# Simulations

*The roadmap to the “perfect” CLAS12 MC detector response*

What we need
What is working
What is not working

# Needs for CLAS12 simulations

- More realistic detector response
  - better digitization
  - use CCDB calibration constants

- FADC, multihit signal types
  - trigger
  - commissioning with and w/o beam

- Documentation
  - can always be improved
  - still installation issues

- Java Detector interfaces

- Code optimization

# Digitization Status

| Detector | Sensitivity | Digitization |
|---|---|---|
| BST | 3/4 regions, 2 layers/region, 3 modules/layer, 256 variable angle strips. Charge sharing. electronic noise. | 3 bit ADC, region/layer/strip |
| Micromegas | 3/4 regions, 2 layers/region, 3 tiles/layer, 1000 strips/tile. Charge sharing. Lorentz angle. | 12 bit ADC, region/layer/tile/strip |
| CTOF | 58 scintillators, PMT q.e., attenuation length, effective velocity | region/paddle ADC TDC |
| CND | 3/4 layers, 48 scintillators each, PMT q.e., attenuation length, effective velocity, birks effect, paddle resolution | region/layer/paddle ADC TDC |
| HTCC | 12 sectors, 4 layers. Wavelength-dependent PMT q.e., gas and mirror refraction indexes | sector/layer, PMT, nphe |
| DC | 3 region, 2 superlayers/region, 6 layers/SL. DOCA, drift velocity, cell resolution | sector/region/SL/layer/wire, TDC |
| LTCC | 6 sectors, 2 regions, 18 PMT / region. Wavelength-dependent PMT q.e., gas and mirror refraction indexes | sector/region, PMT, nphe |
| FTOF | 6 sectors, 3 panels, 5/23/62 paddles/panel, left right PMT | sector, panel, (un) smeared ADC TDC |
| PCAL | 15 layers, u,v,w views, 24 scintillator/view, attenuation length, effective velocity, PMT gain, nphe/charge | sector/stack/view/PMT ADC TDC |
| EC | 39 layers, u,v,w views, 36 scintillator/view, attenuation length, effective velocity, PMT gain, nphe/charge | sector/stack/view/PMT ADC TDC |
| RICH | Wavelength-dependent PMT q.e., gas and mirror refraction indexes, multi-channel PMT | PMT, ADC, TDC |
| FT | Light Yield for PbW04, APD q.e, gain, noise | PMT, ADC, TDC |

# Detector Response

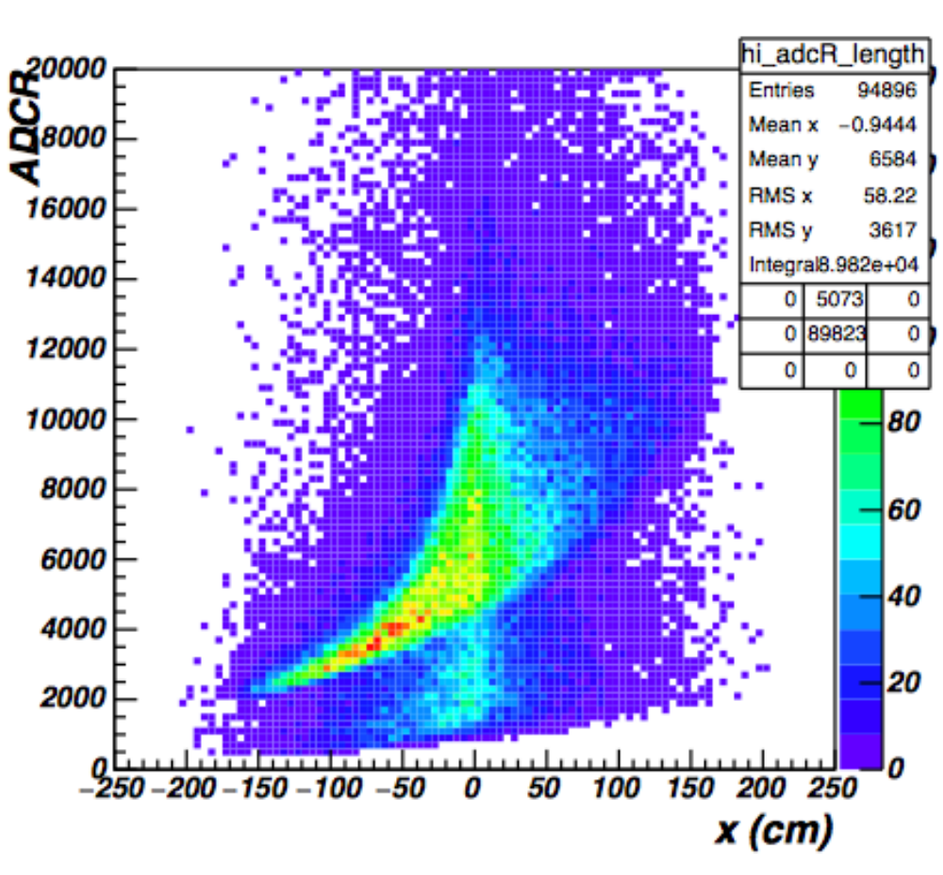
- Working Templates (FTOF, EC):
  - “advanced” digitization
  - use CCDB calibration constants
  - detector “status”

ADC:
- Attenuation according to exponential law
- Conversion from energy to ADC based on MIP signal (dEdxMIP=2 MeV/cm, countsForAMinimumIonizing=2000)

TDC:
- Delay due to light propagation in the paddle (effective velocity)
- Parameterized  Time Walk
- Gaussian time spread based on parameters that will be matched to data (σ2=σ02+σ12/√EPMT Conversion from time to TDC (time2tdc=20ns-1)

Output: both “smeared” and “unsmeared” TDCs

Status:
0 – fully functioning
1 – noADC
2 – noTDC
3 – noADC, noTDC (PMT is dead)
5 – any other reconstruction problem



# Status Weighted by each run luminosity

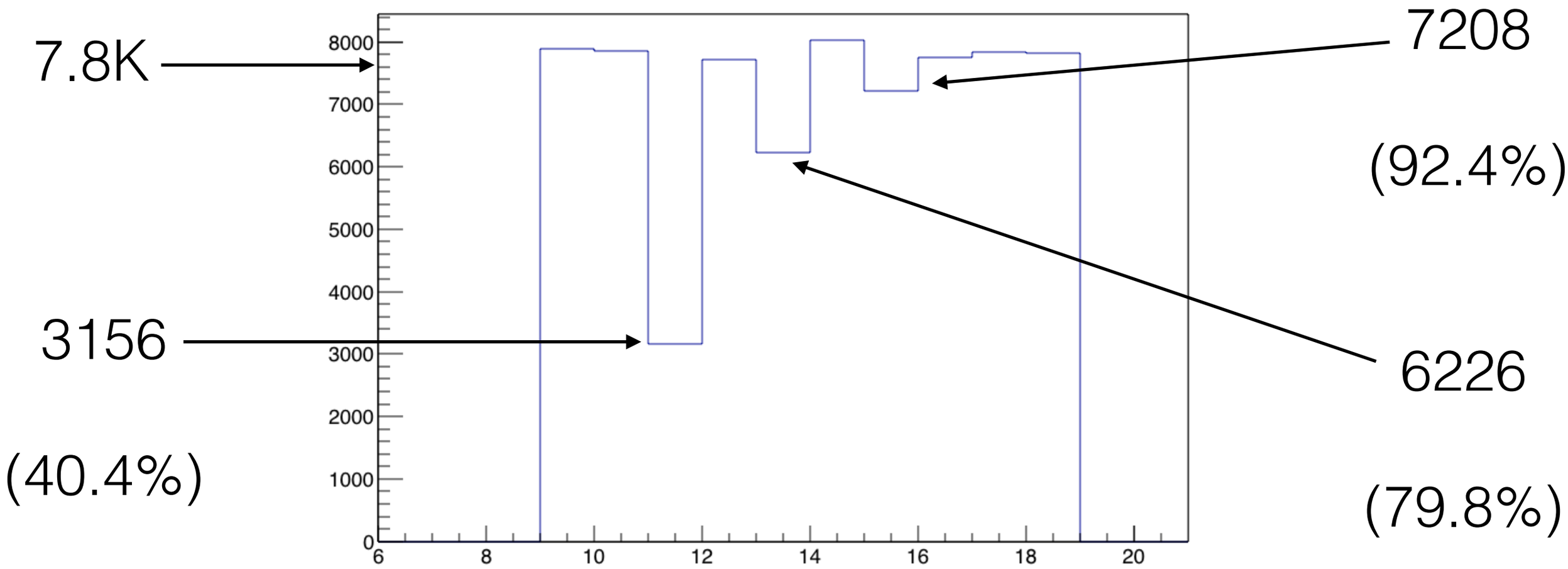
TEST: FTOF paddle

**-RUN_WEIGHTS="runs.txt" -N=100000**

```
> Run weights table loaded:
   - run: 2     weight: 0.1      n. events: 10043
   - run: 13    weight: 0.6      n. events: 59901
   - run: 22    weight: 0.2      n. events: 20034
   - run: 30    weight: 0.1      n. events: 10022
```

DB status “3”

for paddle 11 in run 13
for paddle 13 in run 22
for paddle 15 in run 30



# Detector Response

- Working Templates (FTOF, EC):
  - “advanced” digitization
  - use CCDB calibration constants
  - run-dependent luminosity-weighted “status”

**Working?**

In principle (software-wise), but:
Needs parameters tweaking / optimization

**Plan:**

- Work more closely with CALCOM (especially during commissioning)
- Have dedicated meetings for each detector
- Needs streamlined mechanism for misalignments
- Reconstruction is helping with this

# FADC, multi-hit signal types

```
> BST  100, 0

  > True Step by Step infos   (101, 0)
    - Edep                    (101, 1)
    - Pid                     (101, 2)
    - positions               (101, 3)

  > Dgtz Step by Step infos   (102, 0)
    - ADCL                    (102, 1)
    - ADCR                    (102, 2)

  > True Integrated infos     (103, 0)
    - Edep                    (103, 1)
    - Pid                     (103, 2)
    - positions               (103, 3)

> Dgtz Integrated infos      (104, 0)
    - ADCL                    (104, 1)
    - ADCR                    (104, 2)

> Voltage as a function of time (105, 0)
    - Identifier                (105, 1)
    - Time                      (105, 2)
    - Voltage                   (105, 3)

> FADC Bank                (106, 0)
    - Identifier              (106, 1)
    - Time                    (106, 2)
    - Voltage                 (106, 3)
```

Detector Time Window (DTW)

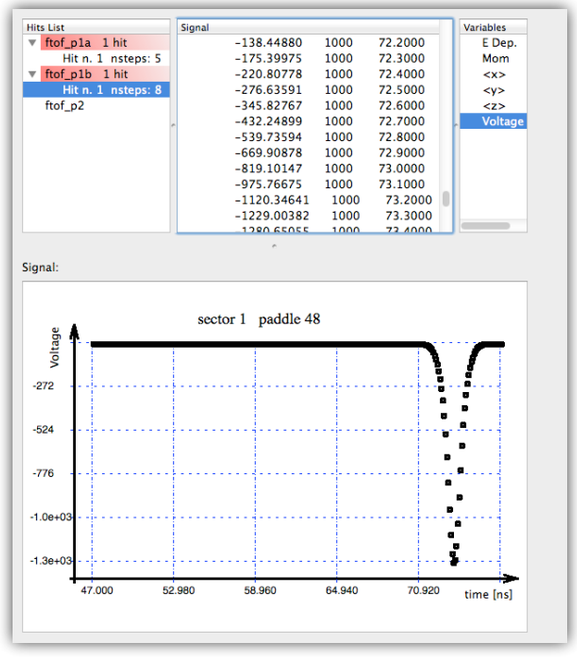
Individual geant4 steps are integrated over DTW

```
 pid",     "Ri", "ID of the first particle entering the sensitive volume");
"mpid",    "Ri", "ID of the mother of the first particle entering the sensitive volume");
"tid",     "Ri", "Track ID of the first particle entering the sensitive volume");
"mtid",    "Ri", "Track ID of the mother of the first particle entering the sensitive volume");
"otid",    "Ri", "Track ID of the original track that generated the first particle entering the sensitive volume");
"trackE",  "Rd", "Energy of the track");
"totEdep", "Rd", "Total Energy Deposited");
"avg_x",   "Rd", "Average X position in global reference system");
"avg_y",   "Rd", "Average Y position in global reference system");
"avg_z",   "Rd", "Average Z position in global reference system");
"avg_lx",  "Rd", "Average X position in local reference system");
"avg_ly",  "Rd", "Average Y position in local reference system");
"avg_lz",  "Rd", "Average Z position in local reference system");
"px",      "Rd", "x component of momentum of the particle entering the sensitive volume");
"py",      "Rd", "y component of momentum of the particle entering the sensitive volume");
"pz",      "Rd", "z component of momentum of the particle entering the sensitive volume");
“vx",      "Rd", "x component of primary vertex of the particle entering the sensitive volume");
"vy",      "Rd", "y component of primary vertex of the particle entering the sensitive volume");
"vz",      "Rd", "z component of primary vertex of the particle entering the sensitive volume");
"mvx",     "Rd", "x component of primary vertex of the mother of the particle entering the sensitive volume");
"mvy",     "Rd", "y component of primary vertex of the mother of the particle entering the sensitive volume");
"mvz",     "Rd", "z component of primary vertex of the mother of the particle entering the sensitive volume");
"avg_t",   "Rd", "Average time");
"hitn",    "Ri", "Hit Number");
```

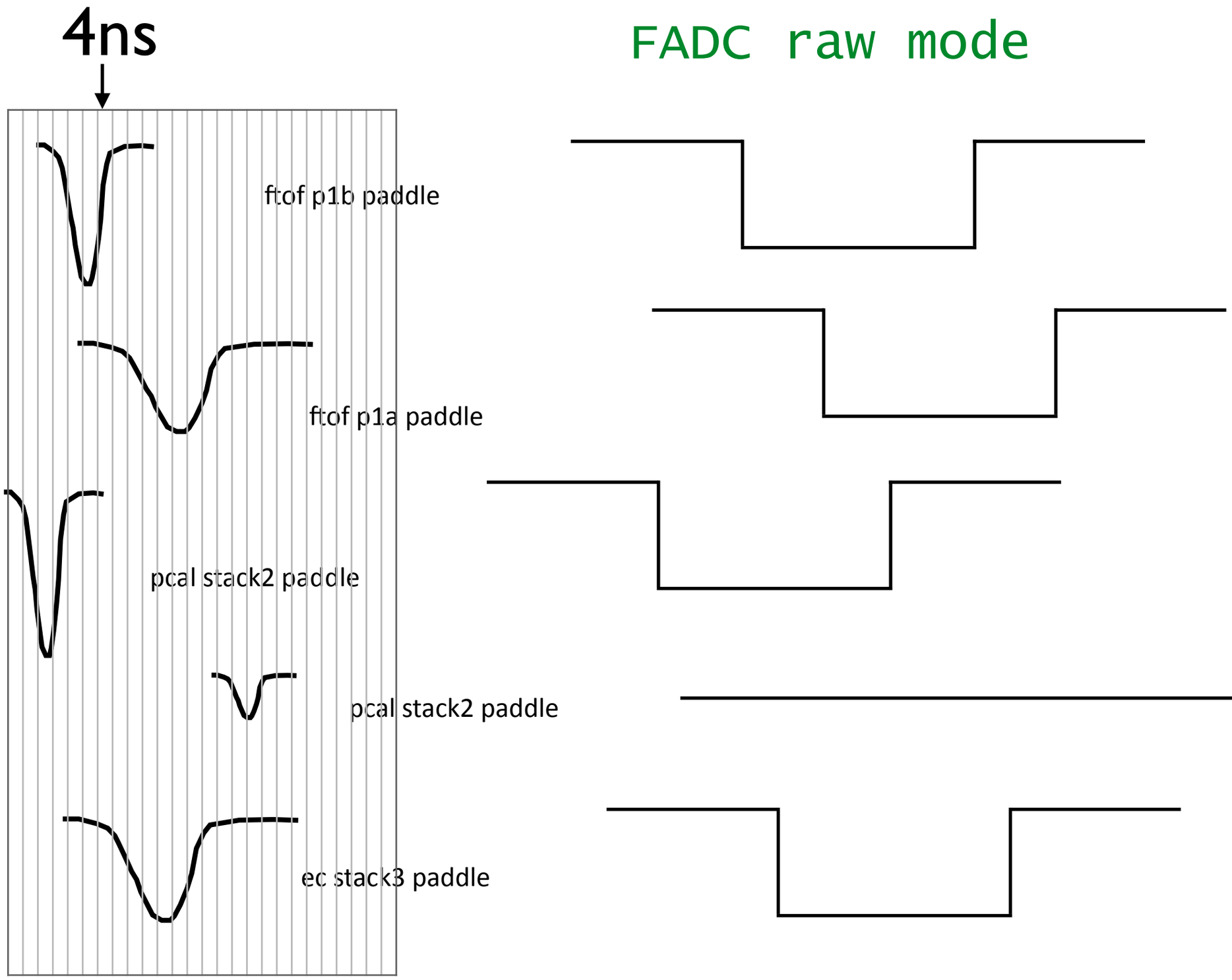
Automatic

sector
SuperLayer
Layer
wire
LR
Doca
SDoca
time
Stime

user-defined



# FADC, multi-hit signal types




*Coming Christmas 2015. Needs commissioning of parameters!*

# Documentation

Hall-B web page > CLAS12 > Offline
gemc.jlab.org
github.com/gemc

## Loading / Unloading detectors

A detector can be loaded with several variations (a variation could have a different material, or dimensions, or positioning).

To load a detector in the gcard:

```
<detector name="<path>/bst" factory="TEXT" variation="original"/>
```

Where **name** points to the name (with path) of the detector system and **variation** points to its variation. To remove a detector simply remove or comment its corresponding line.

**Note:** The true information for each system is saved in the output with the **INTEGRATEDRAW** - by default it would otherwise not be saved.

# Documentation

Hall-B web page > CLAS12 > Offline

gemc.jlab.org

github.com/gemc

## Using a custom generator

gemc support the LUND format . To generate events using a LUND file:

```
-INPUT_GEN_FILE="LUND, filename"
```

***Header Infos***

| Column | Quantity |
|---|---|
| **1** | **Number of particles** |
| **2*** | Number of target nucleons |
| **3*** | Number of target protons |
| **4*** | Target Polarization |
| **5** | **Beam Polarization** |
| **6*** | *x* |
| **7*** | *y* |
| **8*** | *W* |
| **9*** | *Q2* |
| **10*** | *nu* |

***Particle Infos***

| Column | Quantity |
|---|---|
| **1** | index |
| **2** | charge |
| **3** | **type**(=1 is active) |
| **4** | **particle id** |
| **5** | parent id (decay bookkeeping) |
| **6** | daughter (decay bookkeeping) |
| **7** | $\mathbf{p_x}$ [GeV] |
| **8** | $\mathbf{p_y}$ [GeV] |
| **9** | $\mathbf{p_z}$ [GeV] |
| **10** | E [GeV] |
| **11** | mass (not used) |
| **12** | **x vertex** [cm] |
| **13** | **y vertex** [cm] |
| **14** | **z vertex** [cm] |

# Documentation

Hall-B web page > CLAS12 > Offline

gemc.jlab.org

github.com/gemc

## Generating Background

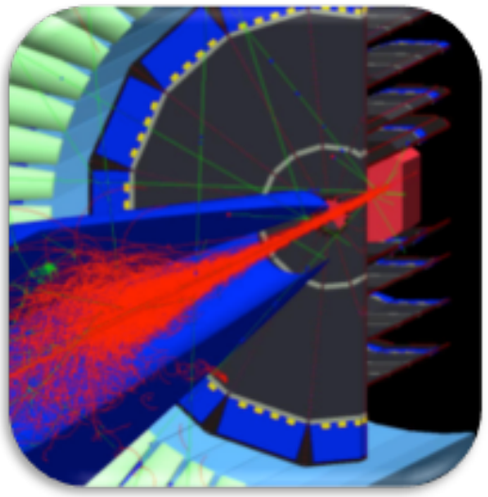
To add background coming from the beam the following quantities must be defined

1. a time window: the total time of one event
2. the number of beam particles for each event
3. the number of beam bunches

These quantities are defined with the **LUMI_EVENT** option. For example for clas12 $10^{35}$ luminosity on 5cm LH2 target:

```
<option name="LUMI_EVENT"      value="124000, 250*ns, 2*ns" />
<option name="LUMI_P"          value="e-, 11*GeV, 0*deg, 0*deg" />
<option name="LUMI_V"          value="(0.,0.,-10.)cm" />
<option name="LUMI_SPREAD_V"   value="(0.01, 0.01)cm" />
```

Adds 124000 e- in 250 ns time window, grouped in 2 ns bunches. That would produce 125 bunches with 992 particles each bunch. The beam is 100 micron wide and starts 10 cm upstream of the center of the target.



# Documentation / Installation

Documentation:

- doxygen
- gemc.jlab.org
- Hall/B
- Github
- docs not complete, can be improved
- need better mechanism on how to run massive jobs on farm (web interactive)

Installation:

- Better than in the past
- DMG for MAC provides app
- No RPM for Linux, we need it
- Re-compiling everything still a pain

# Interfaces, Optimization

- Detector interfaces
  - add java to build geant4 volumes, materials, mirrors, optical properties, etc.
  - same parameters as reconstruction
  - pass materials to reconstruction (simplified TGEO?)

- Code optimization
  - better, faster algorithms
  - standalone hit process routines
  - C++ 11
  - geant4 multithreading

- Background Studies
  - higher luminosity?

# gemc collaboration (gemc.github.io)

## How to contribute

### Feel free to ask

So let's say that you have an idea for a great feature. It's a good idea to open an issue describing the feature and its implementation and ask the code author's opinion. If they agree, go for it! They might even have some good suggestions for changes or additions to the feature as well.

If it's a bug you found, occasionally it can be ok to just create a **pull request** (PR), as long as it's clearly a bug with a straightforward fix, but it's also not a bad idea to file the bug as an issue first.

Finally, if you want to contribute but are not sure where, you can ask the author if they need help with anything – it could be as simple as helping improve the documentation.

### Forking the repo

Ok, so we have a great feature idea (or we found a bug), we opened an issue to check with the author, and they signed off on it. Whoo! Time to get to coding. First thing you do is create a fork, that is a copy of the main repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Forking a repository is a simple two steps process:

1. On GitHub, navigate to the gemc repository.
2. In the top-right corner of the page, click Fork.

You now have a copy of the repo you just forked, available in your GitHub account; its **fork-url-address** can be found on the right menu.

You can create a pull request based on this fork. If you are working on several new features at once, you can create a branch for each feature.

### Code Standards

When writing both commits and code, it's important to do so in harmony with a project's existing style. If the project uses camelCase variable naming, this is how you should name your variables as well. If the project has a test suite, you should be writing tests for any changes you make.

Even if you don't agree with some of the author's stylistic decisions, you should adhere to them in your PR. If you have a solid reason why they should be changed, open up an issue and discuss it there. Never ever change an author's existing code style to something you prefer, this is in extremely poor taste.

### Create a Pull Request

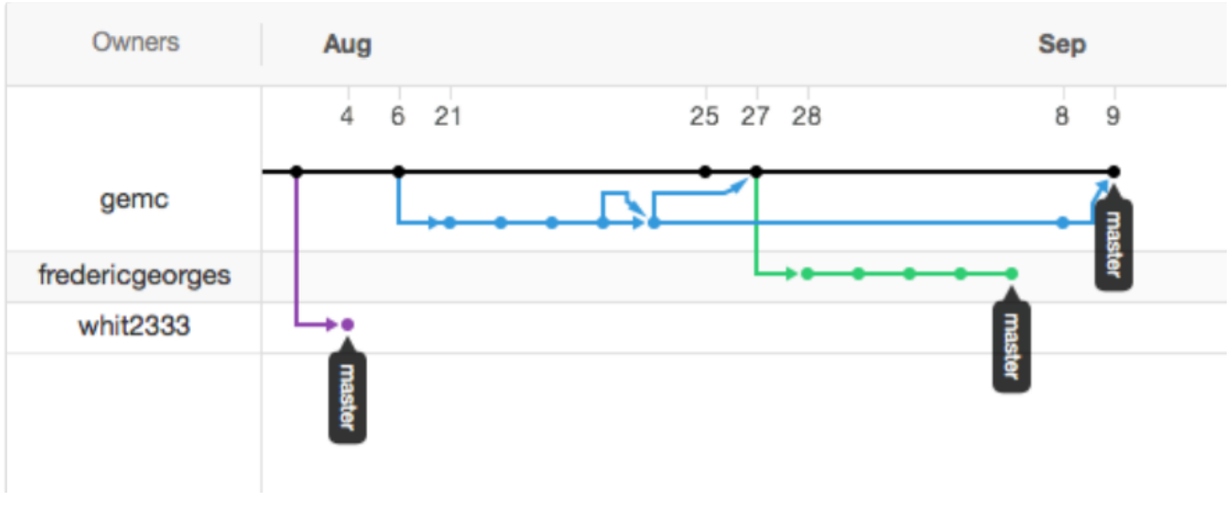
To create the pull request, navigate in github to your fork, and click on the PR button:

You will be presented with a page with a summary of your changes. Once you're ready, go ahead and press the PR button to provide additional informations:

- Make sure you selected the correct branch name ("master" if it's the main fork)
- Make sure the title and description are clear and concise
- If the change is visual, make sure to include a screenshot or gif
- If the PR closes an issue, make sure to put Closes #X at the end of the

description on a newline

1. Create an "issue"
2. Fork
3. Modify
4. Pull request

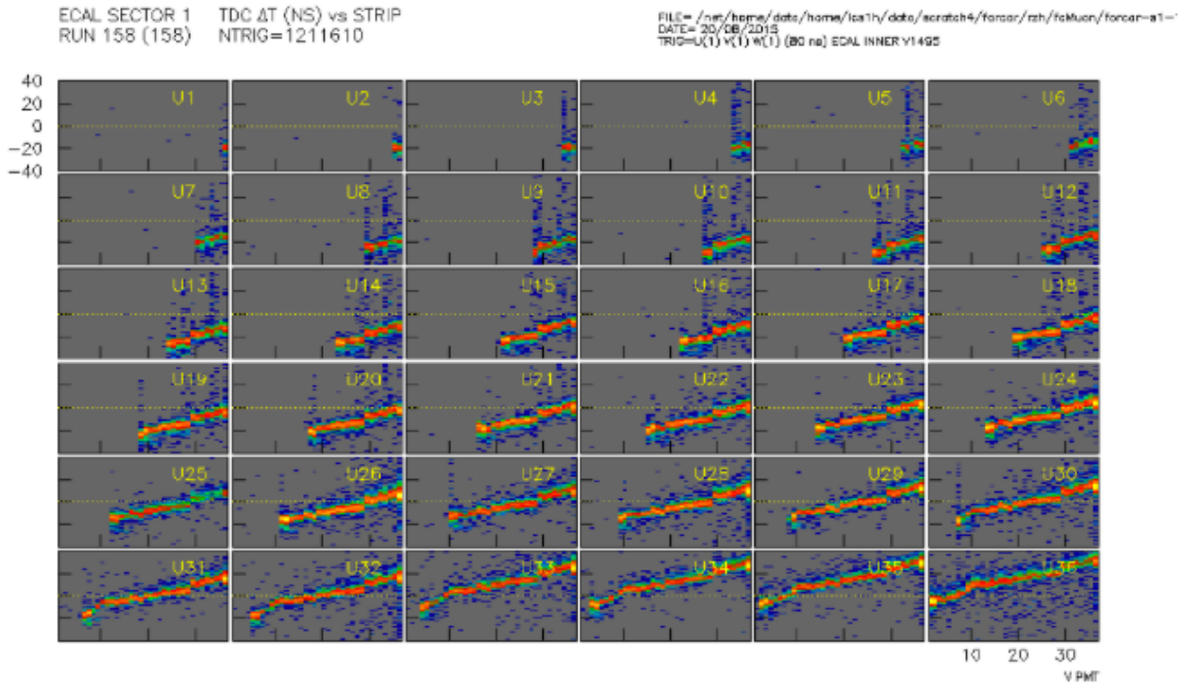


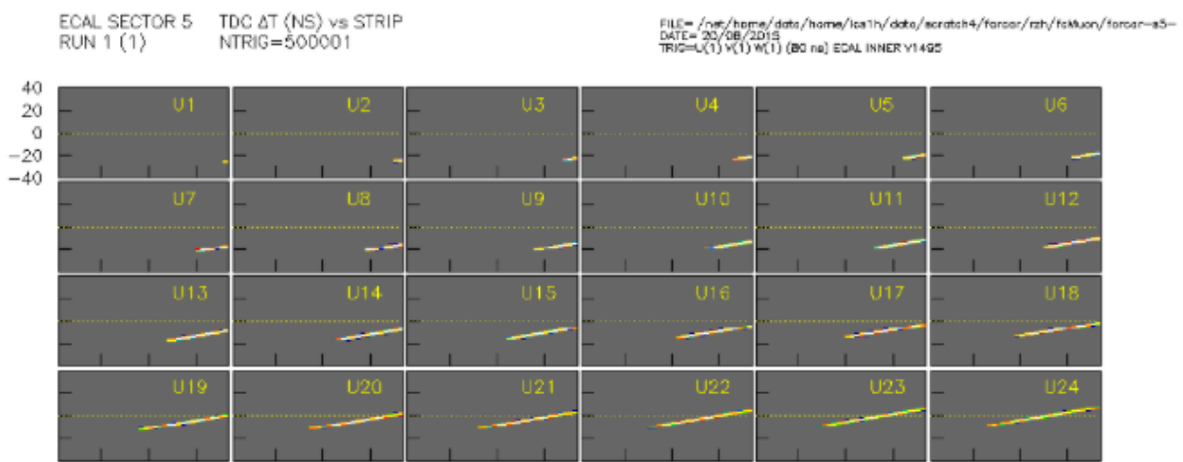

source: https://github.com/gemc

# gemc collaboration (gemc.github.io)

1. I added pcConstants class to organize hardwired constants similar to ecConstants.
2. Renamed some constants.
3. Added new constants for effective velocity.
4. Previous hitprocess did not propagate light along strip. This has been added for both EC and PCAL.
5. TDC dgtz now corresponds to current online EVIO format (24 ps / count). EVIO TDC data in composite bank is in ps, not counts, from both 1190 and 1290 TDCs. This is Sergey's choice.

Plots below show test of these changes. Top plot shows EC data from forward carriage, bottom panel show GEMC simulation throwing muons from region of target. Each panel shows U time plotted vs. V strip (as proxy for distance). Y axis is actually U-V, time difference of U and V in ns. For example, U22 plot is time difference of U22 PMT and each of the V strips that the muon went through in coincidence.




Cole Smith Pull Request for EC/PCAL digitization

# gemc collaboration (gemc.github.io)

*Test Suite: compare github branches*

> Running tests on Darwin_macosx10.10-x86_64-gcc4.2.1

> Compilation test
 > Compilation options:  -j16 OPT=1
 > Compilation: PASSED
 > Warnings: NO WARNINGS

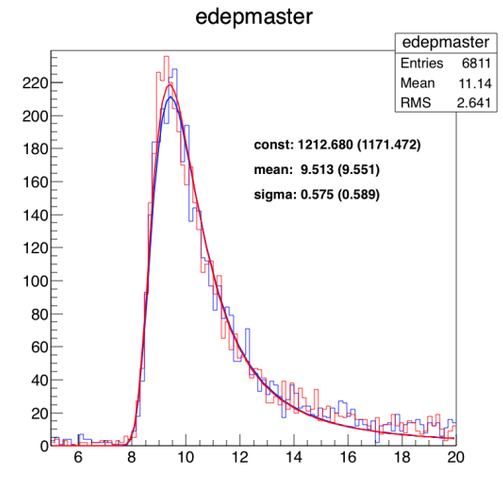
> FTOF Running test for 5000 events
 > master time: 59  -  branch time: 58
 > Edep Landau Constant percentage difference: 3.51759 %
 > Edep Landau MPV percentage difference: -0.393551 %
 > Edep Landau Sigma percentage difference: -2.29966 %

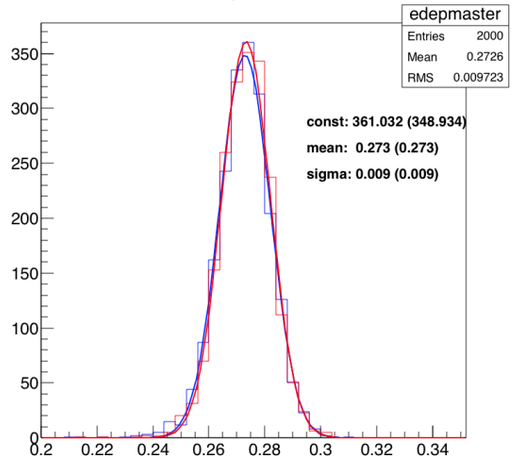
> EC Running test for 2000 events
 > master time: 1345  -  branch time: 1360
 > Edep Sampling Fraction Constant percentage difference: 3.46719 %
 > Edep Sampling Fraction MPV percentage difference: 0.141233 %
 > Edep Sampling Fraction Sigma percentage difference: -3.1772 %

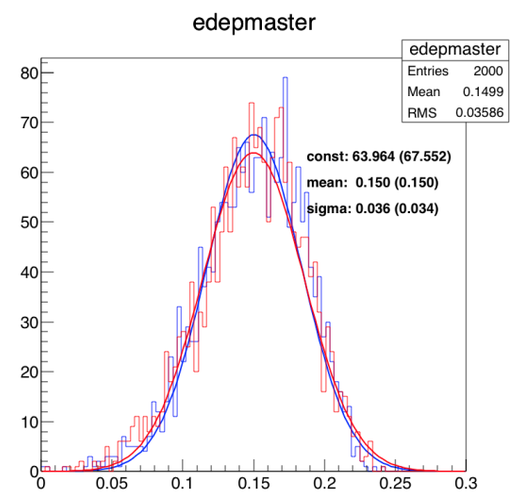
> Full CLAS12 Running test for 2000 events: PCAL
 > master time: 714  -  branch time: 715
 > Edep Sampling Fraction Constant percentage difference: -5.31061 %
 > Edep Sampling Fraction MPV percentage difference: -0.339036 %
 > Edep Sampling Fraction Sigma percentage difference: 6.63857 %

> Solenoid test
 > master time: 1  -  branch time: 1
 > Solenoid Position Test: SAME
 > Solenoid Values Test: SAME

> Torus test
 > master time: 21.5132  -  branch time: 21.4379
 > Torus Position Test: SAME
 > Torus Values Test: SAME



# gemc collaboration (gemc.github.io)

Kickoff meeting in Dec. (CLAS, EIC, Solid)

- Ways to improve gemc framework
- Each experiment: ways to optimize detectors
- user/developers feedbacks

*github provides nice tools to collaborate, lot of work to do*

# Summary: lots of work to do

- More realistic detector response
  - better digitization
  - use CCDB calibration constants
  - streamlined mechanism for misalignments

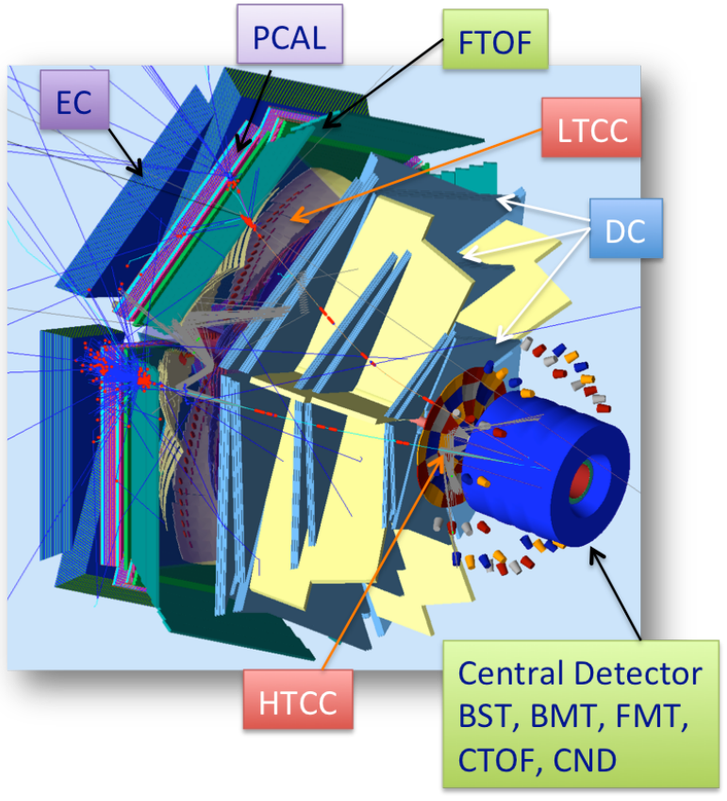
- FADC, multihit signal types
  - trigger
  - commissioning with and w/o beam

- Documentation
  - can always be improved
  - still installation issues

- Java Detector interfaces
  - doxygen
  - gemc.jlab.org
  - docs not complete, can be optimized
  - need better mechanism on how to run massive jobs on farm (web interactive)

- Code optimization
  - better, faster algorithms
  - standalone hit process routines
  - C++ 11
  - geant4 multithreading

- Background Studies
  - higher luminosity?




*gemc collaboration: please participate / volunteer!*