

MIP Calibration

Basic Principles and a How to Do

Daniel Heuchel

AHCAL Testbeam Analysis Workshop

Tokyo, 8. August, 2018

HELMHOLTZ RESEARCH FOR
GRAND CHALLENGES



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386



CALICE
Calorimeter for ILC



Outline

- Introduction to MIP Calibration
- Overview MIP Calibration
- Preparation procedure
 - ➔ RootTreeGenerator
 - ➔ Pedestal Extraction
- The Extraction_MIP software package
 - ➔ Filling Histograms
 - ➔ Fitting Histograms
- Hands-on exercise: From a RunX.slcio to a mip.fits.tsv

Introduction to MIP Calibration

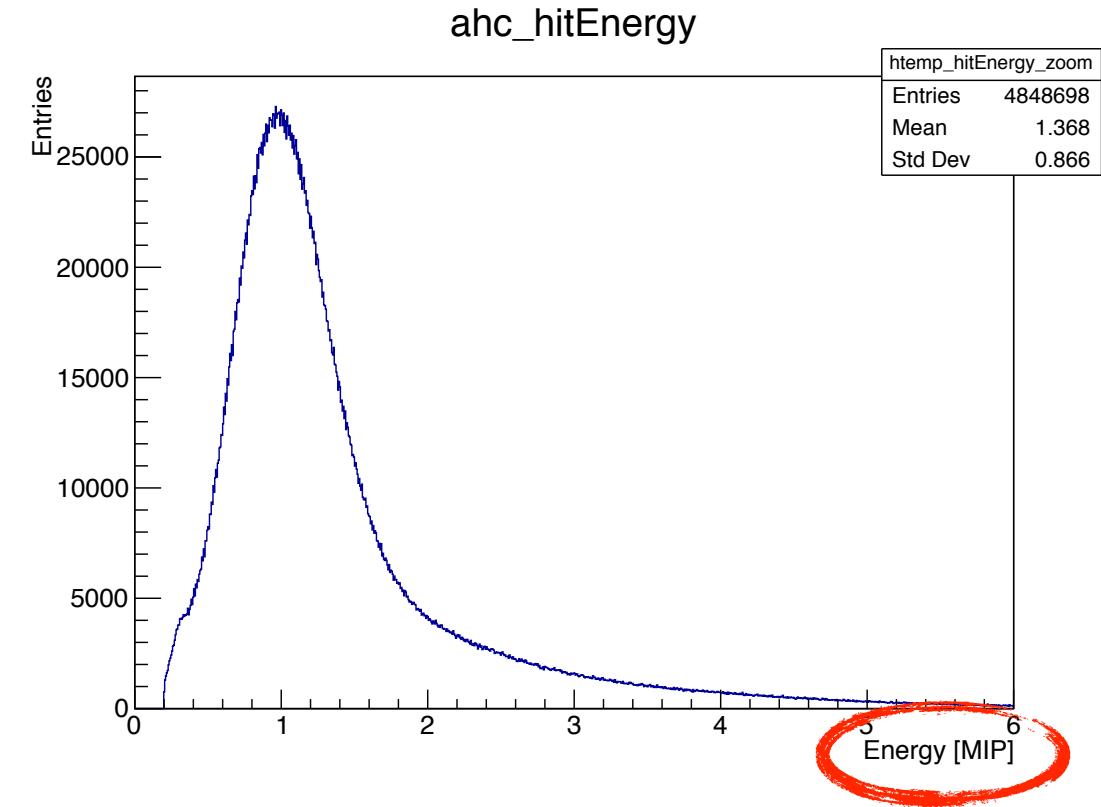
What, Why and in Which Context?

- MIP Calibration: Extract the MIP-constant for each AHCAL channel
 - ➔ MIP-constant: Most probable energy deposition for a minimum-ionizing particle for the individual channel
 - ➔ One of the calibration constants to translate:
Analogue to Digital Counts (ADC) → MIP

Introduction to MIP Calibration

What, Why and in Which Context?

- MIP Calibration: Extract the MIP-constant for each AHCAL channel
 - MIP-constant: Most probable energy deposition for a minimum-ionizing particle for the individual channel
 - One of the calibration constants to translate: Analogue to Digital Counts (ADC) → MIP
 - Enable a statement about the energy deposition of particles through the calorimeter in the units of MIP



Introduction to MIP Calibration

From ADC to MIP

- Formula to do the ADC to MIP conversion for the individual channels:

$$A_i = \frac{f_{\text{sat}}^{i-1} \left((a_i - P_i) \cdot \frac{IC_i}{G_i^{\text{CM}}(T)} \right)}{LY_i}$$

Hit amplitude in ADC

Hit amplitude in MIP

The diagram illustrates the formula for converting hit amplitude from ADC to MIP. It features two red arrows: one pointing from the term a_i in the formula to the text 'Hit amplitude in ADC', and another pointing from the variable A_i in the formula to the text 'Hit amplitude in MIP'.

From: https://bib-pubdb1.desy.de/record/301691/files/main_print.pdf

Introduction to MIP Calibration

From ADC to MIP

- Formula to do the ADC to MIP conversion for the individual channels:

Pedestal offset in ADC (individual memory-cells)



$$A_i = \frac{f_{\text{sat}}^i{}^{-1} \left((a_i - P_i) \cdot \frac{IC_i}{G_i^{\text{CM}}(T)} \right)}{LY_i}$$

From: https://bib-pubdb1.desy.de/record/301691/files/main_print.pdf

Introduction to MIP Calibration

From ADC to MIP

- Formula to do the ADC to MIP conversion for the individual channels:

$$A_i = \frac{f_{\text{sat}}^i{}^{-1} \left((a_i - P_i) \cdot \frac{IC_i}{G_i^{\text{CM}}(T)} \right)}{LY_i}$$

SiPM Gain



From: https://bib-pubdb1.desy.de/record/301691/files/main_print.pdf

Introduction to MIP Calibration

From ADC to MIP

- Formula to do the ADC to MIP conversion for the individual channels:

$$A_i = \frac{f_{\text{sat}}^i{}^{-1} \left((a_i - P_i) \cdot \frac{IC_i}{G_i^{\text{EM}}(T)} \right)}{LY_i}$$

Intercalibration Factor



From: https://bib-pubdb1.desy.de/record/301691/files/main_print.pdf

Introduction to MIP Calibration

From ADC to MIP

- Formula to do the ADC to MIP conversion for the individual channels:

$$A_i = \frac{f_{\text{sat}}^{i-1} \left((a_i - P_i) \cdot \frac{IC_i}{G_i^{\text{CM}}(T)} \right)}{LY_i}$$

Inverse of SiPM pixel saturation function



From: https://bib-pubdb1.desy.de/record/301691/files/main_print.pdf

Introduction to MIP Calibration

From ADC to MIP

- Formula to do the ADC to MIP conversion for the individual channels:

$$A_i = \frac{f_{\text{sat}}^i{}^{-1} \left((a_i - P_i) \cdot \frac{IC_i}{G_i^{\text{CM}}(T)} \right)}{LY_i}$$

Light-yield: Most probable triggered SiPM pixels for a MIP

From: https://bib-pubdb1.desy.de/record/301691/files/main_print.pdf

Introduction to MIP Calibration

From ADC to MIP

- Formula to do the ADC to MIP conversion for the individual channels:

$$A_i = \frac{f_{\text{sat}}^i{}^{-1} \left((a_i - P_i) \cdot \frac{IC_i}{G_i^{\text{CM}}(T)} \right)}{LY_i}$$

MIP Constant: Most probable energy deposition for a MIP in ADC

$$LY_i = \frac{M_i(T) \cdot IC_i}{G_i^{\text{CM}}(T)}$$

From: https://bib-pubdb1.desy.de/record/301691/files/main_print.pdf

Introduction to MIP Calibration

Where to Find the Constants: DataBase and SteeringFile

```
<!-- XXX: change to correct database entries and tags-->
<processor name="GeoConditions" type="ConditionsProcessor">
  <parameter name="DBInit" type="string" value="flccaldb02.desy.de:calice:caliceon:Delice.1:3306"/>
  <parameter name="DBCondHandler" type="StringVec">
    Ahc2ModuleDescription          /test_lan/Ahc2_June2018/ModuleDescription          HEAD
    Ahc2ModuleConnection           /test_lan/Ahc2_June2018/ModuleConnection           HEAD
    Ahc2ModuleLocationReference   /test_lan/Ahc2_June2018/ModuleLocationReference   HEAD
    Ahc2HardwareConnection         /test_lan/Ahc2_June2018/HardwareConnection        HEAD
    Ahc2DetectorTransformation    /test_lan/Ahc2_June2018/DetectorTransformation     HEAD
    E4DPedestal                   /test_lan/Ahc2_June2018/Pedestal                  HEAD
    E4DGainConstants              /test_lan/Ahc2_June2018/gain_constants            HEAD
    E4DGainSlopes                 /test_lan/Ahc2_June2018/gain_slopes              HEAD
    E4DMipConstants               /test_lan/Ahc2_June2018/mip_constants             HEAD
    E4DMipSlopes                  /test_lan/Ahc2_June2018/mip_slopes                HEAD
    E4DDeadCellMap                /cd_calice_Ahc2/TestbeamMay2018/DeadCellMap      HEAD
    E4DSaturationParameters       /cd_calice_Ahc2/TestbeamMay2018/SaturationParameters HEAD
    E4DIntercalibration           /cd_calice_Ahc2/TestbeamMay2018/Intercalibration   HEAD
    E4DPhysicsCalibIntercalibration /cd_calice_Ahc2/TestbeamMay2018/PhysicsCalibIntercalibration HEAD
    E4DTimeSlopes                 /cd_calice_Ahc2/TestbeamMay2018/TimeSlope          ahc2_006
    E4DTimeOffset                 /cd_calice_Ahc2/TestbeamMay2018/TimeOffset          ahc2_006
  </parameter>
</processor>
```

This talk

Introduction to MIP Calibration

Where to Find the Constants: DataBase and SteeringFile

```
<!-- XXX: change to correct database entries and tags-->
<processor name="GeoConditions" type="ConditionsProcessor">
  <parameter name="DBInit" type="string" value="flccaldb02.desy.de:calice:caliceon:Delice.1:3306"/>
  <parameter name="DBCondHandler" type="StringVec">
    Ahc2ModuleDescription          /test_lan/Ahc2_June2018/ModuleDescription          HEAD
    Ahc2ModuleConnection           /test_lan/Ahc2_June2018/ModuleConnection           HEAD
    Ahc2ModuleLocationReference   /test_lan/Ahc2_June2018/ModuleLocationReference   HEAD
    Ahc2HardwareConnection         /test_lan/Ahc2_June2018/HardwareConnection        HEAD
    Ahc2DetectorTransformation    /test_lan/Ahc2_June2018/DetectorTransformation     HEAD
    E4DPedestal                   /test_lan/Ahc2_June2018/Pedestal                  HEAD
    E4DGainConstants              /test_lan/Ahc2_June2018/gain_constants            HEAD
    E4DGainSlopes                 /test_lan/Ahc2_June2018/gain_slopes              HEAD
    E4DMipConstants               /test_lan/Ahc2_June2018/mip_constants             HEAD
    E4DMipSlopes                  /test_lan/Ahc2_June2018/mip_slopes                HEAD
    E4DDeadCellMap                /cd_calice_Ahc2/TestbeamMay2018/DeadCellMap      HEAD
    E4DSaturationParameters       /cd_calice_Ahc2/TestbeamMay2018/SaturationParameters HEAD
    E4DIntercalibration           /cd_calice_Ahc2/TestbeamMay2018/Intercalibration   HEAD
    E4DPhysicsCalibIntercalibration /cd_calice_Ahc2/TestbeamMay2018/PhysicsCalibIntercalibration HEAD
    E4DTimeSlopes                 /cd_calice_Ahc2/TestbeamMay2018/TimeSlope          ahc2_006
    E4DTimeOffset                  /cd_calice_Ahc2/TestbeamMay2018/TimeOffset          ahc2_006

```

See Olin's talk

Introduction to MIP Calibration

Where to Find the Constants: DataBase and SteeringFile

```
<!-- XXX: change to correct database entries and tags-->
<processor name="GeoConditions" type="ConditionsProcessor">
    <parameter name="DBInit" type="string" value="flccaldb02.desy.de:calice:caliceon:Delice.1:3306"/>
    <parameter name="DBCondHandler" type="StringVec">
        Ahc2ModuleDescription /test_lan/Ahc2_June2018/ModuleDescription HEAD
        Ahc2ModuleConnection /test_lan/Ahc2_June2018/ModuleConnection HEAD
        Ahc2ModuleLocationReference /test_lan/Ahc2_June2018/ModuleLocationReference HEAD
        Ahc2HardwareConnection /test_lan/Ahc2_June2018/HardwareConnection HEAD
        Ahc2DetectorTransformation /test_lan/Ahc2_June2018/DetectorTransformation HEAD
        E4DPedestal /test_lan/Ahc2_June2018/Pedestal HEAD
        E4DGainConstants /test_lan/Ahc2_June2018/gain_constants HEAD
        E4DGainSlopes /test_lan/Ahc2_June2018/gain_slopes HEAD
        E4DMipConstants /test_lan/Ahc2_June2018/mip_constants HEAD
        E4DMipSlopes /test_lan/Ahc2_June2018/mip_slopes HEAD
        E4DDeadCellMap /cd_calice_Ahc2/TestbeamMay2018/DeadCellMap HEAD
        E4DSaturationParameters /cd_calice_Ahc2/TestbeamMay2018/SaturationParameters HEAD
        E4DIntercalibration /cd_calice_Ahc2/TestbeamMay2018/Intercalibration HEAD
        E4DPhysicsCalibIntercalibration /cd_calice_Ahc2/TestbeamMay2018/PhysicsCalibIntercalibration HEAD
        E4DTimeSlopes /cd_calice_Ahc2/TestbeamMay2018/TimeSlope ahc2_006
        E4DTimeOffset /cd_calice_Ahc2/TestbeamMay2018/TimeOffset ahc2_006
```

This talk

Introduction to MIP Calibration

Where to Find the Constants: DataBase and SteeringFile

```
<!-- XXX: change to correct database entries and tags-->
<processor name="GeoConditions" type="ConditionsProcessor">
    <parameter name="DBInit" type="string" value="flccaldb02.desy.de:calice:caliceon:Delice.1:3306"/>
    <parameter name="DBCondHandler" type="StringVec">
        Ahc2ModuleDescription /test_lan/Ahc2_June2018/ModuleDescription HEAD
        Ahc2ModuleConnection /test_lan/Ahc2_June2018/ModuleConnection HEAD
        Ahc2ModuleLocationReference /test_lan/Ahc2_June2018/ModuleLocationReference HEAD
        Ahc2HardwareConnection /test_lan/Ahc2_June2018/HardwareConnection HEAD
        Ahc2DetectorTransformation /test_lan/Ahc2_June2018/DetectorTransformation HEAD
        E4DPedestal /test_lan/Ahc2_June2018/Pedestal HEAD
        E4DGainConstants /test_lan/Ahc2_June2018/gain_constants HEAD
        E4DGainSlopes /test_lan/Ahc2_June2018/gain_slopes HEAD
        E4DMipConstants /test_lan/Ahc2_June2018/mip_constants HEAD
        E4DMipSlopes /test_lan/Ahc2_June2018/mip_slopes HEAD
        E4DDeadCellMap /cd_calice_Ahc2/TestbeamMay2018/DeadCellMap HEAD
        E4DSaturationParameters /cd_calice_Ahc2/TestbeamMay2018/SaturationParameters HEAD
        E4DIntercalibration /cd_calice_Ahc2/TestbeamMay2018/Intercalibration HEAD
        E4DPhysicsCalibIntercalibration /cd_calice_Ahc2/TestbeamMay2018/PhysicsCalibIntercalibration HEAD
        E4DTimeSlopes /cd_calice_Ahc2/TestbeamMay2018/TimeSlope ahc2_006
        E4DTimeOffset /cd_calice_Ahc2/TestbeamMay2018/TimeOffset ahc2_006
```



See Yuji's talk

Introduction to MIP Calibration

Where to Find the Constants: DataBase and SteeringFile

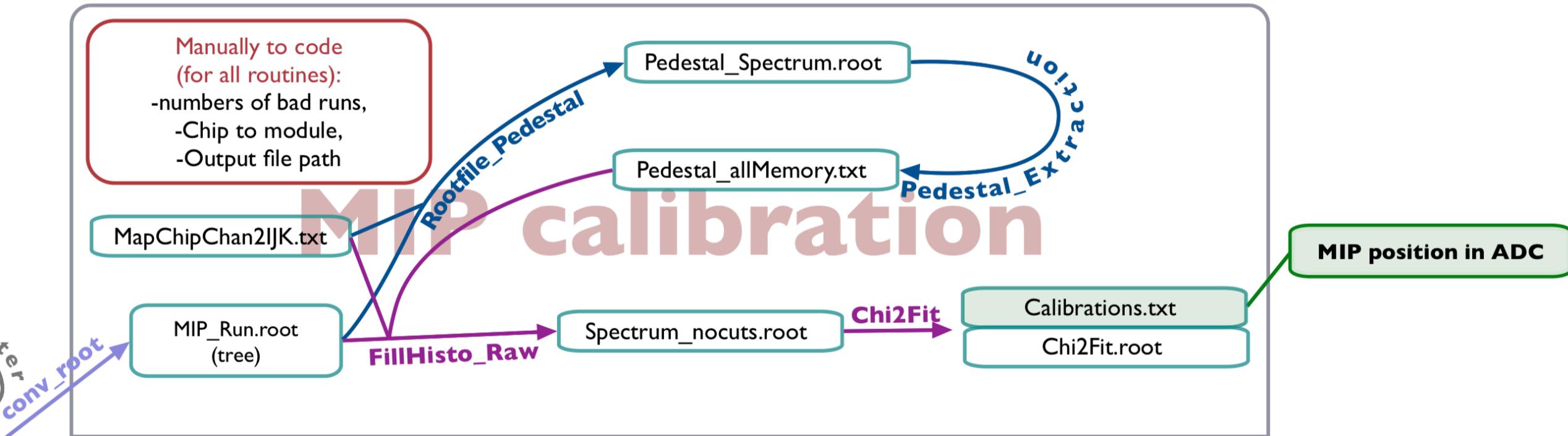
```
<!-- XXX: change to correct database entries and tags-->
<processor name="GeoConditions" type="ConditionsProcessor">
    <parameter name="DBInit" type="string" value="flccaldb02.desy.de:calice:caliceon:Delice.1:3306"/>
    <parameter name="DBCondHandler" type="StringVec">
        Ahc2ModuleDescription /test_lan/Ahc2_June2018/ModuleDescription HEAD
        Ahc2ModuleConnection /test_lan/Ahc2_June2018/ModuleConnection HEAD
        Ahc2ModuleLocationReference /test_lan/Ahc2_June2018/ModuleLocationReference HEAD
        Ahc2HardwareConnection /test_lan/Ahc2_June2018/HardwareConnection HEAD
        Ahc2DetectorTransformation /test_lan/Ahc2_June2018/DetectorTransformation HEAD
        E4DPedestal /test_lan/Ahc2_June2018/Pedestal HEAD
        E4DGainConstants /test_lan/Ahc2_June2018/gain_constants HEAD
        E4DGainSlopes /test_lan/Ahc2_June2018/gain_slopes HEAD
        E4DMipConstants /test_lan/Ahc2_June2018/mip_constants HEAD
        E4DMipSlopes /test_lan/Ahc2_June2018/mip_slopes HEAD
        E4DDeadCellMap /cd_calice_Ahc2/TestbeamMay2018/DeadCellMap HEAD
        E4DSaturationParameters /cd_calice_Ahc2/TestbeamMay2018/SaturationParameters HEAD
        E4DIntercalibration /cd_calice_Ahc2/TestbeamMay2018/Intercalibration HEAD
        E4DPhysicsCalibIntercalibration /cd_calice_Ahc2/TestbeamMay2018/PhysicsCalibIntercalibration HEAD
        E4DTimeSlopes /cd_calice_Ahc2/TestbeamMay2018/TimeSlope ahc2_006
        E4DTimeOffset /cd_calice_Ahc2/TestbeamMay2018/TimeOffset ahc2_006
```



See Lorenz's talk

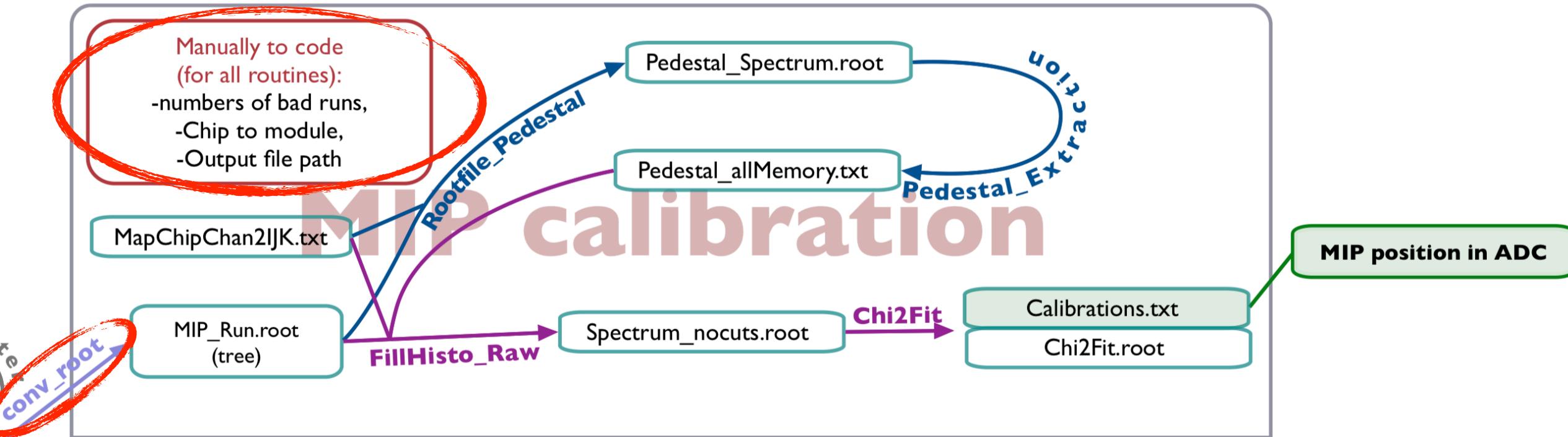
One of the Old MIP Calibrations

Schematic of Procedure



One of the Old MIP Calibrations

Schematic of Procedure

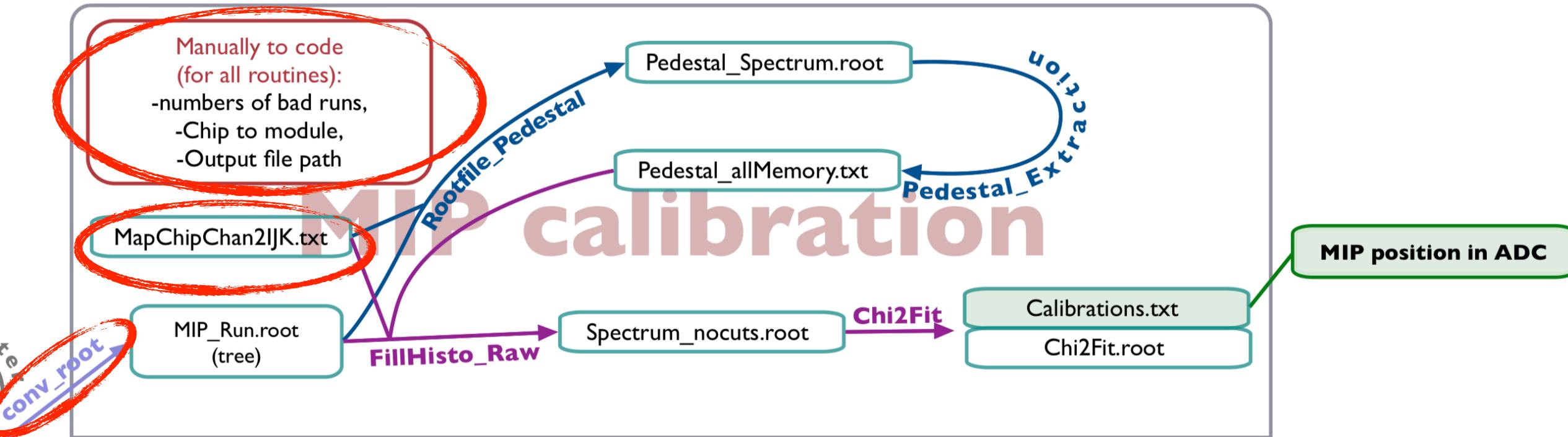


Problems:

- Different versions of routines, hardcoded parts

One of the Old MIP Calibrations

Schematic of Procedure

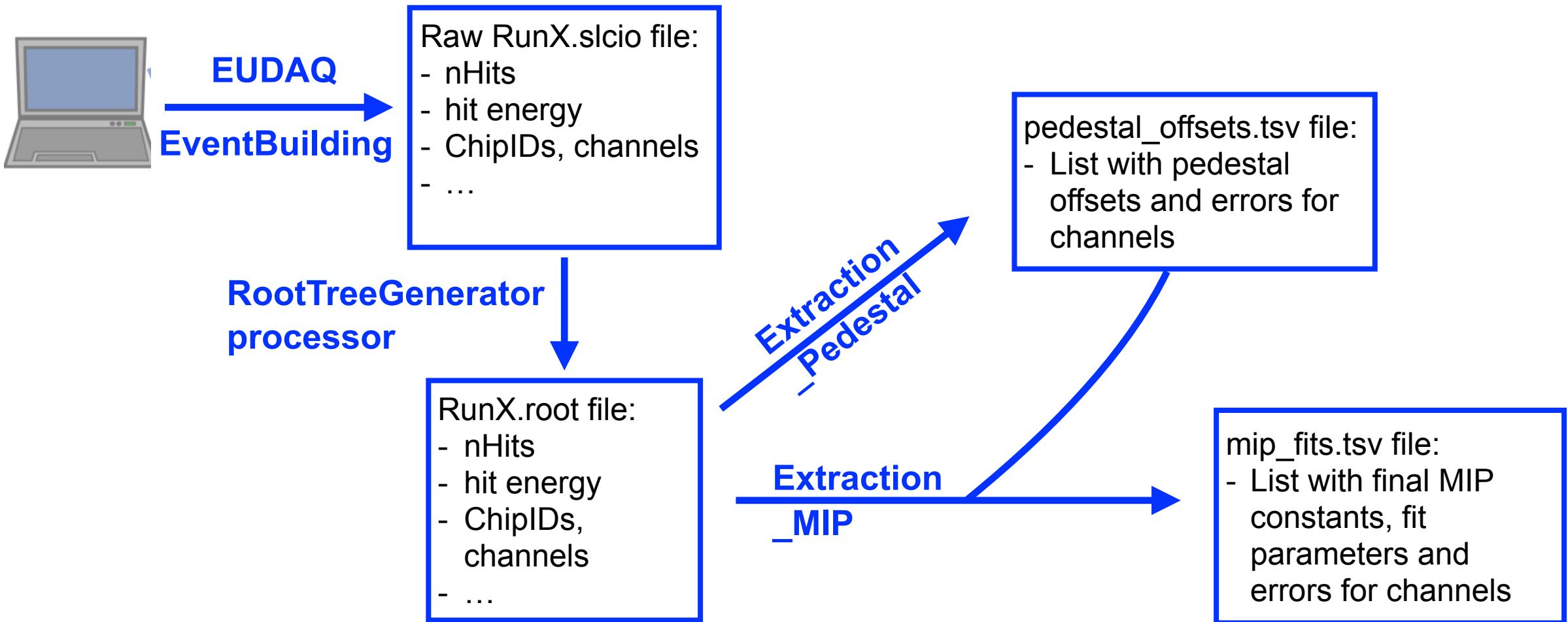


Problems:

- Different versions of routines, hardcoded parts
- Reconstruction with mapping, afterwards translation back to Chip/Channel

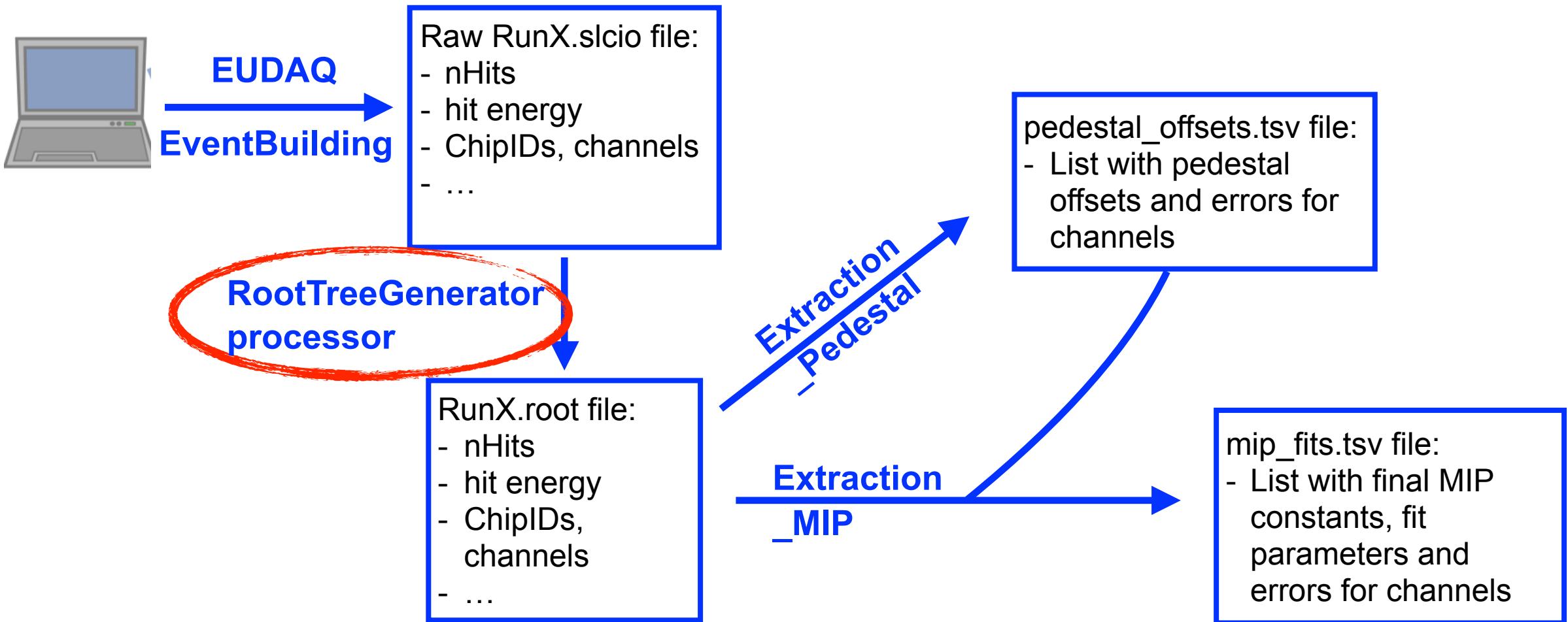
Overview MIP Calibration

Schematic of Procedure



Overview MIP Calibration

Schematic of Procedure



Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Goal: Convert the RunX.slcio file (stored on pnfs) to a RunX.root file for further processing:

- Event building already done by EUDAQ (latest software feature, no processor required!)
 - Sorting parameters like hits, ADC, ChipID, Channel, etc. according to events
- Generation of a root-tree structure (mytree) with the RootTreeGeneratorEUDAQ2016

Execute: ./myMarlin steering_mip_calib.xml

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Goal: Convert the RunX.slcio file (stored on pnfs) to a RunX.root file for further processing:

- Event building already done by EUDAQ (latest software feature, no processor required!)
 - Sorting parameters like hits, ADC, ChipID, Channel, etc. according to events
- Generation of a root-tree structure (mytree) with the RootTreeGeneratorEUDAQ2016

Execute: ./myMarlin steering_mip_calib.xml



Linking latest CaliceSoft libraries
(pre-release v04-11), initialize ILCsoft
and setting Marlin parameters

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Goal: Convert the RunX.slcio file (stored on pnfs) to a RunX.root file for further processing:

- Event building already done by EUDAQ (latest software feature, no processor required!)
 - Sorting parameters like hits, ADC, ChipID, Channel, etc. according to events
- Generation of a root-tree structure (mytree) with the RootTreeGeneratorEUDAQ2016

Execute: ./myMarlin steering_mip_calib.xml



The steering file for the
RootTree generation

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Steering file for running the processor:

- Global: Input RunX.slcio file

```
1 <?xml version="1.0" encoding="us-ascii"?>
2
3 <!--#####
4 #                               #
5 # Steering file for slcio -> RootTree   #
6 #                               #
7 #####-->
8
9
10<marlin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://ilcsoft.desy.de/marlin/marlin.xsd">
11
12  <execute>
13    <processor name="RootTreeGeneratorEUDAQ2016"/>
14    <processor name="ProgressHandler"/>
15  </execute>
16
17  <global>
18    <parameter name="LCIOInputFiles">
19      /pnfs/desy.de/calice/tb-cern/native/cernAhcalMay2018/slcio/Muon/40GeV/run060300_20180510_222723.slcio
20    </parameter>
21    <!-- Set the number of processed records (run+evt): -->
22    <parameter name="MaxRecordNumber" value="0" />
23    <parameter name="SkipNEvents" value="0" />
24    <parameter name="SupressCheck" value="false" />
25    <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> ERROR </parameter>
26  </global>
27
28  <processor name="ProgressHandler" type="ProgressHandler">
29    <!-- Report event/run numbers at fixed time intervals, and handle SIGINT (Ctrl-C) for a graceful exit.-->
30    <!--The number of seconds after which the number of processed events will be shown.-->
31    <parameter name="ReportInterval" type="int" value="10"/>
32  </processor>
33
34
35  <!-- Write root file containing Temp data -->
36  <processor name="RootTreeGeneratorEUDAQ2016" type="RootTreeGeneratorEUDAQ2016">
37    <!--Name of the output ROOT file-->
38    <parameter name="OutputRootFileName" type="string">
39      /afs/desy.de/group/flc/pool/heucheld/workspace/mip_calib/tb_cern_may18/Run60300.root
40    </parameter>
41    <parameter name="InputCollectionName" type="string">
42      EUDAQDataScCAL
43    </parameter>
44    <parameter name="BranchPrefix" type="string">
45      ahcal_
46    </parameter>
47  </processor>
48</marlin>
```

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Steering file for running the processor:

- Global: Input RunX.slcio file
- RootTreeGeneratorEUDAQ 2016

```
1 <?xml version="1.0" encoding="us-ascii"?>
2
3 <!--#####
4 #                               #
5 # Steering file for slcio -> RootTree   #
6 #                               #
7 #####-->
8
9
10<marlin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://ilcsoft.desy.de/marlin/marlin.xsd">
11
12  <execute>
13    <processor name="RootTreeGeneratorEUDAQ2016"/>
14    <processor name="ProgressHandler"/>
15  </execute>
16
17  <global>
18    <parameter name="LCIOInputFiles">
19      /pnfs/desy.de/calice/tb-cern/native/cernAhcalMay2018/slcio/Muon/40GeV/run060300_20180510_222723.slcio
20    </parameter>
21    <!-- limit the number of processed records (run+evt): -->
22    <parameter name="MaxRecordNumber" value="0"/>
23    <parameter name="$kipNEvents" value="0" />
24    <parameter name="SupressCheck" value="false" />
25    <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> ERROR </parameter>
26  </global>
27
28  <processor name="ProgressHandler" type="ProgressHandler">
29    <!--Report event/run numbers at fixed time intervals, and handle SIGINT (Ctrl-C) for a graceful exit.-->
30    <!--The number of seconds after which the number of processed events will be shown.-->
31    <parameter name="ReportInterval" type="int" value="10"/>
32  </processor>
33
34
35  <!-- write root file containing Temp data -->
36  <processor name="RootTreeGeneratorEUDAQ2016" type="RootTreeGeneratorEUDAQ2016">
37    <!-- name of the output ROOT file -->
38    <parameter name="OutputRootFileName" type="string">
39      /afs/desy.de/group/flc/pool/heuchelD/workspace/mip_calib/tb_cern_may18/Run60300.root
40    </parameter>
41    <parameter name="InputCollectionName" type="string">
42      EUDAQDataScCAL
43    </parameter>
44    <parameter name="BranchPrefix" type="string">
45      ahcal_
46    </parameter>
47  </processor>
48</marlin>
```

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Steering file for running the processor:

- Global: Input RunX.slcio file
- RootTreeGeneratorEUDAQ 2016
- Input Collection: EUDAQDataScCAL

```
1 <?xml version="1.0" encoding="us-ascii"?>
2
3 <!--#####
4 #                               #
5 # Steering file for slcio -> RootTree   #
6 #                               #
7 #####-->
8
9
10<marlin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://ilcsoft.desy.de/marlin/marlin.xsd">
11
12  <execute>
13    <processor name="RootTreeGeneratorEUDAQ2016"/>
14    <processor name="ProgressHandler"/>
15  </execute>
16
17  <global>
18    <parameter name="LCIOInputFiles">
19      /pnfs/desy.de/calice/tb-cern/native/cernAhcalMay2018/slcio/Muon/40GeV/run060300_20180510_222723.slcio
20    </parameter>
21    <!-- limit the number of processed records (run+evt): -->
22    <parameter name="MaxRecordNumber" value="0"/>
23    <parameter name="SkipNEvents" value="0" />
24    <parameter name="SupressCheck" value="false" />
25    <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> ERROR </parameter>
26  </global>
27
28  <processor name="ProgressHandler" type="ProgressHandler">
29    <!--Report event/run numbers at fixed time intervals, and handle SIGINT (Ctrl-C) for a graceful exit.-->
30    <!--The number of seconds after which the number of processed events will be shown.-->
31    <parameter name="ReportInterval" type="int" value="10"/>
32  </processor>
33
34
35  <!-- Write root file containing Temp data -->
36  <processor name="RootTreeGeneratorEUDAQ2016" type="RootTreeGeneratorEUDAQ2016">
37    <!--Name of the output ROOT file-->
38    <parameter name="OutputRootFileName" type="string">
39      /afs/desy.de/group/flc/pool/heuchel/workspace/mip_calib/tb_cern_may18/Run60300.root
40    </parameter>
41    <parameter name="InputCollectionName" type="string">
42      EUDAQDataScCAL
43    </parameter>
44    <parameter name="BranchPrefix" type="string">
45      ahcal_
46    </parameter>
47  </processor>
48</marlin>
```

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Steering file for running the processor:

- Global: Input RunX.slcio file
- RootTreeGeneratorEUDAQ 2016
 - Input Collection: EUDAQDataScCAL
 - Output path RunX.root

```
1 <?xml version="1.0" encoding="us-ascii"?>
2
3 <!--#####
4 #                               #
5 # Steering file for slcio -> RootTree   #
6 #                               #
7 #####-->
8
9
10<marlin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://ilcsoft.desy.de/marlin/marlin.xsd">
11
12  <execute>
13    <processor name="RootTreeGeneratorEUDAQ2016"/>
14    <processor name="ProgressHandler"/>
15  </execute>
16
17  <global>
18    <parameter name="LCIOInputFiles">
19      /pnfs/desy.de/calice/tb-cern/native/cernAhcalMay2018/slcio/Muon/40GeV/run060300_20180510_222723.slcio
20    </parameter>
21    <!-- limit the number of processed records (run+evt): -->
22    <parameter name="MaxRecordNumber" value="0"/>
23    <parameter name="$SkipNEvents" value="0" />
24    <parameter name="SupressCheck" value="false" />
25    <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> ERROR </parameter>
26  </global>
27
28  <processor name="ProgressHandler" type="ProgressHandler">
29    <!--Report event/run numbers at fixed time intervals, and handle SIGINT (Ctrl-C) for a graceful exit.-->
30    <!--The number of seconds after which the number of processed events will be shown.-->
31    <parameter name="ReportInterval" type="int" value="10"/>
32  </processor>
33
34
35  <!-- Write root file containing Temp data -->
36  <processor name="RootTreeGeneratorEUDAQ2016" type="RootTreeGeneratorEUDAQ2016">
37    <!--Name of the output ROOT file-->
38    <parameter name="OutputRootFileName" type="string">
39      /afs/desy.de/group/flc/pool/heuchelD/workspace/mip_calib/tb_cern_may18/Run60300.root
40    </parameter>
41    <parameter name="InputCollectionName" type="string">
42      EUDAQDataScCAL
43    </parameter>
44    <parameter name="BranchPrefix" type="string">
45      ahcal_
46    </parameter>
47  </processor>
48</marlin>
```

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Steering file for running the processor:

- Global: Input RunX.slcio file
- RootTreeGeneratorEUDAQ 2016
 - Input Collection: EUDAQDataScCAL
 - Output path RunX.root
 - Prefix for the tree entries

```
1 <?xml version="1.0" encoding="us-ascii"?>
2
3 <!--#####
4 #                               #
5 # Steering file for slcio -> RootTree   #
6 #                               #
7 #####-->
8
9
10<marlin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://ilcsoft.desy.de/marlin/marlin.xsd">
11
12  <execute>
13    <processor name="RootTreeGeneratorEUDAQ2016"/>
14    <processor name="ProgressHandler"/>
15  </execute>
16
17  <global>
18    <parameter name="LCIOInputFiles">
19      /pnfs/desy.de/calice/tb-cern/native/cernAhcalMay2018/slcio/Muon/40GeV/run060300_20180510_222723.slcio
20    </parameter>
21    <!-- limit the number of processed records (run+evt): -->
22    <parameter name="MaxRecordNumber" value="0"/>
23    <parameter name="SkipNEvents" value="0" />
24    <parameter name="SupressCheck" value="false" />
25    <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> ERROR </parameter>
26  </global>
27
28  <processor name="ProgressHandler" type="ProgressHandler">
29    <!--Report event/run numbers at fixed time intervals, and handle SIGINT (Ctrl-C) for a graceful exit.-->
30    <!--The number of seconds after which the number of processed events will be shown.-->
31    <parameter name="ReportInterval" type="int" value="10"/>
32  </processor>
33
34
35  <!-- Write root file containing Temp data -->
36  <processor name="RootTreeGeneratorEUDAQ2016" type="RootTreeGeneratorEUDAQ2016">
37    <!--Name of the output ROOT file-->
38    <parameter name="OutputRootFileName" type="string">
39      /afs/desy.de/group/flc/pool/heucheld/workspace/mip_calib/tb_cern_may18/Run60300.root
40    </parameter>
41    <parameter name="InputCollectionName" type="string">
42      EUDAQDataScCAL
43    </parameter>
44    <parameter name="BranchPrefix" type="string">
45      ahcal_
46    </parameter>
47  </processor>
48</marlin>
```

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Steering file for running the processor:

- Global: Input RunX.slcio file
- RootTreeGeneratorEUDAQ 2016
 - Input Collection: EUDAQDataScCAL
 - Output path RunX.root
 - Prefix for the tree entries
- ProgressHandler

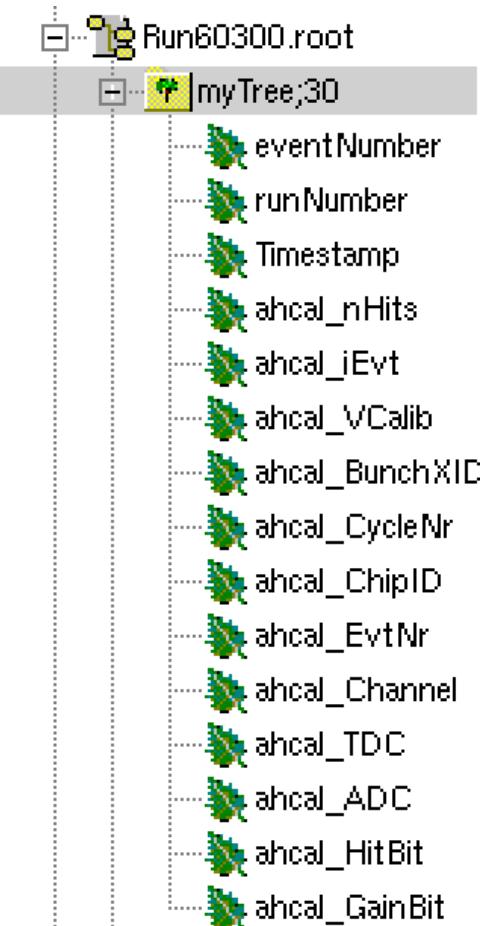
```
1 <?xml version="1.0" encoding="us-ascii"?>
2
3 <!--#####
4 #                               #
5 # Steering file for slcio -> RootTree   #
6 #                               #
7 #####-->
8
9
10<marlin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://ilcsoft.desy.de/marlin/marlin.xsd">
11
12  <execute>
13    <processor name="RootTreeGeneratorEUDAQ2016"/>
14    <processor name="ProgressHandler"/>
15  </execute>
16
17  <global>
18    <parameter name="LCIOInputFiles">
19      /pnfs/desy.de/calice/tb-cern/native/cernAhcalMay2018/slcio/Muon/40GeV/run060300_20180510_222723.slcio
20    </parameter>
21    <!-- limit the number of processed records (run+evt): -->
22    <parameter name="MaxRecordNumber" value="0"/>
23    <parameter name="SkipNEvents" value="0" />
24    <parameter name="SupressCheck" value="false" />
25    <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> ERROR </parameter>
26  </global>
27
28  <processor name="ProgressHandler" type="ProgressHandler">
29    <!--Report event/run numbers at fixed time intervals, and handle SIGINT (Ctrl-C) for a graceful exit.-->
30    <!--The number of seconds after which the number of processed events will be shown.-->
31    <parameter name="ReportInterval" type="int" value="10"/>
32  </processor>
33
34
35  <!-- Write root file containing Temp data -->
36  <processor name="RootTreeGeneratorEUDAQ2016" type="RootTreeGeneratorEUDAQ2016">
37    <!--Name of the output ROOT file-->
38    <parameter name="OutputRootFileName" type="string">
39      /afs/desy.de/group/flc/pool/heucheld/workspace/mip_calib/tb_cern_may18/Run60300.root
40    </parameter>
41    <parameter name="InputCollectionName" type="string">
42      EUDAQDataScCAL
43    </parameter>
44    <parameter name="BranchPrefix" type="string">
45      ahcal_
46    </parameter>
47  </processor>
48</marlin>
```

Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Output RunX.root tree:

- Showing the basic parameters for all events/hits

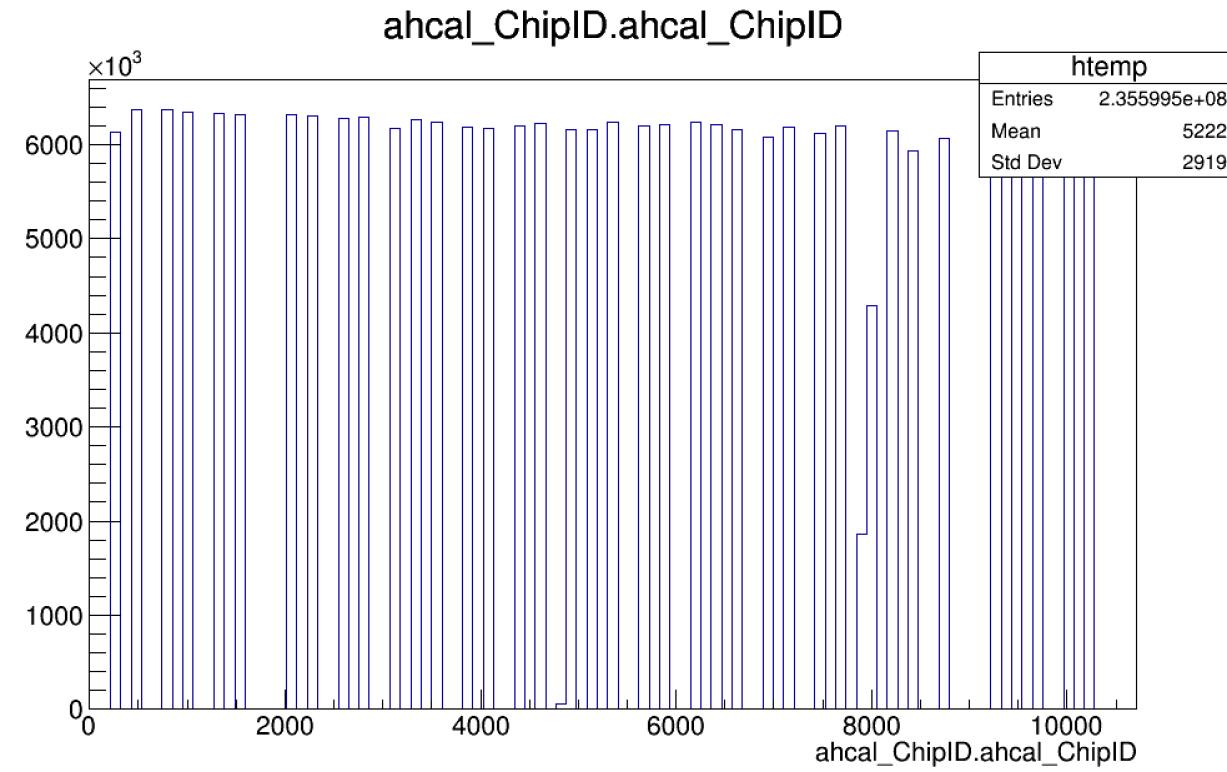
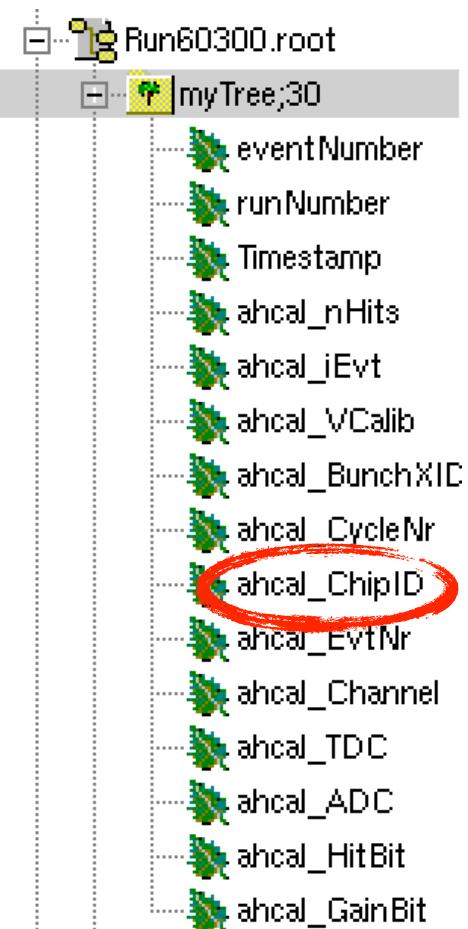


Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Output RunX.root tree:

- Showing the basic parameters for all events/hits

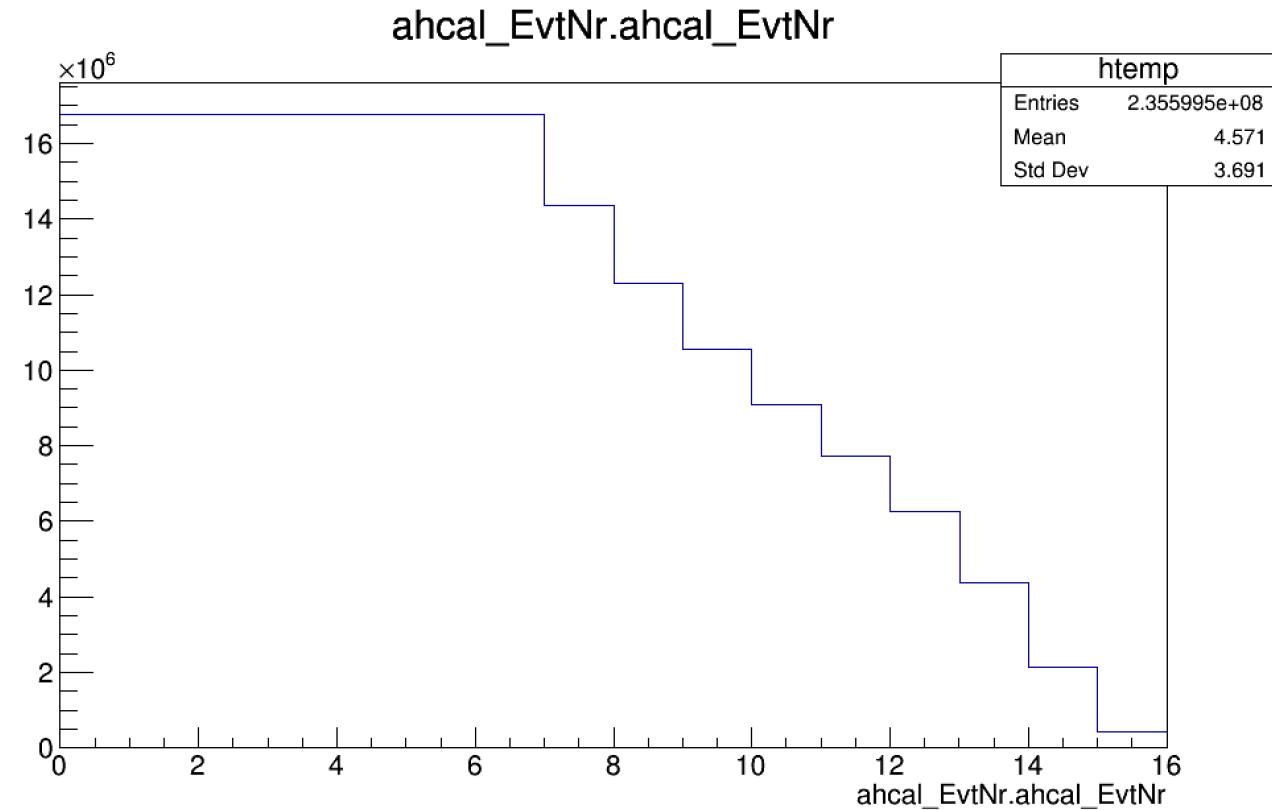
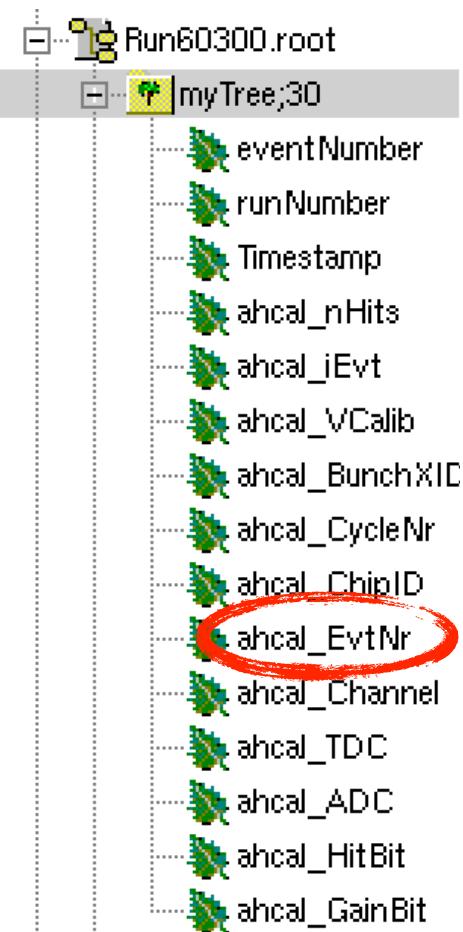


Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

Output RunX.root tree:

- Showing the basic parameters for all events/hits

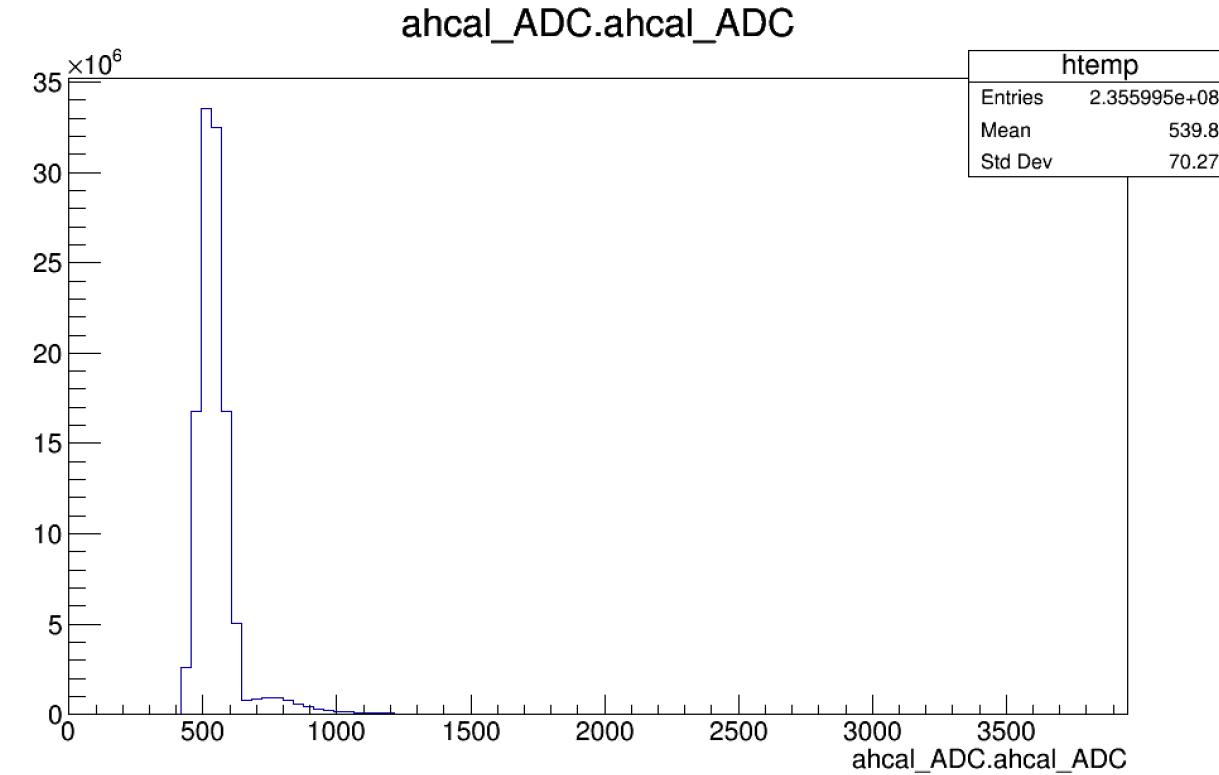
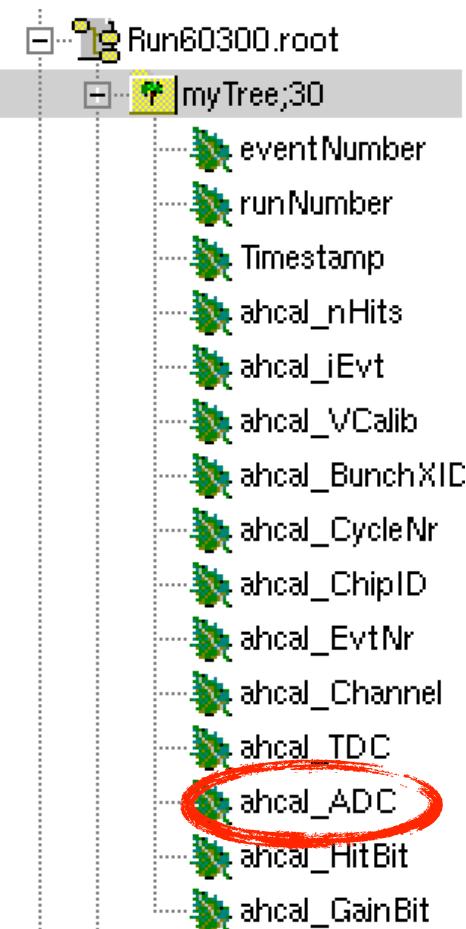


Step 1: Steering - RootTreeGenerator

From RunX.slcio to RunX.root

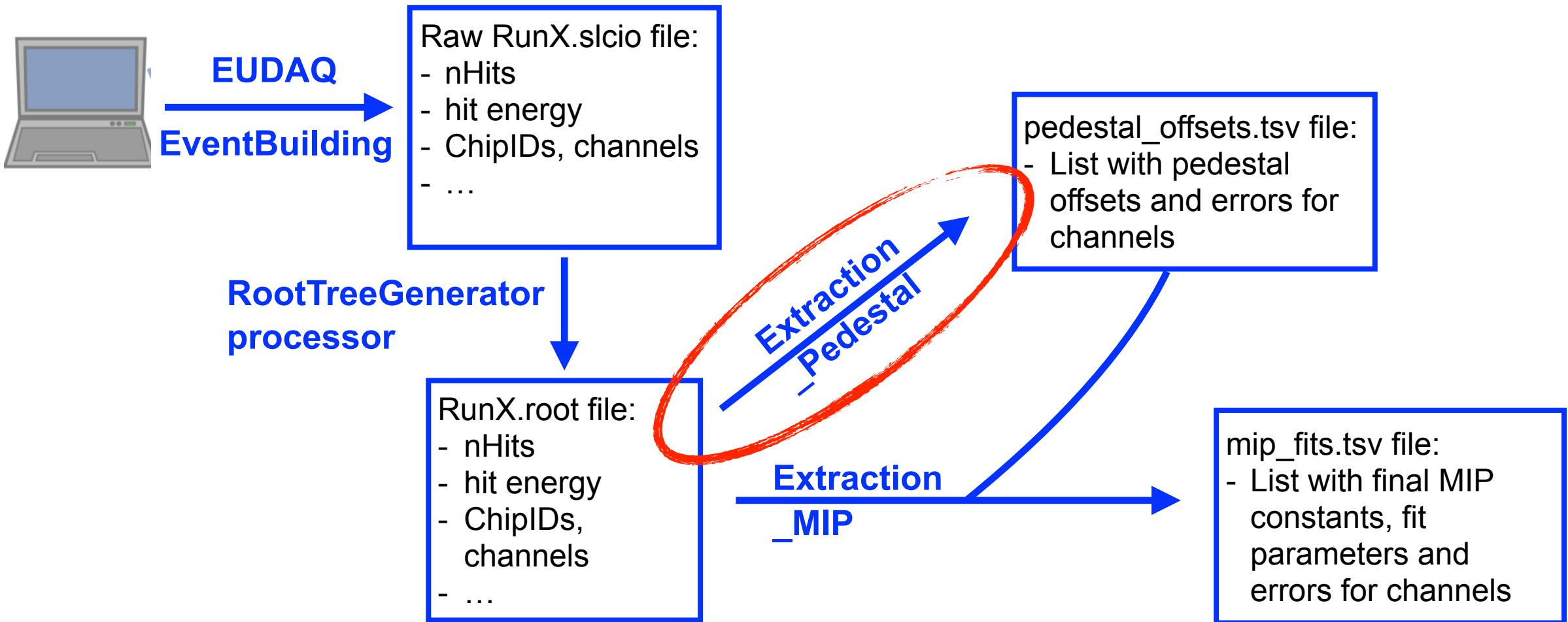
Output RunX.root tree:

- Showing the basic parameters for all events/hits



Overview MIP Calibration

Schematic of Procedure



Step 2: Pedestal Extraction

From RunX.root to `pedestal_offsets_in.tsv`

- Software package: Extraction_Pedestal set up by Eldwan and Olin
- Can be found in: [/afs/desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware](https://afs/desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware)
- Used for pedestal extraction for MIP calibration (AT), as well as gain calibration (ET)

Step 2: Pedestal Extraction

From RunX.root to pedestal_offsets_in.tsv

- Software package: Extraction_Pedestal set up by Eldwan and Olin
- Can be found in: [/afs/desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware](https://afs.desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware)
- Used for pedestal extraction for MIP calibration (AT), as well as gain calibration (ET)

```
Execute: ./pedestal_memcell [INPUT FILE] [OUTPUT FOLDER] [MODE 0(HG/LG) or 1(HG/TDC)] [TRIGGER 1(AT) or 0(ET)]
```

In this example:

```
Execute: ./pedestal_memcell Run60300.root /afs/desy.de/group/flc/pool/heuchel/workspace/mip_calib/tb_cern_may18/ 1 1
```

Step 2: Pedestal Extraction

What is happening?

- Based on two main functions:
 - ➔ MakePedestalHistograms()
 - ➔ Read in mytree
 - ➔ Loops over events, nhits to fill histograms
 - ➔ Create histograms for each triggered ChipID channels (extract pedestal and width for channel) and channel memory cells (extract the memory cells offset)
 - ➔ Fill ADC values of hits

Step 2: Pedestal Extraction

What is happening?

- Based on two main functions:
 - ➔ MakePedestalHistograms()
 - ➔ Read in mytree
 - ➔ Loops over events, nhits to fill histograms
 - ➔ Create histograms for each triggered ChipID channels (extract pedestal and width for channel) and channel memory cells (extract the memory cells offset)
 - ➔ Fill ADC values of hits
 - ➔ Result: Map with ChipID, vector with 36 histograms for channels and 16 histograms each for memory cells filled with corresponding ADC values

Step 2: Pedestal Extraction

What is happening?

- Based on two main functions:
 - ➔ ExtractATPedestals()
 - ➔ Loop over all histograms (channels, individual memorycells)
 - ➔ Two conditions for pedestal extraction:
 1. nEntries > 100
 2. TSpectrum::GetNPeaks() > 0
 - ➔ If histo does not pass this: pedestal, pedestal width = 0
 - ➔ Else: GetMean and GetWidth as pedestal and pedestal width
 - ➔ Before: 2x Re-iteration by SetRangeUser(Mean - 3 * RMS, Mean + 3 * RMS)

Step 2: Pedestal Extraction

What is happening?

- Based on two main functions:
 - ➔ ExtractATPedestals()
 - ➔ Loop over all histograms (channels, individual memorycells)
 - ➔ Two conditions for pedestal extraction:
 1. nEntries > 100
 2. TSpectrum::GetNPeaks() > 0
 - ➔ If histo does not pass this: pedestal, pedestal width = 0
 - ➔ Else: GetMean and GetWidth as pedestal and pedestal width
 - ➔ Before: 2x Re-iteration by SetRangeUser(Mean - 3 * RMS, Mean + 3 * RMS)
 - ➔ Calculate pedestal_offset for each memory cell of the channel
 - ➔ Save results in list

Step 2: Pedestal Extraction

From RunX.root to pedestal_offsets_in.tsv

Output pedestal_offsets_in.tsv file:

- ChipID
- Channel #
- Pedestal position all
- Pedestal position all error
- Pedestal offset memory cell

Output Plots_comp.root file:

- All pedestal spectra: ChipID, Channel, MemCell

Step 2: Pedestal Extraction

From RunX.root to pedestal_offsets_in.tsv

Output pedestal_offsets_in.tsv file:

- ChipID
- Channel #
- Pedestal position all
- Pedestal position all error
- Pedestal offset memory cell

Output Plots_comp.root file:

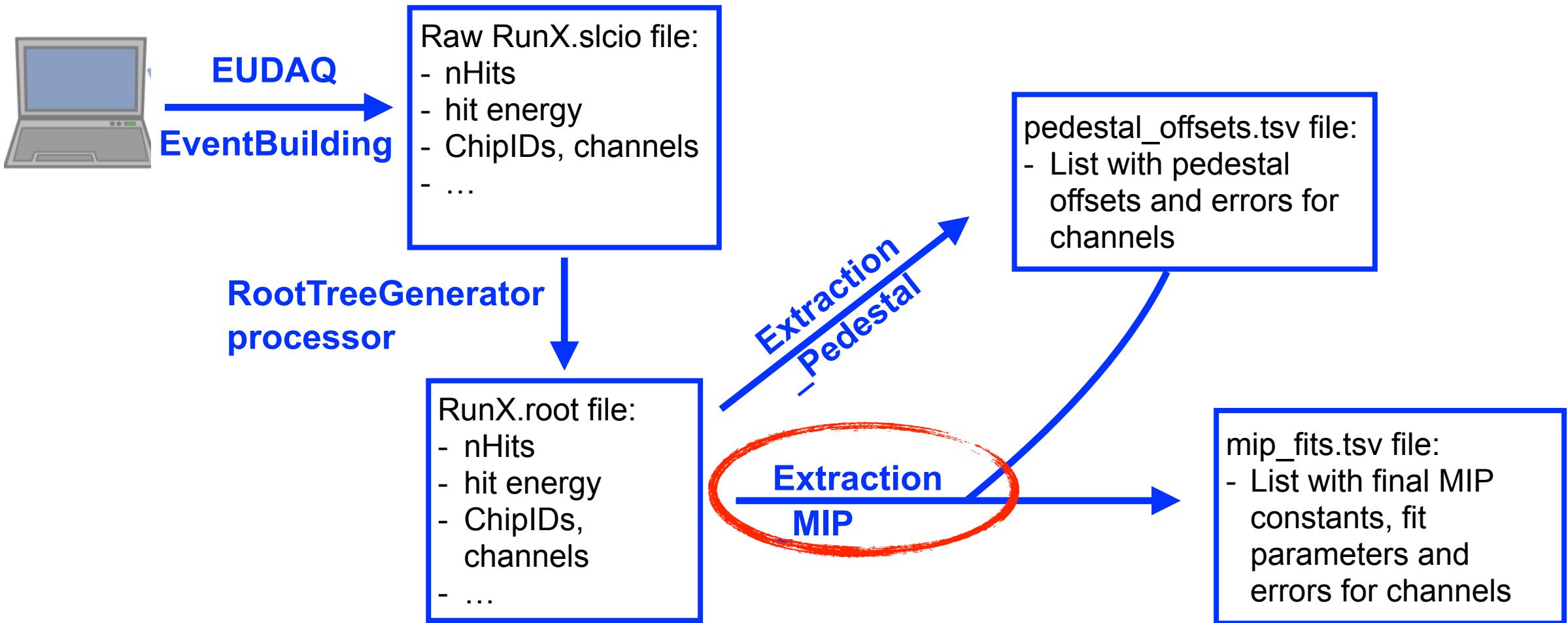
- All pedestal spectra: ChipID, Channel, MemCell

```
#pedestal positions & memory cell dependent offsets (tpedOffsetcellX = tpedOffsetcell2 - tpedOffsetcell1) from file "/afs/desy.de/group/flc/pool/heuchel/workspace/mip_calib/tb_cern_may18/test_Run60300//pedestal_offsets_in.tsv"
#format: the ordering of memory cells is inverted for DAQ HBU
#chip  chn      pedposall      pedwidthall      pedOffsetcell1  pedOffsetcell2  pedOffsetcell3 ...      pedOffsetcell16
271    9      558.551 13.4964 7.74366 0      6.66062 11.9182 4.25365 -5.00803      0.97751 7.53793 9.21107 17.9237 -1.70654      1.7547 12.0196 7.67841 13.4744 -1.82571
271   10      591.152 14.7697 3.33799 0      7.93552 8.639 -2.95822      4.23467 10.7301 11.0614 16.7624 3.93781 -2.27164      18.0629 9.58617 18.6462 -11.0692      15.7176
271   11      582.666 15.3139 6.4772 0      19.8698 12.6352 15.0605 2.39024 11.596      4.56609 18.1837 14.6171 6.26952 7.13976 16.0034 14.1037 4.15006 11.1487
271   12      590.893 14.6599 -2.02211      0      13.1367 10.7085 -5.02162      1.14644 3.66356 4.80892 0.979461      4.83139 1.81398 3.5496 12.4306 12.6894 2.49631 3.63031
271   13      592.848 15.0129 1.64906 0      10.3883 -1.42588      1.05478 9.9227 6.06266 1.96027 7.40683 6.71649 -6.88181      2.06115 3.84188 -4.62579      3.7894 4.72878
271   14      561.369 13.745 -4.48405      0      8.12329 8.16825 10.3668 12.123 20.5284 8.41518 10.222 14.1574 9.5587 4.05037 -2.77539      3.1929 6.51363 4.93882
271   15      591.966 15.557 9.04309 0      9.26293 12.123 -3.5276 17.8027 11.2772 -2.91484      -0.0347748      3.52134 5.30886 11.2422 8.93613 6.85962 -4.46172      4.1089
```

- This list normally contains a lot of 0 for a single run, since not all channels survived the statistics condition; not used in further MIP calibration

Overview MIP Calibration

Schematic of Procedure



Step 3: MIP Constant Extraction

The Extraction_MIP Software Package

- Software package: Extraction_MIP set up by Eldwan and me
- Can be found in: /afs/desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware
- Used for the MIP constant extraction with errors for ChipID, channel

```
Execute: ./mip_extractor [INPUT ROOT FILE] [INPUT PEDESTAL FILE] [OUTPUT FOLDER]
```

In this example:

```
Execute: ./mip_extractor Run60300.root pedestal_offsets_in.tsv /afs/desy.de/group/flc/pool/heuchel/workspace/mip_calib/tb_cern_may18/
```

Step 3: MIP Constant Extraction

Filling Histograms

- Based on two main functions:
 - ➔ MakeHistograms()
 - ➔ Read in mytree and pedestal_offsets_in.tsv
 - ➔ Loops over events and nhits to fill histograms
 - ➔ Create 36 histograms for each channel of a triggered Chip according to ChipID
 - ➔ Memory cell-wise pedestal subtraction on ADC values, depending in which memory cell the hit was saved
 - ➔ Fill ADC value into histogram

Step 3: MIP Constant Extraction

Filling Histograms

- Based on two main functions:
 - ➔ MakeHistograms()
 - ➔ Read in mytree and pedestal_offsets_in.tsv
 - ➔ Loops over events and nhits to fill histograms
 - ➔ Create 36 histograms for each channel of a triggered Chip according to ChipID
 - ➔ Memory cell-wise pedestal subtraction on ADC values, depending in which memory cell the hit was saved
 - ➔ Fill ADC value into histogram
 - ➔ Result: Map with ChipID, vector with 36 histograms for channels filled with memory cell-wise pedestal subtracted ADC values

Step 3: MIP Constant Extraction

Fitting Histograms

- Based on two main functions:
 - ➔ FitHistograms()
 - ➔ Loop over all histograms for channels
 - ➔ Statistical rejection of channels with histogram entries < 1000
 - ➔ Each histogram fitted with 3-step Landau convoluted Gaussian-fit

Step 3: MIP Constant Extraction

Fitting Histograms

- Based on two main functions:
 - ➔ FitHistograms()
 - ➔ Loop over all histograms for channels
 - ➔ Statistical rejection of channels with histogram entries < 1000
 - ➔ Each histogram fitted with 3-step Landau convoluted Gaussian-fit
 - ➔ 1. Only with free area (integral)-parameter, others fixed, larger range, start parameters
 - ➔ 2. Release other fit parameters (landau width, mpv and gauss width), adapt and set limits
 - ➔ 3. Apply corrected fit range (+/- 0.3*max), if necessary correct gaussian and landau width parameters and set parameter limits for last fit

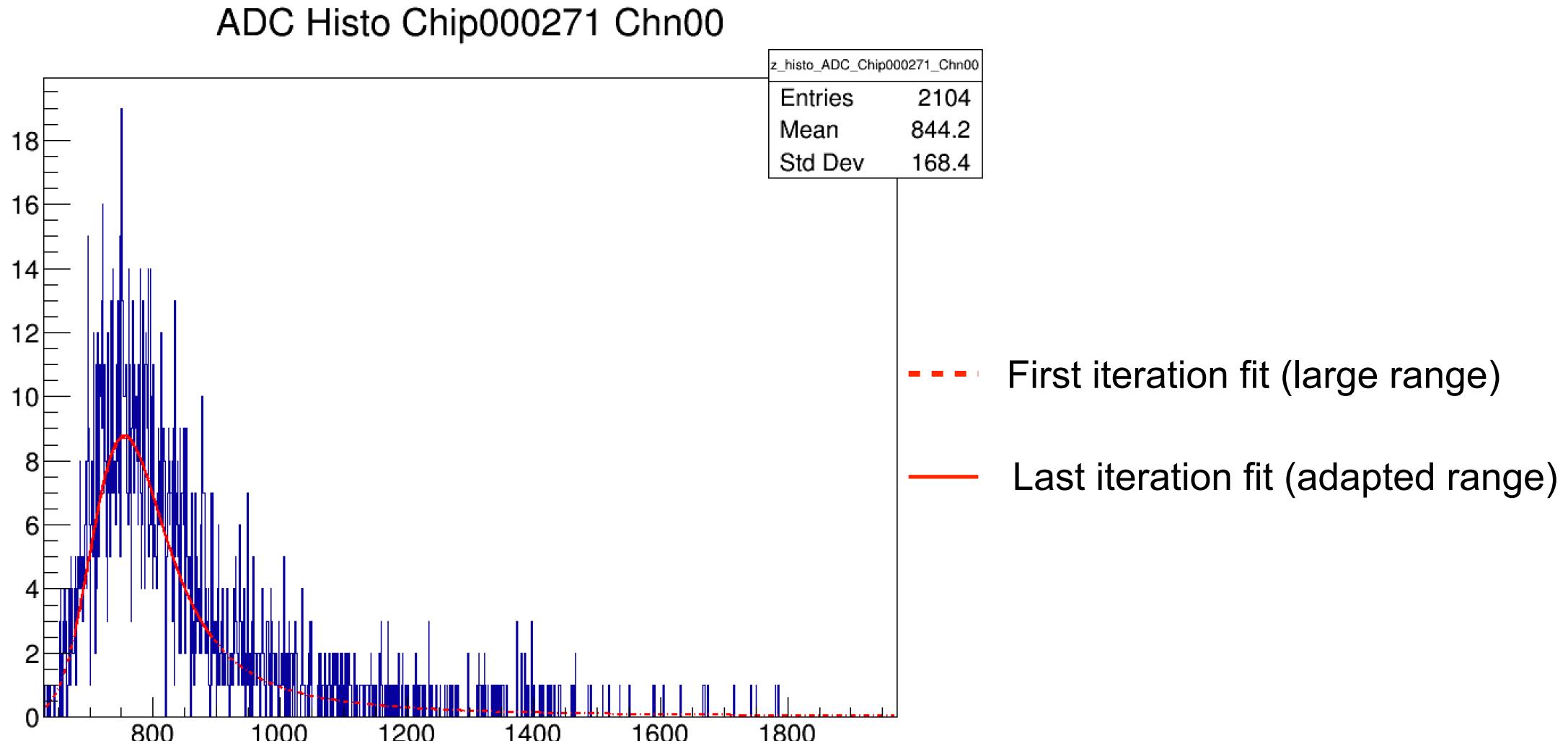
Step 3: MIP Constant Extraction

Fitting Histograms

- Based on two main functions:
 - ➔ FitHistograms()
 - ➔ Loop over all histograms for channels
 - ➔ Statistical rejection of channels with histogram entries < 1000
 - ➔ Each histogram fitted with 3-step Landau convoluted Gaussian-fit
 - ➔ 1. Only with free area (integral)-parameter, others fixed, larger range, start parameters
 - ➔ 2. Release other fit parameters (landau width, mpv and gauss width), adapt and set limits
 - ➔ 3. Apply corrected fit range (+/- 0.3*max), if necessary correct gaussian and landau width parameters and set parameter limits for last fit
 - ➔ Return: MPV, MPV error, other fit function parameters, Chi2, Ndf
 - ➔ Quality condition: If $\text{Chi2}/\text{Ndf} < 2$ = good fit, else: bad fit
 - ➔ Save all fitted spectra, result list with good and bad fits

Step 3: MIP Constant Extraction

Fitting Histograms



Step 3: MIP Constant Extraction

Results

- Output:
- mip_fits.tsv with the good fit results: Chi2/Ndf < 2

ChipID	Chn	MPV	MPV_Error	lw	gw	Chi2	NDF
268	0	673.103	2.5292	25.0184	41.2934	278.962	239
268	1	680.337	4.71912	24.7469	59.5448	246.522	255
268	6	629.398	2.99281	21.594	39.126	204.567	209
268	7	657.538	5.31974	39.6927	43.1706	190.059	256
268	12	661.448	10.5321	5.00001	72.6658	184.369	227
269	0	808.073	4.87204	27.7327	73.1938	361.243	358
269	1	773.803	4.14243	21.151	62.0799	380.41	277
269	2	810.83	7.15798	16.2876	69.8939	306.428	288
269	3	769.095	3.12708	30.0203	56.7156	323.15	287
269	4	784.512	2.6415	29.5828	44.7732	276.157	241
269	5	801.825	2.86863	32.3082	51.3993	319.942	325
269	6	784.235	4.3716	36.7797	64.3415	270.578	357
269	7	808.24	4.70117	27.9698	67.3279	289.045	335
269	8	785.323	3.27884	35.2598	51.7595	355.361	289
269	9	770.932	3.00798	32.3835	46.7244	257.637	275
269	10	795.516	3.54293	28.6606	57.8761	329.908	300
269	11	805.473	4.20684	27.3118	65.473	324.703	348
269	12	807.459	6.87508	40.7626	85.5432	206.337	327
269	13	797.481	6.40213	58.6203	59.8241	319.296	371
269	14	788.65	4.10911	41.983	47.7888	254.956	344
269	15	776.177	4.70733	44.0805	59.2747	271.022	345
269	16	783.049	6.11773	48.1061	62.8718	286.222	329
269	17	771.273	5.28215	48.0608	39.7703	245.63	315
270	4	841.049	28.8819	19.1168	136.575	310.14	412
270	5	772.82	3.0083	5	70.4966	237.693	237
270	10	773.936	4.93563	43.8424	42.5714	274.828	309
270	11	796.478	3.50063	41.7425	31.5341	320.35	286
270	16	871.709	4.54623	47.2426	64.3116	311.387	403
270	17	835.342	5.01187	31.0672	78.9023	383.045	371
270	22	848.503	4.87697	48.5332	51.6434	278.936	339

Step 3: MIP Constant Extraction

Results

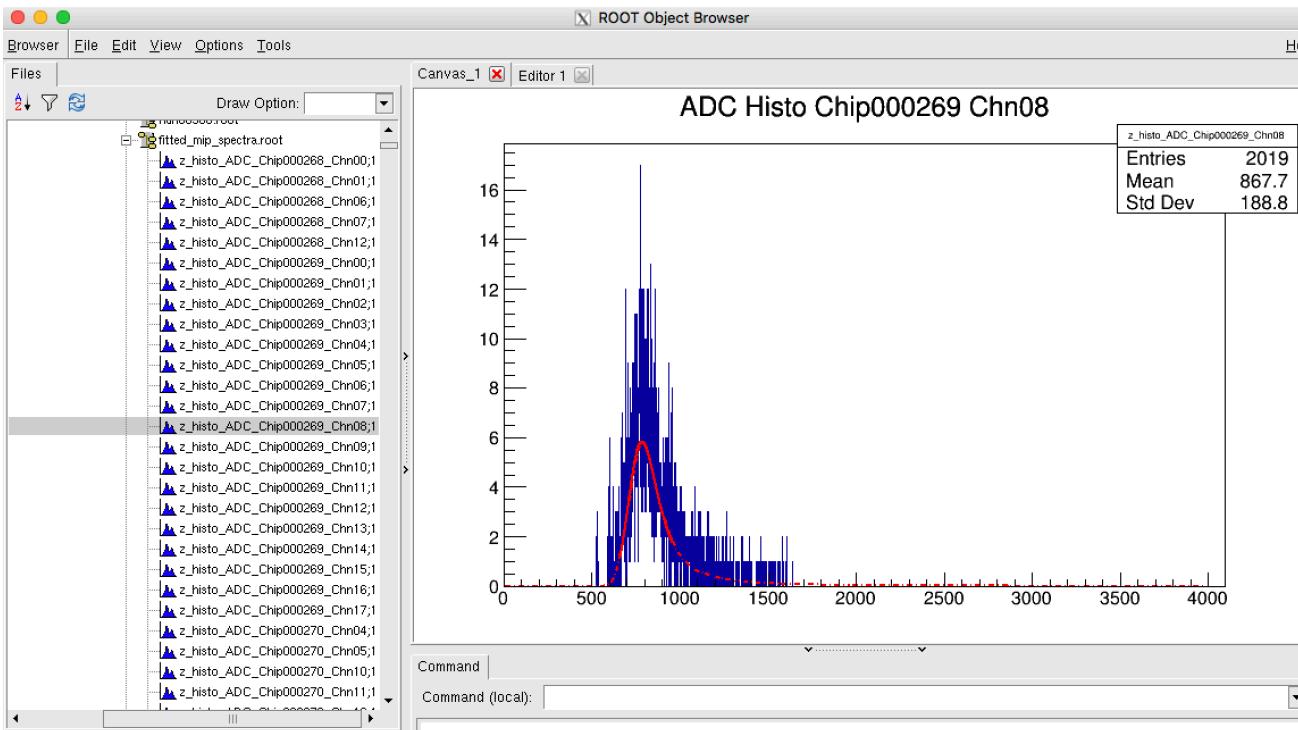
- Output:
 - `mip.fits.tsv` with the good fit results: $\text{Chi2/Ndf} < 2$
 - `mip.fits_bad.tsv` with bad fit result: $\text{Chi2/Ndf} > 2$

ChipID	Chn	MPV	MPV_Error	lw	gw	Chi2	NDF
5900	31	1098.97	4.26705	5	235.31	939.007	413.03

Step 3: MIP Constant Extraction

Results

- Output:
 - `mip.fits.tsv` with the good fit results: $\text{Chi}^2/\text{Ndf} < 2$
 - `mip.fits_bad.tsv` with bad fit result: $\text{Chi}^2/\text{Ndf} > 2$
 - `fitted_mip_spectra.root` with fitted spectra for all channels $n\text{Hits} \geq 1000$



Summary

MIP Calibration

- Far more automatized and less hard-coded Pedestal Extraction/MIP calibration package developed
 - No back and forth mapping
 - Easy steering of .slcio file and directly addressing only ChipID and channel
 - Only 3 commands in shell for full MIP and pedestal constants extraction!
 - MIP calibration output: Two lists with good and bad fits and corresponding results, fitted spectra
 - Big thank you to Eldwan!
 - Documentation on Confluence to be done!

Summary

MIP Calibration

- Far more automatized and less hard-coded Pedestal Extraction/MIP calibration package developed
 - ➔ No back and forth mapping
 - ➔ Easy steering of .slcio file and directly addressing only ChipID and channel
 - ➔ Only 3 commands in shell for full MIP and pedestal constants extraction!
 - ➔ MIP calibration output: Two lists with good and bad fits and corresponding results, fitted spectra
 - ➔ Big thank you to Eldwan!
 - ➔ Documentation on Confluence to be done!
- Software (ready) for performing MIP calibration for May/June muon runs of test beam
 - First: implementing fit for pedestal extraction instead of mean, rms
 - ➔ All .slcio files of muon scan, big .root file, mip constant extraction of all channels passing cuts!
 - ➔ Crosscheck and upload to data base

Hands-on Exercise

Perform a MIP Calibration

- Your task: Perform a MIP calibration for a specific muon run of the May 2018 testbeam campaign!
 - The files can be found here: `/pnfs/desy.de/calice/tb-cern/native/cernAhcalMay2018/slcio/Muon/40GeV/`
 - Use the different software packages to extract the MIP constants saved in the `mip.fits.tsv` file
- A few advices:
 - Check in the FLC wiki good/bad run list if it was a good run with a high number of events first!
 - In the may 2018 testbeam campaign at SPS only the 38 big AHCAL-layers were installed
 - Of course not all ~22000 channels can be calibrated with this single muon run (single beam position); The statistics- and the fit quality-cut in the software will automatically select the appropriate channels illuminated by the muon beam (only one run, due to time reasons)
 - Everything you need for step 1 can be found here: https://stash.desy.de/projects/CALICE/repos/calice_tutorial/browse/MIP_Calibration; Do a git clone to your local!
 - Software packages for step 2 and 3 are located here: [/afs/desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware](https://afs.desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware)

Hands-on Exercise

Perform a MIP Calibration - Guideline

First clone the calice_tutorial folder to your local machine:

```
git clone https://stash.desy.de/scm/calice/calice_tutorial.git
```

Chose a RunX from the pnfs and good/bad run list an modify the steering file input and output, then:

```
./myMarlin steering_mip_calib_may2018.xml
```

Now you should have the RunX.root (root tree) file. Check it in TBrowser, then:

```
cd /afs/desy.de/group/flc/hcal/calice_soft/pre-v04-11/flchcalsoftware/myInstall/bin
```

Now you are in the flcahcalsw environment, first do the pedestal extraction for AT

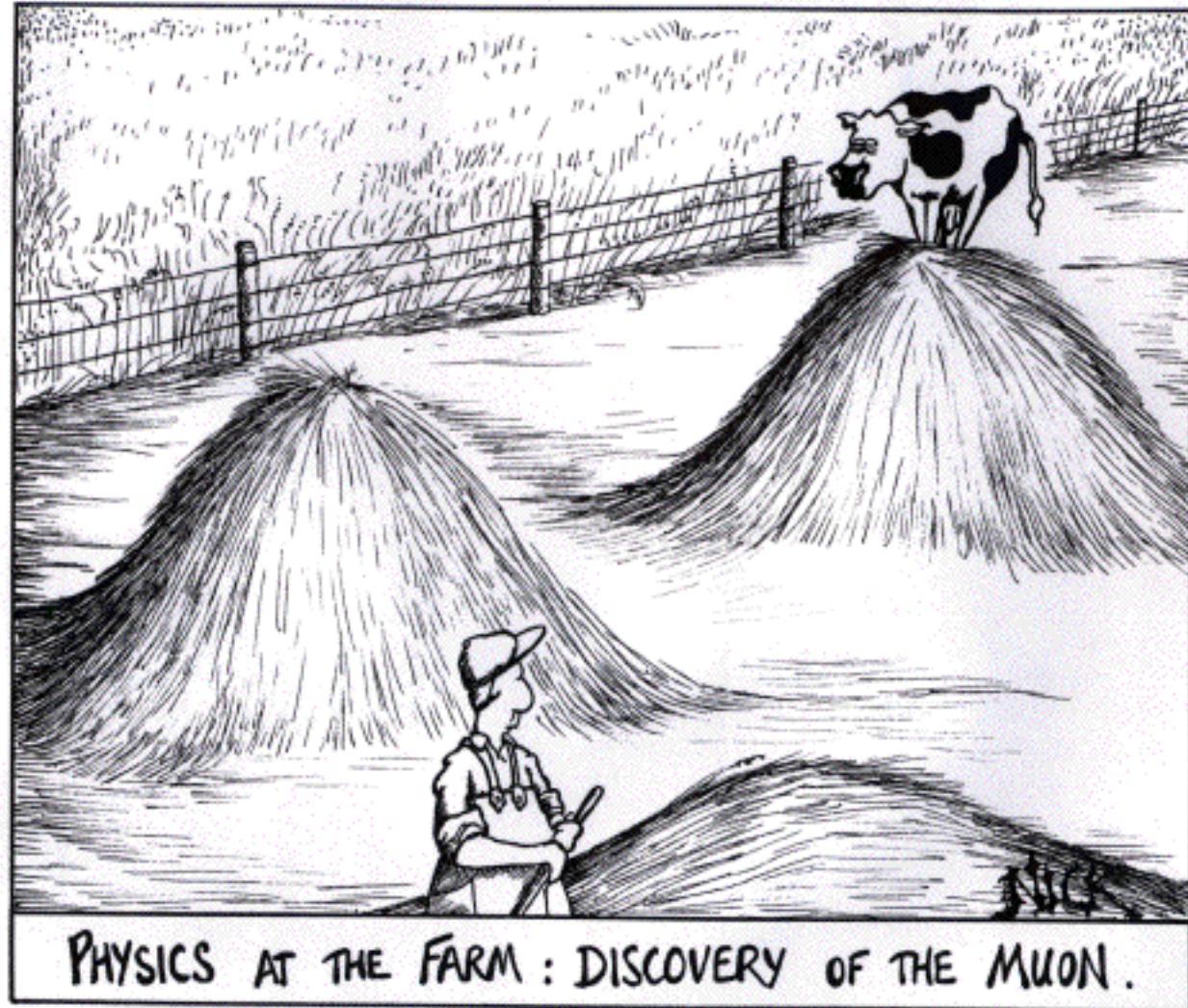
```
./pedestal_memcell path/to/RunX.root path/to/your/output/folder/ 1 1
```

Check the outcome for the pedestal_offsets_in.tsv file, then do the MIP constant extraction:

```
./mip_extractor path/to/RunX.root path/to/pedest_offsets_in.tsv /path/to/your/output/folder/
```

After finishing investigate the mip.fits.tsv, mip.fits_bad.tsv file and the fitted_mip_spectra.root and check the number of good/bad channels.

Thank you!



Backup

Pedestal Extraction Details

Mode Selection to Fill Histograms

```
if(this->getTriggerMode() == 0)
{
    if(this->getMode() == 1)
    {
        if (ADC->at(i) > 150 && GainBit->at(i) == 1){
            histos[ChipID->at(i)][chn->at(i)]->Fill(ADC->at(i), EvtNr->at(i));
        }
    }
    else
    {
        if (TDC->at(i) > 150 && GainBit->at(i) == 1){
            histos[ChipID->at(i)][chn->at(i)]->Fill(TDC->at(i), EvtNr->at(i));
        }
    }
}

if(this->getTriggerMode() == 1)
{
    if(HitBit->at(i) != 0) continue;

    //Should be running by default in HG/TDC
    if (ADC->at(i) > 150 && GainBit->at(i) == 1){
        histos[ChipID->at(i)][chn->at(i)]->Fill(ADC->at(i), EvtNr->at(i));
    }
}
```

Pedestal Extraction Details

Calculation of PedestalAll and PedestalOffsets:

```
s->Search(thisHisto, 4, "", 0.5);
int NPeaks = s->GetNPeaks();

if (NPeaks > 0)
{
    if(thisHisto->GetEntries() > 100)//For a reliable extraction of the pedestal value...
    {
        //Computing the pedestal
        double RMS = thisHisto->GetRMS();
        double Mean = thisHisto->GetMean();
        thisHisto->GetXaxis()->SetRangeUser(Mean - 3 * RMS, Mean + 3 * RMS);

        //Re-iterate range
        Mean = thisHisto->GetMean();
        RMS = thisHisto->GetMean();
        thisHisto->GetXaxis()->SetRangeUser(Mean - 3 * RMS, Mean + 3 * RMS);

        *thisPedestalPosition = thisHisto->GetMean();
        *thisPedestalWidth = thisHisto->GetRMS();
    }
    else
    {
        *thisPedestalPosition = 0;
        *thisPedestalWidth = 0;
    }
}

for (int i=0; i<16; i++) _pedestalOffsets.at(i) = _pedestalPositions.at(1) - _pedestalPositions.at(i);
```

MIP Extraction Details

Landau-Gauss-Fit, last iteration

```
// 3rd fit with corrected range
TH1F tmp("tmp", "tmp", 2501, -.5, 2500.5);
max = landau_gauss_function->GetMaximum();
tmp.Eval(landau_gauss_function);
lower_bound = tmp.GetBinCenter(tmp.FindFirstBinAbove(.30*max));
upper_bound = tmp.GetBinCenter(tmp.FindLastBinAbove (.30*max));
landau_gauss_function->SetRange(lower_bound, upper_bound);

// correct parameters landau and gauss width
landau_gauss_function->GetParameters(parameters);
if(parameters[3] >= 90. || parameters[3] <= 35.) parameters[3] = 50;
if(parameters[0] < 30.) parameters[3] = 30;
landau_gauss_function->SetParameters(parameters);
landau_gauss_function->SetParLimits(0, 5, 1000);
landau_gauss_function->SetParLimits(3, 15, 500);
TFitResultPtr result = hist.Fit(landau_gauss_function, "RQBS");

landau_gauss_function->GetParameters(parameters);
landau_gauss_nobound ->SetParameters(parameters);

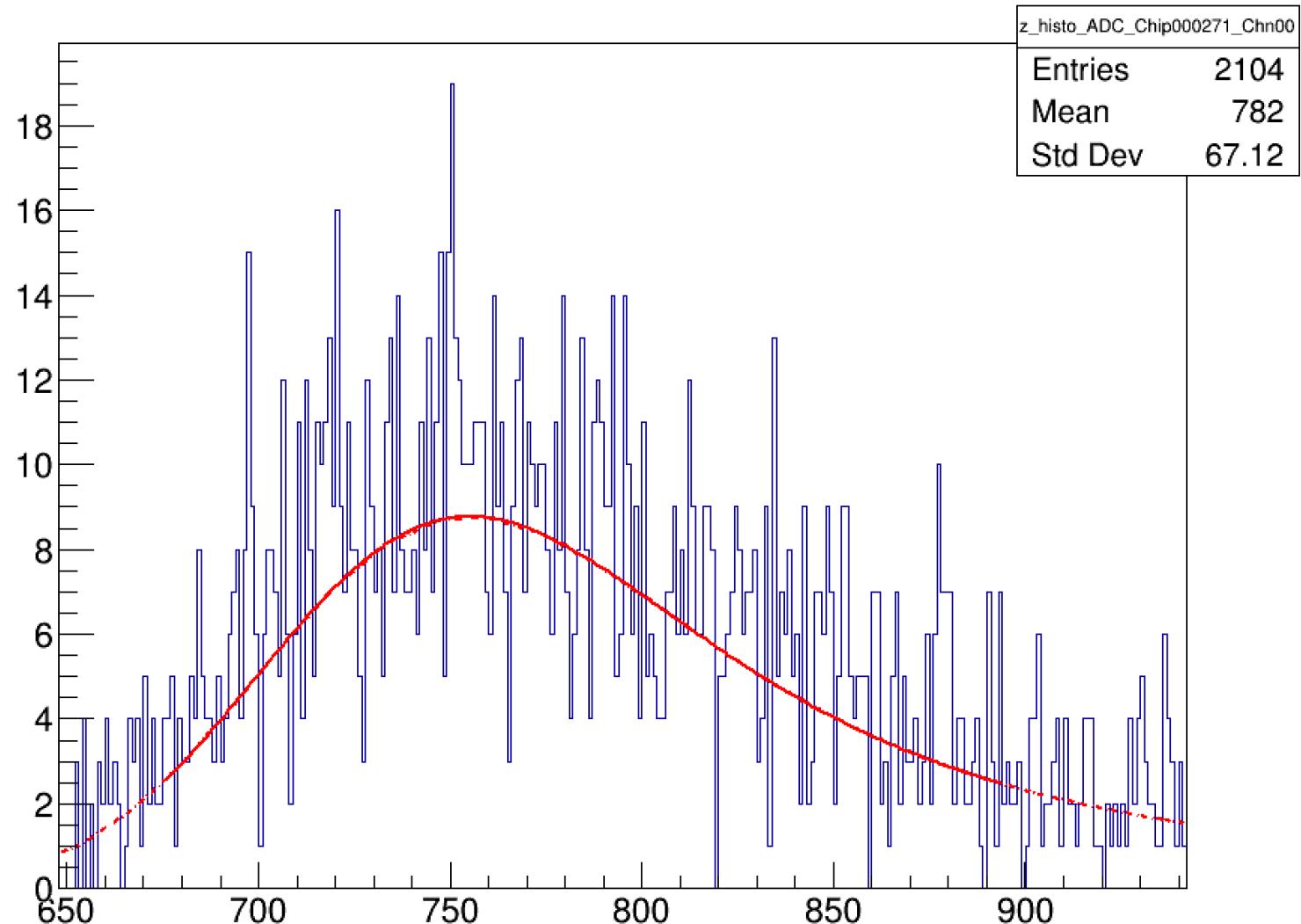
// add functions to histograms to be saved in root file
//hist.GetListOfFunctions()->Add(landau_gauss_iterations[0]);
//hist.GetListOfFunctions()->Add(landau_gauss_iterations[1]);
//hist.GetListOfFunctions()->Add(landau_gauss_iterations[2]);
hist.GetListOfFunctions()->Add(landau_gauss_nobound);
hist.GetListOfFunctions()->Add(landau_gauss_function, "R");

return result;
```

MIP Extraction Details

Landau-Gauss-Fit, illustration different fit iterations

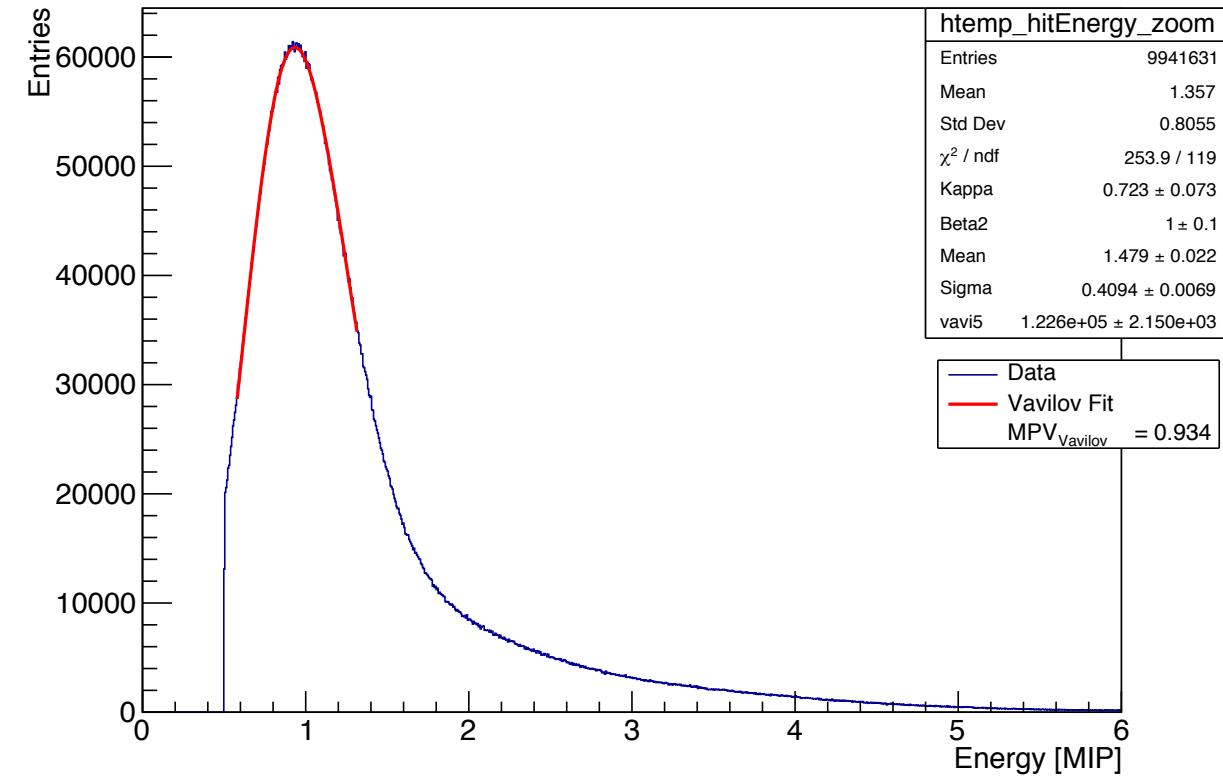
ADC Histo Chip000271 Chn00



Landau-Gauss vs. Vavilov

MIP check - Run 60247 - All channels, Hits

ahc_hitEnergy_zoom_all_channels



ahc_hitEnergy_zoom_all_channels

