CAL Software Code Review Report
June 28, 2002

Reviewers: Riccardo Giannitrapani, Berrie Giebels, Traudl Hansl-Kozanecka, Heather Kelly, Karl Young

Attendees: Toby Burnett, Alexandre Chekhtman, Richard Dubois, Eric Grove, Michael Kuss, Leon Rochester, Mark Strickman, Tracy Usher

Introduction
The CAL software review was the third in our continuing series of code reviews. We would like to thank the CAL software team who clearly spent much time to prepare for this review. This is still a learning process, but with each review it seems both the developers and the reviewers have a better idea of what to expect.
In this review, we took a look at both the documentation and the code from selected files. It was fairly easy to choose the classes to study: CalDigi, CalCluster, CalXtalRecData, CalDigiAlg, CalXtalRecAlg, CalClustersAlg.

Doxygen Documentation
General Comments:
The issue of brief versus detailed descriptions was discussed. In much of the CalDigi documentation, the whole description was part of the brief section. This is rather undesirable. It was unclear how to determine where the brief description ended and the detailed began. After reviewing the Doxygen manual, it seems this can be accomplished in a number of ways. Certainly, inserting a newline, at the end of the brief description will do the trick, as was done in the CalRecon code. In addition, since we have the JAVADOC_AUTOBRIEF option set to YES in our default Doxyfile, we can remove the @brief keyword altogether. This option forces the brief description to include only the first sentence in the description. The remaining sentences will be part of the detailed description. This will have to be more clearly explained in the code documentation recommendations.

The Doxygen output mainpage of CalDigi did not include the version tag. Upon investigation, it seems that while the version statement is in the CalDigi requirements file, there is a blank line between the package name and the version line. This could be the source of the trouble. The CalDigi requirements file should be updated and the code (in glastpack and VCMT) used to extract the version should be checked to see if this is the root of the problem.

The mainpages should contain more information about the test routines. How can they be used? What is the expected output?

When specifying TDS INPUT and TDS OUTPUT comments, one should either include the full TDS path or the full EventModel constant. This will allow readers to truly understand where this data is located. The code documentation recommendations should be updated to state this explicitly.

The remaining comments concerning the Doxygen documentation will be split according to package.

Event
As noted in the code documentation recommendations, obvious comments such as, // default constructor and //include files, should be removed, for example, in the CalXtalRecData class.

CalDigi
Make sure that source comments such as TDS INPUT are used rather than INPUT, when denoting TDS data, this includes the CalDigiAlg and CalXtalRecData code.
More details concerning the ADC and the possible range values should appear somewhere in the Doxygen documentation. Perhaps the description of the ranges belongs in CalXtalId - but at the very least such documentation should be referenced somewhere from CalDigi, as these constants are used often in the CalDigi code.

CalRecon
The release.notes need to be enclosed in a doxygen block, as has been done for the CalDigi package, and linked into the mainpage.

Code Review

General comments
We should avoid hard-coded constants. Both packages should utilize XML files as input for parameters. This will require use of the IFile class available in the xml package.

Developers should remember to follow the data member naming conventions in our coding standards:
http://www-glast.slac.stanford.edu/software/CodeHowTo/codeStandards.html
The first letter after the "m_" should be in lowercase.

As for the test routines in these packages, it is recommended that the standard GlastPolicy test_package pattern be used, rather than the defining a local main routine in the local package.

Methods should be defined in the source file in the order they are declared in the header, see CalClusterAlg for an example where this is not done.

Much more generally, common utilities such as unit conversions should be done within the facilities package. The core group should determine a set of common utilities and these should be added to the facilities package.

We have code that uses both the CLHEP and the geometry package's vector and point classes. It would be best to standardize. Unfortunately, there is much code already written, and it would take some time to convert. Rather than force all existing code to be modified, it was decided that all new code should use the CLHEP classes.

When retrieving data from the TDS, we should use const. This is good practice that should be applied to all of our code.

Event
Enumerations, where available, should be used whenever possible. For example, CalXtalRecData::getSelectedEnergyRange should probably use idents::CalXtalId::AdcRange rather than a char.
The non-const get routines should be removed from the CalXtalRecData class. In addition, it would be clearer to store the range as an int, rather than a char.
Use of the standard Gaudi fillStream method should be preferred over a user-defined method such as CalXtalRecData::writeOut.
The CalClusterCol class should be removed and replaced with an ObjectVector of CalCluster

CalDigi
Programming Details
It was noted that crystal segmentation needs to be tested and further discussed.
An additional readout mode is required, *forced*, which will allow the user to force a specific range to be read out.
Variable names should differ from type names by more than capitalization, SignalMap and signalMap for example. Use of an enumeration of the diodes is recommended within the CalDigiAlg.

Implementation Details
A calibration constants database is some months off. How will noise constants, for example, be handled in the short-term? The CAL software team should continue the discussion of handling noise.

CalRecon
Programming Details
CLHEP routines should be used whenever possible. For example, is the gamma function truly necessary or can the CLHEP routine LogGamma be used instead?

It was decided to remove the CalClusterAlg.CalNumber parameter. It is currently unused and unnecessary.

Whenever a call returns a StatusCode, it should be checked before execution proceeds. In CalXtalRecAlg::execute there should be a check of the status code after the call to the retrieve method.

The CalRep class should be nested in CalDisplay.

Implementation Details
Currently, CalRecon can process ideal simulated data; it is not yet ready for realistic data that requires calibration. The CAL software team recognizes this need and is working to implement new calibration classes.

A number of CalClustersAlg implementation details were discussed. The CalClustersAlg::Leak constants are based on a different geometry. The constants should be extracted and stored in an input XML file. In addition, the constants need to be re-estimated for the new geometry.
The current linear approximation for light yield is probably inaccurate for high energy protons, recent test results from CERN will prove to be useful.

Conclusions

The CAL software team has laid the groundwork for their code documentation. There are some loose ends to tie up, but the documentation looks good. There will undoubtedly be updates to the CAL code over the coming months. The developers are urged to keep the documentation up to date as modifications are implemented. There will certainly be further CAL software reviews in the future.

The issue of manpower was never directly addressed, however, it is clear that the CAL software effort would benefit from additional contributors. Alexandre Chekhtman is clearly the cornerstone of the programming effort; his hard work should be commended. The efforts of SAS manager, Richard Dubois, should be noted - the CalDigi code was his first contribution to the CVS repository. Reorganization within the CAL programming team has been taking place over the last few months. Naming Mark Strickman CAL software czar is some evidence that the dust is finally settling. Help from France should be on the way in the coming months, and this will help mitigate the manpower issue.

There is also additional work for the Documentation Task Force and the core group in general. The existing coding standards should be updated. Among the items to consider:

- Use of const when retrieving data from the TDS
- Defining access methods as const.
- Use of CLHEP methods for all new code - and update existing code as possible

This set of items should be reviewed by the core group in the near term.
The Documentation Task Force should review the existing code documentation recommendations and update those portions that require more detail.

Action Items for CAL Software Group

- *Self-Imposed Items - details mentioned by the CAL group themselves*
    - Update energy correction algorithm with coefficients based on the current geometry
    - Implement calibration classes to accept input calibration data.
    - Implement new algorithms
        - Cluster search
        - Low energy corrections
        - Corrections to position and direction calculations for off-axis events
    - Add "forced" range mode to CalXtalId and use it in the Event::CalDigi TDS classes
    - Replace CalClusterCol with ObjectVector
    - Remove Minuit optimizer from CalRecon code - either put the code in a common utility package like facilities - or consider using ROOT's TMinuit directly.
- *Documentation and Programming Items*

    - Fix typo in Event::CalDigi class description: ACD => ADC
    - Add $Header$ to Event::CalDigi
    - CalDigi package mainpage: Add more info on test routine
    - Make sure TDS INPUT/OUPUT comments provide full path or full EventModel const name
    - CalDigi::CalDigiAlg - separate execute method into multiple methods
    - CalRecon::CalClusterAlg - remove CalNumber jobOptions parameter
    - Event::CalXtalRecData - Get rid of non-const versions of getEnergy and getPosition methods
    - Event::CalCluster - Use fillStream instead of writeOut
    - CalRecon::CalDisplay - nest CalRep class in CalDisplay
    - Use XML files to store constant parameters
- *Implementation Items*
    - Test to see if xtal segmentation, as planned, gives adequate positions.
    - Use direction determined by TKR (if available) to initialize direction fit.
    - Study recent CERN test results and consider the use of non-linear light propagation.
    - Discuss use of multiple scattering information for energy estimate, starting with the existing TkrRecon algorithm. Also consider the EGRET method of correcting low energy estimations.

Action Items for Documentation Task Force

- Update the existing description of brief versus detailed descriptions so it is clear how to separate the two in the resulting Doxygen output.
- Does the code that extracts the version from the requirements file require that the version line appear directly after the package <packageName> line?