GLAST LAT TECHNICAL DOCUMENT	Document # **LAT-TD-05595-01**	Date 12 January 2005
	Author(s) Mark Strickman	
	Subsystem/Office Calorimeter	
Document Title **calibGenCAL v3 Description**		

Gamma Ray Large Area Space Telescope (GLAST) Large Area Telescope (LAT) Calorimeter Calibration Software calibGenCAL v3 Description

Change History Log

Revision	Effective Date	Description of Changes

Contents

1. Purpose

This document describes the LAT calorimeter calibration process in the SAS environment, as applied during the early Integation and Test phase of the mission, using version v3 of the calibGenCAL package.

2. Scope

This document covers calibration procedures to be used on the flight calorimeters prior to launch. In particular, it covers calibrations performed by the calibGenCAL v3 package.

This document includes discussions of both charge injection and muon calibration processes, but does not discuss on-orbit calibrations using galactic cosmic rays. It will discuss the calibration algorithms and products, and how to run the calibration software. It deals primarily with software in the Science Analysis Software (SAS) environment, although topics from on-line analysis will be touched on as required.

3. Definitions

3.1. Acronyms

CAL	Calorimeter
CDE	Crystal Detector Element
CSV	Comma Separated Values
DAC	Digital to Analog Converter
EM	Engineering Model
LAT	Large Area Telescope
GLAST	Gamma-ray Large Area Space Telescope
PDA	Pin Diode Assembly
SAS	Science Analysis Software
TKR	Tracker

3.2. Definitions

mm	millimeter
Simulation	To examine through model analysis or modeling techniques to verify conformance to specified requirements

4. References

LAT-TD-00035-1 “LAT Coordinate System.”

LAT-MD-4187 “Electronic and Muon Calibration Definition”

5. Measurement and Labeling Conventions

5.1. Coordinate system

Dimensions and values shall use the LAT Coordinate System as their reference for describing orientations and directions (if applicable). This is detailed in LAT-TD-00035-1 “LAT Coordinate System.”

5.2. Alternating layers

Calorimeter crystals are arranged in alternating layers. In the top layer, the long axis of each crystal is parallel to the X-axis, in the next, the long axis of each crystal is parallel to the Y-axis, on so on. The former are referred to as X layers and the latter as Y layers. Dimensions referred to as “transverse” without a specification of X or Y dimension are typically the same for crystals in X and Y layers. In X layers, a transverse length would be parallel to the Y-axis, while in Y layers, it would be parallel to the X-axis. Frequently, the variable name will specify X or Y, but the dimension can apply to both.

5.3. Units

The dimensions listed here are nominal and in millimeters and do not reflect tolerances. When a number is underlined in a drawing (e.g. .360.5) it means that its value has been changed by hand for the purposes of this document.

5.4. Diode labeling

Diode labeling has not been absolutely consistent throughout the development process. As a result synonym exist for both diode size and crystal end notation. In particular, the low energy diode corresponding to the LEX8 and LEX1 channels can be referred to as “low energy”, “large” or “big”. These terms are used interchangeably. The high energy diode, corresponding to the HEX8 and HEX1 channels, can be referred to as “high energy” or “small” interchangeably.

The crystal end in the negative coordinate (X or Y) direction is referred to as “negative”, “N”, “minus” or “M” interchangeably. The crystal end in the positive coordinate direction is referred to as “positive”, “plus” or “P” interchangeably.

6. Description of the Calorimeter Calibration Process

The calorimeter calibrations are intended to allow the calculation of various calorimeter responses which, in turn, allow conversion of measurements in “instrument units” such as ADC units into “physical units” such as MeV. This is accomplished by performing measurements that produce known input and measuring the response.

For the calorimeter, the primary quantities that require calibration are:

1. The energy scale (what deposited energy corresponds to a given output in ADC units)
2. The event position scale (where along a crystal was the energy deposited) and
3. The trigger threshold performance (position and efficiency of the trigger threshold)

The calibrations that are required flow from the process by which we reconstruct these quantities from data for a single CDE. In addition, the process of simulating data also places requirements on the calibration data types. These are described in LAT-TD-xxxx and summarized below.

6.1. DAC Energy Scale

Note that many of the processing steps and calibrations described use the "DAC" energy scale. This scale, named after the DAC settings that control the size of a charge injection calibration pulse, is a linear charge scale, similar to the "electrons" scale used previously, although with different units. DAC units are proportional to the number of electrons (i.e. charge) deposited at the input of the front end processor. There are actually two distinct DAC scales, one for the large (LEX) diode and one for the small (HEX) diode. Within each diode, the software that converts to DAC units (e.g. adc2dac) automatically scales for range, resulting in a single DAC scale (e.g. for LEX8 and LEX1). Using the DAC scale removes dependence on the front end electronics calibration.

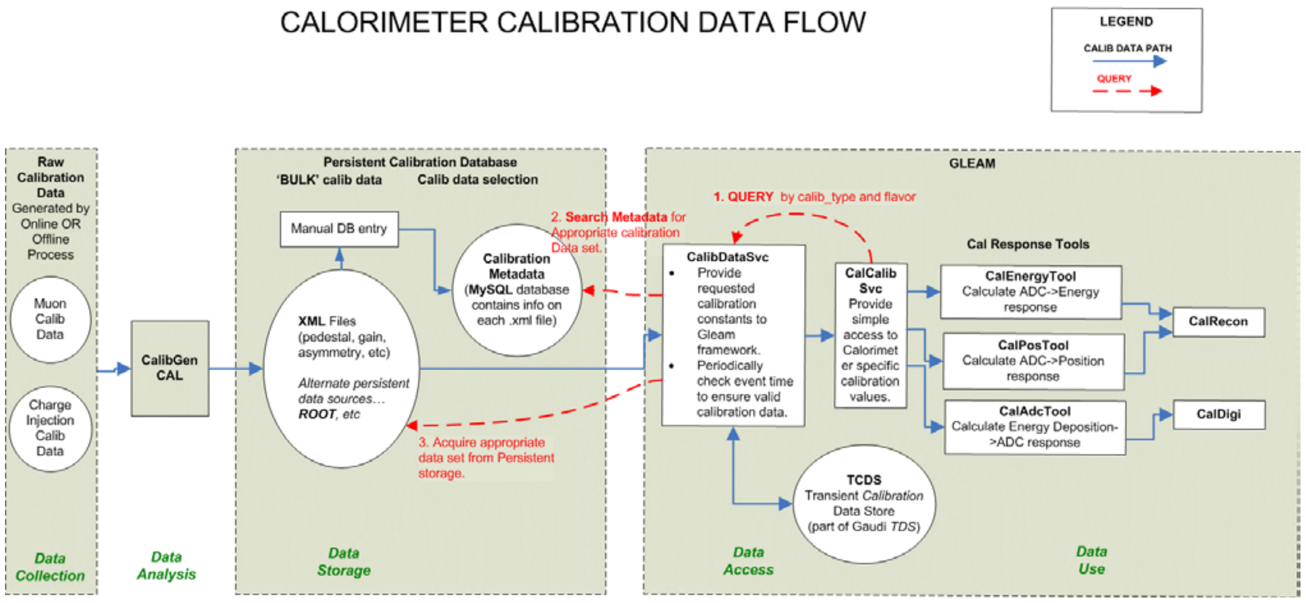

CALORIMETER CALIBRATION DATA FLOW
LEGEND
CALIB DATA PATH
QUERY
Raw Calibration Data Generated by Online OR Offline Process
Muon Calib Data
Charge Injection Calib Data
Data Collection
CalibGen CAL
Data Analysis
Persistent Calibration Database
'BULK' calib data
Calib data selection
Manual DB entry
XML Files (pedestal, gain, asymmetry, etc)
Alternate persistent data sources… ROOT, etc
Calibration Metadata (MySQL database contains info on each .xml file)
2. Search Metadata for Appropriate calibration Data set.
3. Acquire appropriate data set from Persistent storage.
Data Storage
GLEAM
1. QUERY by calib_type and flavor
CalibDataSvc
• Provide requested calibration constants to Gleam framework.
• Periodically check event time to ensure valid calibration data.
CalCalib Svc Provide simple access to Calorimeter specific calibration values.
Cal Response Tools
CalEnergyTool Calculate ADC->Energy response
CalPosTool Calculate ADC->Position response
CalAdcTool Calculate Energy Deposition->ADC response
CalRecon
CalDigi
TCDS Transient Calibration Data Store (part of Gaudi TDS)
Data Access
Data Use

Figure 1 Overall flow of CAL calibration data

6.2. Flow of Calibration Data

Figure 1 summarizes the overall flow of calibration data in the SAS environment. This flow can be broken down into five basic steps:

1. Data Collection: Data are collected via a variety of on-line test procedures described in LAT-MD-04187. These procedures produce a variety of outputs, including html-formatted reports, CSV lists, LDF files and (after an off-line conversion) Root Files.
2. Data Analysis: Data analysis tasks are split between on-line and SAS environments. Analyses performed online normally require information not available in the off-line environment (e.g. charge injection DAC settings in the integral nonlinearity charge injection tests). Most analyses, however, are performed in the SAS environment, using the CalibGenCAL package, described in detail below. Even analyses performed on-line are "passed through" CalibGenCAL in order to convert them to standard output formats. CalibGenCAL outputs results in XML files recognized by the overall SAS calibration system.
3. Data Storage: LAT calibration data persistent storage is currently in the form of XML files, each of which contains a single calibration type for all channels and towers present during the time interval for which it is valid. calibGenCAL writes these XML files directly. Samples for various calibration types are shown in figures below. The data access system is designed to read only a single XML file for each data type. At some future point, the XML files currently used may be replaced with Root files.
4. Data Access: As indicated in Figure 1, access to the calibration data by the CAL response tools is via a hierarchy of interfaces. CalCalibSvc is the API that the tools access directly. Information is passes that uniquely identifies the type, channels and time period of the required calibration data. In addition, multiple versions are supported via the "flavor" parameter.

   CalibDataSvc is the GLEAM service that, given information supplied by CalCalibSvc, searches the MYSQL metadata database to find the appropriate persistent files, reads those files, and moves the data to the Gaudi Transient Calibration Data Store (TCDS), from which they can be accessed by Gaudi applications such as CAL Response Tools.

   Further information on the Gaudi calibration system can be found at:

   http://www-glast.slac.stanford.edu/software/calib
5. Data Use: Section 6.3 below describes the processes by which energy and position are determined for each hit crystal. These processes are implemented by the CAL Response Tools.

6.3. Single CDE Recon Summary

The following steps are executed by the CALEnergyTool and CALPosTool to compute the amount of and position of energy deposition in a single hit CDE (**bold** quantities are results of calibrations; *italicized* quantities are functions):

1. Rescale to DAC scale
   a. Subtract **Pedestals (CAL_Ped)** from ADC values from each end of CDE.
   b. Convert pedestal subtracted ADC values to DAC scale using function DAC = *adc2dac*(ADC). This function makes use of the **Integral Nonlinearity**

(CAL_IntNonlin) calibration results, corrected for **FLE Threshold** crosstalk effects. The function uses a cubic spline functional representation of the calibration datapoints.

2. Position

   a. Calculate the "Asymmetry" LightAsym = *log*($\mathrm{DAC}_{\mathrm{plus}}$/$\mathrm{DAC}_{\mathrm{minus}}$) where $\mathrm{DAC}_{\mathrm{plus/minus}}$ is the signal in DAC units from the plus or minus end of the CDE respectively.

   b. Calculate position of hit along CDE using Position = *LightAsym2posPM*(LightAsym), where *P* and *M* assume values of large or small depending on whether the signals from the plus or minus ends of the CDE are taken from the large or small diodes, respectively. This function makes use of the **Asymmetry (CAL_LightAsym, CAL_muPbigMsmallAsym, CAL_muPsmallMbigAsym)** calibration results using a cubic spline functional representation of the calibration datapoints.

3. Energy

   a. If signals from CDE ends come from different size diodes, convert signals from small diode (in DAC units) to large diode equivalent (also in DAC units) using the ratio of two asymmetries (note LP is signal from large diode, positive end, LP is large diode, negative end, similar for SP, SN for small diode):

      Exp(AsymLL) = LP/LN

      Exp(AsymLS) = LP/SN

      LN/SN = Exp(AsymLS)/Exp(AsymLL)

      Similar for LP/SP

   b. Compute the geometric average of the DAC values from each end of the CDE. Depending on which diodes are available, this could be:
      i. $\mathrm{DAC}_{\mathrm{geom\ mean}}$ = *sqrt*($\mathrm{DAC}_{\mathrm{big\ plus}}$* $\mathrm{DAC}_{\mathrm{big\ minus}}$)
      ii. $\mathrm{DAC}_{\mathrm{geom\ mean}}$ = *sqrt*($\mathrm{DAC}_{\mathrm{small\ plus}}$* $\mathrm{DAC}_{\mathrm{small\ minus}}$)
      iii. $\mathrm{DAC}_{\mathrm{geom\ mean}}$ = *sqrt*($\mathrm{DAC}_{\mathrm{big\ equiv\ plus}}$* $\mathrm{DAC}_{\mathrm{big\ minus}}$)
      iv. $\mathrm{DAC}_{\mathrm{geom\ mean}}$ = *sqrt*($\mathrm{DAC}_{\mathrm{big\ plus}}$* $\mathrm{DAC}_{\mathrm{big\ equiv\ minus}}$)

   c. Calculate the deposited energy using the **MevPerDac (CAL_muBigDiodeMevPerDac, CAL_muSmallDiodeMevPerDac)** for the appropriate diode (Big/Big equiv, Small respectively): Energy (MeV) = **MevPerDac** * $\mathrm{DAC}_{\mathrm{geom\ mean}}$

   Note: **MevPerDac** is the inverse of the gain; it is, in effect, the width of a DAC scale bin in MeV.

4. In the following cases, position and energy will have to be computed using only one end of a CDE:

   - Dead PDA or partial electronics failure
   - Position reconstructed off end of CDE implying direct diode deposit
   - TKR or CAL indication of event too close to CDE end to have good asymmetry due to transverse dependence of light yield near ends

     a. In these cases, the position must be supplied externally. In the first case, TKR or CAL hodoscope data can provide a position, at least to within one CDE width. In the second and third cases, assumptions can be made that the event occurred near the end of the CDE.

     b. Given a position estimate, PlusMinusRatio = *exp*(*pos2LightAsymPM*(position)). *pos2LightAsymPM*(position) is the inverse of LightAsym2posPM. It uses the inverse of the **Asymmetry** calibration, represented by a cubic spline function, to return $\log(DAC_{plus}/DAC_{minus})$ as a function of position. P and M are as described in 2.b above.

     c. If the valid end signal is DAC_{minus}, compute $DAC_{plus\ estimated} = DAC_{minus} * PlusMinusRatio$

     d. If the valid end signal is DAC_{plus}, compute $DAC_{minus\ estimated} = DAC_{plus} / PlusMinusRatio$

     e. Compute energy as in 3.b above using the measured and estimated ends.

7. calibGenCAL release plans

The calorimeter muon and charge injection calibration processes have been gathered together in the calibGenCAL package. Although this package is a part of GlastRelease and EM releases, it is independent of the Gaudi framework and GLEAM. The current version of calibGenCAL, as described in this document, is v3r1p1.

As part of the CAL calibration effort in support of LAT I&T, the calibGenCAL functionality in v3 was substantially modified from previous versions, involving some fundamental changes to the parameterization of calibration quantities. These changes improve performance and allow the calibration system to handle event configurations likely to appear in flight (although not during ground calibration). An example is a hit CDE, read out in single range readout mode, that uses a LEX range on one end and a HEX range on the other.

With the release of v3, new calibration packages shipped with each CAL module, plus those run previously, will be analyzed or reanalyzed using the new version of calibGenCAL. In addition, calibration performed as part of I&T will use the v3, and previously run calibrations will be reanalyzed.

8. calibGenCAL v3 Overview

8.1. Muon Calibrations (muonCalib)

calibGenCAL v3 performs a variety of muon calibrations. These include:

1. Pedestals – Noise pedestal position and width are measured.
2. Light Asymmetry – Light asymmetry vs position represented by a table of 10 points, each position bin representing 1/12 of the crystal length (the end bins are not saved due to large systematic uncertainties). Asymmetry is saved for all possible combinations of big and small diode for each CDE.
3. MevPerDac – MevPerDac is the ratio of energy deposited by a vertical incidence muon to the muon Landau peak most probable value in sqrt(DAC_{minus}*DAC_{plus}) units. It is, in effect, the binwidth of the DAC scale in MeV and, as such, is proportional to the inverse of the gain. Because the geometric mean of the signals from each end (i.e. sqrt(DAC_{minus}*DAC_{plus})) estimates energy deposition very close to independent of hit position along the crystal, MevPerDac is not a function of position.

8.2. Charge Injection Calibrations

In v3r0, the only charge injection calibration that is analyzed is the integral linearity (INTNONLIN) calibration. Future versions will also analyze threshold position calibrations done with charge injection.

The INTNONLIN calibration analysis returns a table of ADC vs DAC values (or the inverse). Hence, it describes the relationship between the nonlinear ADC pulse height scale that comes out of the instrument and the linear DAC scale described in Section 6.

9. calibGenCAL v3 Algorithms and Products

9.1. Class structure

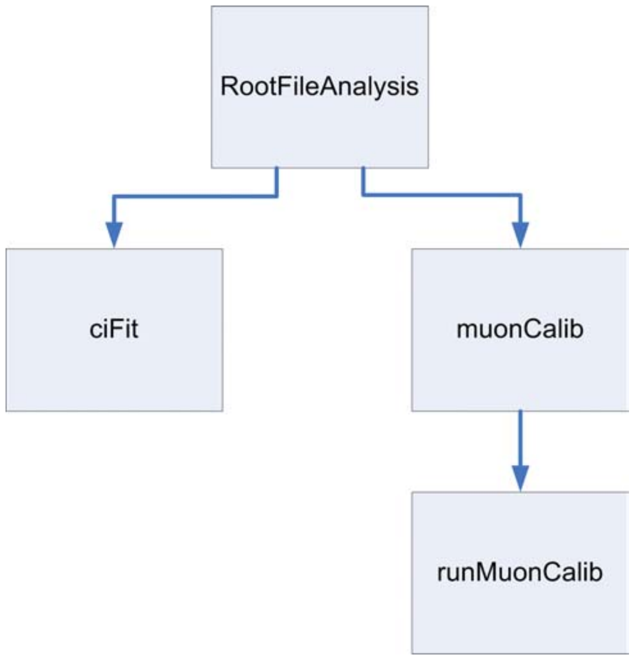


Figure 2: Class dependency tree for RootFileAnalysis

The RootFileAnalysis class dependency tree is shown in xxx.

RootFileAnalysis is a generic class containing:

- Digi/recon/mc root file input
- The event collection – allows event/playback functionality.
- Chain structure which allows single or multiple file runs

The muonCalib class contains:

- muon specific Histograms
- muonCalib data arrays
- muonCalib text/XML file output
- all muon calibration algorithms

The ciFit class contains

- INTNONLIN calibration specific vectors and data arrays
- INTNONLIN text/XML file output
- INTNONLIN calibration algorithms

9.2. Overall Flow

calibGenCAL is divided into two independent applications, MuonCalib and ciFit. Each of these is run independently and uses its own configuration file.

9.3. MuonCalib

The overall flow of MuonCalib is controlled by the runMuonCalib application. Figure 4 and Figure 5 show its overall functional flow. Analysis is controlled by an input file and is divided into three "phases," all described below.

9.3.1. Inputs

Inputs consist of one or more digiRoot files containing the data and an options file (not shown in the figures).

9.3.1.1 digiRoot Files

The input data files are produced by the calu_collect_ext script, a unit test run in the LATTE ground test environment. These are then converted from LDF format files produced by the ground test environment to digiRoot files. This and other unit tests and test suites are described in LAT-MD-4187, "Electronic and Muon Calibration Definition".

9.3.1.2 Options File

The options file (muonCalib_option.xml) contains the following sections: (see Figure 3 for a sample):

- TEST_INFO – Metadata describing test
- PATHS – File paths and names for input and output files
- CONSTANTS – Various thresholds and geometry parameters used in analysis
- GENERAL – Various setup parameters for controlling analysis

If multiple input files are desired, add more file names to first line, separated by spaces. The input files will be concatenated. If a specific output filenames is not specified (which is the case in the default muonCalib_option.xml file), the output file is created in the folder specified by OUTPUT_FOLDER, using a file name created from the input file name.

```
<?xml version="1.0" ?>
<!-- Little test ifile -->

<!DOCTYPE ifile SYSTEM "$(XMLROOT)/xmldata/ifile.dtd" >
```

```
<ifile cvs_Header="$Header: /home/cvs/SLAC/calibGenCAL/src/muonCalib_option.xml,v 1.1 2004/12/23 02:13:29
       fewtrell Exp $"
 cvs_Revision="$Revision: 1.1 $" >

  <section name="TEST_INFO"> Test configuration info.
    <item name="TIMESTAMP"    value="2004-07-31-06:50"> Timestamp for test run.</item>

    <item name="INSTRUMENT"   value="LAT"> Instrument name</item>
    <item name="TOWER_LIST"   value="1"> Listof towers currently installed. </item>
  </section>

  <section name="PATHS"> Intput/Output file paths
    <item name="INPUTFILE_LIST"  value=" D:\GLAST\RELEASE\FM101\041022230030_FM101_Pshp_calu_collect_ext.root">
       Space delimited list of input ROOT digi files</item>

    <item name="INTNONLINFILE_TXT" value="../output/intnonlin.txt"> Input integral non-linearity text file</item>
    <item name="DTD_FILE"          value="../xml/calCalib_v2r2.dtd"> Data description file for XML ouptut </item>

    <item name="OUTPUT_FOLDER" value="..\output\"> Folder for auto-named output files </item>

    <item name="PEDFILE_XML"     value=""> Optional path for output ADC pedestal xml file (default ""
       autogenerates filename from input filename)</item>
    <item name="ASYMFILE_XML"    value=""> Optional path for output asymetry xml file (default "" autogenerates
       filename from input filename)</item>
    <item name="MPDFILE_XML"      value=""> Optional path for output MevPerDac  xml  file (default ""
       autogenerates filename from input filename)</item>

    <item name="PED_HISTFILE"    value=""> Optional path for ROOT histogram filename for pedestal calibration
       phase. (default "" autogenerates filename from input filename)</item>
    <item name="ASYM_HISTFILE"    value=""> Optional path for ROOT histogram filename (default "" autogenerates
       filename from input filename)</item>
    <item name="MPD_HISTFILE"    value=""> Optional path for ROOT histogram filename (default "" autogenerates
       filename from input filename)</item>

    <item name="PEDFILE_TXT"     value=""> Optional path for output ADC pedestal text file (default ""
       autogenerates filename from input filename)</item>
    <item name="ASYMFILELL_TXT"    value=""> Optional path for output LL asymetry text file  (default ""
       autogenerates filename from input filename)</item>
    <item name="ASYMFILELS_TXT"    value=""> Optional path for output LS asymetry text file (default ""
       autogenerates filename from input filename)</item>
    <item name="ASYMFILESL_TXT"    value=""> Optional path for output SL asymetry text file (default ""
       autogenerates filename from input filename)</item>
    <item name="ASYMFILESS_TXT"    value=""> Optional path for output SS asymetry text file (default ""
       autogenerates filename from input filename)</item>
    <item name="LARGEMPD_TXT"      value=""> Optional path for output mev/dac conversion text file (Large diodes)
       (default "" autogenerates filename from input filename)</item>
    <item name="SMALLMPD_TXT"      value=""> Optional path for output mev/dac conversion text file (Small diodes)
       (default "" autogenerates filename from input filename)</item>

    <item name="LOGFILE"     value=""> application log file duplicates stdout (default "" autogenerates filename
       from input filename)</item>
  </section>

  <section name="CONSTANTS"> Constants
    <item name="HIT_THRESH"    value="100"> Threshold val for counting a 'hit xtal'.  (ADCM + ADCP) </item>

    <item name="CELL_HOR_PITCH" value="27.84"> Horizontal pitch (in mm) between 2 neighboring xtals mounted in
       the Cal structure</item>
    <item name="CELL_VERT_PITCH" value="27.84"> Horizontal pitch (in mm) between 2 neighboring xtals mounted in
       the Cal structure</item>

    <item name="MAX_ASYM_LL" value="0.5"> Used to throw out bad asymmetry logratios</item>
    <item name="MAX_ASYM_LS" value="2.5"> Used to throw out bad asymmetry logratios</item>
    <item name="MAX_ASYM_SL" value="-1.0"> Used to throw out bad asymmetry logratios</item>
    <item name="MAX_ASYM_SS" value="1.0"> Used to throw out bad asymmetry logratios</item>
    <item name="MIN_ASYM_LL" value="-0.5"> Used to throw out bad asymmetry logratios</item>
    <item name="MIN_ASYM_LS" value="1.0"> Used to throw out bad asymmetry logratios</item>
    <item name="MIN_ASYM_SL" value="-2.5"> Used to throw out bad asymmetry logratios</item>
    <item name="MIN_ASYM_SS" value="-1.0"> Used to throw out bad asymmetry logratios</item>

  </section>

  <section name="GENERAL"> General program options
    <item name="NEVENTS_ROUGHPED" value="10000"> Number of events to run for rough pedestal calibration </item>
    <item name="NEVENTS_PED"      value="10000"> Number of events to run for final pedestal calibration</item>
    <item name="NEVENTS_ASYM"     value="1000000"> Number of events to run for asymmetry calibration</item>
    <item name="NEVENTS_MPD"      value="1000000"> Number of events to run for MevPerDac calibration</item>

    <item name="READ_IN_PEDS"     value="false"> (boolean) Skip pedestal analysis and read pedestal .txt file
        from previous run</item>
    <item name="READ_IN_ASYM"     value="false"> (boolean) Skip asymetry analysis and read pedestal .txt file
        from previous run</item>
    <item name="SKIP_MPD"         value="false"> (boolean) Skip MevPerDac analysis pass.</item>

    <item name="GENERATE_XML" value="true"> (boolean) generate xml output</item>
    <item name="GENERATE_TXT" value="true"> (boolean) generate text output</item>
    <item name="GENERATE_HISTOGRAM_FILES" value="true"> (boolean) generate histogram output</item>
    <item name="GENERATE_LOGFILE" value="true"> (boolean) generate logfile which duplicates stdout </item>

    <item name="GEN_OPT_ASYM_HISTS" value ="false"> (boolean) generate optional asymetry histograms</item>

  </section>

</ifile>
```

Figure 3: Sample muonCalib_option.xml file

9.3.2. Data Selection

The summarizeHits method performs access, logging and preliminary screening of hits for each event. It is used for both asymmetry and MevPerDac analysis. For each particle event, summarizeHits loops over all CAL crystal hits. For each hit it extracts crystal ID information and ADC values, then pedestal subtracts and performs a series of tests. Note the following definitions used in selection:

- Test direction – Layer set (either X or Y) for which the one and only one hit crystal per layer align in a single column
- Longitudinal direction – The layer set (either X or Y) that is not the test direction i.e. Y if the test direction is X and vice versa

For each hit crystal, summarizeHits performs the following tests:

- Compares the sum of positive and negative ends to a threshold to determine if the hit is valid
- Rejects any event with more than 2 hit crystals in any layer (to prevent events that escape the sides of crystals)
- Defines good[XY]Track to be true if:
    - Number of hit [XY] layers (i.e. test direction layers) = 4
    - Number of hit [XY] columns (i.e. test direction columns) = 1
    - Number of hit [YX] layers (i.e. longitudinal direction layers) > 0

Where goodXTrack defines X to be the test direction and Y the longitudinal direction and goodYTrack defines Y to be the test direction and X the longitudinal direction.

9.3.3. Phase I -- Pedestals

The pedestal analysis flow is shown in Figure 4. It consists of two iterations. In the first (Phase Ia), pedestals histograms are filled using LEX8 channels from all large diodes, whether the crystal involved was hit or not. The resulting distributions are slightly distorted by hit crystal signals, but are still representative of the pedestal distributions. The resulting histograms (one per large diode) are fit with a Gaussian model and the results output to a temporary text file.

The second phase (Phase Ib) uses the rough pedestals from Phase Ia to select crystals with signals within 5 (rough pedestal distribution) sigmas of the pedestal position. The analysis is then repeated as above, but for all channels, resulting in 4 pedestal values for each crystal end (two for each diode). These are then written out to text and XML files.

9.3.4. Phase II -- Asymmetry

The asymmetry analysis flow is shown in Figure 5. Preliminary data selection is performed in the summarizeHits method as described in Section 9.3.2. In addition, hits where the DAC energy scale value ≤ 0 for either face of a hit crystal are rejected. For asymmetry analysis, events that have a "goodXTrack" are used to measure asymmetry in the Y layers and vice versa i.e. X is the "test direction" for the measurement of asymmetry in Y-layer crystals. This nomenclature makes more sense if we consider that the asymmetry in a Y-layer crystal is a function of X position. Particles that traverse a single column in X, for example, all cross Y-layer crystals at more or less the same longitudinal position and with little or no longitudinal component to their track, even though the Y crystals sampled by such an event may or may not be in a single column.

Asymmetry measurements are only performed for the middle 10 crystal-width-bins along each crystal. In other words, any event with a track in the first or last column of the "test direction" is rejected. Asymmetry at the ends of the crystal is susceptible to multiple systematics and is not calibrated. Since crystal end hits can be detected by other means than asymmetry (e.g. direct diode deposition), and, knowing that the hit was near the end of the crystal, assumptions made about hit position, asymmetry calibration near the ends of crystal is not required.

9.3.5. Phase III – MevPerDac

The MevPerDac analysis flow is shown in Figure 6. MevPerDac defines the bin width in MeV of the linear DAC unit energy scale described in Section 6.1. As for asymmetry, the summarizeHits method is used to perform preliminary data selection. However, in this case, events with a "goodXTrack" are used to calibrate X crystals and those with a "goodYTrack" are used to calibrate Y crystals i.e. X is the "test direction" for X crystals and Y is the "test direction" for Y crystals. This selection restricts pathlength corrections to a single dimension. In addition, the "longitudinal direction" layers must have at least two layers hit to obtain a good pathlength correction.

Following selection, a rough track is fit to each event, using the X and Y crystal positions (i.e. using the hodoscopic nature of the calorimeter). Events where this track is more than 52.5 degrees from the vertical are rejected. Likewise, events within a crystal width of the end of any crystals are rejected.

For events not rejected above, using INTNONLIN results from ciFit, ADC signals from each diode are converted to the DAC energy scale (recall that there is one DAC scale for each diode). The DAC values from either end are then used to form an asymmetry, which in turn can be used to determine a hit position along the crystal. These positions are used to refit the particle track and get a better track estimate. Using the better track fit, energy deposits are corrected for path length through each crystal.

For each crystal, histograms are collected of deposited energy measured with the large diode. The energy is estimated by the geometric mean of the DAC values at either end of the crystal:

meanDAClarge = sqrt(DAC large pos * DAC large neg)

Fitting the resulting peak with a Landau distribution yields a most probable value parameter, which in turn gives:

MevPerDac(large diode) = 11.2 MeV/MPV of Landau

Since muon signals are rather small in the small diode, Landau fits to individual energy deposit histograms are vulnerable to systematic and statistical errors. Instead, we form a profile histogram of meanDACsmall vs meanDAClarge using all crystals and selected events. The slope of this histogram is the ratio of MevPerDac(small diode)/MevPerDac(large diode) = large2small, which gives, for each crystal:

MevPerDac(small) = MevPerDac(large) * large2small

The resulting MevPerDac(large/small) for each crystal are written out to text and XML files.

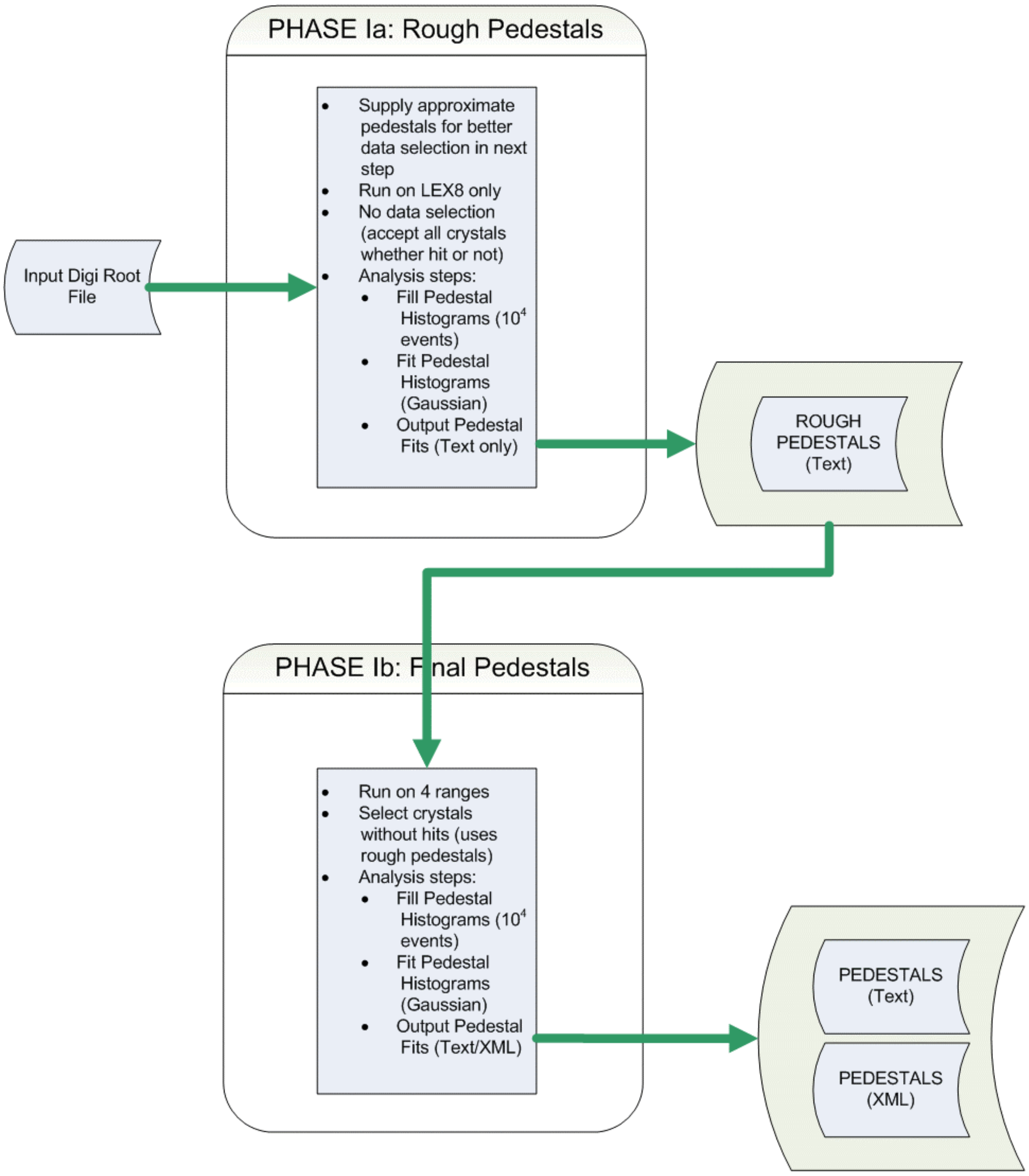

PHASE Ia: Rough Pedestals
• Supply approximate pedestals for better data selection in next step
• Run on LEX8 only
• No data selection (accept all crystals whether hit or not)
• Analysis steps:
• Fill Pedestal Histograms (10^4 events)
• Fit Pedestal Histograms (Gaussian)
• Output Pedestal Fits (Text only)
Input Digi Root File
ROUGH PEDESTALS (Text)
PHASE Ib: Final Pedestals
• Run on 4 ranges
• Select crystals without hits (uses rough pedestals)
• Analysis steps:
• Fill Pedestal Histograms (10^4 events)
• Fit Pedestal Histograms (Gaussian)
• Output Pedestal Fits (Text/XML)
PEDESTALS (Text)
PEDESTALS (XML)

Figure 4: muonCalib Phase I (Pedestals)

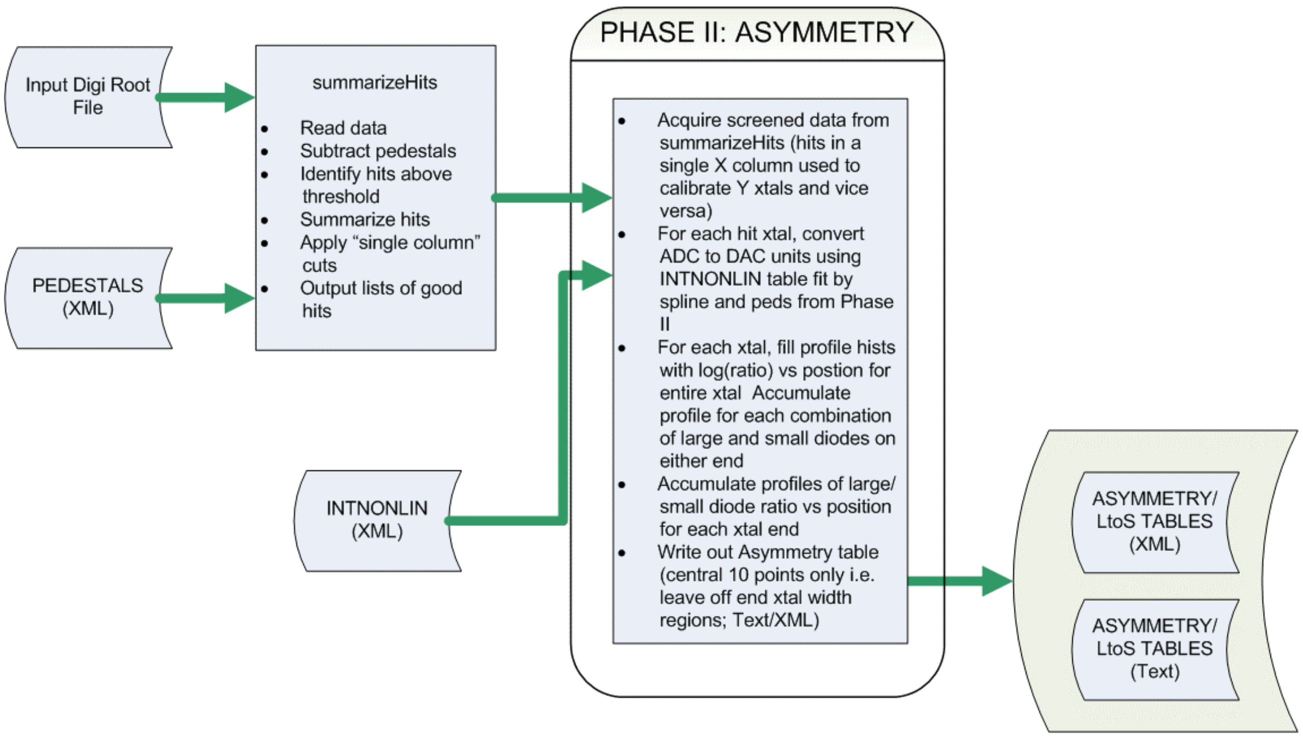

Input Digi Root File
PEDESTALS (XML)
summarizeHits
• Read data
• Subtract pedestals
• Identify hits above threshold
• Summarize hits
• Apply "single column" cuts
• Output lists of good hits
INTNONLIN (XML)
PHASE II: ASYMMETRY
• Acquire screened data from summarizeHits (hits in a single X column used to calibrate Y xtals and vice versa)
• For each hit xtal, convert ADC to DAC units using INTNONLIN table fit by spline and peds from Phase II
• For each xtal, fill profile hists with log(ratio) vs postion for entire xtal Accumulate profile for each combination of large and small diodes on either end
• Accumulate profiles of large/ small diode ratio vs position for each xtal end
• Write out Asymmetry table (central 10 points only i.e. leave off end xtal width regions; Text/XML)
ASYMMETRY/ LtoS TABLES (XML)
ASYMMETRY/ LtoS TABLES (Text)

Figure 5: muonCalib Phase II (Asymmetry)

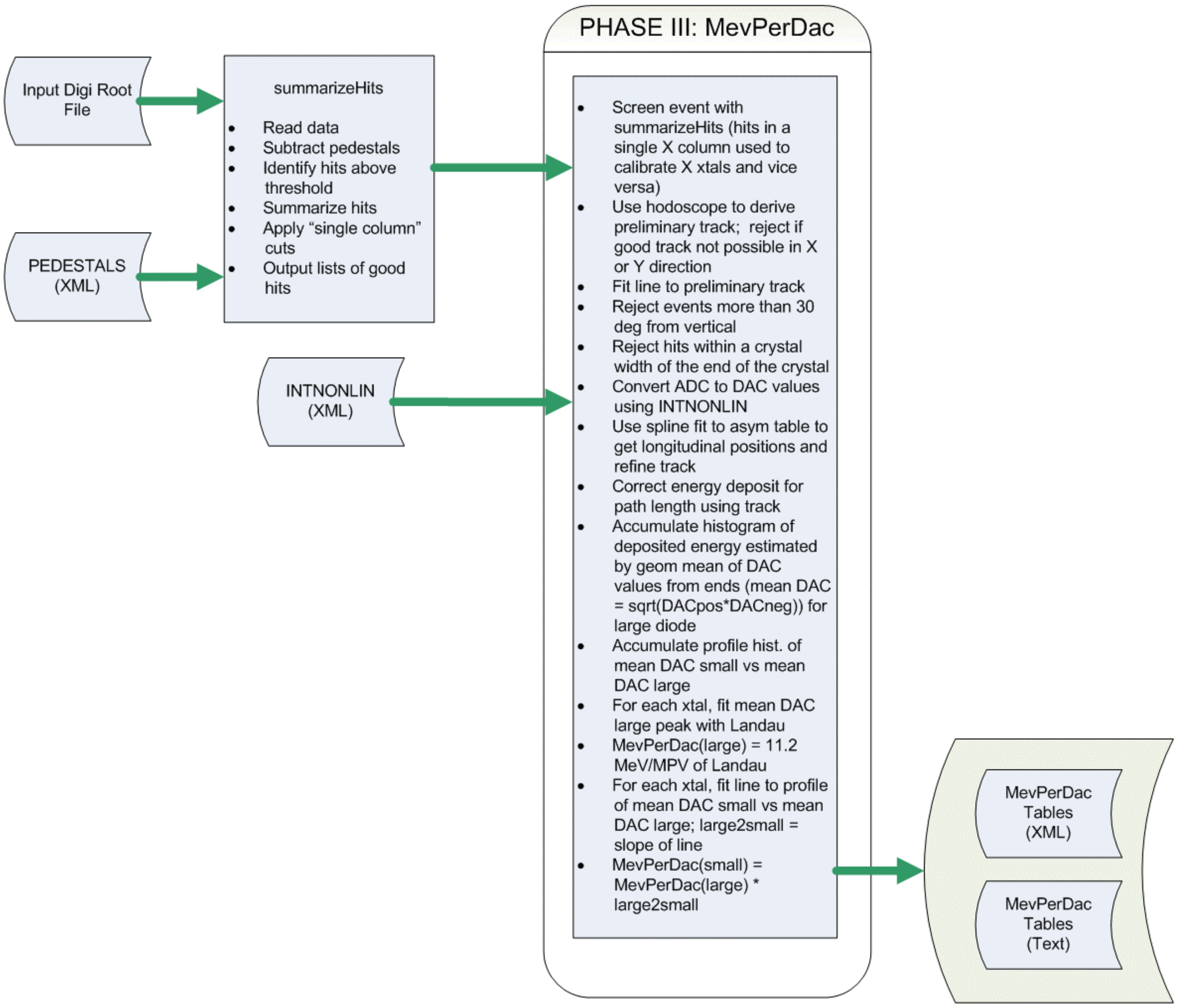

PHASE III: MevPerDac
Input Digi Root File
PEDESTALS (XML)
INTNONLIN (XML)
summarizeHits
• Read data
• Subtract pedestals
• Identify hits above threshold
• Summarize hits
• Apply "single column" cuts
• Output lists of good hits
• Screen event with summarizeHits (hits in a single X column used to calibrate X xtals and vice versa)
• Use hodoscope to derive preliminary track; reject if good track not possible in X or Y direction
• Fit line to preliminary track
• Reject events more than 30 deg from vertical
• Reject hits within a crystal width of the end of the crystal
• Convert ADC to DAC values using INTNONLIN
• Use spline fit to asym table to get longitudinal positions and refine track
• Correct energy deposit for path length using track
• Accumulate histogram of deposited energy estimated by geom mean of DAC values from ends (mean DAC = sqrt(DACpos*DACneg)) for large diode
• Accumulate profile hist. of mean DAC small vs mean DAC large
• For each xtal, fit mean DAC large peak with Landau
• MevPerDac(large) = 11.2 MeV/MPV of Landau
• For each xtal, fit line to profile of mean DAC small vs mean DAC large; large2small = slope of line
• MevPerDac(small) = MevPerDac(large) * large2small
MevPerDac Tables (XML)
MevPerDac Tables (Text)

Figure 6: muonCalib Phase III (MevPerDac)

9.3.6. Products

Note that all values are accompanied by an error estimation.

9.3.6.1 Pedestals

Pedestal results, including peak positions and widths, are stored in both a text file and an XML file. A sample pedestal XML file fragment for the first two crystals is shown in Figure 7. The pedestal distribution peak and width (sigma) are in adc units. They are supplied for each range of each crystal end.

```
<calCalib>
 <generic instrument="LAT" timestamp="2004-07-31-06:50" calibType="CAL_Ped" fmtVersion="v2r2">
 </generic>
 <dimension nRow="1" nCol="1" nLayer="8" nXtal="12" nFace="2" nRange="4"/>
 <tower iRow="0" iCol="0">
  <layer iLayer="0">
   <xtal iXtal="0">
    <face end="POS">
     <calPed avg="634.621" sig="5.80922" range="LEX8"/>
     <calPed avg="203.72" sig="0.859742" range="LEX1"/>
     <calPed avg="618.369" sig="4.38384" range="HEX8"/>
     <calPed avg="238.365" sig="0.724878" range="HEX1"/>
    </face>
    <face end="NEG">
     <calPed avg="419.113" sig="5.73681" range="LEX8"/>
     <calPed avg="225.506" sig="0.867359" range="LEX1"/>
     <calPed avg="514.761" sig="4.36158" range="HEX8"/>
     <calPed avg="257.83" sig="1.07284" range="HEX1"/>
    </face>
   </xtal>
   <xtal iXtal="1">
    <face end="POS">
     <calPed avg="551.955" sig="5.8208" range="LEX8"/>
     <calPed avg="219.701" sig="0.858616" range="LEX1"/>
     <calPed avg="610.17" sig="4.38969" range="HEX8"/>
     <calPed avg="217.475" sig="0.686978" range="HEX1"/>
    </face>
    <face end="NEG">
     <calPed avg="576.909" sig="5.70945" range="LEX8"/>
     <calPed avg="223.803" sig="0.842729" range="LEX1"/>
     <calPed avg="645.69" sig="4.36241" range="HEX8"/>
     <calPed avg="224.608" sig="0.485693" range="HEX1"/>
    </face>
   </xtal>
```

Figure 7: Sample from Pedestals XML file for first two crystals

9.3.6.2 Asymmetry

The asymmetry product is just the histogram of asymmetry (log(P/M)) vs position for LE and HE diodes for each crystal. Each table contains ten numbers representing ten positions defined by the centers of the orthogonal crystals, excluding the end crystals. Currently, the read routine for the table linearly extrapolates from the last two positions to get values at the ends. Figure 8 shows a fragment of an asymmetry XML file for the first two crystals.

```
<calCalib>
 <generic instrument="LAT" timestamp="2004-07-31-06:50" calibType="CAL_Asym" fmtVersion="v2r2">
 </generic>
 <dimension nRow="1" nCol="1" nLayer="8" nXtal="12" nFace="1" nRange="1"/>
 <tower iRow="0" iCol="0">
  <layer iLayer="0">
   <xtal iXtal="0">
    <face end="NA">
     <asym
           bigVals=" -0.328393 -0.246047 -0.161543 -0.0881425 -0.0142309 0.0614083 0.129262 0.20902
0.28039 0.357443"
           bigSigs=" 0.00173584 0.00162079 0.00177822 0.0016393 0.00169392 0.00149337 0.00177718
0.00177895 0.00166996 0.0017155"
           NsmallPbigVals=" 1.3105 1.39387 1.48458 1.56068 1.6272 1.70299 1.78077 1.85838 1.932
2.00787"
           NsmallPbigSigs=" 0.00359729 0.00420086 0.00432564 0.00426201 0.00463923 0.00417912
0.00454567 0.00470233 0.00446112 0.00427022"
           PsmallNbigVals=" -1.95181 -1.87059 -1.78453 -1.7177 -1.64051 -1.56288 -1.49604 -1.4149 -
1.34698 -1.26676"
           PsmallNbigSigs=" 0.00431784 0.00418917 0.00446958 0.0041342 0.00475571 0.00464651
0.00473123 0.00393864 0.00421022 0.00405998"
           smallVals=" -0.312923 -0.230668 -0.13841 -0.0688759 0.000916836 0.0787046 0.155466
0.234469 0.304625 0.383674"
           smallSigs=" 0.00533777 0.0055795 0.00602977 0.00517078 0.00533871 0.00556939 0.00581306
0.00562109 0.00544018 0.00539649" />
     </asym>
    </face>
   </xtal>
   <xtal iXtal="1">
    <face end="NA">
     <asym
           bigVals=" -0.329877 -0.286631 -0.224763 -0.156905 -0.0719117 0.0310668 0.126919 0.204485
0.272213 0.329418"
           bigSigs=" 0.00132653 0.00122162 0.00145306 0.00146868 0.00159212 0.00169487 0.0015836
0.00169492 0.00138676 0.00126705"
           NsmallPbigVals=" 1.34245 1.38316 1.44804 1.51152 1.59483 1.69884 1.79495 1.87838 1.94021
1.99137"
           NsmallPbigSigs=" 0.00302683 0.00282602 0.00345221 0.00315471 0.00317027 0.00333854
0.00331445 0.0037883 0.00340748 0.0033179"
           PsmallNbigVals=" -1.9917 -1.95235 -1.89105 -1.8288 -1.74271 -1.64079 -1.54858 -1.47347 -
1.40521 -1.35053"
           PsmallNbigSigs=" 0.00358986 0.00318854 0.00341839 0.003396 0.00360037 0.00342791
0.00321552 0.0033818 0.00286631 0.00264047"
           smallVals=" -0.319381 -0.282561 -0.218244 -0.160382 -0.0759707 0.0269884 0.11945 0.200424
0.262794 0.311418"
           smallSigs=" 0.00457156 0.00409381 0.00449331 0.00428005 0.00454184 0.00449121 0.00422685
0.00459749 0.00430711 0.0040312" />
     </asym>
    </face>
   </xtal>
```

Figure 8: Sample from an asymmetry XML file showing values for the first two crystals

9.3.6.3 MevPerDac

The MevPerDac product is a table of MevPerDac values for each diode size for each crystal (total of two per crystal). It is stored in XML and text files. A sample for two crystals is shown in Figure 9.

```
<calCalib>
 <generic instrument="LAT" timestamp="2004-07-31-06:50" calibType="CAL_MevPerDac" fmtVersion="v2r2">
 </generic>
 <dimension nRow="1" nCol="1" nLayer="8" nXtal="12" nFace="1" nRange="1"/>
 <tower iRow="0" iCol="0">
```

```
  <layer iLayer="0">
   <xtal iXtal="0">
    <face end="NA">
     <mevPerDac bigVal="0.395167" bigSig="0.0279662" smallVal="0.0783085" smallSig="0.00554195">
    </face>
   </xtal>
   <xtal iXtal="1">
    <face end="NA">
     <mevPerDac bigVal="0.39043" bigSig="0.0270145" smallVal="0.0726614" smallSig="0.00502757">
    </face>
   </xtal>
```

Figure 9 Sample from an MevPerDac XML file showing values for the first two crystals

9.4. Charge Injection INTNONLIN Calibration (ciFit)

The ciFit routine analyses data from charge injection tests designed to measure the integral linearity of the pulse height scale. Deviations from linearity, or INTNONLIN, are used as part of the energy scale calibration process. The INTNONLIN results are used in both the asymmetry and MevPerDac calibrations described above, and are an integral part of CALrecon as well.

9.4.1. Inputs

9.4.1.1 digiRoot Files

ciFit uses the results of the calibGen suite of tests run in the LATTE environment. See LAT-MD-04187, "CAL Electronic and Muon Calibration Suite Definition" for details. Output of these tests are LDF (or, earlier, CED) data files, which are then converted to Root using either ced2root or follow-on LDF converters. Some information necessary for analysis of these data (in particular, the DAC values controlling the amplitudes of the charge injection pulses) are not currently stored in the LDF and/or Root files (although this will change in the near future). Hence, ciFit currently passes those values manually from the options file described below.

A part of calibGen runs a series of six charge injection elements using the CALF_COLLECT_CI_SINGLEX16 test script with various settings, which produce output files with names like:

YYMMDDHHMMSS_FMNNN_Pshp_calu_collect_ci_singlex16.ced

Where YYMMDDHHMMSS is the file time tag, NNN is the CAL module number, and .ced indicates a CED file, soon to be replaced by .ldf (LDF) files. Note that the file name does not identify which element the file contains. This can be determining by time-ordering the files from a given calibGen run.

For ciFit, we use element 1 of the CALF_COLLECT_CI_SINGLEX16 runs for the low energy linearity calibration and element 6 for the high energy linearity calibration. Both runs are set up with leGain = 5, heGain = 0, tackDelay = 104, fleDAC = 127 and fheDAC = 127. Element 1 is set up for leOnly, while element 6 is set up for heOnly and has Calib Gain Off.

9.4.1.2 Options File

The options file (ciFit_option.xml) contains the following sections: (see Figure 10 for a sample):

- TEST_INFO – Metadata describing test. In particular, this section includes the charge injection DAC values used in the test along with the number of pulses per DAC value
- PATHS – File paths and names for input and output files
- SPLINE_CFG – Parameters that control data smoothing and output

The output file name is created in the folder specified by OUTPUT_FOLDER, using a file name created from the input file name. INFILE_TYPE specifies ROOT or CSV file input (not currently used).

```
?xml version="1.0" ?>
<!-- Little test ifile -->

<!DOCTYPE ifile SYSTEM "$(XMLROOT)/xmldata/ifile.dtd" >

<ifile cvs_Header="$Header: /nfs/slac/g/glast/ground/cvs/calibGenCAL/src/ciFit_option.xml,v 1.4 2005/01/06
05:32:16 fewtrell Exp $"
       cvs_Revision="$Revision: 1.4 $" >

  <section name="TEST_INFO"> Test configuration info.
    <item name="TIMESTAMP"      value="2004-11-12-12:27"> Timestamp for test run.</item>
    <item name="STARTTIME"      value="2004-11-12-12:27"> Start of test run</item>
    <item name="STOPTIME"       value="2004-11-12-12:49"> Stop of test run</item>

    <item name="INSTRUMENT"     value="LAT"> Instrument name</item>
    <item name="TOWER_LIST"     value="1"> List of towers currently installed. </item>

    <item name="TRIGGER_MODE"     value="FORCED_TRIGGER"> Trigger mode.</item>
    <item name="INST_MODE"        value="ONBOARD_CAL"> Instrument mode</item>
    <item name="TEST_SOURCE"      value="ONBOARD_CAL_DAC"> Test source</item>

    <item name="DAC_SETTINGS"
value="0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,80,96,11
2,128,144,160,176,192,208,224,240,256,272,288,304,320,336,352,368,384,400,416,432,448,464,480,496,512,543,575
,607,639,671,703,735,767,799,831,863,895,927,959,991,1023,1055,1087,1119,1151,1183,1215,1247,1279,1311,1343,1
375,1407,1439,1471,1503,1535,1567,1599,1631,1663,1695,1727,1759,1791,1823,1855,1887,1919,1951,1983,2015,2047,
2079,2111,2143,2175,2207,2239,2271,2303,2335,2367,2399,2431,2463,2495,2527,2559,2591,2623,2655,2687,2719,2751
,2783,2815,2847,2879,2911,2943,2975,3007,3039,3071,3103,3135,3167,3199,3231,3263,3295,3327,3359,3391,3423,345
5,3487,3519,3551,3583,3615,3647,3679,3711,3743,3775,3807,3839,3871,3903,3935,3967,3999,4031,4063,4095}"> list
of dac settings used in test.</item>
    <item name="N_PULSES_PER_DAC" value="50"> Number of pulses per DAC setting</item>

  </section>

  <section name="PATHS"> Intput/Output file paths
    NOTE: All environment variables will be expanded with facilities::Util::expandEnvVar()
    <item name="OUTPUT_FOLDER" value="X:\FM\Module\FM102\FM102_calibGenCAL_analysis\"> Folder for auto-named
output files </item>

    <item name="DTD_FILE"     value="$(CALIBUTILROOT)/xml/calCalib_v2r1.dtd"> description file for output
.xml file </item>
    <item name="XMLPATH"     value=""> Optional path for output .xml file (default "" autogenerates filename
from input filename)</item>
    <item name="TXTPATH"     value=""> Optoinal path for output .txt file (default "" autogenerates filename
from input filename)</item>

    <item name="INFILE_TYPE" value="ROOT">Input file type (ROOT,CSV)</item>

    <item name="ROOTFILE_LE1"
value="X:\FM\Module\FM102\FM102_calibGenCAL_analysis\041112122650_FM102_PSHP_CALU_COLLECT_CI_SINGLEX16.root">
input DIGIROOT file LE #1 </item>
```

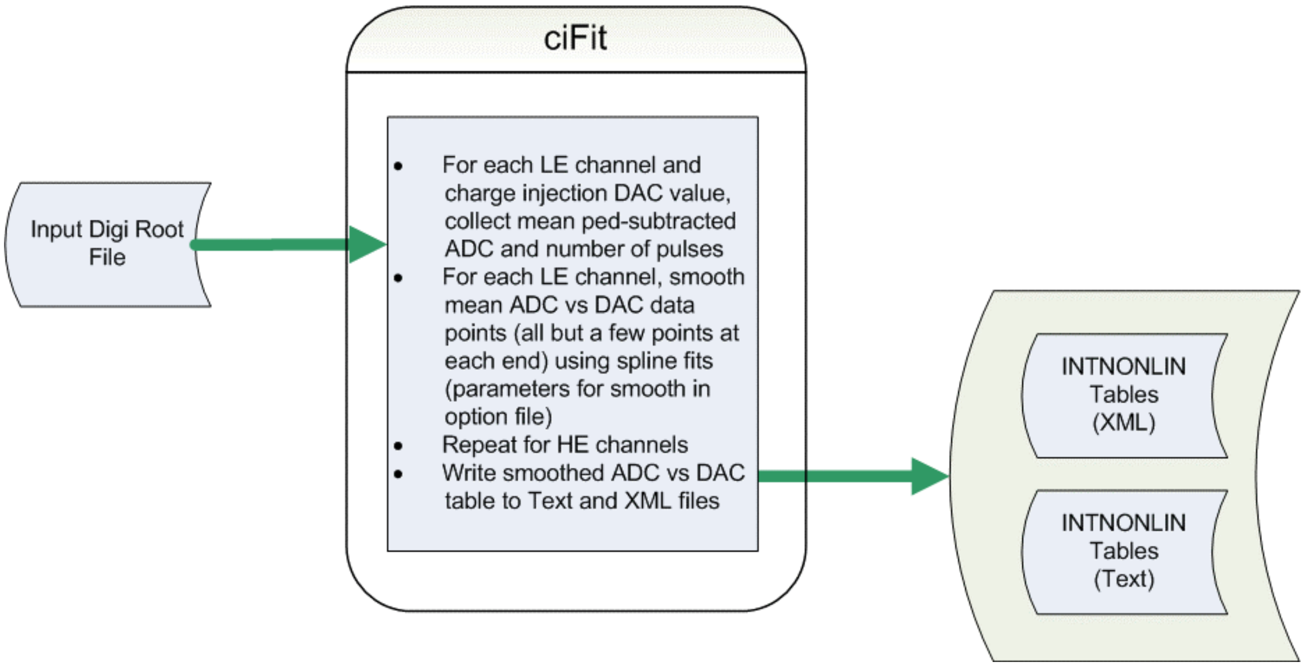
```
    <item name="ROOTFILE_HE1"
value="X:\FM\Module\FM102\FM102_calibGenCAL_analysis\041112124530_FM102_PSHP_CALU_COLLECT_CI_SINGLEX16.root">
input DIGIROOT file HE #1 </item>

    <item name="ROOTFILE_LE2"   value=""> input DIGIROOT file LE #2 (currently unused)</item>
    <item name="ROOTFILE_HE2"   value=""> input DIGIROOT file HE #2 (currently unused)</item>
    <item name="ROOTFILE_LE3"   value=""> input DIGIROOT file LE #3 (currently unused)</item>
    <item name="ROOTFILE_HE3"   value=""> input DIGIROOT file HE #3 (currently unused)</item>
  </section>

  <section name="SPLINE_CFG">
    <item name="GROUP_WIDTH"   value="3,4,3,4"> # of points to use for each smoothing step, 1 entry per RANGE
</item>
    <item name="SKIP_LOW"      value="3,1,3,1"> # of points at beggining of array to copy verbatim and skip
smoothing step </item>
    <item name="SKIP_HIGH"     value="6,10,6,10"> # of points at end of array to copy verbatim and skip
smoothing step </item>
    <item name="N_PTS_MIN"     value="23,45,23,48"> minimum # of spline points to output per range </item>
  </section>
</ifile>
```

Figure 10 Sample ciFit_option.xml file

9.4.2. Algorithm



- For each LE channel and charge injection DAC value, collect mean ped-subtracted ADC and number of pulses
- For each LE channel, smooth mean ADC vs DAC data points (all but a few points at each end) using spline fits (parameters for smooth in option file)
- Repeat for HE channels
- Write smoothed ADC vs DAC table to Text and XML files

Figure 11 ciFit flow

ciFit is a relatively simple application that computes the mean ADC value for each channel in a CAL module for each charge injection DAC value, then smooths the resulting table of ADC vs DAC using spline functions and outputs the smoothed table of values. The smoothing process removes small, nonsignificant fluctuations from the data.

ciFit currently does not make corrections due to the difference in cross-talk-induced nonlinearities between charge injection and muon calibrations. This is because the bulk of data currently collected are collected with FLE set high (i.e. externally triggered data), for which this phenomenon is well clear of the muon peak in the spectrum. This correction will be included in a future version of the code.

9.4.3. Products

ciFit produces both text and XML calibration files containing both the DAC values used in the test and the smoothed mean ADC values corresponding to those DAC values. These tables of values can then be used by functions such as adc2dac to convert ADC results into the linearized “DAC” pulse-height scale. A sample (DAC values and one crystal) of the XML output is shown in Figure 12.

```
<calCalib>
  <generic instrument="LAT" timestamp="2004-11-18-03:17"
           calibType="CAL_IntNonlin" fmtVersion="v2r1">

    <inputSample startTime="2004-11-18-03:17" stopTime="2004-11-18-03:38"
triggers="FORCED_TRIGGER" mode="ONBOARD_CAL" source="ONBOARD_CAL_DAC" >

Times are start and stop time of calibration run.
Other attributes are just made up for code testing.
    </inputSample>
  </generic>

<!-- EM instrument: 8 layers, 12 columns -->

<!-- number of collections of dac settings should normally be
     0 (the default), if dacs aren't used to acquire data, or
     equal to nRange -->
 <dimension nRow="1" nCol="1" nLayer="8" nXtal="12" nFace="2" nRange="4"
           nDacCol="4" />

 <dac range="LEX8"
     values="0 2 4 10 16 22 28 34 40 46 52 58 64 112 160 208 256 272 288 304 320 336 "
     error="0.1" />
 <dac range="LEX1"
     values="0 8 16 24 32 40 48 56 64 128 192 256 320 384 448 512 639 767 895 1023 1151 1279 1407 1535 1663 1791
1919 2047 2175 2303 2431 2559 2687 2815 "
     error="0.1" />
 <dac range="HEX8"
     values="0 2 4 10 16 22 28 34 40 46 52 58 64 112 160 208 256 272 288 304 320 "
     error="0.1" />
 <dac range="HEX1"
     values="0 8 16 24 32 40 48 56 64 128 192 256 320 384 448 512 639 767 895 1023 1151 1279 1407 1535 1663 1791
1919 2047 2175 2303 2431 2559 2687 2815 "
     error="0.1" />

 <tower iRow="0" iCol="0">
  <layer iLayer="0">
   <xtal iXtal="0">
    <face end="POS">
     <intNonlin range="LEX8"
```

```
            values="0.00 21.74 42.04 106.10 172.60 238.24 304.12 369.22 435.96 503.02 568.77 636.18 702.57
1237.92 1780.54 2325.87 2875.32 3059.80 3245.14 3429.98 3614.52 3741.72 "

            error="0.10" />

     <intNonlin range="LEX1"

            values="0.00 9.31 19.10 28.81 38.39 48.25 58.00 67.81 77.58 156.74 237.01 318.18 400.21 482.73
565.93 650.28 817.95 989.94 1164.44 1340.84 1518.83 1697.73 1877.19 2056.83 2236.73 2416.74 2596.67 2776.59
2956.46 3136.28 3316.14 3495.74 3675.44 3854.70 "

            error="0.10" />

     <intNonlin range="HEX8"

            values="-0.00 22.96 44.76 112.60 181.32 249.31 318.51 387.63 456.12 525.38 594.48 663.48 732.07
1289.55 1850.07 2413.44 2982.73 3173.68 3362.62 3554.10 3608.46 "

            error="0.10" />

     <intNonlin range="HEX1"

            values="-0.00 9.94 20.08 30.23 40.46 50.68 60.79 71.10 81.36 164.12 247.47 331.90 416.62 501.71
586.99 672.64 842.71 1016.38 1191.09 1366.76 1542.72 1719.02 1895.43 2072.03 2248.54 2425.07 2601.60 2778.02
2954.50 3130.90 3307.36 3485.03 3664.79 3841.33 "

            error="0.10" />

    </face>

    <face end="NEG">

     <intNonlin range="LEX8"

            values="0.00 21.66 43.50 112.48 180.54 248.13 317.15 386.24 455.53 524.79 592.47 662.40 731.32
1287.34 1849.43 2412.87 2994.98 3175.30 3365.20 3436.20 "

            error="0.10" />

     <intNonlin range="LEX1"

            values="0.00 9.98 20.09 30.06 40.30 50.44 60.66 70.79 80.99 163.68 247.21 331.63 417.05 503.01
589.54 676.63 850.39 1028.44 1207.85 1387.98 1568.28 1748.95 1929.72 2110.67 2291.63 2472.53 2653.48 2834.31
3015.49 3196.22 3377.17 3558.14 3739.26 "

            error="0.10" />

     <intNonlin range="HEX8"

            values="-0.00 22.46 47.06 115.62 185.48 256.00 325.33 396.07 467.34 537.09 607.46 677.84 748.75
1317.98 1890.27 2467.13 3047.24 3241.58 3435.74 3630.98 3701.18 "

            error="0.10" />

     <intNonlin range="HEX1"

            values="0.00 10.40 20.83 31.16 41.63 52.20 62.58 73.04 83.64 168.47 254.02 340.42 427.16 514.30
601.54 689.06 862.56 1039.36 1216.61 1394.12 1571.77 1749.60 1927.52 2105.15 2283.05 2460.83 2638.66 2816.45
2994.31 3171.85 3351.89 3532.63 3710.92 "

            error="0.10" />

    </face>

   </xtal>
```

Figure 12 Sample INTNONLIN XML file fragment

10. calibGenCAL Run Procedure

The following is a sample procedure for running calibGenCAL on NRL-environment preship test data in a Windows environment. It refers to CED files and the ced2root converter, which will change to LDF files and an appropriate LDF to root converter in the near future.

Note that the applications run in this procedure can be run from MRvcmt, or from the command line prompt (after running the appropriate setup.bat file for that application). Whereas data files here are referred to as ced files, recent and future tests will generate ldf files.

1. Data are on X:\FM\Module\FMNNN where X: is \\xeus.nrl.navy.mil\glastcal and NNN is the CAL module number
2. Create X:\FM\Module\FMNNN\FMNNN_calibGenCAL_analysis folder to hold root files and analysis results
3. Identify CED files containing charge injection calibration data. These are have the format:

```
…\FMNNN\FMNNN_Pshp_calibGen\Event
Collections\YYMMDDHHMMSS_FMNNN_Pshp_calu_collect_ci_singlex16.ced
```

There are six files with this format, representing Elements 1-6 of the test. For the low energy channels, we want to use Element 1, which is the first file measured. For the high energy channels, we want to use Element 6, the sixth file measured. Both tests have the following settings:

- LE gain = 5
- HE gain = 0
- TAC delay = 104
- FLE DAC = 127
- For low energy, LEonly; for high energy, HEonly
- High energy has Calib Gain Off

4. Run ced2root (or appropriate converter) on the two charge injection ced files. Pass the input and output file paths/names in as external parameters (e.g. in the external parameters box in MRvcmt or as command line parameters if running in a command line environment). The first parameter (input file) will look like:

   ```
   "…\FMNNN\FMNNN_Pshp_calibGen\Event
   Collections\YYMMDDHHMMSS_FMNNN_Pshp_calu_collect_ci_singlex16.ced"
   ```

   Where … is replaced by the preceding portion of the path, XXX is the module number and YYMMDDHHMMSS is the time tag portion of the filename. The parameter is enclosed in double quotes to deal with spaces in the path name.

   The second parameter (output file), separated from the first by a space, will look like:

   ```
   "…\FMNNN\FMNNN_calibGenCAL_analysis\
   YYMMDDHHMMSS_FMnnn_Pshp_calu_collect_ci_singlex16.root"
   ```

5. If the data file is ldf rather than ced, run the LatIntegration application (part of the LatIntegration package) from EM v3r0407p1em0 or later (these versions are set to ignore the UDF record in the ldf file – this will change eventually). Input is an options file that specifies input and output files and input file type (LDF in this case). Following is a sample options file:

   ```
   // One might want to include Gleam/basicoptions and update accordingly..but
   // this works for now
   ```

```
ApplicationMgr.DLLs+= { "GaudiAlg", "GaudiAud"};
ApplicationMgr.ExtSvc += {"ChronoStatSvc"};
AuditorSvc.Auditors = {"ChronoAuditor"};
ApplicationMgr.DLLs += {"LdfConverter"};

EventSelector.Instrument = "EM";

// ----------------------------
// setup basic event loop stuff
//
ApplicationMgr.ExtSvc += {
    "LdfEventSelector/EventSelector" ,
    "LdfCnvSvc/EventCnvSvc"
    };

EventPersistencySvc.CnvServices = {"EventCnvSvc"};


// ----------------------------
//  a structure for the topalg, using sequencer steps
ApplicationMgr.TopAlg = {
      "Sequencer/Top" };


Generator.Members = {};
Digitization.Members = {};


//the top sequence loop
Top.Members={
      "Sequencer/Output"
};


// ----------------------------
//  Geometry definition
//
ApplicationMgr.DLLs += {"GlastSvc"};
ApplicationMgr.ExtSvc += { "GlastDetSvc"};
GlastDetSvc.xmlfile="$(XMLGEODBSROOT)/xml/em/emSegVols.xml";
GlastDetSvc.visitorMode="recon";

ApplicationMgr.DLLs +={
    "RootIo"
};
ApplicationMgr.ExtSvc += { "RootIoSvc" };

digiRootWriterAlg.OutputLevel=3;

Output.Members = {
    "digiRootWriterAlg"
};



// Output DIGI file name
digiRootWriterAlg.digiRootFile =
"X:\FM\Module\FM107\FM107_calibGenCAL_analysis\050113230259_FM107_Pshp_calu_c
ollect_ci_singlex16.root";

// Input EBF File
EventSelector.StorageType = "LDFFILE";
EventSelector.InputList = {"X:\FM\Module\FM107\FM107_Pshp_calibGen\Event
Collections\050113230259_FM107_Pshp_calu_collect_ci_singlex16.ldf"};

EventSelector.OutputLevel = 4;

// Set output level threshold (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL )
MessageSvc.OutputLevel = 3;
ApplicationMgr.EvtMax  = 1000000000;
```

Relevant parameters are all at the end of the file.

6. Identify externally triggered muon collection ced file. This will have the format:

```
"…\FMNNN\FMNNN_Pshp_extMuons\Event
Collections\041217224225_FM104_Pshp_calu_collect_ext.ced"
```

Where the FMNNN_Pshp_extMuons folder may have various names.

7. Run ced2root (or appropriate converter) on the muon file, using a procedure similar to 4 above. Output file should go in the FMNNN_calibGenCAL_analysis folder (in this scenario – other storage organization schemes are possible).
8. Prepare to run ciFit by modifying ciFit_option.xml (see Section 9.4.1.2). Make sure to set the OUTPUT_FOLDER to the desired location (FMNNN_calibGenCAL_analysis in this case). The INFILE_TYPE should be left at ROOT and the ROOTFILE_LE1 and ROOTFILE_HE1 files should point to the calibGen Element 1 and Element 6 converted root files. None of the other ROOTFILE inputs are currently used. Set the TIMESTAMP, STARTTIME, STOPTIME and INSTRUMENT variables to match the files being analyzed. Other settings can generally be left at default. DAC_SETTINGS default matches the standard set of DAC settings currently in use. This may change in the future, but future versions should allow DAC settings to be extracted from the data file. ciFit creates a text logfile that contains the setup options file and all the screen output from the run.
9. Run ciFit. The only parameter is the modified options file path. Output files should be created according to the OUTPUT_FOLDER parameter in the options file.
10. Prepare to run muonCalib by modifying muonCalib_option.xml (see Section 9.3.1.2). Make sure to set the OUTPUT_FOLDER to the desired location (FMNNN_calibGenCAL_analysis in this case). The INPUTFILE_LIST should point to the converted root file generated in 7 above. INTNONLINFILE_TXT points to the text file output of ciFit. All the various output file names will be autogenerated in the OUTPUT_FOLDER unless overridden by being explicitly set in the options file. Set the TIMESTAMP and INSTRUMENT variables to match the files being analyzed. Other settings can generally be left at default. Note that the required version of the dtd file (part of the XML output file generation) may not be present depending on what version of glastrelease or EM has been installed. If the requested version is not available, download it from the CVS site. muonCalib creates a text logfile that contains the setup options file and all the screen output from the run.
11. Run muonCalib. The only parameter is the modified options file path. Output files should be created according to the OUTPUT_FOLDER parameter in the options file.