LAr Noise Monitoring Tools

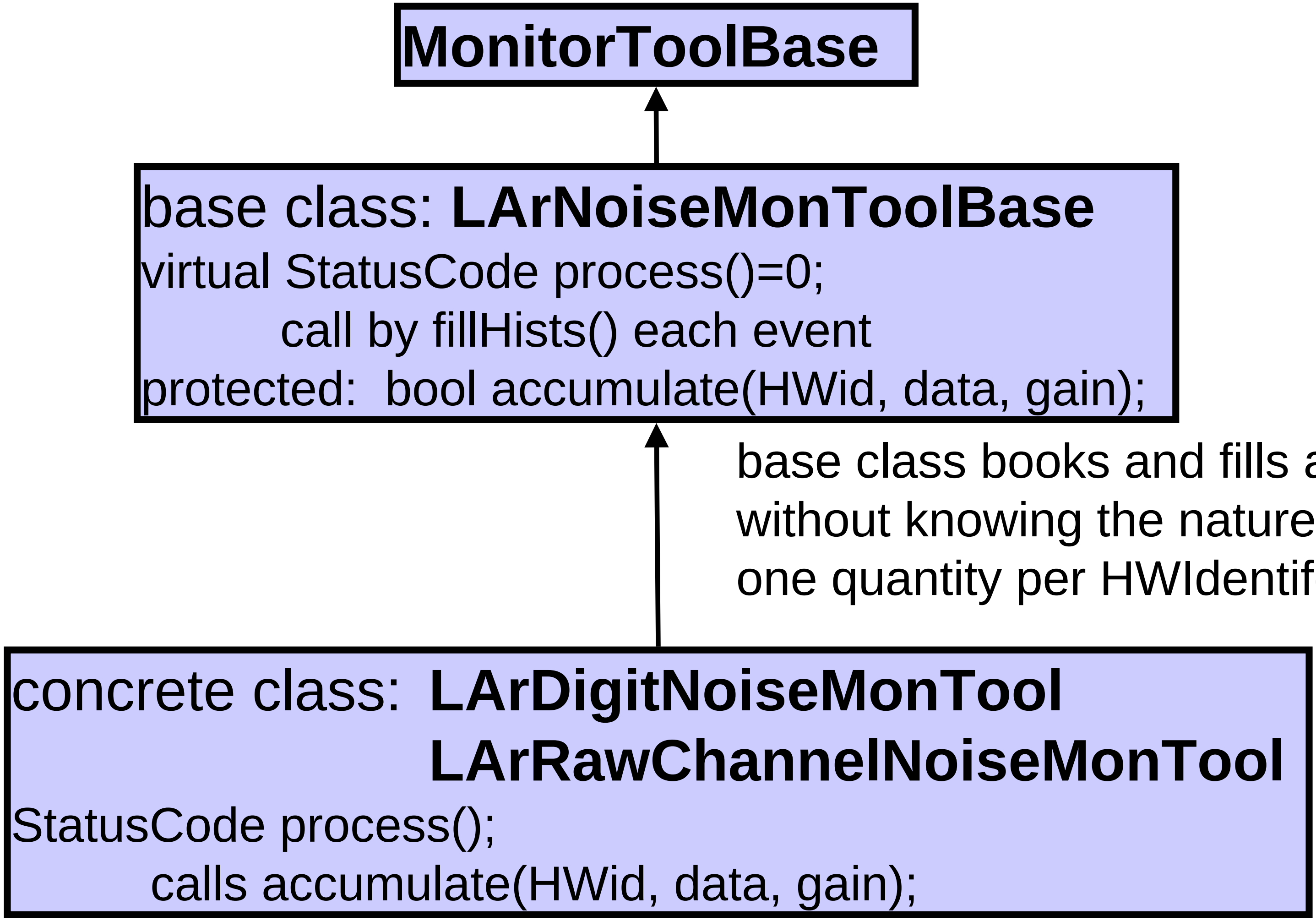
T.Ince, M.Lefebvre, R.McPherson,
R.Seuster, K.Voss
University of Victoria

LAr Week 14-18 Nov 2005

Goals

- Common framework for LArDigit, LArRawChannel (and possibly CaloCell) LAr noise monitoring in ATLAS
- Minimize code duplication
- Efficient and maintainable code

Simple Class Design for Noise Monitoring



MonitorToolBase

base class: **LArNoiseMonToolBase**
virtual StatusCode process()=0;
call by fillHists() each event
protected: bool accumulate(HWid, data, gain);

base class books and fills all histos,
without knowing the nature of the data,
one quantity per HWIdentifer and gain

concrete class: **LArDigitNoiseMonTool**
LArRawChannelNoiseMonTool
StatusCode process();
calls accumulate(HWid, data, gain);

concrete class implements process(), which loops over data container
and calls accumulate(HWid, data, gain) once per channel

Current Implementation

- First version in LArCalorimeter/LArMonTools
- Two histogram contexts
  - FEB context (one bin per channel)
  - Feedthrough context (one bin per slot)
- Histograms filled each event
  - data profile
  - bin integrated data profile (only used internally)
- Histograms refreshed in checkHists()
  - bin integrated relative coherent noise
- Features
  - LArDigitNoiseMonTool: monitor one time sample, or the average over all time samples
  - LArRawChannelNoiseMonTool: monitor the energy, or the time

Current Implementation

$$d_\alpha = \underbrace{\begin{cases} \text{LArDigitNoiseMonTool:} & \text{a given time sample} \\ & \text{the average over all time samples} \\ \text{LArRawChannelNoiseMonTool:} & \text{energy} \\ & \text{time (not clear if this will be useful)} \end{cases}}$$

FEB,gain context: α = channel #

Feedthrough,gain context: averaged per α = slot #
(1FEB per slot)

$$D_\alpha \equiv \sum_{\text{first }\beta}^{\alpha} \left(d_\alpha - K \right)$$

where *K* is fixed (for a job) to control the growth of *D*

data profile histograms: $\mu\left[d_\alpha\right] \pm \sigma\left[d_\alpha\right]$ vs α

integrated data profile histogram: $\mu\left[D_\alpha\right] \pm \sigma\left[D_\alpha\right]$ vs α

Current Implementation

$$R_\alpha \equiv \frac{(\text{total noise})_\alpha}{(\text{incoherent noise})_\alpha} = \frac{\sigma[D_\alpha]}{\sqrt{\sum_{\text{first } \beta}^{\alpha} \sigma^2[d_\beta]}}$$

thanks to Petr Gorbounov input

integrated relative coherent noise histogram:

$$(R_\alpha - 1) \pm \sigma[R_\alpha] \text{ vs } \alpha$$

LArDigitNoiseMonTool

- Example jobOption use on commissioning phase1 data

```
theApp.Dlls += [ "AthenaMonitoring"]
theApp.Dlls += [ "LArMonTools"]
theApp.TopAlg += ["AthenaMon/LArMon1"]
LArMon1 = Algorithm( "LArMon1" )
# AthenaMon
LArMon1.CheckEveryNoEvents = 100
LArMon1.AthenaMonTools += ["LArDigitNoiseMonTool/digitNoiseMon"]
# LArMonToolBase
ToolSvc.digitNoiseMon.histoPathBase           = "/Digit0Noise"
ToolSvc.digitNoiseMon.OutputLevel             = INFO
ToolSvc.digitNoiseMon.dataNameBase            = "Digit0"
ToolSvc.digitNoiseMon.febIDs                  = [0]
ToolSvc.digitNoiseMon.feedthroughIDs          = [0]
ToolSvc.digitNoiseMon.monitorCoherentNoise = True
# LArDigitNoiseMonTool
ToolSvc.digitNoiseMon.LArDigitContainerKey  = "LArDigitContainer_MC"
ToolSvc.digitNoiseMon.sampleNumber            = 0
```

label for data type

choose all FEBs

choose all FTs

enable coherent noise monitoring

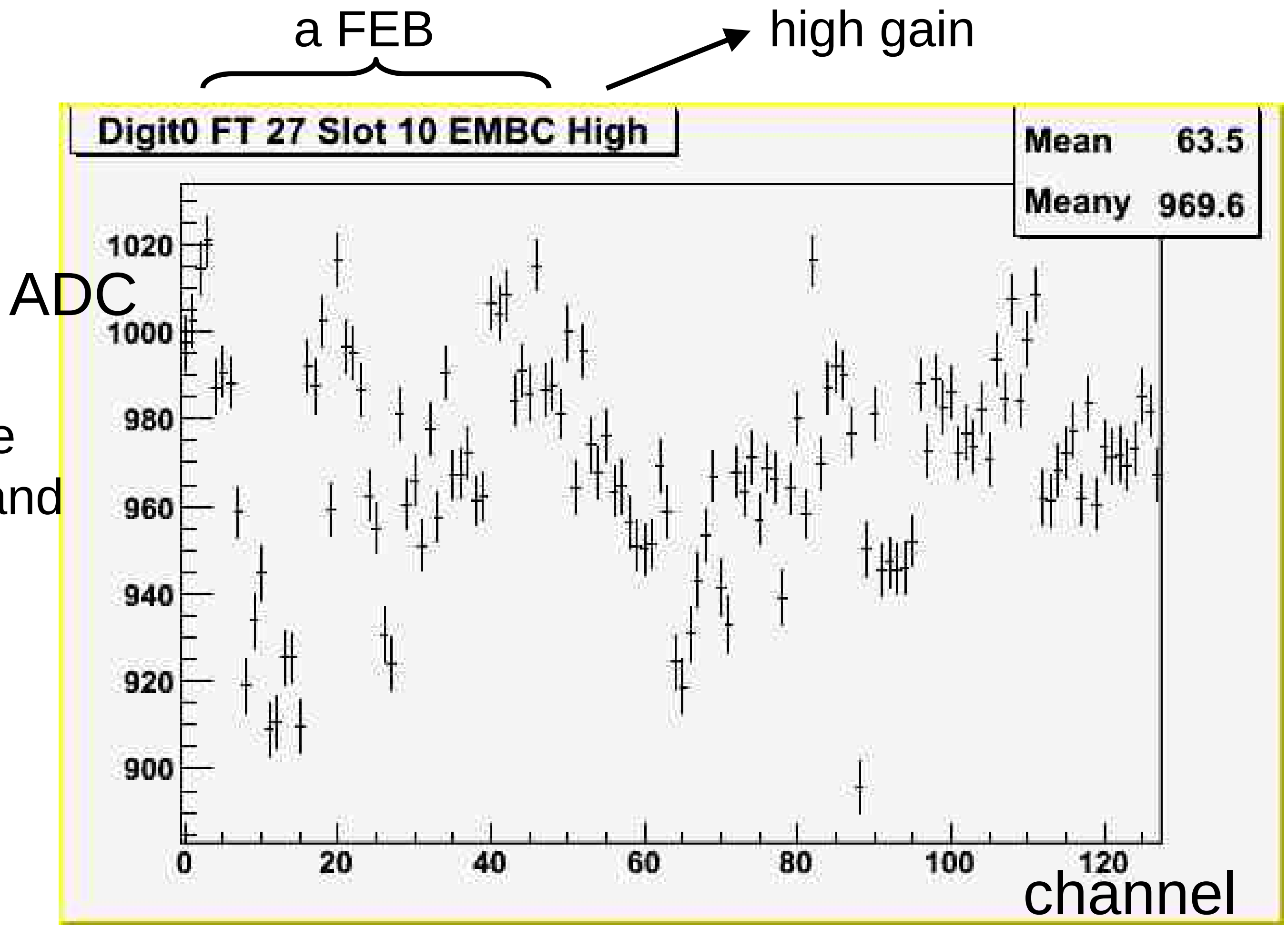
monitor digit0

LArDigitNoiseMonTool
example histograms per FEB

- sample 0 (ADC) vs channel (phase1 run 18720)

Profile histo of d_α, the digit0 ADC.

So here each point gives the channel ped and ped rms



LArDigitNoiseMonTool
example histograms per FEB

- sample 0 (ADC) vs channel (phase1 run 18720)

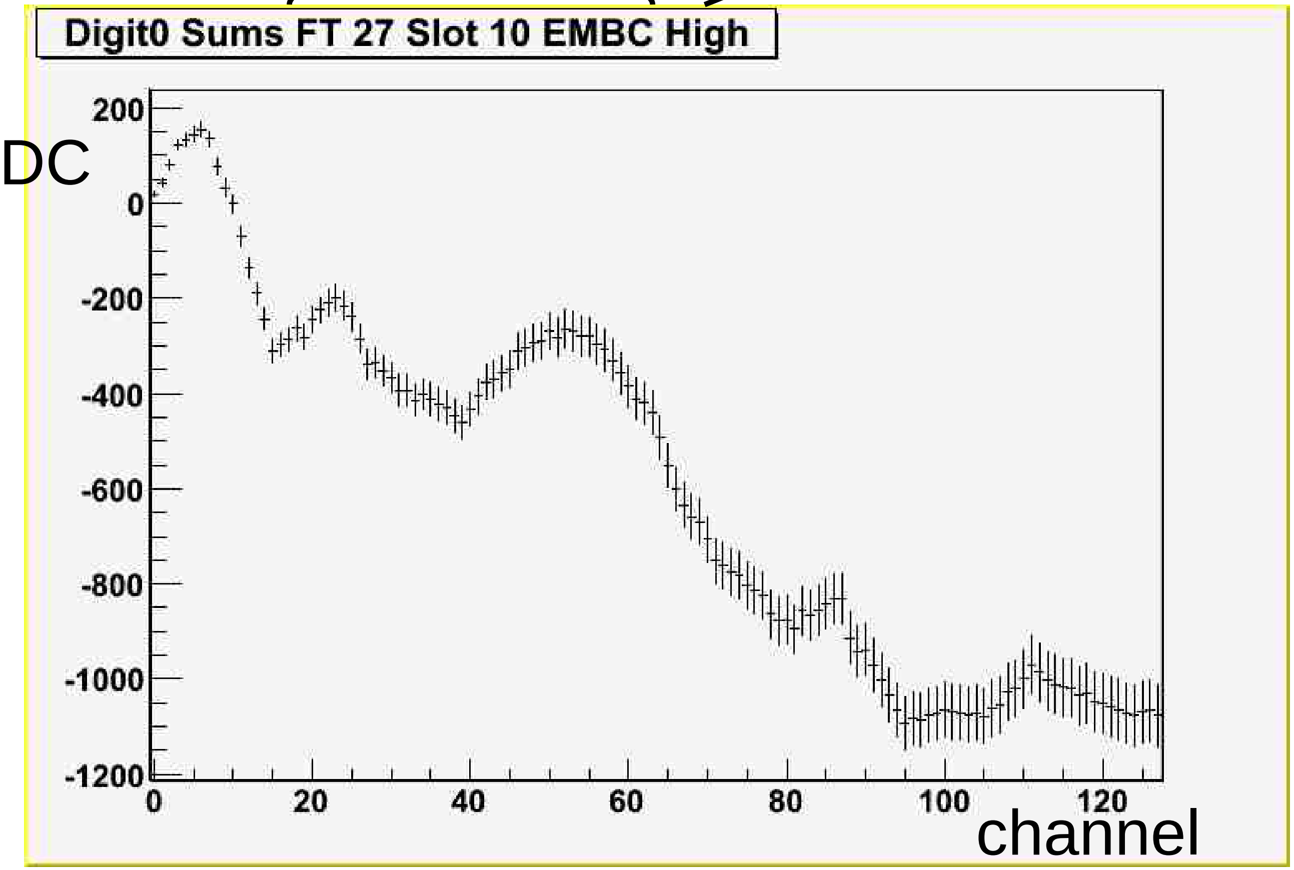
Profile histo of

$$D_{\alpha} \equiv \sum_{\beta=0}^{\alpha} \left(d_{\alpha} - K \right)$$

a FEB

high gain

ADC



channel

LArDigitNoiseMonTool
example histograms per FEB

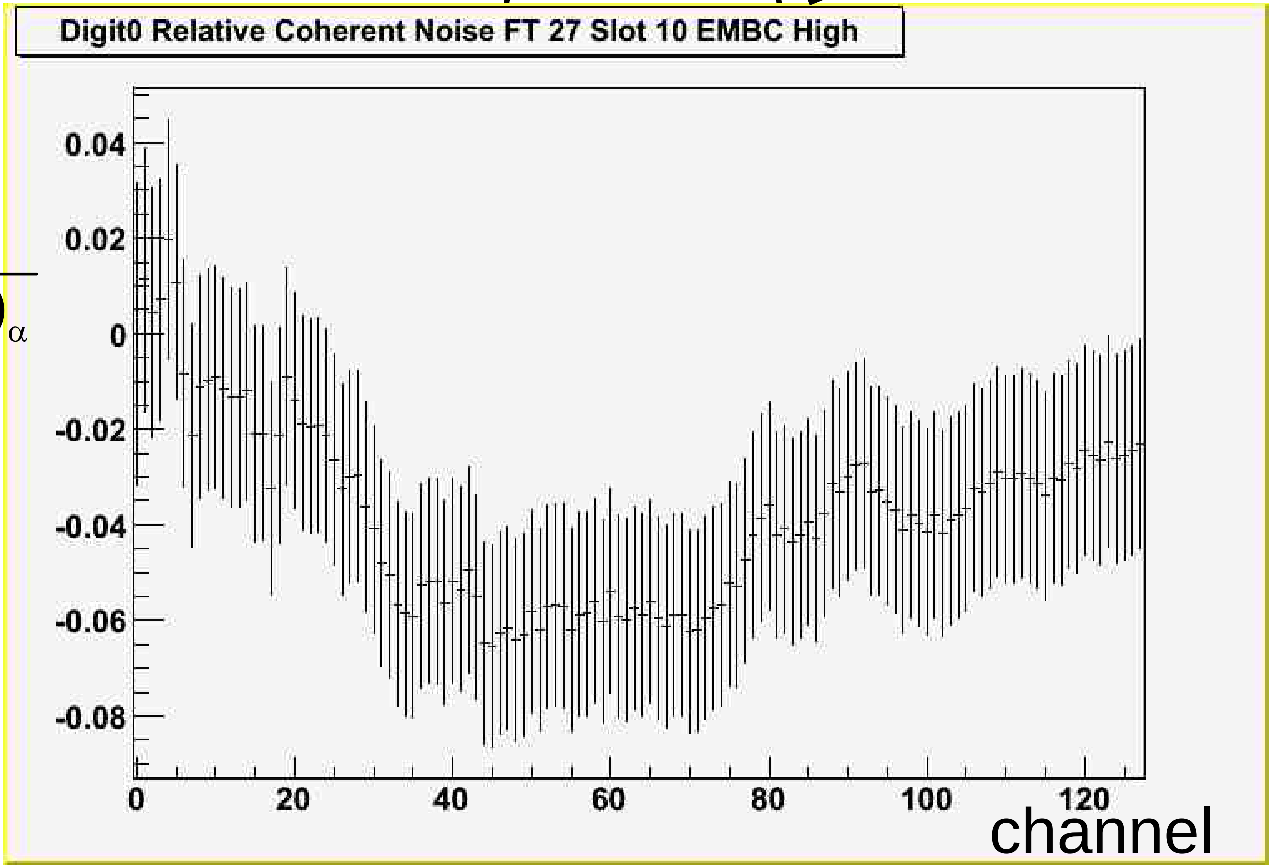
- sample 0 (ADC) vs channel (phase1 run 18720)

histo of R_α - 1

$$R_\alpha \equiv \frac{(\text{total noise})_\alpha}{(\text{incoherent noise})_\alpha} = \frac{\sigma\left[D_\alpha\right]}{\sqrt{\sum_{\beta=0}^{\alpha}\sigma^2\left[d_\beta\right]}}$$

a FEB

high gain



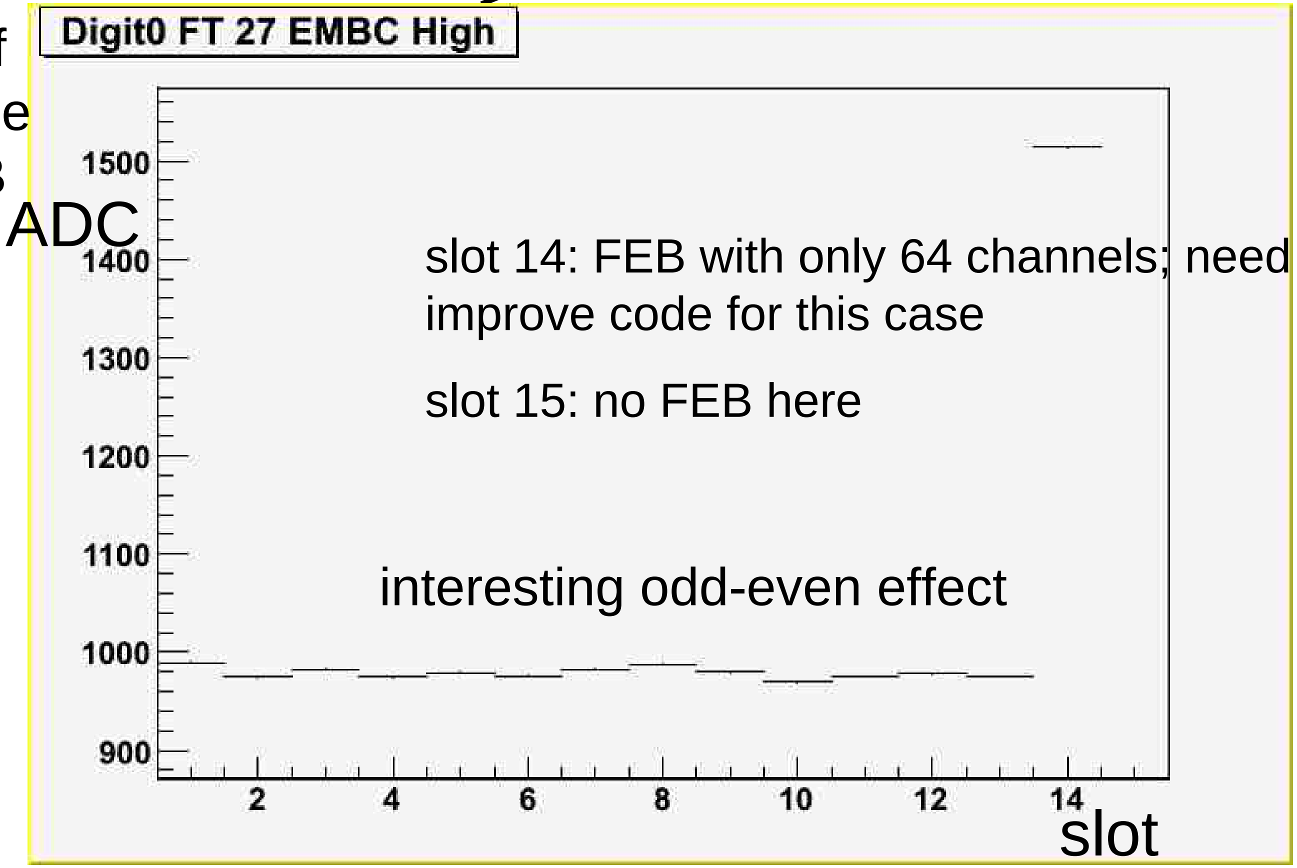
LArDigitNoiseMonTool
example histograms per Feedthrough

- average sample 0 (ADC) vs slot (phase1 run 18720)

a Feedthrough

high gain

Profile histo of d_α, the average digit0 per FEB



slot 14: FEB with only 64 channels; need to improve code for this case

slot 15: no FEB here

interesting odd-even effect

LArDigitNoiseMonTool

example histograms per Feedthrough

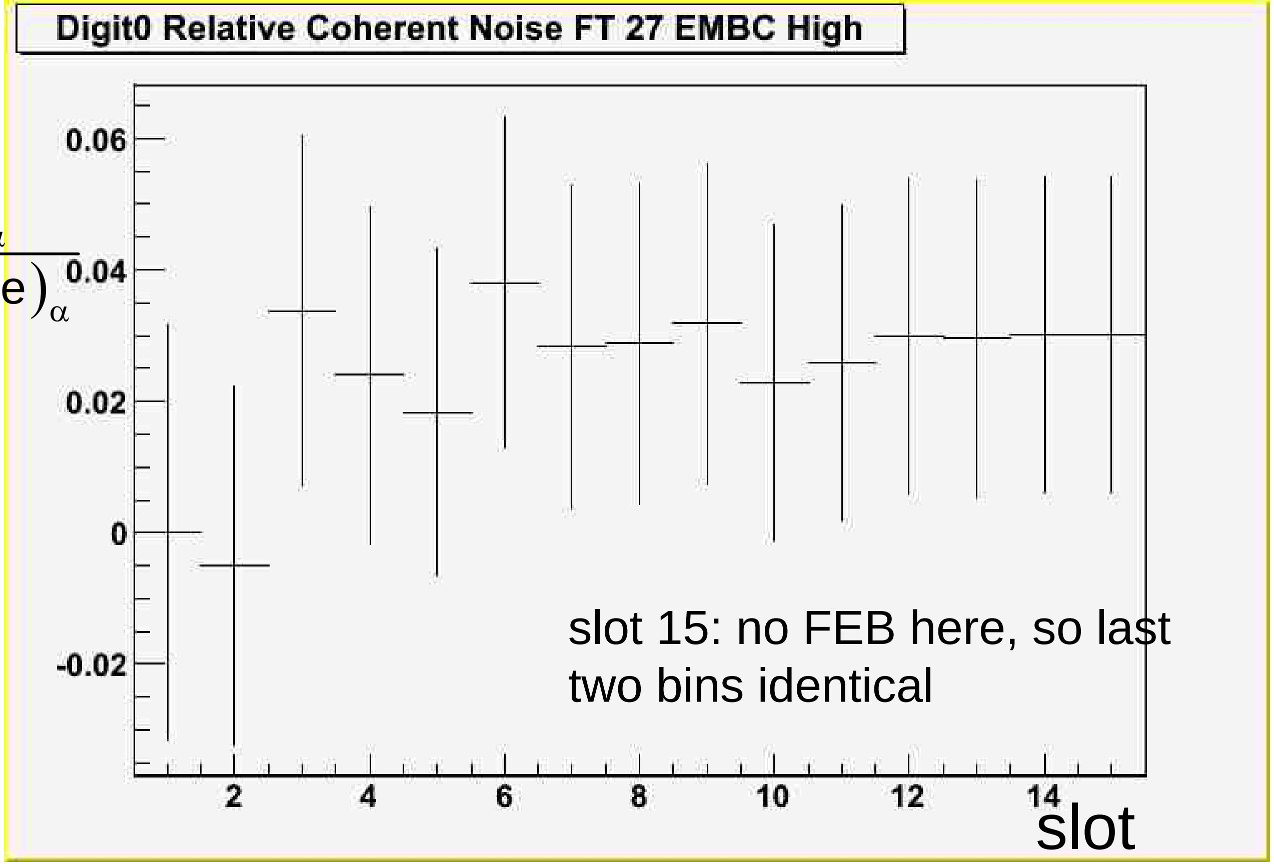
- average sample 0 (ADC) vs slot (phase1 run 18720)

a Feedthrough

high gain

histo of R_α - 1

$$R_\alpha \equiv \frac{(\text{total noise})_\alpha}{(\text{incoherent noise})_\alpha} = \frac{\sigma[D_\alpha]}{\sqrt{\sum_{\beta=1}^{\alpha}\sigma^2[d_\beta]}}$$



slot 15: no FEB here, so last two bins identical

slot

LArRawChannelNoiseMonTool

- Example jobOption use on commissioning phase1 data

```
# AthenaMon
LArMon1.CheckEveryNoEvents = 100
LArMon1.AthenaMonTools += ["LArRawChannelNoiseMonTool/rawChannelNoiseMon"]
# LArMonToolBase
ToolSvc.rawChannelNoiseMon.histoPathBase          = "/LArRawChannelNoise"
ToolSvc.rawChannelNoiseMon.OutputLevel            = INFO
ToolSvc.rawChannelNoiseMon.dataNameBase           = "LArRawChannel"
ToolSvc.rawChannelNoiseMon.febIDs                 = [0]
ToolSvc.rawChannelNoiseMon.feedthroughIDs         = [0]
ToolSvc.rawChannelNoiseMon.monitorCoherentNoise = True
# LArRawChannelNoiseMonTool
ToolSvc.rawChannelNoiseMon.LArRawChannelContainerKey = "LArRawChannels"
ToolSvc.rawChannelNoiseMon.energyNotTime      = True
ToolSvc.rawChannelNoiseMon.energyUnits         = GeV
ToolSvc.rawChannelNoiseMon.timeUnits           = picosecond
```

label for data type

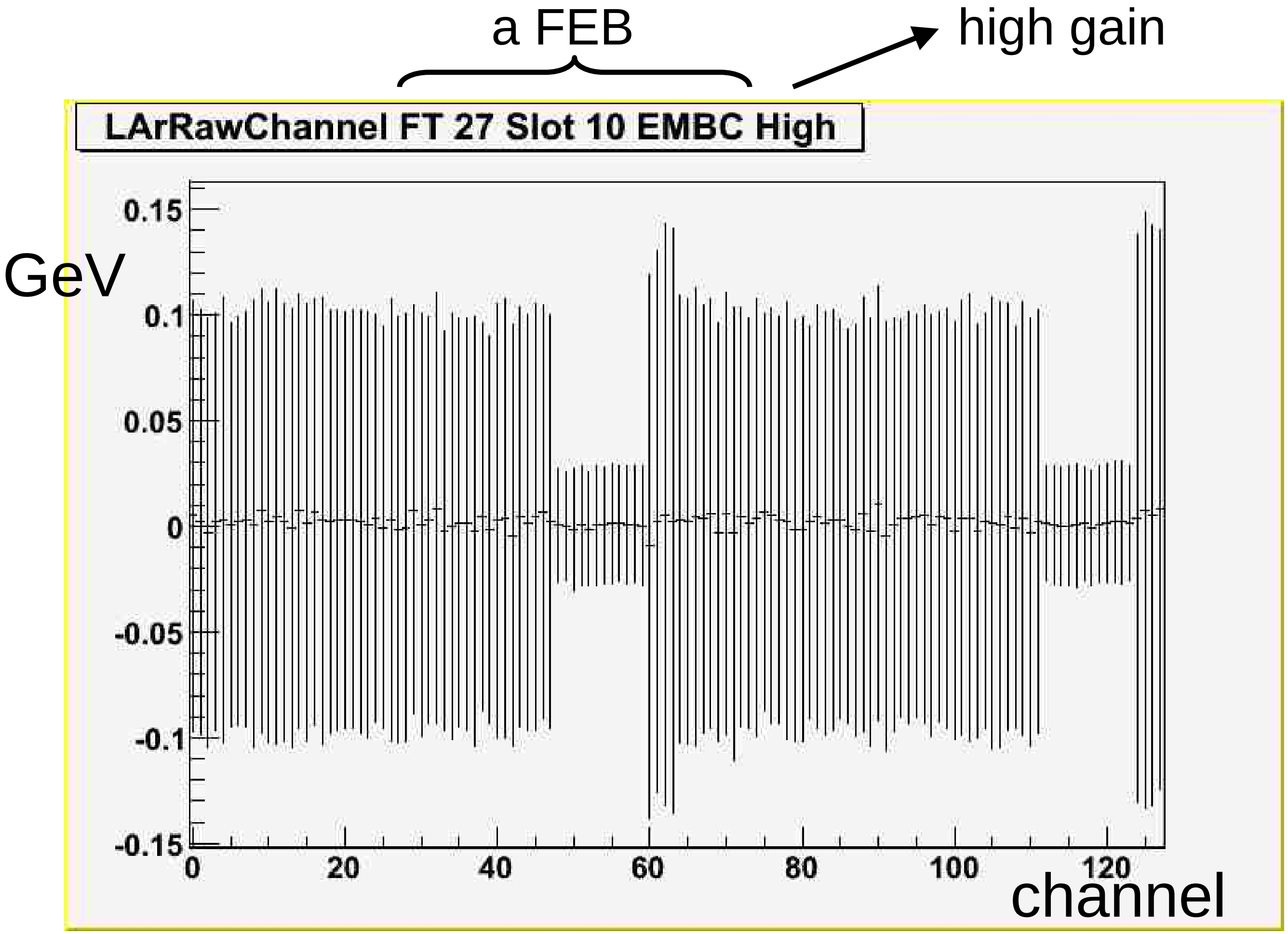
monitor energy

LArRawChannelNoiseMonTool

example histograms per FEB

- Energy (GeV) vs channel (phase1 run 18720)

Profile histo of d_{α}, the channel E(GeV).



a FEB

high gain

GeV

channel

LArRawChannelNoiseMonTool
example histograms per FEB

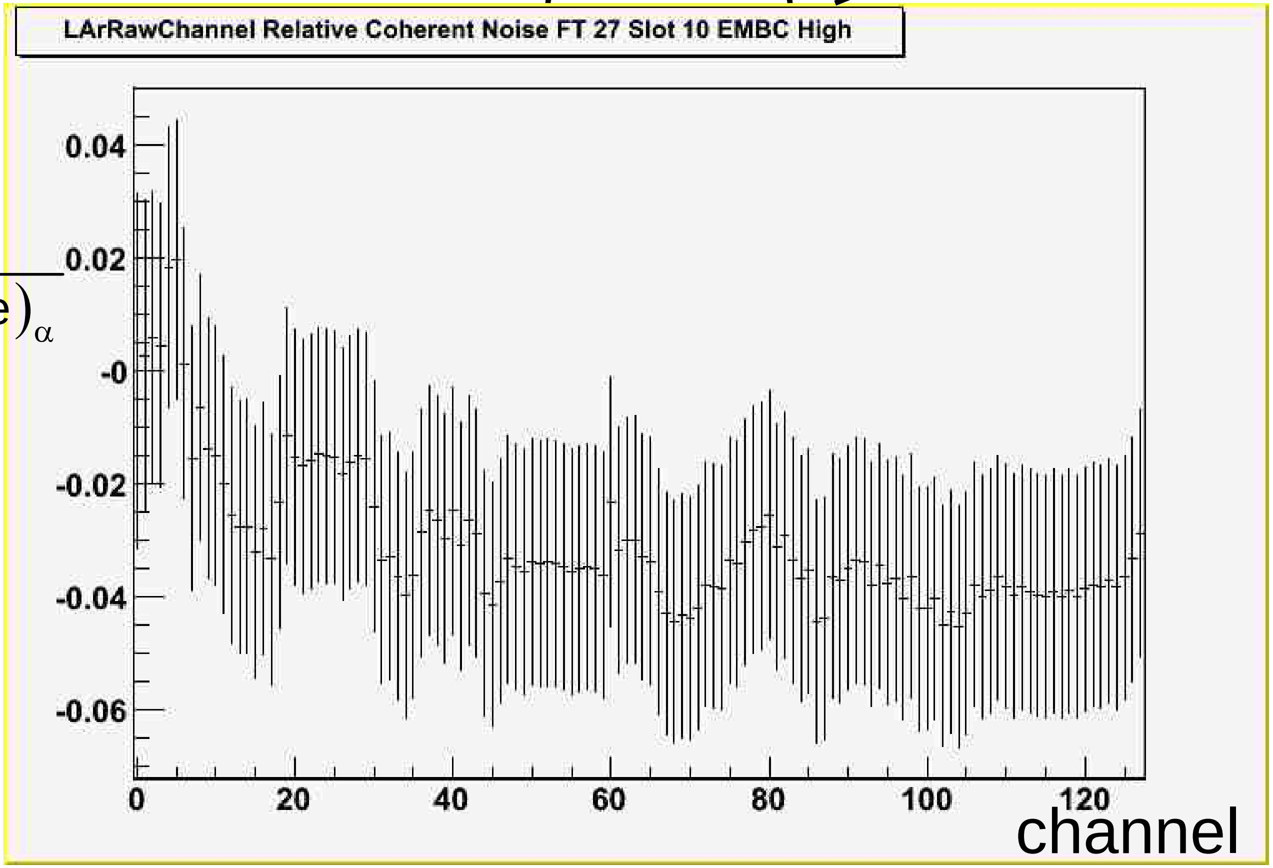
- Energy (GeV vs channel (phase1 run 18720)

a FEB

high gain

histo of R_α - 1

$$R_\alpha \equiv \frac{(\text{total noise})_\alpha}{(\text{incoherent noise})_\alpha} = \frac{\sigma[D_\alpha]}{\sqrt{\sum_{\beta=0}^{\alpha} \sigma^2[d_\beta]}}$$



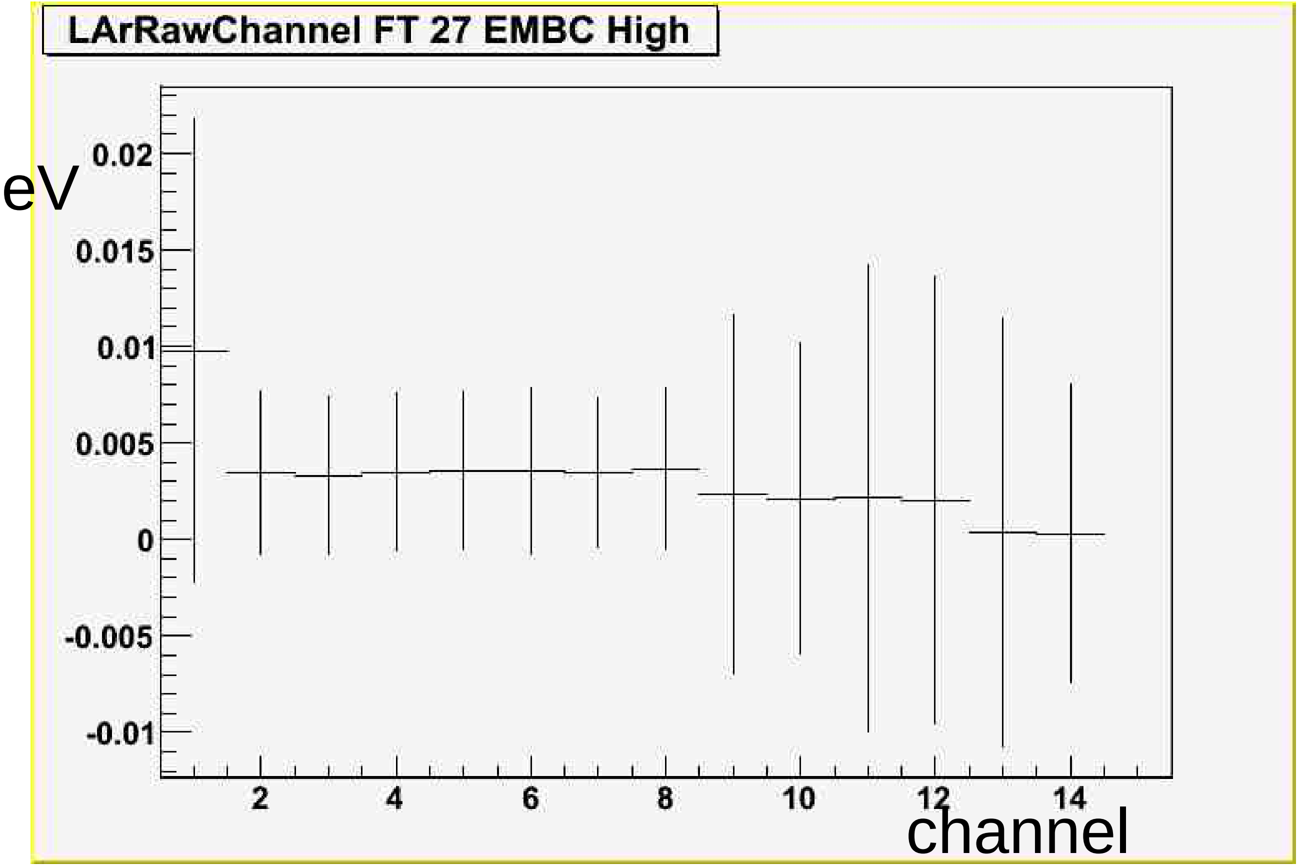
LArRawChannelNoiseMonTool
example histograms per Feedthrough

- Energy (GeV) vs channel (phase1 run 18720)

Profile histo of d_α, the average channel E(GeV) per FEB

a Feedthrough

high gain



GeV

channel

LArRawChannelNoiseMonTool
example histograms per Feedthrough

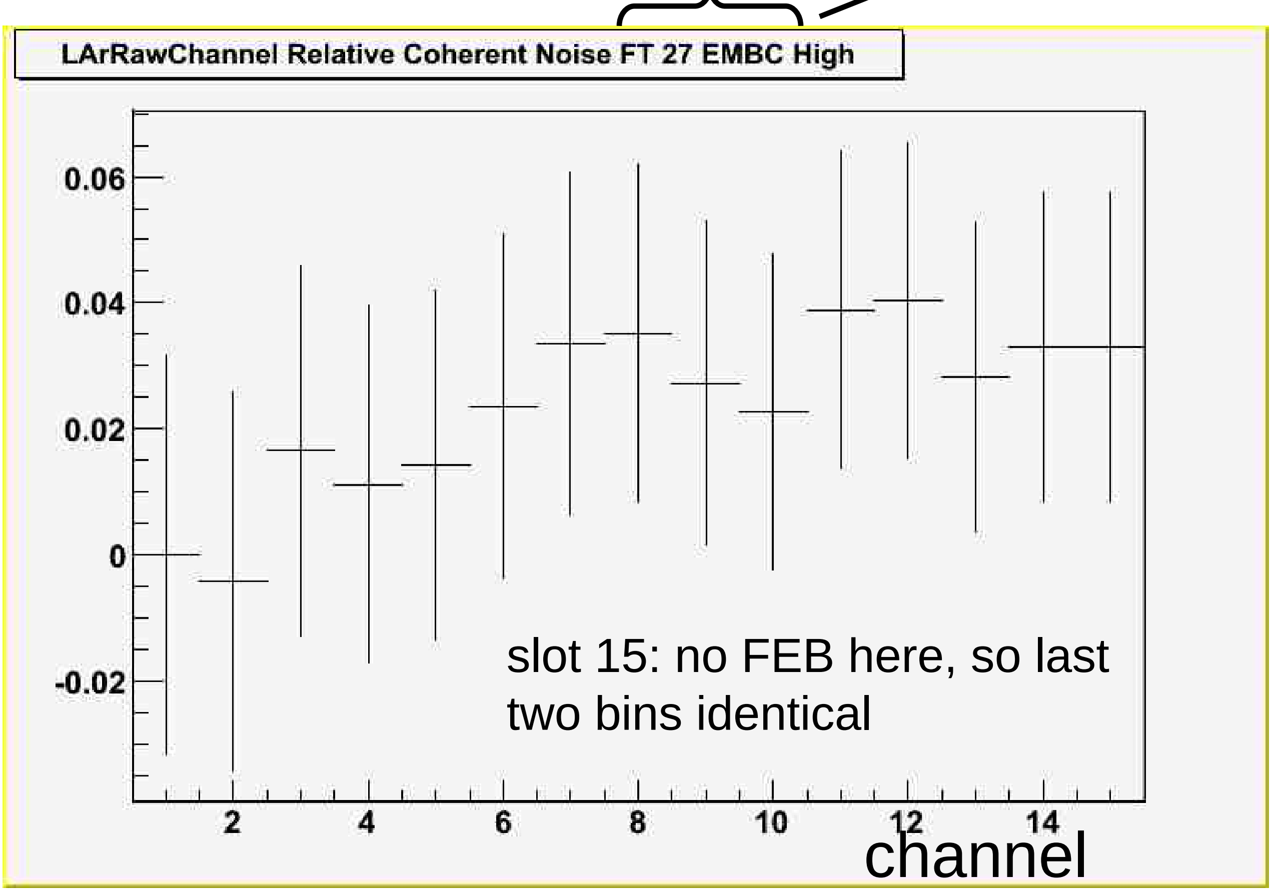
- Energy (GeV) vs channel (phase1 run 18720)

a Feedthrough

high gain

histo of R_α - 1

$$R_\alpha \equiv \frac{(\text{total noise})_\alpha}{(\text{incoherent noise})_\alpha} = \frac{\sigma[D_\alpha]}{\sqrt{\sum_{\beta=1}^{\alpha} \sigma^2[d_\beta]}}$$



slot 15: no FEB here, so last two bins identical

Further improvements

- Add a “offline” context
  - detector oriented, averaged over phi, per eta bins
  - would this be useful???
- Macro development
  - will be easier when histogram types have stabilized
- Other improvements envisaged
  - improve how to choose feb/feedthrough in jobOption
  - improve Doxygen docs
- Comments welcome!

Noise Monitoring from “calib” data objects

- Consider implementing noise monitoring using dedicated calibration data objects
  - Pedestal and their rms are to be computed in dedicated algorithms and, optionally, loaded in the database
  - LArDigit: LArPedestalMaker produces LArPedestal
    - needs improvements (Kai)
  - LArRawChannel: a similar algorithm to be written
  - One way to monitor the pedestals and their rms is to take info from these objects
    - LArDigit: from LArPedestal
  - LArNoiseMonToolBase would need some modifications
    - since data in calib classes are running means (and rms)
  - Is this a good idea???