Cal Calibration Software

Zachary Fewtrell, NRL
March '05

Overview

1. Calibration Types Roundup
2. Software & Data Flow
3. Software HOWTO.
4. What's Next?

SECTION 1

Calibration Types

Calibration Types

1. Pedestals
2. ADC -> DAC *(aka IntNonlin)*
3. Asymmetry
4. MeV per DAC
5. Thresholds

Pedestals

1. ADC units

2. Measured w/ muon calib

3. 1 per ADC range per Xtal.

Average values from FM101:

LEX8/HEX8 ≈ 510

LEX1/HEX1 ≈ 210

ADC -> DAC

> What's a DAC scale?

- Units from onboard ***charge-injection DAC*** circuit

> Why use it?

- effectively linear, unlike ADC

ADC -> DAC

1. Multiple points measured via *charge-injection*
2. Data is smoothed & stored as spline function points.

Avg. ADC/DAC from FM101

LEX8/HEX8 ≈ 11.2

LEX1/HEX1 ≈ 1.3

Asymmetry

1. log(posDAC/negDAC)

2. Computed for all *diode-size* combinations

POS FACE	NEG FACE
Large Diode	Large Diode
Small Diode	Small Diode
Large Diode	Small Diode
Small Diode	Large Diode

3. Measured w/ muon calib.

4. 10 spline points along xtal length.

Asymmetry

Average values from FM101

POS FACE	NEG FACE	Asym
Large Diode	Large Diode	-0.3 -> 0.3
Small Diode	Small Diode	-0.3 -> 0.3
Large Diode	Small Diode	1.3 -> 2.0
Small Diode	Large Diode	-1.3 -> -2.0

MeV per DAC

1. Inverse of gain calib type.
2. One val per diode, per xtal.
3. Measured w/ muon calib.

Average MeV/DAC for FM101

Large Diode ≈ 0.37

Small Diode ≈ 2.0 (using muon gain)

Thresholds

1. ULD - Upper Level Discriminator

2. LAC - Log accept

3. FLE, FHE

4. Measured w/ muons *and* charge-injection.

SECTION 2

Software & Data Flow

Relevant CMT Packages

1. **calibGenCAL:**

   generate **XML** calib tables from **online** output.

2. **CalibDataSvc** (Joanne Land)

   Allow Gleam to **select** proper XML files based on ***time***, ***instrument***, **&** ***flavor***.

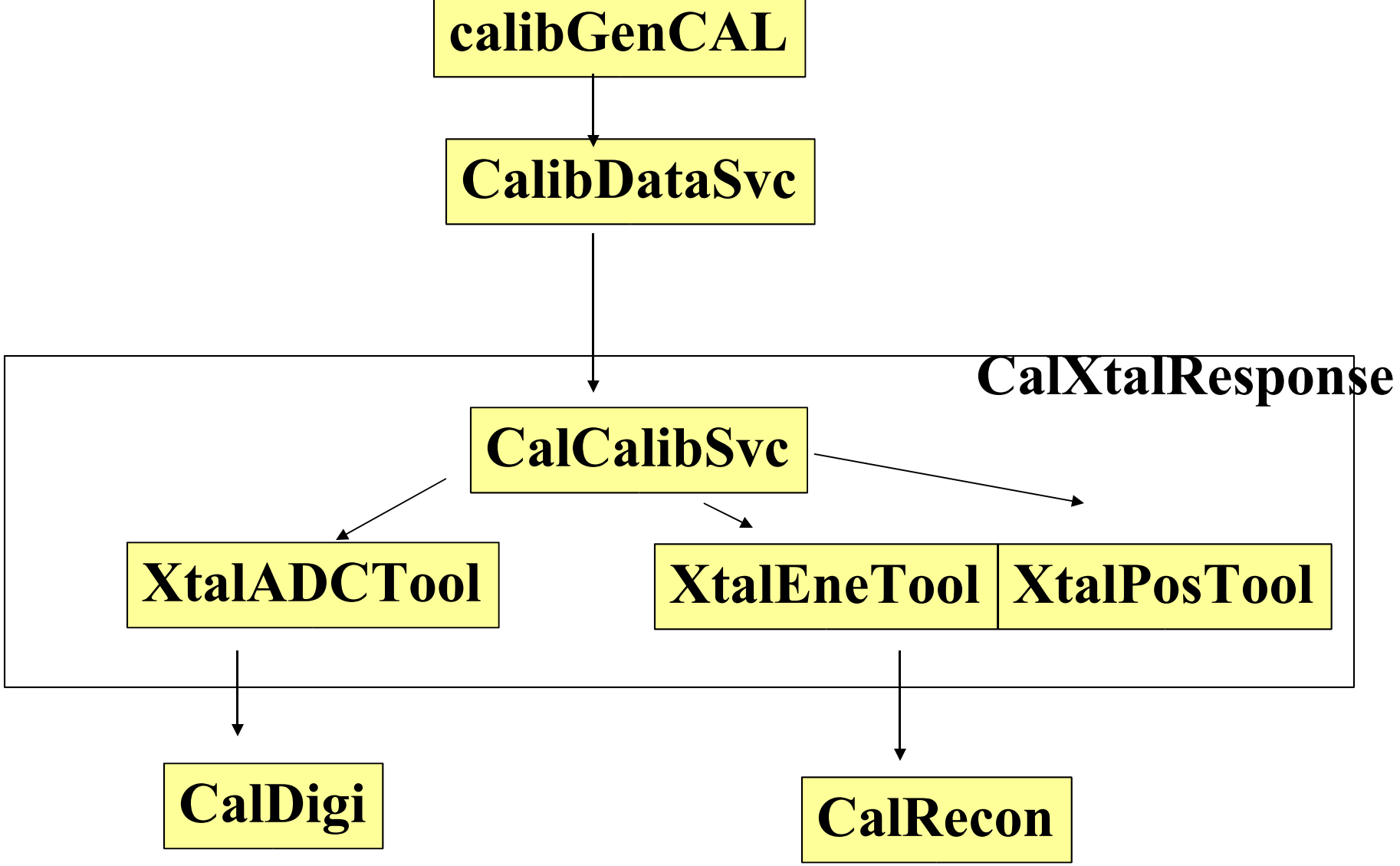
3. **CalXtalResponse (Zach Land):**

   digi <-> energy *conversions*.

   "Everything that goes on inside a **single xtal**." -Mark

4. **CalDigi & CalRecon**

   Use CalXtalResponse work w/ ***full*** Cal.


calibGenCAL
CalibDataSvc
CalXtalResponse
CalCalibSvc
XtalADCTool
XtalEneTool
XtalPosTool
CalDigi
CalRecon

calibGenCAL Apps

App	Inputs	Output
runCIFit	**Charge injection** digi-root	**ADC -> DAC**
runMuonCalib	- ciFit output - **Muon** collect digi-root	**- Pedestals** **- Asymmetry** **- MeV-Per-DAC**

HOWTO – runCIFit

ciFit_option.xml

C:\Documents and Settings\fewtrell.HESE\Desktop\ciFit_option.xml - Notepad2

File Edit View Settings ?

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

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

<ifile cvs_Header="$Header: /home/cvs/SLAC/calibGenCAL/src/ciFit_option.xml,v 1.7 2005/02/16 20:09:11 few
       cvs_Revision="$Revision: 1.7 $" >

  <section name="TEST_INFO"> Test configuration info.
    <item name="TIMESTAMP"      value="2004-11-11-14:39"> Timestamp for test run.</item>
    <item name="STARTTIME"      value="2004-11-11-14:39"> Start of test run</item>
    <item name="STOPTIME"       value="2004-11-11-14:52"> 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=
"0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,80,96,112,
480,496,512,543,575,607,639,671,703,735,767,799,831,863,895,927,959,991,1023,1055,1087,1119,1151,1183,121
791,1823,1855,1887,1919,1951,1983,2015,2047,2079,2111,2143,2175,2207,2239,2271,2303,2335,2367,2399,2431,2
,3039,3071,3103,3135,3167,3199,3231,3263,3295,3327,3359,3391,3423,3455,3487,3519,3551,3583,3615,3647,3679
settings used in test.</item>
    <item name="N_PULSES_PER_DAC" value="50"> Number of pulses per DAC setting</item>
```

runCIFit

1) All settings in single **XML** file.
2) Default location:

   "calibGenCAL/src/ciFit_option.xml"
3) Can specify cfg file in command-line i.e.

   "runCIFit myOptions.xml"

Example Entries

```
<item name="INSTRUMENT" value="LAT"> Instrument name</item>

<item name="TIMESTAMP"  value="2004-11-11-14:39">
     Time of test run.</item>
```

runCIFit

Important config parameters:

Name	Description	Example Vals
TIMESTAMP	Timestamp for test run	2004-11-11-14:39
INSTRUMENT	Instrument name	LAT
TOWER_LIST	List of towers currently installed.	1,2,3,4
OUTPUT_FOLDER	Folder for auto-named output files	"../output/"
ROOTFILE_LE1	input DIGIROOT file Low Energy	041111143546_FM101_PSHP_CALU_COLLECT_CI_SINGLEX16.root
ROOTFILE_HE1	input file High Energy	041111145340_FM101_PSHP_CALU_COLLECT_CI_SINGLEX16.root

`runCIFit`

Features:

2) Auto-generate output filenames from input filenames. *(optionally overridden)*

3) Auto-saves stdout to log-file.

4) Config file is quoted in output log.

5) Can use CMT & other ***environment*** variables in string fields.

6) We hope to ***automate*** as much as possible in the future.

runMuonCalib

1) Same features as runCIFit
2) XML Default location = *"calibGenCAL/src/muonCalib_option.xml"*

runMuonCalib

Important config parameters*:

Name	Description	Example Vals
INPUTFILE_LIST	Space delimited list of input ROOT digi files	041022230030_FM101_Pshp_calu_collect_ext.root
INTNONLINFILE_TXT	IntNonlin TXT file generated by ***runCIFit***	LAT

*shares many params w/ ciFit, such as **TIMESTAMP**, **INSTRUMENT**, **TOWER_LIST**, etc

CalibDataSvc (aka Joanne Land)

1) LAT wide calibration database system.

   (Used by *all* subsystems).

2) constants are stored on local file-system in ***XML*** files.

3) *calibUtil* package defines XML format in *dtd* files.

   At SLAC, David Smith as set up **$LATCalibRoot** for storage of calib files.

3) XML lookup is through a ***MYSQL*** server at *centaurusa.slac.stanford.edu*

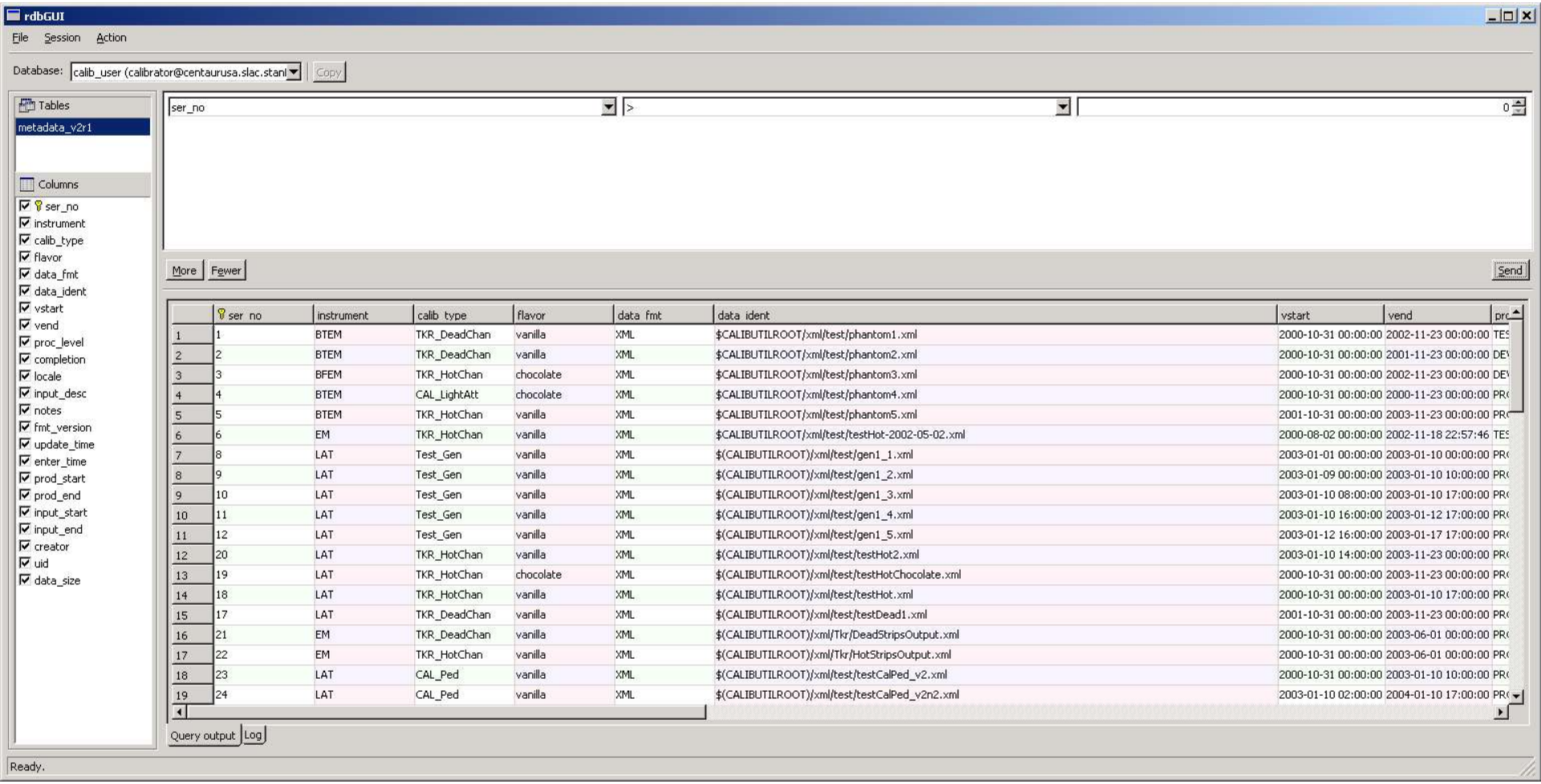
CalibDataSvc

1) You need to know some things about ***CalibDataSvc*** to get it to work w/ Cal software

2) Queries are made to meta-database on following fields:

Name	Description	Example
FLAVOR		**“vanilla”, ”FM101”, ”zachtest”**
INSTRUMENT		**“EM”, “LAT”**
CALIB_TYPE		**CAL_Ped, CAL_MevPerDac, CAL_Asym, CAL_TholdCI**
TIME (optional)	Event time, or ‘fake clock’ time	`CalibDataSvc.CalibTimeSource = “data”;` `CalibDataSvc.startTime = "2003-2-28 23:59:59";`

rdbGUI



1) Use rdbGUI to *register* new **XML** files

2) First place to go if you're having **TROUBLE??** loading a particular calib.

CalibDataSvc

Make sure you have all your ducks in a row!

3) ***ALL calib_types*** for given flavor
   1) **CAL_Ped, CAL_TholdCI, CAL_MevPerDac, CAL_Asym**
4) Entries in ***rdb database***!
5) ***Environment*** variables set up.
   1) **$LATCalibRoot**
6) Time stamp matches the ***validity period*** of your data.
   1) Time ***not*** always present, may need *fake clock*.

CalibDataSvc

Relevant jobOptions settings:

```
// you MUST add your desired calib flavor to list!!

CalibDataSvc.CalibFlavorList += "FM104";



// you MAY need to set up some time info

// "clock" - for data w/out time info

CalibDataSvc.clock = "data";
```

CalXtalResponse

1. **CalCalibSvc:**

   Provide ***easy*** access to any cal calib constant.

   - "Tell me which crystal & I'll give you a float"

   Evaluates ***spline*** functions.

   Used by **Xtal***Tools.

2. **XtalADCTool**

   MC -> digi for single xtal.

3. **XtalEneTool, XtalPosTool**

   Recon energies & positions for single xtal.

4. **defaultOptions.txt:** provides modular options for Gleam to include from inside package.

CalCalibSvc

1. One stop shop for Cal constants
2. Easy interface (no TDS knowledge req'd)
3. Supports 'ideal' flavor which always works & skips the mysql db entirely.
4. Currently only used ***internally***, can provide calib to any package in Gleam, userAlg, etc
5. Supports multiple instances for multiple simultaneous flavors.

CalCalibSvc

'ideal' flavor

1. **No internet req'd.**
2. **No configuration req'd**
3. **No XML req'd**
4. **Always works**
5. **Contains same set of average constants for each xtal.**
6. **Good for testing, yada yada.**
7. **Currently the Gleam default – don't get fooled into thinking your using *real* data.**

CalCalibSvc

interface

1. `CalCalibSvc::getPed(CalXtalId,`
   `pedestal); // output`

2. `CalCalibSvc::evalPos(CalXtalId,`
   `asym, // input`
   `pos); // output`

CalCalibSvc
jobOptions

```
// default flavor for all calib types
```

- `CalCalibSvc.DefaultFlavor = “ideal”;`

```
// override flavor for particular calib
types
```

2. `CalCalibSvc.flavorPeds = “vanilla”;`

XtalADCTool

Interface

```
XtalADCTool::calculate(CalXtalId,
                       MCIntHits, //input
                       rangeP, //outputs
                       rangeN,
                       adcP,
                       adcN,
                       peggedP,
                       peggedN);
```

XtalADCTool

Method

2) Convert each individual hit to ***DAC*** scale

2) Sum DAC vals for each diode

3) Add noise

4) Convert diode DAC vals to **ADC.**

XtalEneTool

Interface

```
XtalEneTool::calculate(CalXtalId,
                       rangeP, //inputs
                       rangeN,
                       adcP,
                       adcN,
                       energy); //output
```

Interface 2 (single face)

```
XtalEneTool::calculate(CalXtalId,
                       adc, //inputs
                       range,
                       position,
                       energy); //output
```

XtalPosTol

Interface

```
XtalPosTool::calculate(CalXtalId,
                       rangeP, //inputs
                       rangeN,
                       adcP,
                       adcN,
                       position); //output
```

More Joanne Land

1) Not you father's ***CalXtalID***

   CalXtalId now contains optional ***range & face*** info.

2) ValSig

   Contains ***value & sigma*** in one class,

   many calib constants are described w/ this.

CalUtil/CalDefs.h

1) Provide ***flat index*** classes for Cal components.
   1) Good for arrays
   2) All xtals in tower
   3) All diodes tower
   4) All ranges in xtal
   5) Etc…
   6) Get(),set() routines (***layer, column, tower***…)
   7) Conversion routines
   8) Iteration.
   9) Range checks.
   10) more complex for multi-tower.

Cal Calibration Software

What's Next?

Zachary Fewtrell, NRL
March '05

What's Missing?

Part 1 (of 2)

1. Multi-tower support.
2. Threshold calibrations.
3. Integration w/ I & T pipeline.
4. Documentation / doxygen.
5. Improved test apps.

Multi-tower Support

1. Current Gleam: replicates single tower.
2. *calibGenCAL* merge separate measurements?
   - merge XML files?
3. Indexing more complicated.

Integration w/ I & T pipeline.

1. Less editing of config files
   - read UDF headers, etc
2. “One script to rule them all?”
   - several steps
   - order will get more complicated

What's Next?

Part 2 (of 3)

1. FLE cross-talk & other voodoo.
   - ask Sasha!
2. Flight *like* calibrations.
   - scaled from muon calib.
3. *Independent position* in CalRecon
4. Updates to Recon data structures
   - at xtal level

Independent position in CalRecon

1) Pos. is *poor* function of asymmetry.

2) Also for dead channels, low signal.

Recon Data Structures

Old Fields

2) enePOS

3) eneNEG

4) Position

5) Multiple range estimates

New Fields

2) *Single* energy

3) *Asymmetry*

4) position

5) *single* range estimate (default)

6) *Method* flags

Xtal Recon Method Flags

1. Records decisions made by **CalXtalResponse**

2. 32-bit bit-field?
   - leave some for posterity

Fields:

2) range/readout(s) used
3) Faces used.
4) Faces below *threshold*
5) Xtal below threshold, *valid* est?
6) External position used?
7) Link to digi, pegged?, which external position?