"Online" Calibration

- Definition of calibration
- "Online" calibration system
  - Conditions database
  - accumulation
  - fitting
  - verification and storage
  - tools
- Offline (PR) use of the "online" calibration system

What is a calibration?

- A physicist's working definition
  - The characterization of the response of the electronics and/or detector to a known stimulus
    - Response is observed through the detector electronic readout
    - The stimulus can be artificial (pulse) or natural (track)
    - Characterization is via a parameterized function
    - A particular calibration results in a specific set of parameters
    - Parameters may translate the raw output of the detector and/or electronics into physical units
  - Examples:
    - Pedestals (reponse to null stimulus)
    - Dirc light gain (ADC response to single photon)
    - EMC energy correction (convert from counts to MeV)
    - DCH time-to-distance (using cosmic tracks)
- What isn't a calibration?
  - Alignment (= position of sensitive parts of the detector)
  - Ambient environment measurements (temperature, pressure, gas composition, magnetic field maps)
- What are the steps of a calibration?
  - The same (or similair) stimuli are repeated many times
  - The response to each stimulus is included in some persistent record (accumulation)
  - After sufficient accumulation, the persistent record is post-processed to extract the parameters (fitting)
  - The parameters are stored for later use (storage)

The Babar Calibration System

- A set of collaborating C++ classes which encapsulate accumulation, fitting, and storage
- An interface to the Conditions Database (Objectivity) which provides:
  - data verification before storage
  - data protection upon retrieval
  - A simplified programmer interface
- A user interface providing familair tools
  - A gui database browser
  - Translation from calibration objects→histograms,ntuples
  - Logical manipulation of calibration data
    - Comparison functions
    - Merger
    - Selection/extraction
- A robust tool which will be used for online calibration (electronically induced stimulus)
  - Can be run in a ROM or in OEP
  - Can be automated
- A potentially useful tool for offline (datastream stimulus) calibration

The Conditions Database

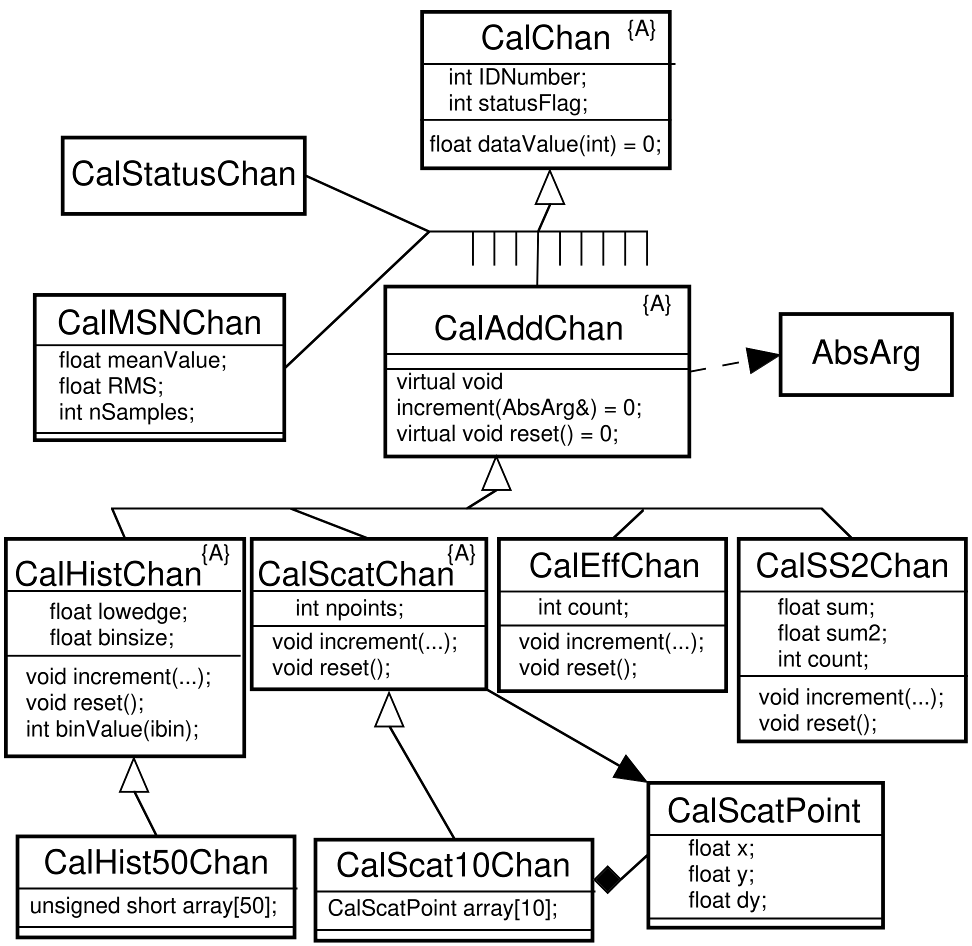
- The work of David Quarrie + Jim Ohnemus
- A particular implementation of an Objectivity database which uses time as a lookup key
  - Objects in the database are associated with a time interval over which they are valid
- Provides a basic read/write interface and protection against unauthorized writing

```
Federated.db
  conditions/
            dch/
             drc/
            emc/
              ifr/
             svt/
                 svtIndex.db
                 svt000000/
                              svt00000.db
                              ...
                 ...
             ...
```

- The index database holds many *containers*
- Each container has a separate interval time-line for the “object” it represents
- Intervals hold pointers to the actual data objects, stored in the numbered databases

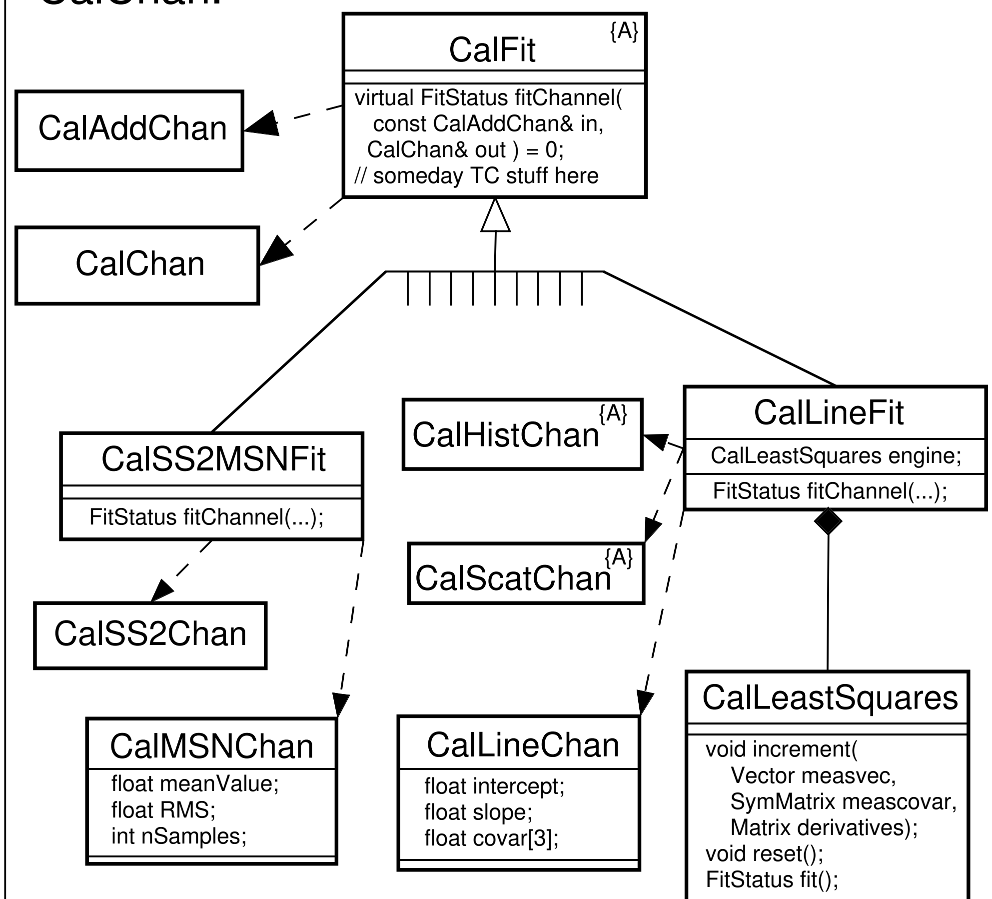
CalChan

The central class of the calibration system is CalChan, an abstract base class which represents the (intermediate) result of calibrating a single 'channel'.



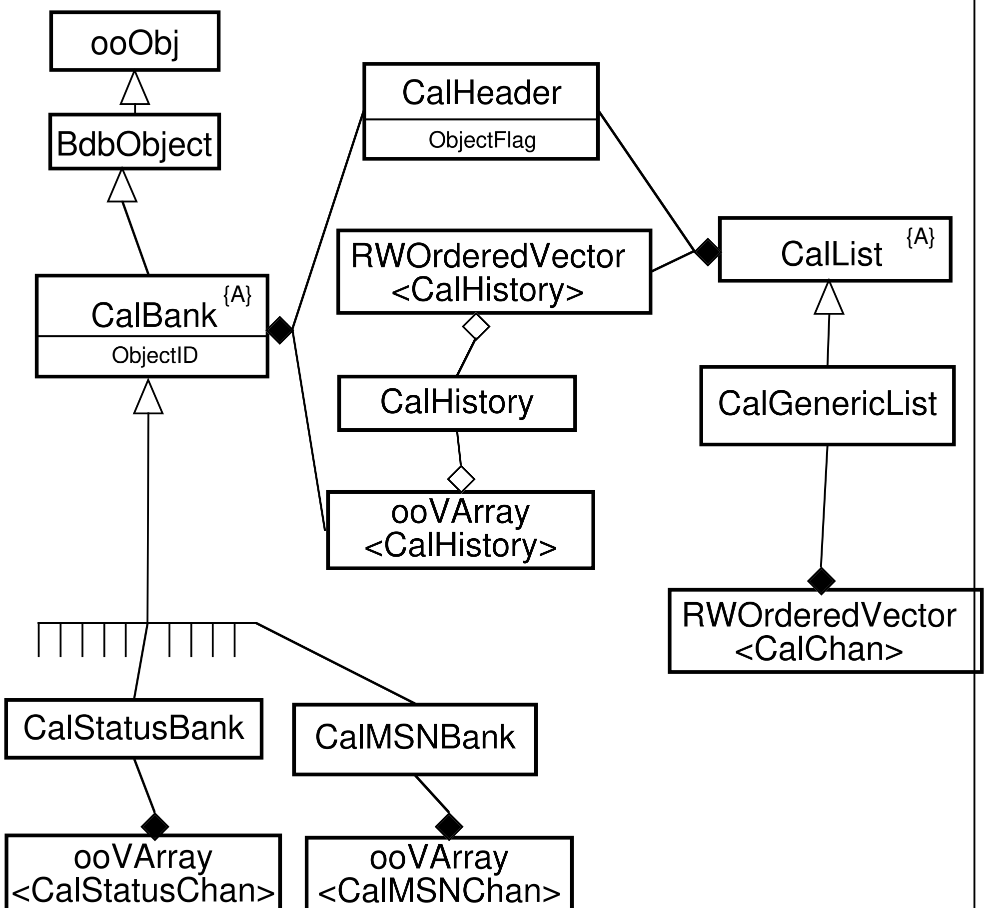
CalFit

The CalFit class performs post-processing after accumulating data. It functions as a *factory*, taking as input an (incremented) CalAddChan and producing as output a CalChan.



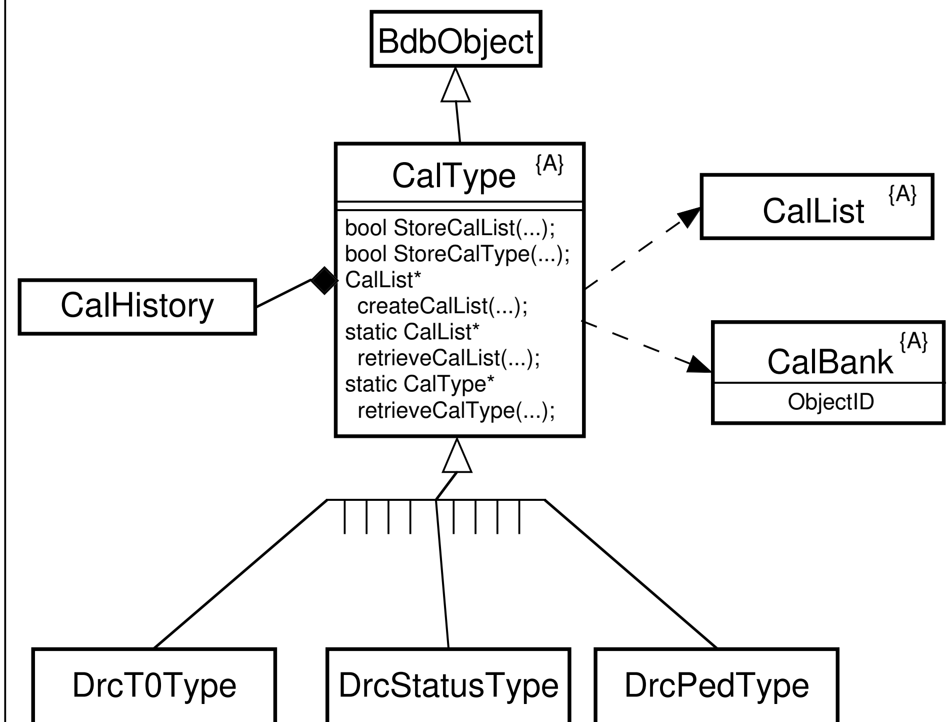
CalChan Collections

There are two (base) collection classes of CalChans: a transient (CalList) and a persistent (CalBank).



CalType

The CalType class embodies the idea of a particular type of calibration (IE Dirc T0 using LED flasher). CalType must be subclassed for each calibration type.

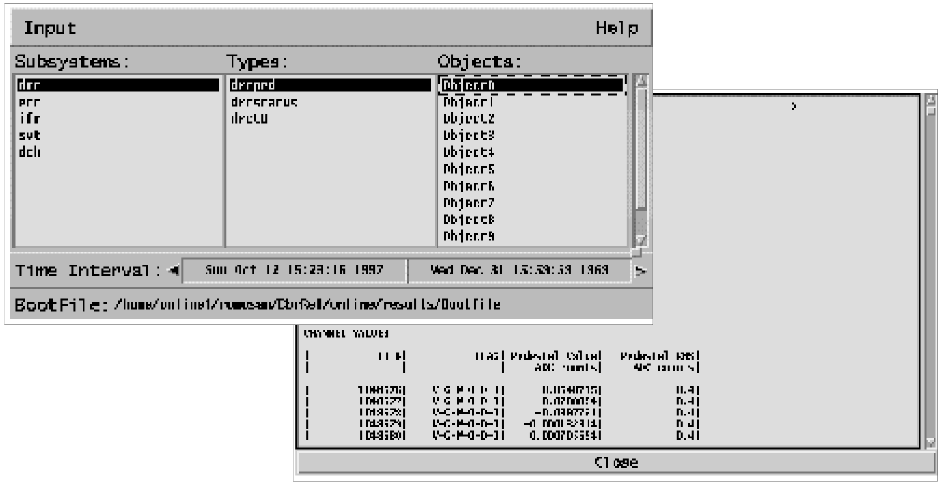


CalType Functions

- Define the storage for the type (CalBank, CalChan)
- Serve as gatekeeper to the database for a type
  - Retrieve data store as a CalBank in the form of a CalList
  - Store data in a CalList
    - Verifies the structure of the collection (fully implemented function as base class level)
    - Verifies the content of the collection (using a user-supplied virtual function) NOTE: the programmer decides how many failing channels in a collection is acceptable.
    - If verification is successful, it copies the collection into a (persistent) CalBank
- CalType is itself persistent to store content check cut values (which can evolve)
- Users never interact directly with CalType objects: (friend) CalList provides access

Tools

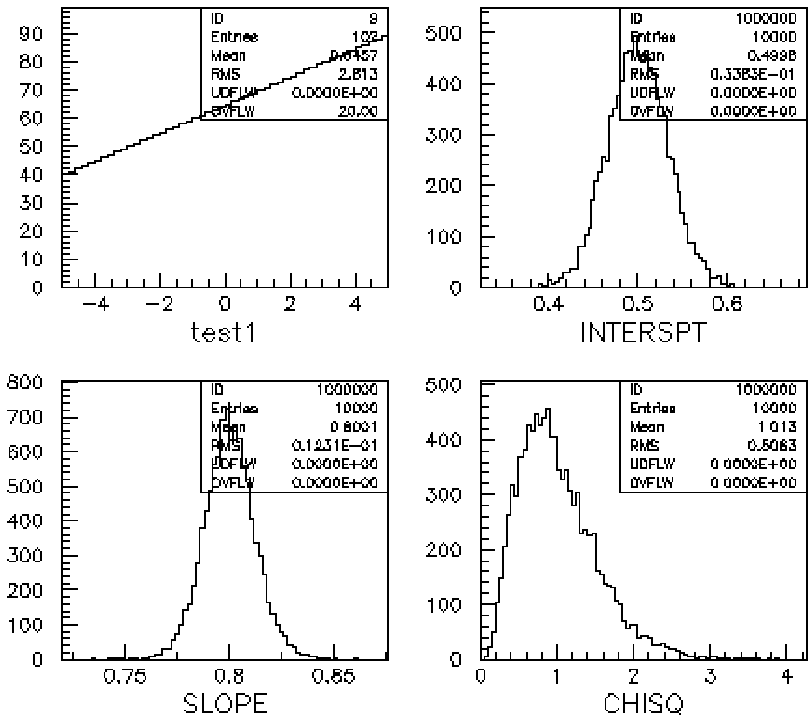
- CalList provides logical manipulations functions (eventually)
  - Compare two lists
  - cut on values in a list
  - merge lists
  - ...
- CalHistBook converts CalLists into histograms or Ntuples (Vasia Shelkov)
- BdbBrowser provides a gui access to the calibration database (through the CalList interface) (Alex Romosan)



Tools

Results from 10K linear fits, stored in an ntuple using CalHistBook

```
* Ntuple ID = 7      Entries = 10000     CalLineChan
*******************************************************************
* Var numb * Type * Packing *    Range     *  Block  *  Name     *
*******************************************************************
*      1   * R*4  *         *              * CHAN_ID  * chan_id
*      1   * R*4  *         *              * INTERSPT * interspt
*      1   * R*4  *         *              * SLOPE    * slope
*      1   * R*4  *         *              * CHISQ    * chisq
*      1   * R*4  *         *              * COVAR1   * covar1
*      1   * R*4  *         *              * COVAR2   * covar2
```



Modifications for use in PR

- CalChan, CalFit, CalBank, CalType interfaces are very general, base implementations are sensible: keep them
- Implement explicit (templated) forms of CalList to avoid type ambiguity (offline equivalent of CalTC)
  - Increment an entire list of CalAddChan
  - Fit an entire list of CalAddChans
- Implement CalIndirectChan
- Implement minuit-based CalFit (???)
- Do whatever else is needed to integrate the needs of online and offline calibration