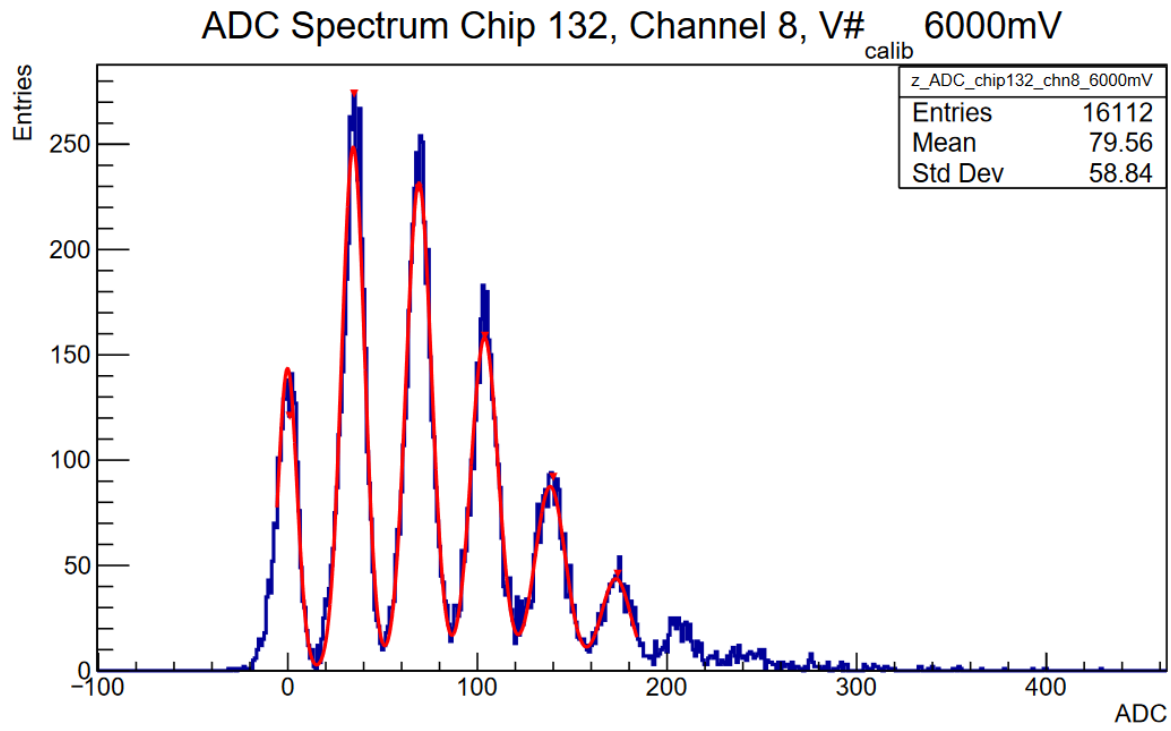
SiPM gain measurement with LED runs

In this exercise you will determine the gain (in arbitrary units) for a SiPM with its readout chain. This will be done by looking into so-called SinglePhotonSpectra (SPS), where very low light intensities are measured in a SiPM, such that you can see individual peaks for the number of fired pixels. The gain is given by the distance between the peaks.



Goal:

- Learn if the gain depends on light intensity
- Learn if the gain depends on SiPM bias voltage

The setting of the light intensity via a voltage for our LED system (Step 1) and the actual measurement (Step 2) are done on a Windows PC with Labview. The data analysis (Steps 3A-D) are done on a Linux PC.

Step 1: Choose the settings you want

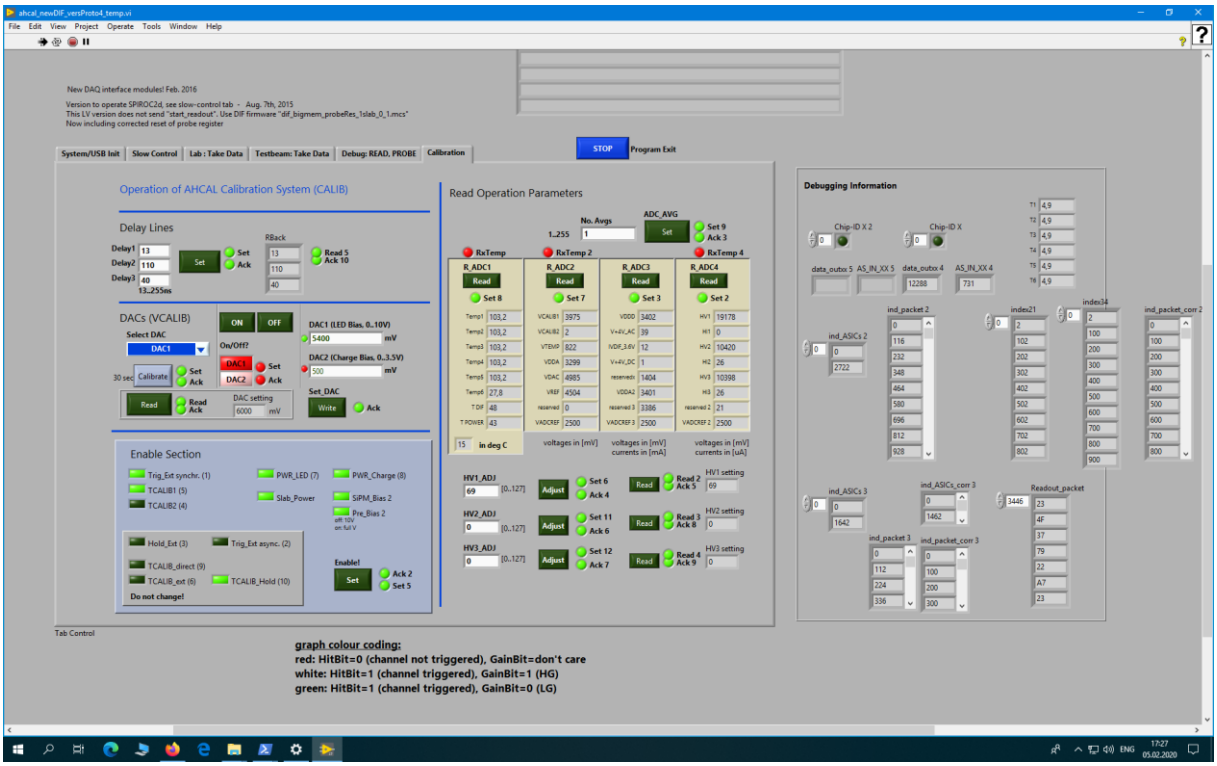
You will take a number of runs with different settings:

1) A pedestal file. The pedestal is the “zero” on your ADC scale, meaning the value you get if there is no signal in the SiPM. You take it with LED voltage set to 0 mV
2) Several runs with different LED voltages. This means the amount of light injected into the scintillator changes.
3) (if time allows) a few runs with the same LED voltage, but changed SiPM bias voltage.

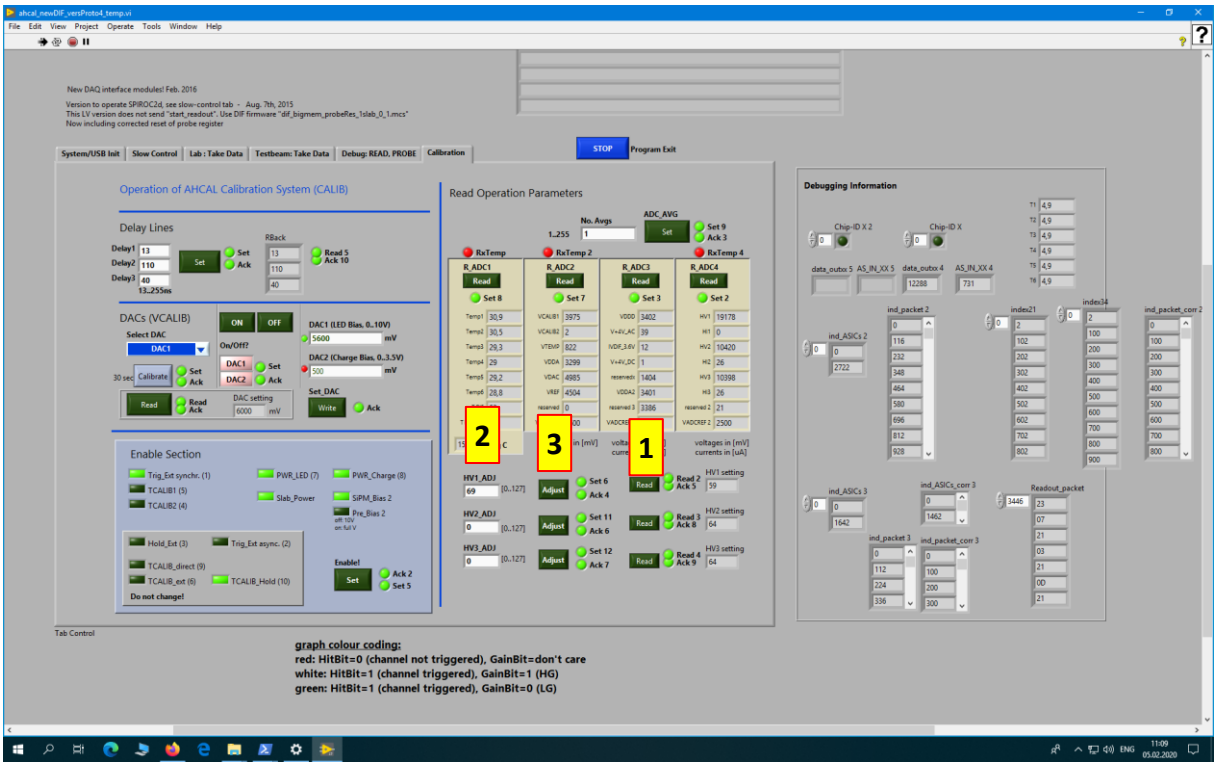
To do the settings

- Go to “Calibration” Tab
    - In the “Delay Lines” section, make sure that values are 13, 110, 40; click on “Set”
    - In “DACs (VCALIB)” section choose “ON”. Enter the LED voltage in the field for “DAC 1” (typically values between 5400 and 6000 mV make sense, 0mV for the pedestal run). Click “Write”
    - in "Enable Section", check that "Trig_Ext synchr. (1)" is on, "TCALIB1 (5)" on and "Pre_Bias 2" on; if not, change and click "Set"
    - the picture below shows the correct settings for a LED run with 5400 mV

  In all steps the indicator next to the “Set” or “Write” should turn (or stay) green. If not, contact instructor.

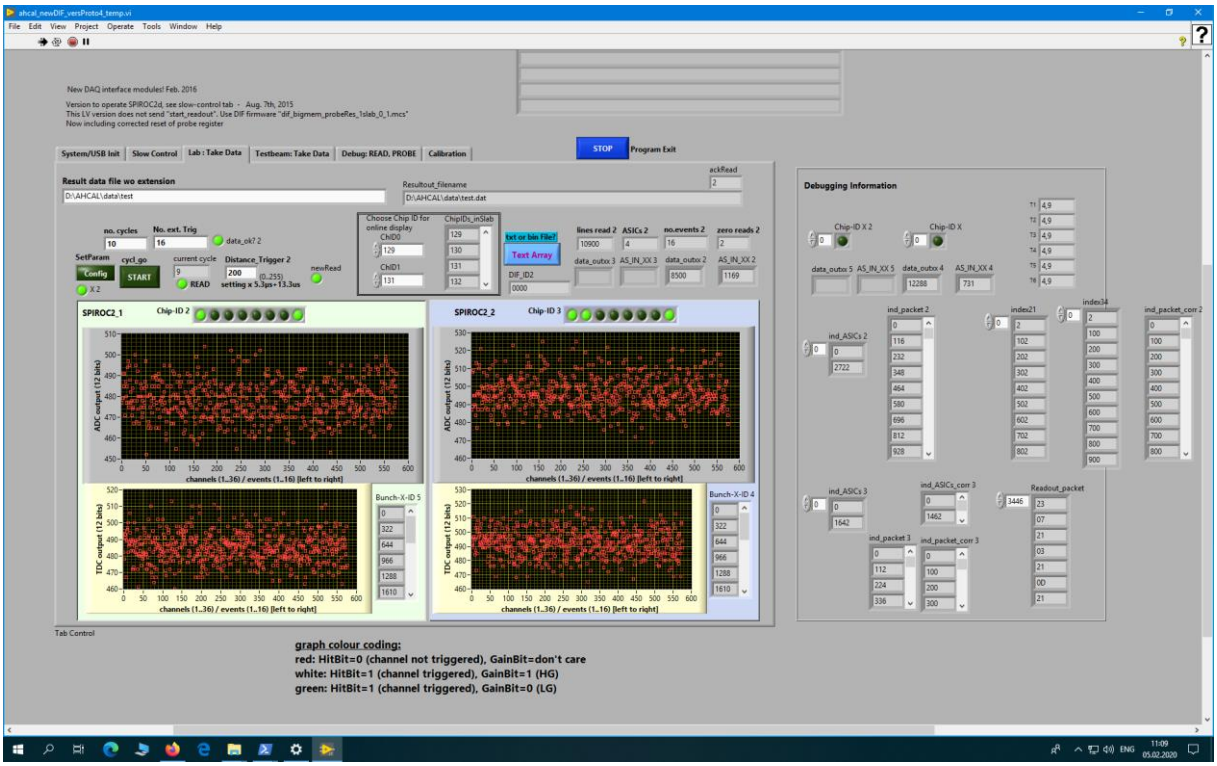


- **if time allows**: change the bias voltage of the SiPM (careful please, this might damage the SiPMs). Go to the “Calibration” tab. The important part now is the lower part of the middle section. The detector provides 3 bias voltage lines, we only use one here, HV1. All three can be adjusted with the help of DACs that take values between 0 and 127 and can change the bias voltage in total by ~4 V. First check the current setting by clicking on “Read”[1] close to HV1 setting. The default value is 69, it corresponds to bias voltage of ~60.9 V (overvoltage of ~5 V). To vary the bias voltage by ~0.5 V up or down, you can change the settings to 83 or 55, respectively. You do that by entering the value below “HV1_ADJ”[2] and clicking on “Adjust” [3]. Check again with “Read”.
    - Then check/change the LED voltage settings, take a run, and analyse it!
    - **Please set the value back to 69 at the end!**



Step 2: Take a run

- if you have not yet created your own data directory, do it now as new folder in D:\AHCAL\data\
- Go to "Lab: Take Data" Tab
    - Enter the new **file name** with the full path for this run (Hint: include “pedestal” or the LED voltage in the file name). **DO NOT FORGET TO CHANGE FOR A NEW RUN!**
    - Choose the number of cycles (will be the number of events per channel and memory cell, should be at least 1000, not more than 3000), make sure that “No. ext. Trig” is 16. Click “Config”.
    - Now you are ready to go! Start the run by clicking on “Start”.
      If everything worked, the “current cycle” should count up, and you should see a moving pattern of red (and maybe white) dots. These are the measured amplitudes for all channels and all memory cells on one ASIC, LowGain in the upper panel, HighGain in the lower panel. Ask your supervisor for more explanations.



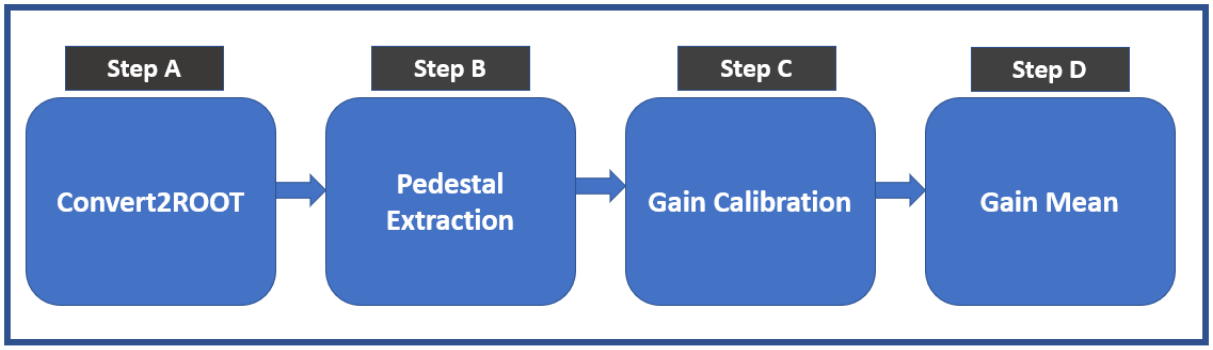
- Take a pedestal run, a run with an LED voltage of 5400mV, and a few more

Step 3: analyze the data

You need a pedestal run and at least one LED run

The analysis of the data is done on the Linux PC, where the data directory of the Labview PC is mounted on /home/calice/EDIT2020/data/LabviewPC/

The analysis proceeds in several steps:



A. **Conversion from text to root file**
B. **Extraction of the pedestal value:** The pedestal defines the "zero" of your measurement. The pedestal position is defined as the position of the amplitude distribution for no signal (LED voltage 0) and is taken as the mean value of a Gaussian fitted to the data. The pedestal position may be different for every channel/chip and also (this is very important) for every memory cell. The spread of the pedestal position among different memory cells in one specific channel affects the data making wider the ADC distributions when we study all memory cell together.
C. **Gain Calibration:** This extracts the gain of the SiPM-ASIC system from so-called SinglePhotonSpectra. To gain statistics we combine all memory cells for a given channel, so we need to subtract the memory-cell-wise pedestal. The program performs a MultiGaussian fit on the Spectra. For comparison it also does a fit with a Fourier Transform (FFT).
D. **Determination of the Mean Gain for each channel:** For many channels we have gain values for more than one LED voltage, so if they are consistent, we can average them.

Step 3A: Convert2ROOT (Converts .txt files to .root files)

The data is saved by the Labview program in ASCII format. This part of the code converts these files to more manageable root files.

The text file contains 12 columns: nHits, iEvt, BunchXID, CycleNr, **ChipID**, **MemoryCell**, **channel**, **HighGain**, **LowGain**, Hit_Bit, Gain_Bit

The relevant for us are:

ChipID: identity of the chip, is not hardcoded in the chip itself but set in the slow control files.

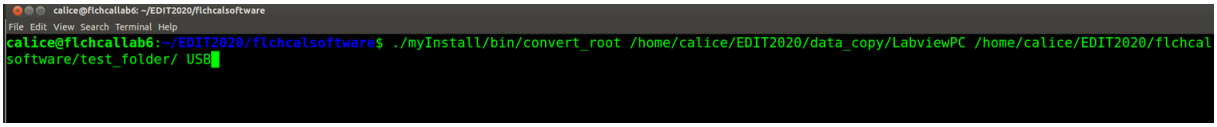
MemoryCell: Memory cell (0-15)

Channel: channel (0-35)

HighGain: signal amplitude measured in the high gain branch of the ASIC, in ADC counts

LowGain: signal measured in the low gain branch of the ASIC, in ADC counts

- Open Terminal
- Go to directory /home/calice/EDIT2020/data and create a new directory for the output files by *mkdir [FOLDER]*
- Go to the folder /home/calice/EDIT2020/flchcalsoftware/ and run the conversion software with
  ./myInstall/bin/convert_root [INPUT FOLDER] [OUTPUT FOLDER] USB



Where:
[INPUT FOLDER] is /home/calice/EDIT2020/data/[directory on windows PC]
[OUTPUT FOLDER] is the new directory you just created

The output is a root file for each *.txt. file in [INPUT FOLDER]. The root files contain a TTree with an entry for each line of the data *.txt file (an entry for each channel that has been stored in the data text file).

Step 3B: Pedestal Extraction

This program extracts a pedestal for each chip, channel and memory-cell. It produces a file that can be used as input for the gain calibration.

The input for this program is the *.root file for a LED scan with 0 voltage.

- Execute the program by *./myInstall/bin/pedestal_memcell [INPUT_PEDESTAL_ROOTFILE] [OUTPUT_FOLDER] 0 0*

Where:

First zero is the (ADC/TDC) mode and second zero is the Trigger mode
The output folder can be the same as before

It produces two output files:

- a .tsv file containing the Mean/RMS pedestal for all memory cells (from 1 to 16) and the relative shift of each memory cell.
- a root file (Pedestal_spectrum.root) with pedestal distributions for all channels, chips and memory cells.

- Look into the Pedestal_spectrum.root and check for a few channels.
- Compare different memory cells for the same channels.

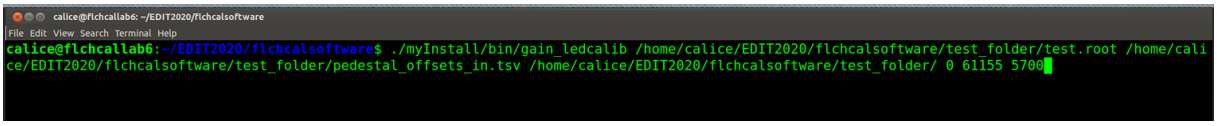
Example of the pedestal_offsets_in.tsv file:

```
#pedestal positions & memory cell dependent offsets (tpedOffsetcellX = tpedOffsetcell2 - tpedOffsetcellX) from file "/
afs/desy.de/group/flc/pool/opinto/tb-cern2018/gain_study_june2018/Long_LED_Run/output_pedestal//pedestal_offsets_in.ts
v"
#format: the ordering of memory cells is inverted for DAQ HBU
#chip   chn     pedposall       pedwidthall     pedOffsetcell1  pedOffsetcell2  pedOffsetcell3  ...     pedOffsetcell1
6
256     0       450.514 5.51522 9.37066 0       2.18018 2.29851 7.85608 2.40712 -0.235079       5.21354 8.6733  12.154
3 3.06178 7.23773 5.48626 3.82292 8.57135 -4.70429
256     1       466.419 5.006   0.474462        0       -4.26731        1.87353 -2.37479        -5.34119        -5.894
9 -7.01072        -6.60015        0.342557        1.51809 -6.92406        0.190545        -2.25133        -4.47039
        -2.03563
256     2       453.306 5.81785 7.03207 0       -0.807887       2.47336 3.01252 2.932   2.50882 -12.6879        12.927
4 -4.01834        -3.16263        2.66048 6.37454 -2.93788        13.92   -0.352516
256     3       463.982 5.13265 -6.48425        0       -0.384504       -0.741546       -10.5726        6.61933 -0.815
641       -8.21029        5.84717 -2.88563        -1.2833 3.63262 -0.363427       -3.49312        -0.997133       -2.1
1912
256     4       457.746 6.34629 -5.61735        0       -5.91627        -12.9096        2.68    -3.97375        1.0999
2 0.698389        -6.27016        -7.53011        4.62408 -3.62077        2.59031 -4.90745        -3.92464        -8.9
533
256     5       465.572 5.49201 -2.80637        0       3.70569 7.33037 3.13887 4.33423 8.44235 -3.54779        2.7530
6 1.03336 2.51668 -2.57602        2.01417 10.1424 4.49813 1.3519
256     6       452.681 4.92537 1.23282 0       6.78861 2.43334 -2.25443        -11.7609        -12.5705        4.9443
1 -3.02123        3.76716 5.2238  -3.86515        0.488468        -0.219628       1.05618 -1.73285
256     7       451.312 6.51683 -9.23429        0       -9.0103 -6.64359        3.56529 -0.208157       2.07766 7.4436
7 -1.00123        4.10543 -0.845514       -7.94528        1.05527 -1.61089        -1.73897        -9.2543
256     8       466.032 5.13085 -6.17827        0       -13.0363        -2.78711        4.97466 -2.98233        -8.813
98        -1.26769        -10.6901        -2.70117        -4.29687        -0.351069       0.846699        -0.724111
        -4.36851        -3.45843
256     9       452.654 5.21179 -3.99949        0       8.6847  1.56489 -5.67668        6.15891 -5.76718        3.2912
1 6.65758 1.37883 4.29299 9.40468 4.12511 4.1181  2.30235 6.34775
256     10      447.302 6.03732 -0.980367       0       -3.13046        -9.45597        -4.29823        -3.70268
        -4.58298        -3.67885        -0.319258       2.55512 2.69124 5.91702 -5.59083        1.56871 5.37862 -8.987
pedestal_offsets_in.tsv
```

Step3C: Gain Calibration

This program performs the extraction of the gain for each chip and channel. It takes a rootfile of a standard LED run as input and the pedestal file (.tsv)) in order to produce ADC spectra for each channel and then fit them.

- Execute the program by *./myInstall/bin/gain_ledcalib [INPUT ROOT FILE] [INPUT_PEDESTAL_FILE] [OUTPUT_FOLDER] 0 [RUN NUMBER] [VCALIB]*

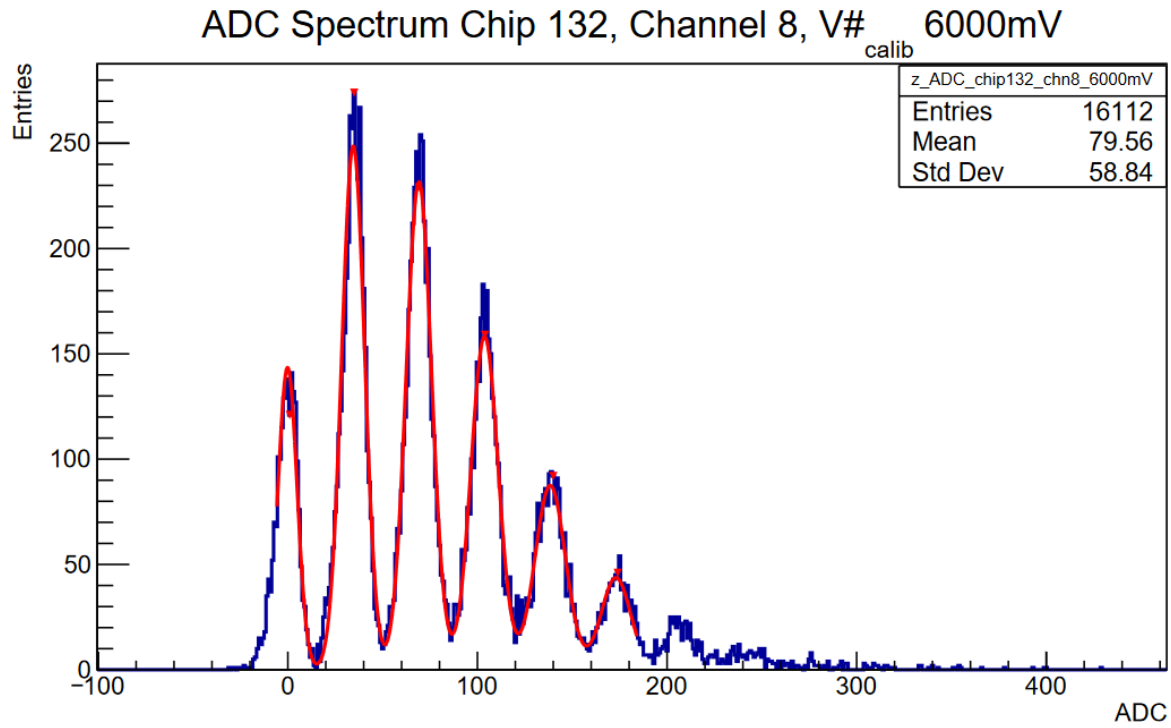


Where:
VCALIB is the LED voltage
RUN NUMBER is arbitrary

The output consists of several files (XXXX is the LED voltage in mV):

- Fit_qualityXXXX.root: contains results related to the quality of the fits
- SPS_allXXXX.root: contains all the ADC histograms (for all channels and chips)
- SPS_fitXXXX.root: contains the fitted histograms and the histograms that satisfy some minimum requirements.
- gainfits_XXXX.tsv: text file with the gain values. Six columns: Chip, channel, gain (multigaussian), gain (FFT), errorgain (multigaussian), Chi2(multigaussian)

Example of a SPS with MultiGaussian fit



Look into SPS_fitXXXX.root and SPS_allXXXX.root

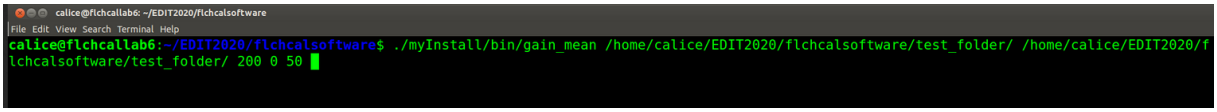
- Why are some channels not fitted?

- Compare one channel for different LED voltages.
    - Does the gain depend on LED voltage?

Step 3D: Gain Mean

If the gain determined for a given channel could be determined for more than 1 LED voltage, we can improve the uncertainty by averaging the gain values. In addition, for channel where no gain could be determined for any LED voltage, the average gain of all channels on that chip is taken.

- *run the code by ./myInstall/bin/gain_mean [INPUT FOLDER] [OUTPUT_FOLDER] 200 0 [MAX GAIN]*

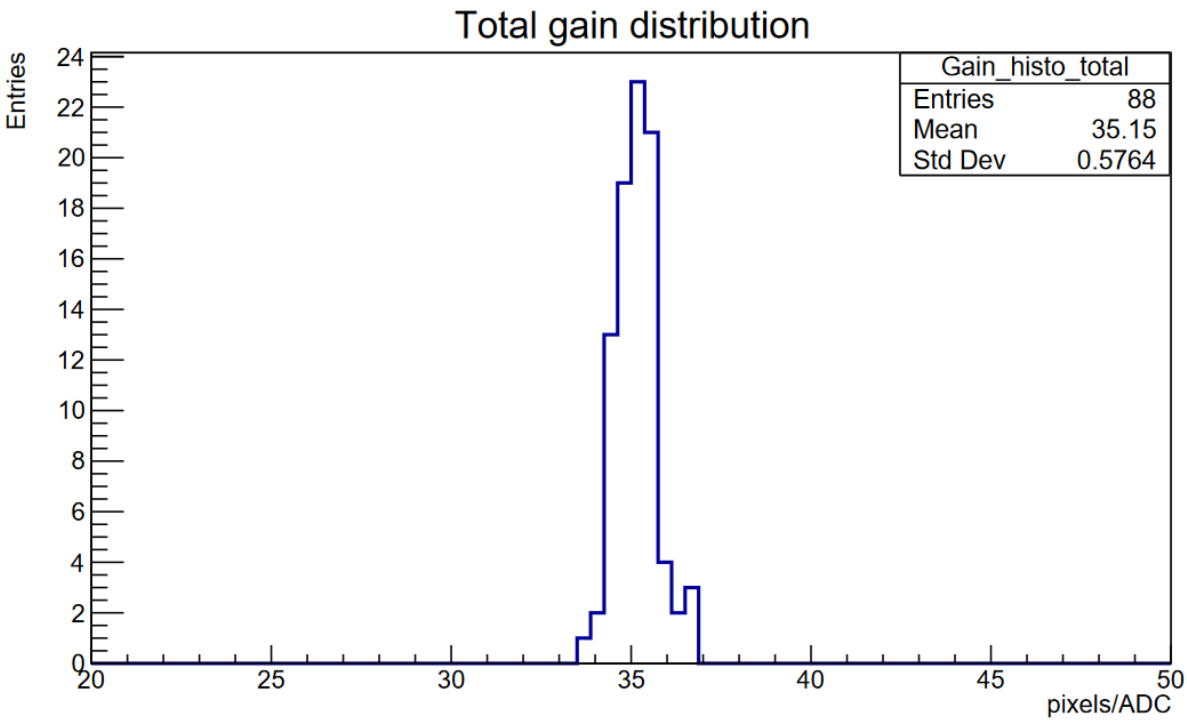


where [MAX GAIN] is the maximum gain you expect (50 is a reasonable value)

The output consists of

- Gain_distribution.root: Contains the gain distribution for all channels and the gains from individual channels
- MeanGainList.txt: text file with the gain values

Example of a gain distribution:



- Look into Gain_distribution.root

    - How much does the gain vary within a chip? How much for the whole board?
- If you have taken data for several SiPM bias voltages: compare them
    - How does the gain depend on the bias voltage? How on the overvoltage? (create a new output folder for different bias voltages)