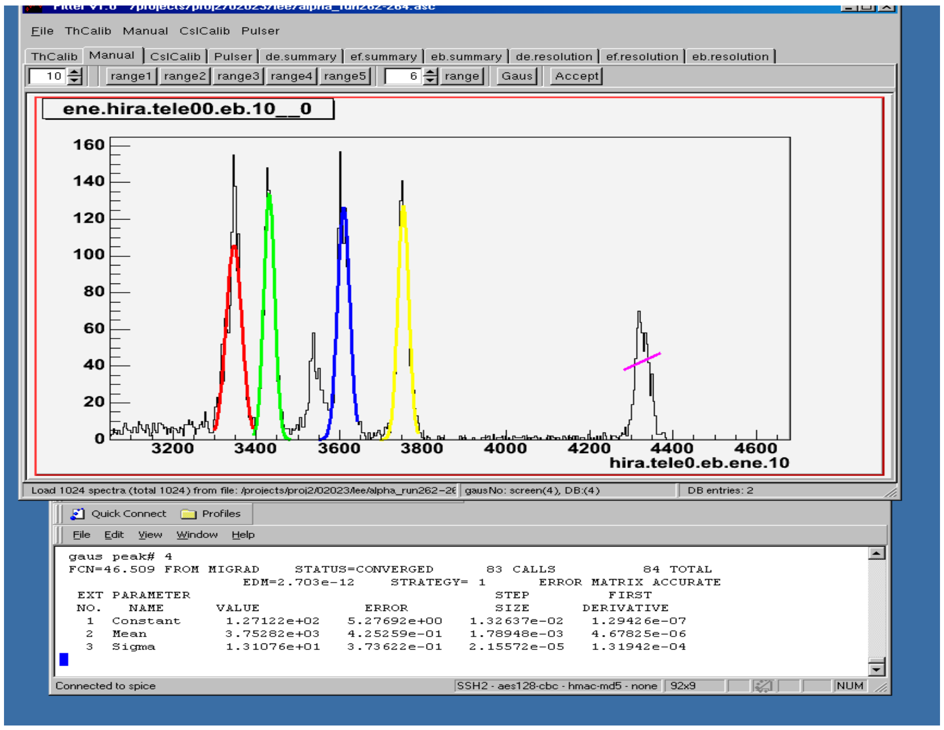
Figure 2.6 Interface of CsI calibration. Red cut is the coutour for proton.

After the calibration is finished, you have to go to "CsICalib" and click "Save offset/slope vdef". If you want to continue the calibration that is not finished before, you can first choose "load offset/slope vdef" and then calibrate the remaining spectra, so that your vdef values will be written in the same vdef file.

You can clear the current loaded Spectra and calibration results in database by clicking "Reset spectra list" and "Reset DB".

2.5 Manual

You can use "Manual" to find the centriod of the peak from Gaussian fitting. Load the spectra from "Maunal" and "Add spectra to list". The icon range1-5 is for peak 1 to peak 5. If more than 5 peaks are needed to be fitted, change the number in the white box next to "range 5" and click "range". After clicking the range icon, go to the corresponding peak and draw a line at the peak (as shown in the figure), then hit the button "Gauss". The peak information (eg mean, sigma) is shown in the SSH window. If the fitting is fine, click "Accept" and the peak information are saved in the database; otherwise re-click the range icon and repeat the procedure.


File ThCalib Manual CsICalib Pulser
ThCalib Manual CsICalib Pulser de.summary ef.summary eb.summary de.resolution ef.resolution eb.resolution
10 range1 range2 range3 range4 range5 6 range Gaus Accept
ene.hira.tele00.eb.10__0
160
140
120
100
80
60
40
20
0
3200
3400
3600
3800
4000
4200
4400
4600
hira.tele0.eb.ene.10
Quick Connect Profiles
File Edit View Window Help
NUM

```
Load 1024 spectra (total 1024) from file: /projects/proj2/02023/lee/alpha_run262-26   gausNo: screen(4), DB:(4)   DB entries: 2
```

```
 gaus peak# 4
 FCN=46.509 FROM MIGRAD    STATUS=CONVERGED      83 CALLS          84 TOTAL
                     EDM=2.703e-12    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE          ERROR          SIZE       DERIVATIVE
   1  Constant     1.27122e+02   5.27692e+00   1.32637e-02   1.29426e-07
   2  Mean         3.75282e+03   4.25259e-01   1.78948e-03   4.67825e-06
   3  Sigma        1.31076e+01   3.73622e-01   2.15572e-05   1.31942e-04
```

```
Connected to spice          SSH2 - aes128-cbc - hmac-md5 - none   92x9
```

Figure 2.7 Interface of peak fitting in manual mode.

Go to "Manual" and "save peak parameters" to save the peak information into file. You can also reset the database and spectra list and print database by clicking the corresponding icons under "Manual"

2.6 Viewer

Viewer provides a nice and quick interface to view specrea. Make sure to put the executive file "viewer" in your bin area.

To run Viewer, go to "File" and "Viewer", then a window will pop up. Go to "Open spectra file" to load the asc file. Click "Disply" and highlight the spectra you want to disply. Click "Apply",then the selected spectra will be displayed one by one automatically.

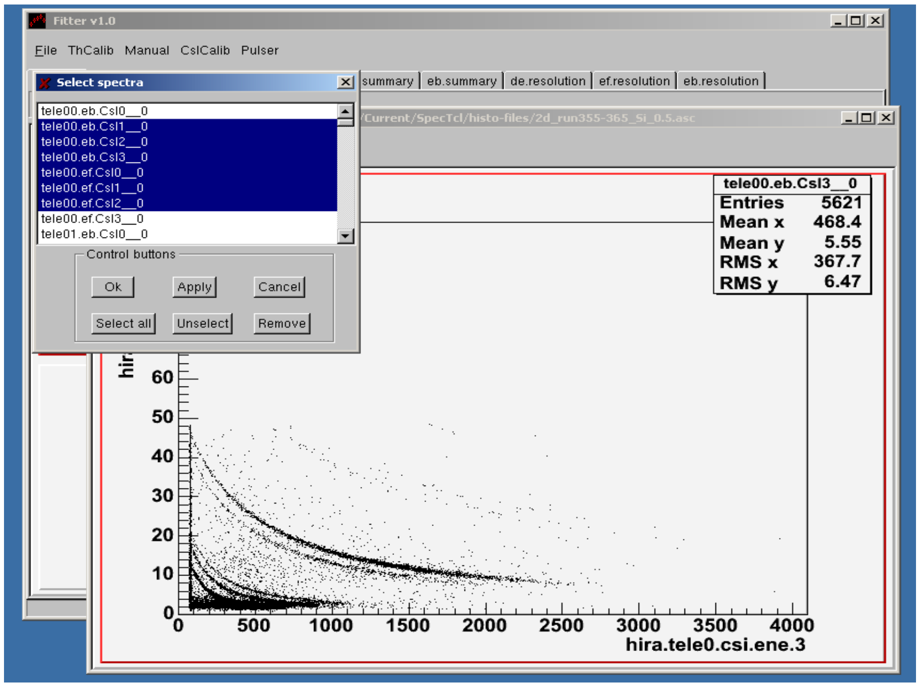


Figure 2.8 Interface of Viewer with selection of spectra.

You can also choose the display geometry, save the active canvas and save the asc file to root file.

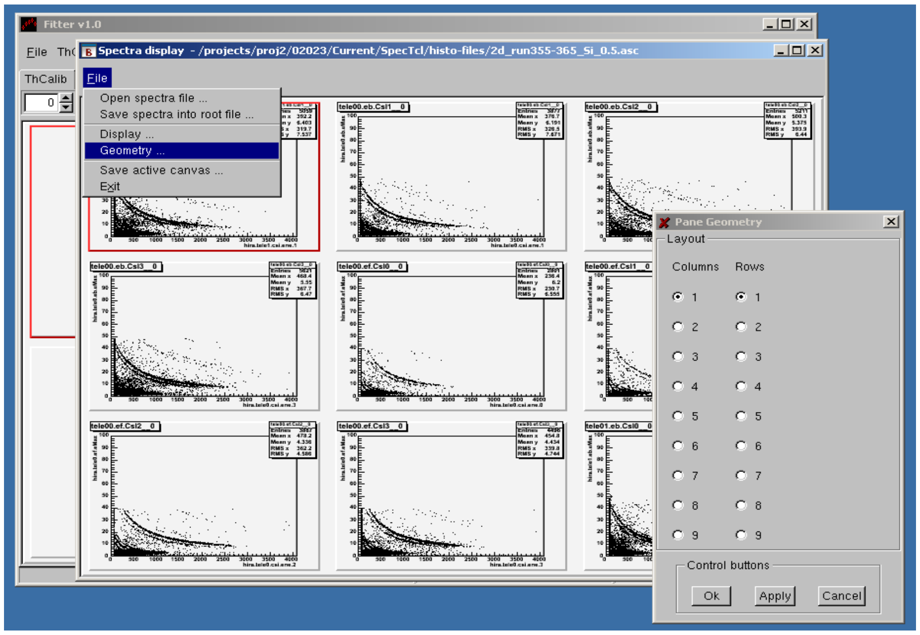


Figure 2.9 Interface of Viewer with display geometry.

Chapter 3

Instructions of UnpackPid Programs

UnpackPid programs are developed to convert a packed file into its original form and unpack the bytes to recreate the structures when reading. Four Unpack programs are developed specifically for HiRA applications. Only histograms are generated (except Etot with tree) after unpacking.

LRootEvent.cfg file contains parameters for Pack and Unpack modes. Parameters used for both Pack and Unpack mode are related to experiment setting. We only need to change the parameters used by Unpack mode only when we run the codes in UnpackPid. Also make sure the root-file path is correct.

Both UnpackPid and SpectclPid use the same configuration file (LRootEvent.cfg). And the configuration files in UnpackPid and SpectclPid are automatically linked.

Meaning of parameters used by Unpack Mode only and file path:

rootfiles.list	Cluster file contains the run numbers (eg. Run200.root)
vdeffiles.list	File contains the names of vdef-files
rootfiles.outflnm	Root file for output
pid.cuts.rootflnm	Root file contain pid cuts
rootfiles.path	Directory where root files are saved

Library: ../SpectclPid/wd/Linux-g++/SpecTcl/libSpecTcl.a

3.1 sum1dRaw

It reads root files listed in "rootfiles.list " and creates 1D histograms for each strip in raw channel.

3.2 sum1d

It reads root files in "rootfiles.list " and creates 1D histograms for each strip in MeV.

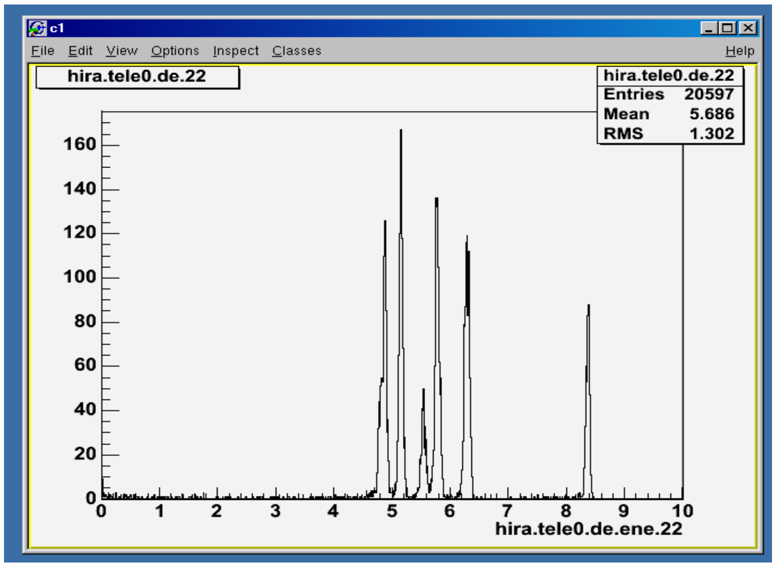


Figure 3.1 Histogram of 1D calibrated alpha peaks for one strip generated by sum1d unpacker.

3.3 Summary

It reads root files in "rootfiles.list " and creates summary histogram for each telescope in MeV.

3.4 CsI_Pid (temp: eb__csi_pid)

It reads root files in "rootfiles.list " and creates 2D histograms of either calibrated Si vs calibrated CsI or calibrated Si vs raw CsI with Pid gates after matching all crystals. Also it creates 1D CsI projection (MeV or Raw) with Pid gates for each particle in every crystal.

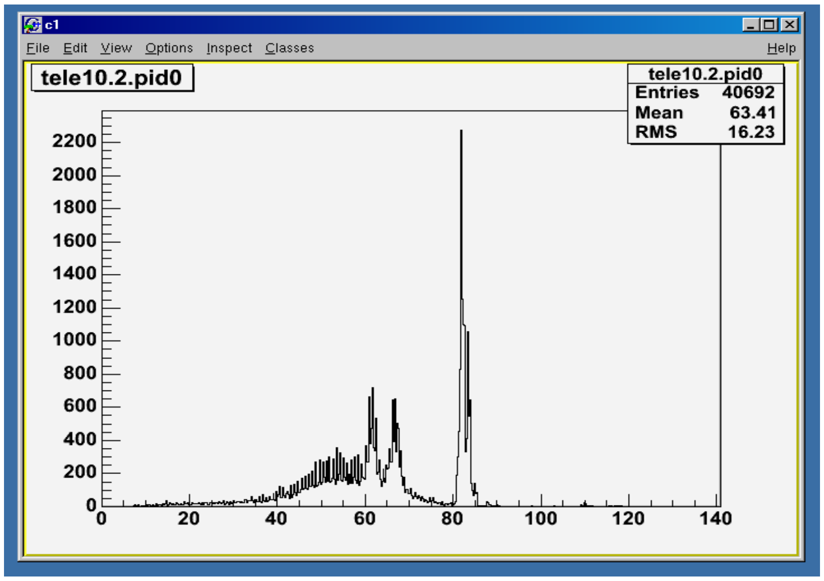
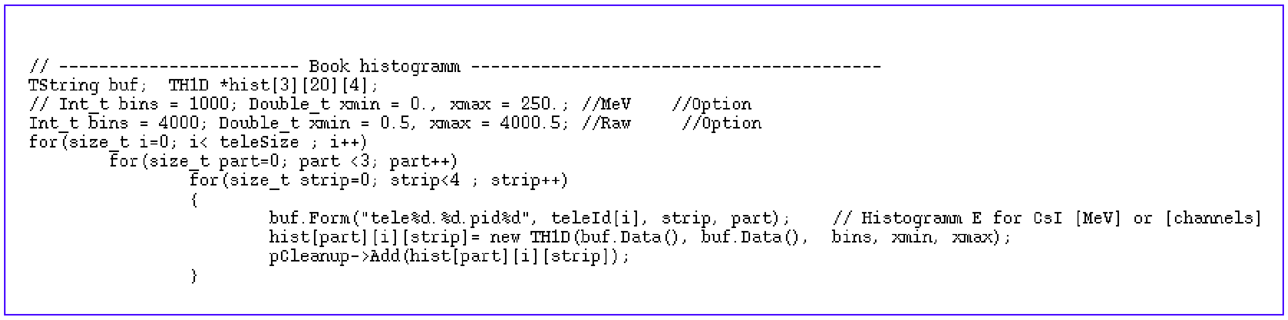
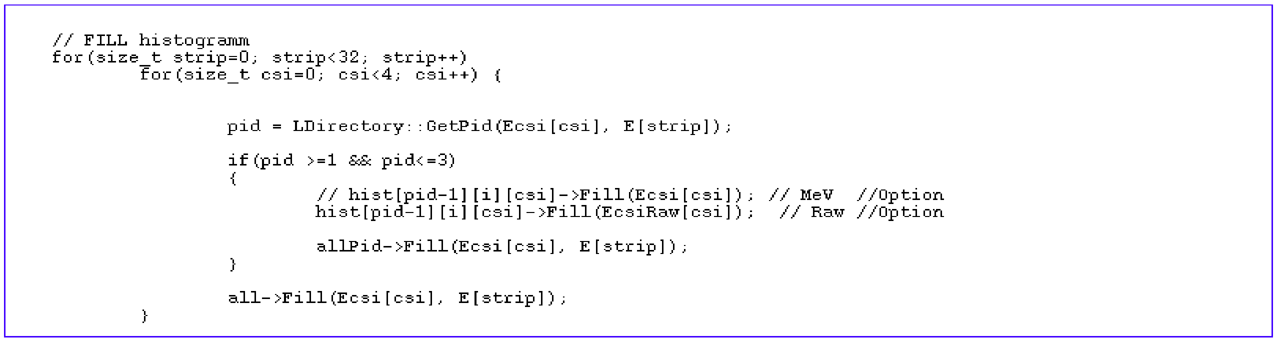


Figure 3.2 Proton peak after proton cut gated on the elastic scattering data.

Edit the source file " CsI_Pid.cc" to choose CsI in MeV or in Raw. See below:



```
// ------------------------ Book histogramm ----------------------------------------
TString buf;  TH1D *hist[3][20][4];
// Int_t bins = 1000; Double_t xmin = 0., xmax = 250.; //MeV    //Option
Int_t bins = 4000; Double_t xmin = 0.5, xmax = 4000.5; //Raw     //Option
for(size_t i=0; i< teleSize ; i++)
        for(size_t part=0; part <3; part++)
                for(size_t strip=0; strip<4 ; strip++)
                {
                        buf.Form("tele%d.%d.pid%d", teleId[i], strip, part);    // Histogramm E for CsI [MeV] or [channels]
                        hist[part][i][strip]= new TH1D(buf.Data(), buf.Data(),  bins, xmin, xmax);
                        pCleanup->Add(hist[part][i][strip]);
                }
```



```
// FILL histogramm
for(size_t strip=0; strip<32; strip++)
        for(size_t csi=0; csi<4; csi++) {


                pid = LDirectory::GetPid(Ecsi[csi], E[strip]);

                if(pid >=1 && pid<=3)
                {
                        // hist[pid-1][i][csi]->Fill(Ecsi[csi]); // MeV  //Option
                        hist[pid-1][i][csi]->Fill(EcsiRaw[csi]);  // Raw //Option

                        allPid->Fill(Ecsi[csi], E[strip]);
                }

                all->Fill(Ecsi[csi], E[strip]);
        }
```

3.5 Si_CsI_Pid (temp: csi_eb_ef_Pid)

It reads root files in "rootfiles.list " and creates 2D histograms of either calibrated Si vs calibrated CsI with Pid gates after matching all crystals. Also it creates 1D Si and CsI projection in MeV with Pid gates for each particle for each Si strip CsI crystal.

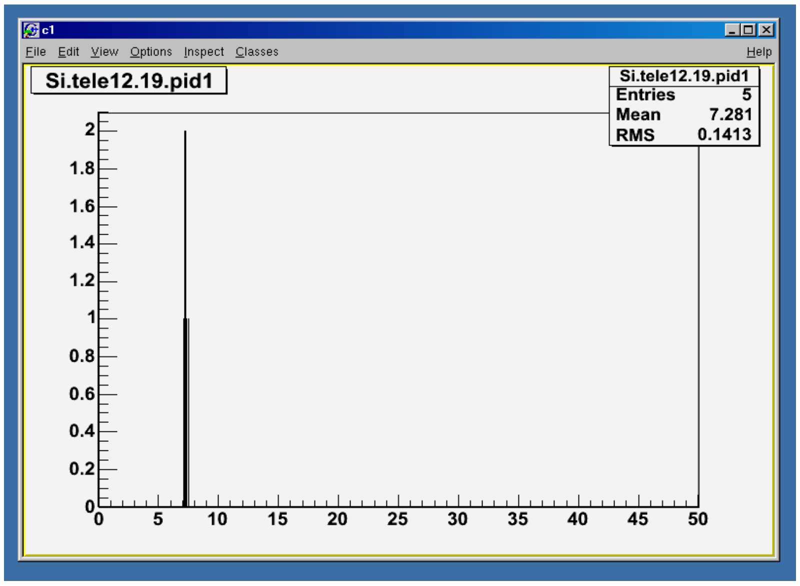
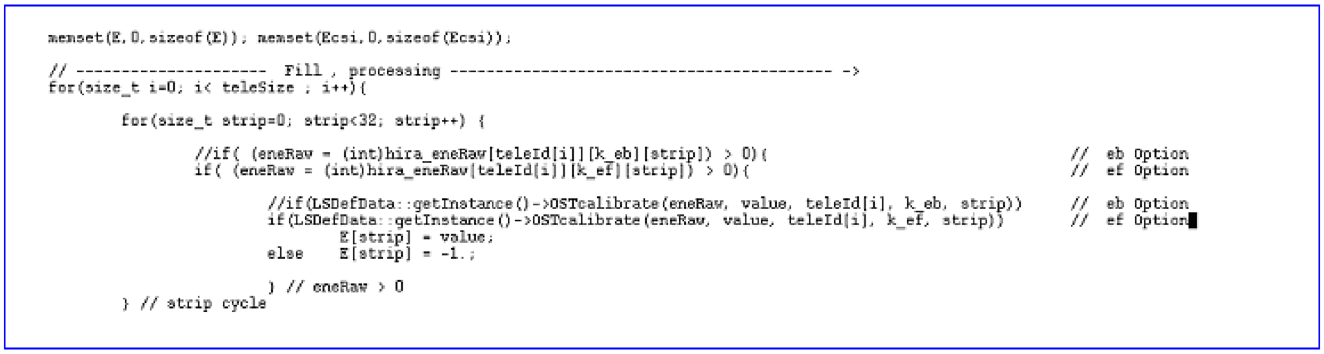


Figure 3.3 Histogram of 1D Si projection from elastic scattering data after gated by deuteron cut.

3.6 E_CsI_mev

It reads root files in "rootfiles.list " and creates 2D histograms of calibrated Si vs CsI for each CsI crystal. Edit the source file " E_CsI_mev" to choose EF vs CsI or EB vs CsI. See below:



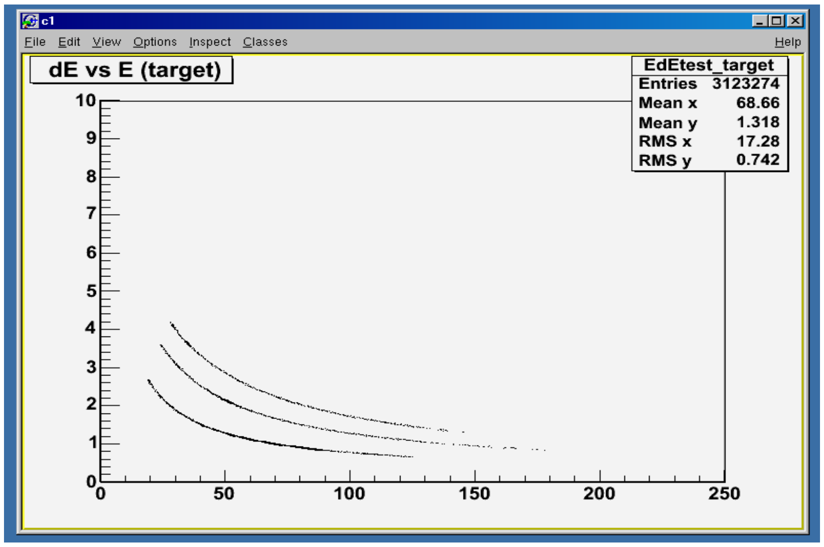
```
memset(E,0,sizeof(E)); memset(Ecsi,0,sizeof(Ecsi));

// ---------------------  Fill , processing ------------------------------------------ ->
for(size_t i=0; i< teleSize ; i++){

        for(size_t strip=0; strip<32; strip++) {

                //if( (eneRaw = (int)hira_eneRaw[teleId[i]][k_eb][strip]) > 0){                              //  eb Option
                if( (eneRaw = (int)hira_eneRaw[teleId[i]][k_ef][strip]) > 0){                               //  ef Option

                        //if(LSDefData::getInstance()->OSTcalibrate(eneRaw, value, teleId[i], k_eb, strip))     //  eb Option
                        if(LSDefData::getInstance()->OSTcalibrate(eneRaw, value, teleId[i], k_ef, strip))      //  ef Option
                                E[strip] = value;
                        else    E[strip] = -1.;

                        } // eneRaw > 0
        } // strip cycle
```

3.7 Etot

Etot reads the final correction file generated from the linear fit of calibrated energy deposited in CsI to Lise calculations and calls the program Ziegler to simulate energy loss in target, deadlayers, mylar foil and Si detectors (or from calibrated Si data).
Then it generates histograms for dE vs E with energy loss in target, deadlayers and mylar foils respectively, plot of Ecsi vs Etot and Trees for the calculated Etotal, Pid for each telescope and the raw data with or without multipity for 1D projection of Si and CsI.

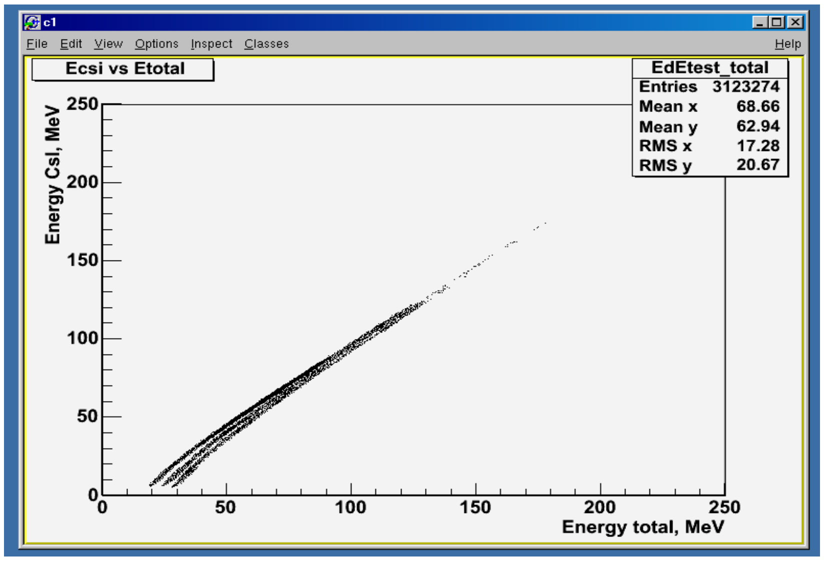
Currently, Etot reads data from /evtdata/02023/lobastov/root-files and write data to root (tree) in /evtdata/02023/lobastov/root-files/calibrated/ directory. You can change path "/evtdata/02023/lobastov/root-files" into LRootEvent.cfg file.

Except L_Ziegler.o, the source, header and library files in Etot are the same as other codes in UnpackPid.



EdEtest_target	
Entries	3123274
Mean x	68.66
Mean y	1.318
RMS x	17.28
RMS y	0.742

Figure 3.4 Histogram of calibrated Si E vs calibrated CsI E with energy loss in the target.



EdEtest_total	
Entries	3123274
Mean x	68.66
Mean y	62.94
RMS x	17.28
RMS y	20.67

Figure 3.5 Plot of energy deposited in CsI vs total energy from Etot Unpacker.

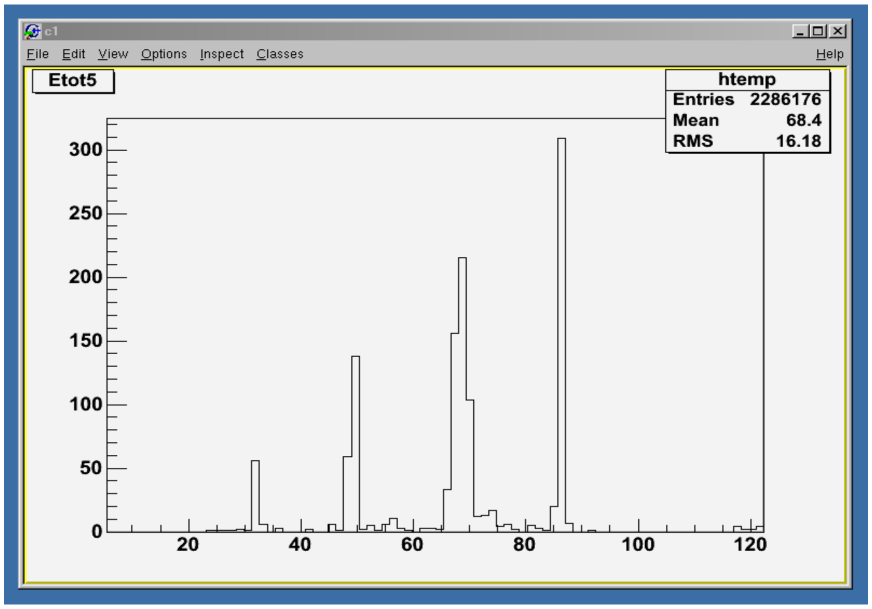


Figure 3.6 Histogram of Pid for each crystal.

Chapter 4

Instructions of Other Relevant Programs

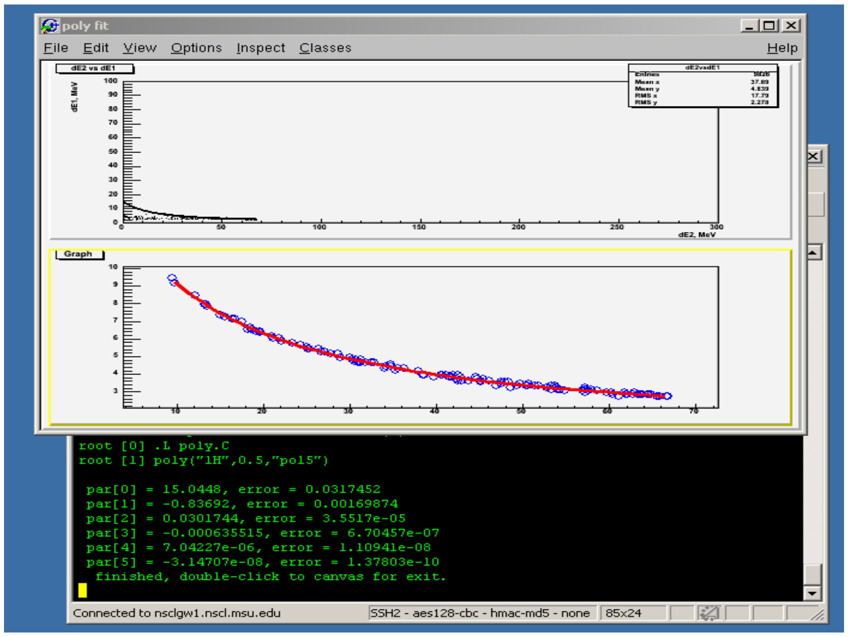
4.1 Geant4 (/projects/proj4/hira/LS_program/Geant4)

Genat4 simulates energies deposited in Si detectors and CsI detectors for different particles. Edit “detector_setup.mac” to change the detector setting (in /mac/ area). To simulate energy deposited:

1. Edit batch.mac to selete which particles will be stimulated . For each particle, serveral commands in batch.mac sare needed.

```
  /user/rootflnm 4Hed.root
  /gps/particle  alpha
  /gps/emin 50. MeV
  /gps/emax 260. MeV
  /run/beamOn 10000
------------------------------
  /user/rootflnm 6Hed.root
  /gps/particle  ion  (except p,d,t and 4He, other particles are named as “ion)
 /gps/ion 2 6 2 0.0 (charge, mass number, charge state)
 /gps/emin 50. MeV
 /gps/emax 260. MeV
 /run/beamOn 10000
```

2. run batch file (batch.run)
3. run root and type ".L poly.C"
4. type "poly("root-filename(no extention)", event threshold,"order of polynomial fitting"). Eg: poly("4He",2,"pol5"). Then the fitting parameters are generated.
5. save the parameters by typing "save(filename)"



```
root [0] .L poly.C
root [1] poly("1H",0.5,"pol5")

 par[0] = 15.0448, error = 0.0317452
 par[1] = -0.83692, error = 0.00169874
 par[2] = 0.0301744, error = 3.5517e-05
 par[3] = -0.000635515, error = 6.70457e-07
 par[4] = 7.04227e-06, error = 1.10941e-08
 par[5] = -3.14707e-08, error = 1.37803e-10
  finished, double-click to canvas for exit.
```

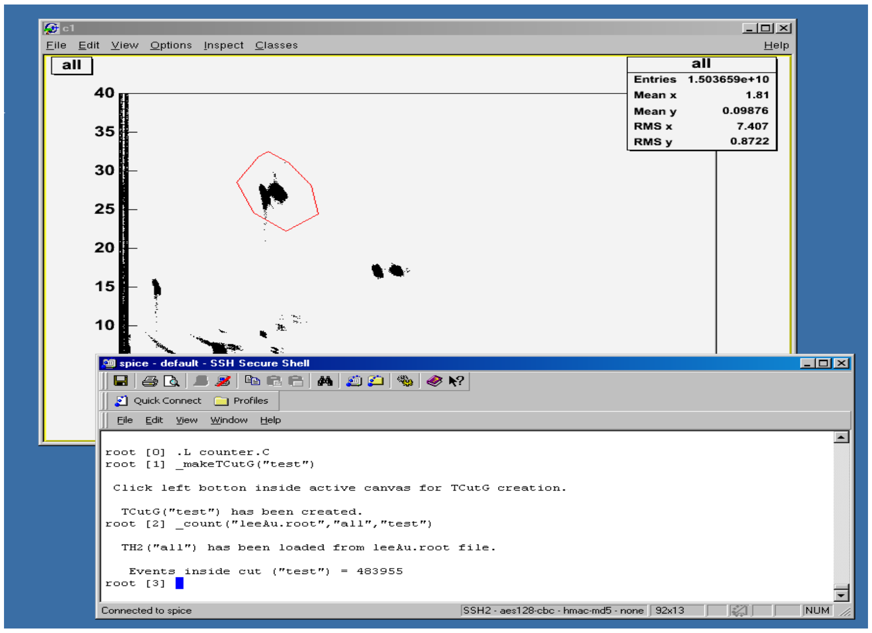
```
Connected to nsclgw1.nscl.msu.edu    SSH2 - aes128-cbc - hmac-md5 - none    85x24
```

Figure 4.1 Geant4 simulations for proton. The parameters are shown in SSH window.

4.2 Count
(/projects/proj4/hira/LS_program/Misc)

Count.C allows us to count the events inside a defined coutour. The procedures are the follows:

1. run root, type "Tbrower b"; open the .root file
2. type ".L counter.C " (draw TH2 into cancas)
3. type "_makeTCutG("cutname")" , then click mouse left botton on TH2 to create coutour.
4. type"_count("filename.root","histname","cutname")",eg _count(   ).The number of event will then displaced.


c1
File Edit View Options Inspect Classes Help
all
40
35
30
25
20
15
10
spice - default - SSH Secure Shell
Quick Connect
Profiles
File Edit View Window Help

```
all
Entries  1.503659e+10
Mean x           1.81
Mean y        0.09876
RMS x           7.407
RMS y          0.8722
```

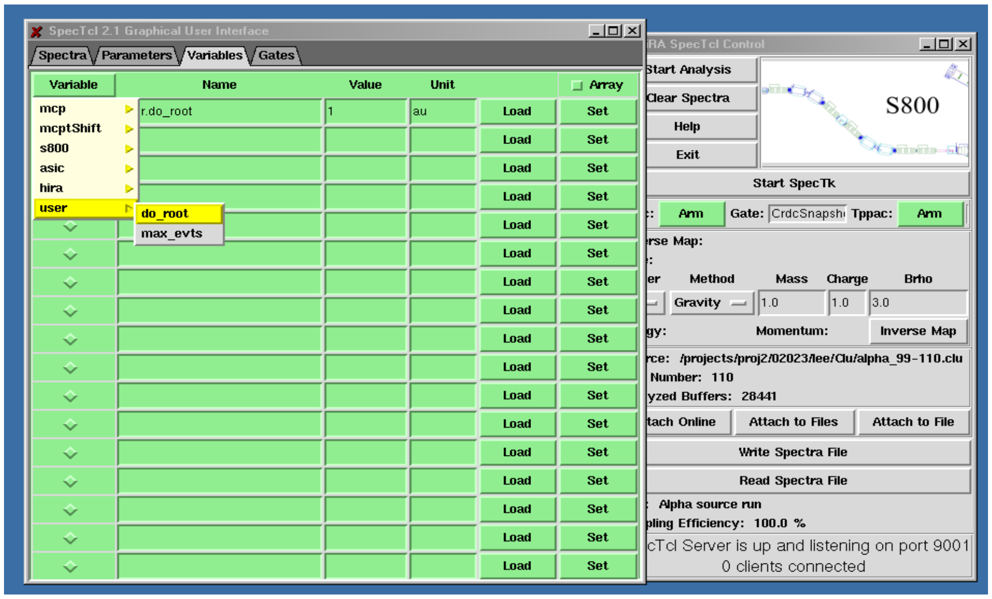
```
root [0] .L counter.C
root [1] _makeTCutG("test")

 Click left botton inside active canvas for TCutG creation.

  TCutG("test") has been created.
root [2] _count("leeAu.root","all","test")

  TH2("all") has been loaded from leeAu.root file.

   Events inside cut ("test") = 483955
root [3]
```

```
Connected to spice          SSH2 - aes128-cbc - hmac-md5 - none   92x13   NUM
```

Figure 4.2 Counting event inside the coutour, number of count is displayed in the SSH window.

4.3 SpectclPid – generation of root files (/projects/proj2/02023/lobastov/SpectclPid)

To generate root file (raw data) with pid parameters, we have to use SpectclPid. First. bring up SpectclPid and go to “Variables” in GUI interface. Then click “Variable” and select “user.do.root”. Change “value” from “0” to “1” and click “Set”. Finally, go to Spectcl Control and attach evt files.



Name	Value	Unit
r.do_root	1	au

```
Gate: CrdcSnapsh
Gravity  1.0  1.0  3.0
rce:  /projects/proj2/02023/lee/Clu/alpha_99-110.clu
 Number:  110
yzed Buffers:  28441
:  Alpha source run
pling Efficiency:  100.0 %
cTcl Server is up and listening on port 9001
0 clients connected
```

Figure 4.3 SpecTclPid interface for generating root files.

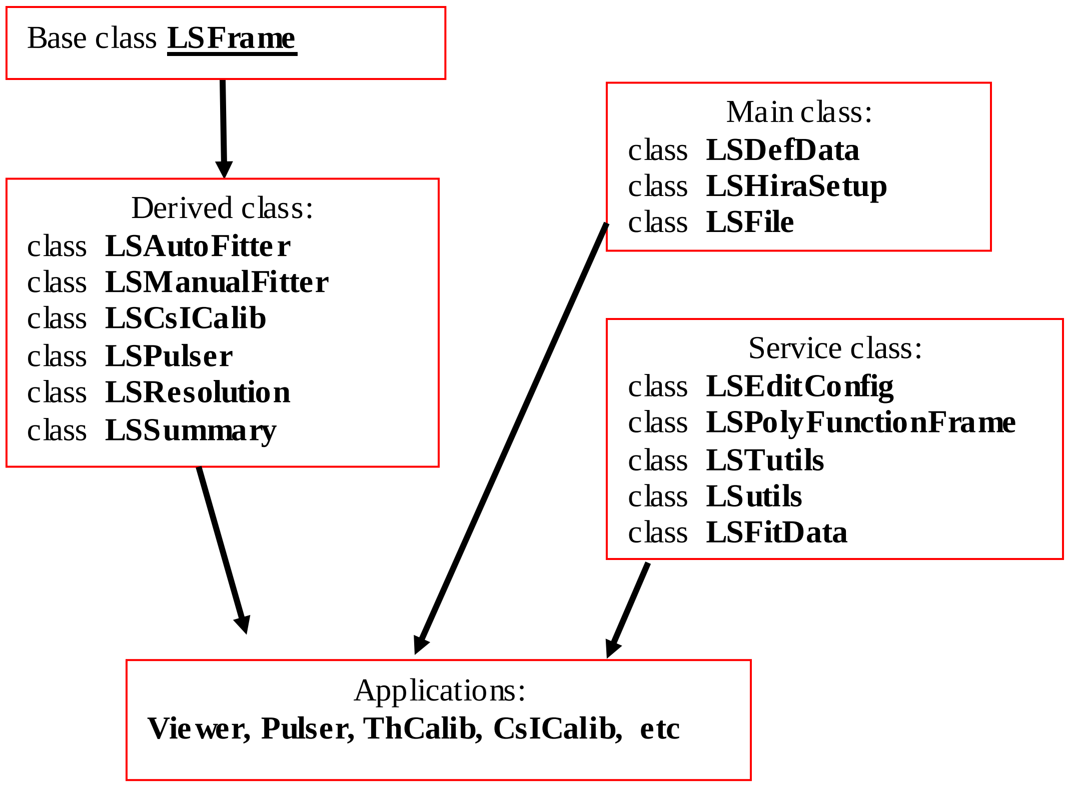
You can edit LRootEvent.cfg file (rootfiles.path) to change path where the root files are saved. Both SpectclPid and UnpackPid use the same configuration file (LRootEvent.cfg). And the configuration files in SpectclPid and UnpackPid are automatically linked.

4.4 Ziegler

A program (no source code) incorprated in Etot code to simulate the energy deposited in deadlayer, mylar foil and detectors.

Chapter 5

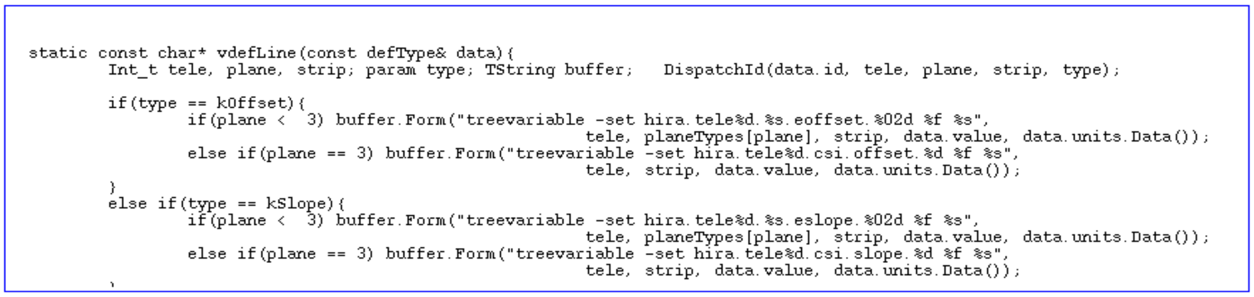
5.1 Structure for Fitter code



Descriptions of Fitter codes:

Header files (include)	**Corresponding source files (src)**	**Descriptions**
LSAutoFitter.hh	LSAutoFitter.cc	ThSource (Auto mode)
LSManualFitter.hh	LSManualFitter.cc	ThSource (Manual mode), Peak100 (for Th or Pulser)
LSResolution.hh	LSResolution.cc	ThSource resolution frame
LSSummary.hh	LSSummary.cc	ThSource summary frame
LSCsICalib.hh	LSCsICalib.cc	CsI Calibration
LSPulser.hh	LSPulser.cc	Pulser
LSFrame.hh	LSFrame.cc	Abstract base class for AutoFitter, ManualFitter,Resolution, Summary, CsICalib and Pulser
LSMainFrame.hh	LSMainFrame.cc	GUI interface for main window (Create main window)
LSDefData.hh	LSDefData.cc	Interface to “database” OST-offset, slope, thres; CORR-correction for CsI calibration; GS-gauss fitting parameters
LSHiraSetup.hh	LSHiraSetup.cc	GUI interface for select tele and plane frame (Create transient window)
LSFile.hh	LSFile.cc	Interface to read, write, looking for, converting from ASCII spectrum to TH1 histogram
LSFitData.hh		Interface for gauss fitting, parameters saving, preparing for fitting
LSEditConfig.hh	LSEditConfig.cc	GUI interface for configuration file editing (Create transient window)
LSPolyFuncFrame.hh	LSPolyFuncFrame.cc	GUI interface for fitting function choice (Create transient window)
LSMultPid.hh	LSMultPid.cc	GUI interface for summary and resolution spectras
LSTutils.hh		Contain some utilits functions (based on ROOT)
LSutils.hh		Contain some utilits functions

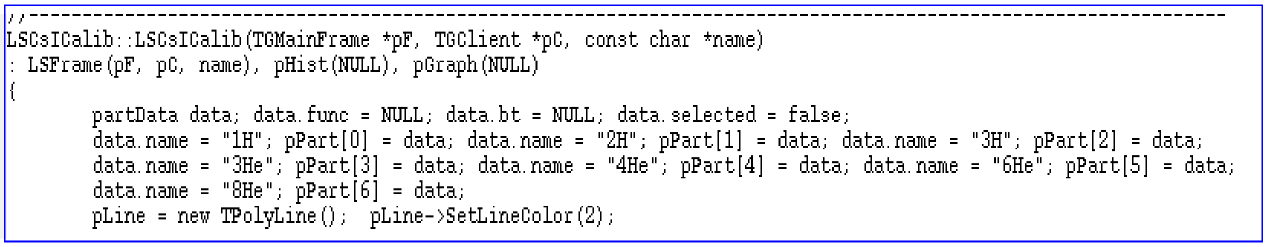
Spectcls often have slightly different naming systems (eg, slope.1 – eslope.01). To be sure the vdef file generated from Fitter can be read by SpecTcl, changes in "LSDefData.hh" is needed (shown as follows) [%02d means two digits]



```
static const char* vdefLine(const defType& data){
        Int_t tele, plane, strip; param type; TString buffer;   DispatchId(data.id, tele, plane, strip, type);

        if(type == kOffset){
                if(plane <  3) buffer.Form("treevariable -set hira.tele%d.%s.eoffset.%02d %f %s",
                                                        tele, planeTypes[plane], strip, data.value, data.units.Data());
                else if(plane == 3) buffer.Form("treevariable -set hira.tele%d.csi.offset.%d %f %s",
                                                        tele, strip, data.value, data.units.Data());
        }
        else if(type == kSlope){
                if(plane <  3) buffer.Form("treevariable -set hira.tele%d.%s.eslope.%02d %f %s",
                                                        tele, planeTypes[plane], strip, data.value, data.units.Data());
                else if(plane == 3) buffer.Form("treevariable -set hira.tele%d.csi.slope.%d %f %s",
                                                        tele, strip, data.value, data.units.Data());
```

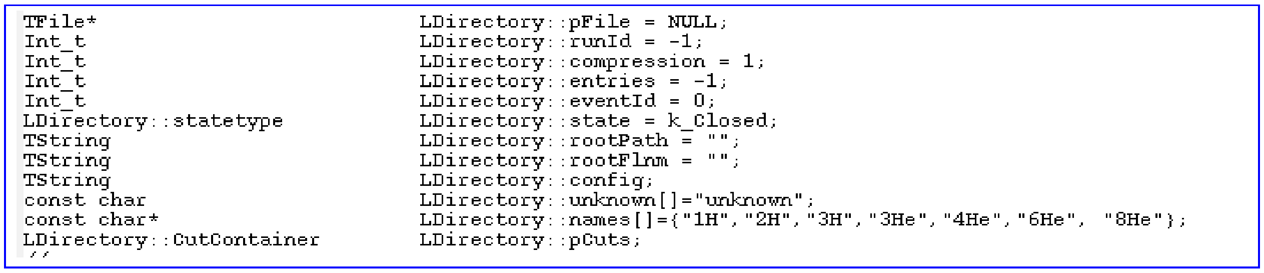
To have more options of particles used for CsI calibrations, you need to run Geant4 first and then edit the file "LSCsICalib.cc" to add the corresponding button in the Fitter interface. Also edit /SpectclPid/src/Ldirectory.cpp/ to include the new particle to Pid gate parameters (See below).

"LSCsICalib.cc":



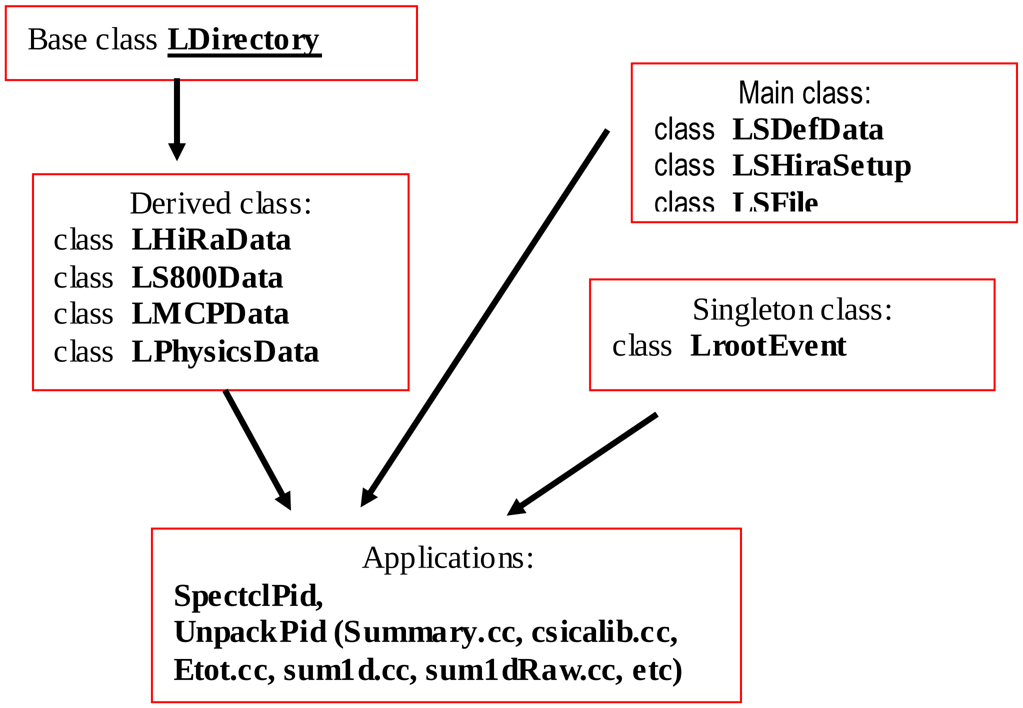
```
//--------------------------------------------------------------------------------------------------------------
LSCsICalib::LSCsICalib(TGMainFrame *pF, TGClient *pC, const char *name)
: LSFrame(pF, pC, name), pHist(NULL), pGraph(NULL)
{
        partData data; data.func = NULL; data.bt = NULL; data.selected = false;
        data.name = "1H"; pPart[0] = data; data.name = "2H"; pPart[1] = data; data.name = "3H"; pPart[2] = data;
        data.name = "3He"; pPart[3] = data; data.name = "4He"; pPart[4] = data; data.name = "6He"; pPart[5] = data;
        data.name = "8He"; pPart[6] = data;
        pLine = new TPolyLine();  pLine->SetLineColor(2);
```

"Ldirectory.cpp":



```
TFile*                          LDirectory::pFile = NULL;
Int_t                           LDirectory::runId = -1;
Int_t                           LDirectory::compression = 1;
Int_t                           LDirectory::entries = -1;
Int_t                           LDirectory::eventId = 0;
LDirectory::statetype           LDirectory::state = k_Closed;
TString                         LDirectory::rootPath = "";
TString                         LDirectory::rootFlnm = "";
TString                         LDirectory::config;
const char                      LDirectory::unknown[]="unknown";
const char*                     LDirectory::names[]={"1H","2H","3H","3He","4He","6He",  "8He"};
LDirectory::CutContainer        LDirectory::pCuts;
```

5.2 Structure for UnpackPid and spectclPid



Descriptions of UnpackPid and SpecTclPid codes:

Header files (include)	**Descriptions**
LRootEvent.hh	Main control class
LDirectory.hh	Abstract base class for HiRA, S800, MCP and Physics; interface for branch creation; open and close root files; read branch list from file; activate(deactivate) branch; etc
LHiRAData.hh	Concrete instance of base class for Hira
LS800Data.hh	Concrete instance of base class for S800
LPhysicsData.hh	Concrete instance of base class for Physics
LMCPData.hh	Concrete instance of base class for MCP

Chapter 6

Others

6.1 How to modify the original SpecTcl to SpecTclPid

To modify the original SpecTcl to the SpecTclPid which contains the Physics Pid parameters, we need to do the following mdifications.

ADD

TreeParameter.h :

```
        class CTreeParameter
        {
        public:
                double*         GetAddress(void){return &value;};          // LSP
                string          GetName(void){return name;};               // LSP
        }
```

CUser.cpp:

```
        #include "LRootEvent.hh"
        CUser::OnBegin(){
                if(do_root) LRootEvent::getInstance()->PackBeginOfRun(RunNumber);
        }
        CUser::OnEnd(){
                cout<<"\n[CUser::OnEnd]"<<flush;
                if(do_root) LRootEvent::getInstance()->PackEndOfRun();
        }
        CUser::OnAttach(){
                vector<CTreeParameter*>::iterator cur; LParameterSet *ptr =
LRootEvent::getInstance()->GetMap();
                for(cur = CTreeParameter::pSelf.begin(); cur != CTreeParameter::pSelf.end(); cur++)
                        ptr->AddNameAddress((*cur)->GetName().c_str(), (*cur)->GetAddress());
                LRootEvent::getInstance()->Initialize();
        }
        CUser::operator(){
                if(do_root && evtNum < max_evts){
                LRootEvent::getInstance()->PackEndOfEvent();
                if( (++evtNum) == max_evts) LRootEvent::getInstance()->PackEndOfRun(); }
        }



void            CUser::Initialize()
{
        do_root.Initialize("user.do_root",0,"au");
        max_evts.Initialize("user.max_evts",1000000,"counts");
}
```

CUser.h:

```
class CUser : public CEventProcessor
{
public:

  CTreeVariable max_evts;
  CTreeVariable do_root;
```

CHiRAS800SpecTclApp.cpp:

```
        #include "LRootEvent.hh"
        CSpecTclApp::~CSpecTclApp(){
                delete LRootEvent::getInstance();
        }
```

Include LphysicsData.cpp

CHANGE

CHiRAS800SpecTclApp.cpp:

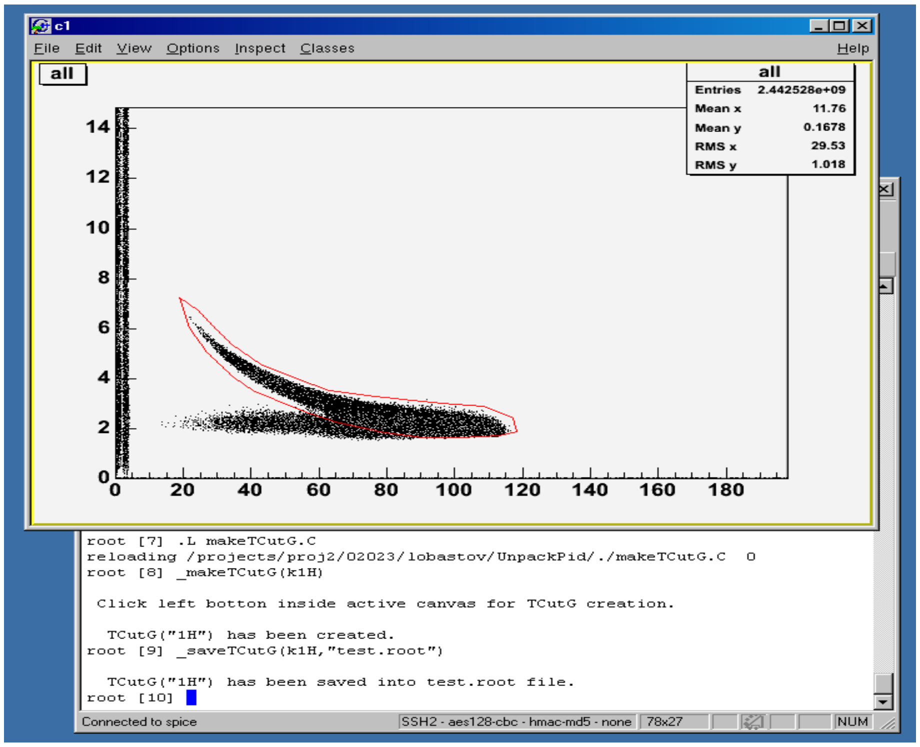
```
        #include "./treeparam/TreeParameter.h"          -->     #include "TreeParameter.h"
        #include "./treeparam/TreeVariable.h"           -->     #include "TreeVariable.h"

        #include "./s800_spectcl/CS800SpecTclApp.h"     -->     #include "CS800SpecTclApp.h"
        #include "./s800_spectcl/CS800.h"               -->     #include "CS800.h"
        #include "./s800_spectcl/CS800Unpacker.h"       -->     #include "CS800Unpacker.h"
        #include "./s800_spectcl/CS800Calibrator.h"     -->     #include "CS800Calibrator.h"
        #include "./s800_spectcl/CS800Calculator.h"     -->     #include "CS800Calculator.h"
        #include "./s800_spectcl/CS800Snapshot.h"       -->     #include "CS800Snapshot.h"
```

CAsic.cpp, CHiracpp, CUser.cpp, CPhysics.cpp, CCsi.cpp, CDecpp, CClassical.cpp:

```
        #include "./s800_spectcl/CS800.h"               -->     #include "CS800.h"
```

6.2 How to create Pid gates from root file

1. root; Tbrowser b; (choose the histogram)
2. type ".L makeTCutG.C"
3. _help()
4. _makeTCutG(k1H) (draw gate)
5. _saveTCut(k1H,"name.root")



```
root [7] .L makeTCutG.C
reloading /projects/proj2/02023/lobastov/UnpackPid/./makeTCutG.C  0
root [8] _makeTCutG(k1H)

 Click left botton inside active canvas for TCutG creation.

  TCutG("1H") has been created.
root [9] _saveTCutG(k1H,"test.root")

  TCutG("1H") has been saved into test.root file.
root [10]
```

Figure 6.1 Generation of cut from histogram.