Crystal Recon Design Description XtalRec

This page describes the design of CAL crystal recon. In particular, it is intended to contain the definition of the xtal status word, the algorithm to mitigate xtal saturation, the algorithms to mitigate electronic failures, etc.

It includes some discussion of mitigation of the failure of the HE channels of a single GCFE. There are 3072 GCFEs in CAL, two for each of the 1536 xtals. The problem occurred in the run starting at MET = 301753824, which is MJD 55402.52108796, or 2010 Jul 25 at 12:30:22 UTC, or 2010 day 206 at 12:30:22 UTC. For more details and related discussion, see ?CAL July 2010 channel failure.

Changes necessary to recon

CalXtalRecAlg and related objects will be changed for Pass 8 (and perhaps earlier) for modify the xtal response in case of failures and for saturated readouts.

Crystal status word

We are adding status bits to xtal recon to indicate that the energy and position information returned by a xtal have been modified, along with some information about why the modification was necessary. The intent here is to fix the reconstruction of events that hit a xtal at the level of xtal recon so that the failure and fix are transparent to downstream energy reconstruction and clustering. Reconstruction code that follows xtal recon could make use of those bits to accept/reject that xtal or modify higher level recon algorithms, if the authors and architects of that code wish.

Each of the 1536 xtals will have a 32-bit status word indicating its hardware and configuration status, and that 1536-element array will be stored as a calibration database quantity. The initial value of the status word for each event comes from the calibration database. In addition, some of the bits in that status word are used by xtal recon to indicate whether the energy and position calculated by xtal recon for a given event are good, need to be corrected, or have been corrected for the appropriate failure mode indicated by the hardware and config status bits.

The status word array needs to have an epoch of validity, like all other calib quantities. Until this failure, the status word for each xtal end was 0x0. Now we have one xtal with two bits set, the bits that indicate bad plus-face HEX8 and HEX1, and that condition began on 25 July.

In this status word, we need to encode both a priori status information – including hardware status that changes slowly in time, easily tracked as a new calibration DB file, and configuration information that can change from run to run and really should be extracted from MOOT – as well as status information from the xtal recon member functions that indicate whether the energy and position data coming out of xtal recon are good or bad, or whether they were derived by a special algorithm.

For example, for our current failure, we have bad HEX8 and HEX1 data, but we've done nothing to the config registers of this GCFE to prevent HEX8 and HEX1 from appearing. We *could* configure to prevent them from being selected as the best (i.e. only) range, and we're building that config now as a contingency. The code to detect whether a given event is suffering from the failure will depend on how we've configured the GCFE, and I'm convinced for this particular failure it's better if we don't touch the config. I'd prefer to read the config information out of MOOT for the appropriate run, but I'll bet for the moment we should just merge the h/w status and config into a calibration DB file. That's fine. We won't be changing the relevant configuration parameters on a run-by-run basis, so it'll change slowly if at all, consistent with our usage of calib DB quantities.

— Usage —

Tracy and Chul have started coding following the thoughts laid out below.

Recall that the current recon sequence has two CAL passes, Cal1 and Cal2, surrounding an intervening TKR pass that is TkrReconAlg/FirstPass. We will modify the second CAL pass to include a CalXtalRecAlg member function that fixes the response of each CAL xtal that has a failure. Thus:

```
Cal1.Members = {"CalXtalRecAlg/first",
              "CalClustersAlg/first",
              "CalEventEnergyAlg/RawEnergy"};
Cal2.Members = {"CalXtalRecAlg/FixXtalResp",
                "CalEventEnergyAlg/second"};
```

On the first pass, for the current failure, CalXtalRecAlg::calculate() will see the status word indicating bad HEX8 and HEX1 data, so it'll set status bits indicating that it has generated bad energy and bad longitudinal (X or Y) position information.

Then the moments analysis will see a status word indicating bad energy and longitudinal position information, so it'll ignore that xtal in the moments calculation. If the energy info is bad (status & 0x01 == 1), then the moments analysis should ignore the xtal entirely. If energy info is good – as it will be if the moments analysis were called a second time – then moments can use the two transverse coordinates for this xtal, i.e. those provided by geometry, along with the energy info in the moments analysis.

The new function that fixes the xtal recon data from this xtal after the moments analysis will calculate a good energy and set the position equal to the closest approach of the shower axis (from the moments trajectory). Thus, it'll clear the bits indicating bad energy and bad position, and it'll set the bit indicating that the position it's returning has been derived from external information (i.e. the moments analysis). Any recon code downstream can then decide whether to use that xtal longitudinal position information or not.

I've divided the 32 bits into 16 for recon status (and used 10), 10 for hardware status (and used 8 so far), and 5+1 for config status (and used 4). Perhaps it'd make more sense to reserve more bits for recon status. Note that I've included two bits for Bill's mods to the longitudinal position calculation to indicate whether he's corrected for direct light in the near diode or corrected for an ambiguous asymmetry. Those can be set in the initial CalXtalRecAlg pass and used as desired later.

Comments?

I'd like to proceed with getting this new calib quantity in the database so we can extract it in recon with CalCalibSvc.

Eric

– bit map for CalXtalRecData status word –

(rev. 16 Feb 2010, Pass 8 discussion Grove, Bruel, Baldini, Usher)

bit	function
	status of recon calculation
0	bad energy
1	bad longitudinal position
2	energy has been provided by external means
3	energy has been calculated by failure mitigation algorithm
4	energy has been calculated for corrected longitudinal position
5	minus-face energy measurement is saturated
6	plus-face energy measurement is saturated
7	position has been provided by external means
8	longitudinal position has been corrected for direct light
9	longitudinal position has been corrected for ambiguous ratio
10-15	unused
	status of h/w
16	bad minus-face LEX8
17	bad minus-face LEX1
18	bad minus-face HEX8
19	bad minus-face HEX1
20	bad plus-face LEX8
21	bad plus-face LEX1
22	bad plus-face HEX8
23	bad plus-face HEX1
24-25	unused
	status of config
26	minus-face LE autoranging disabled
27	minus-face HE autoranging disabled
28	plus-face LE autoranging disabled
29	plus-face HE autoranging disabled
30-31	unused

Algorithms in CalXtalRecData

Algorithms for correcting current and not-unlikely future failures in FixXtalResp

On 5 Aug 2010 (Day 217), at 5:24 PM EDT, J. Eric Grove wrote:

Chul,
Here I've detailed the actions for FixXtalResp in the cases we currently understand. I've added a new CalXtalRecData status word definition [at the bottom of this email] adding Bill's corrected-energy bit and a bit to indicate that we've mitigated the failure.
Eric

```
case == bad plus-face HEX8 && bad plus-face HEX1 && ! plus-face HE autoranging disabled
/* this is the case for the current failure and current configuration */
/* if one of the HEX ranges is best range, fix it */
/* if plus-face first range == (HEX8 || HEX1),
        calculate E using the opposite face and the externally provided longitudinal position
        clear the bad energy bit
        set the externally provided longitudinal position bit
        set the failure-mitigation energy bit
*/


case == bad plus-face HEX8 && bad plus-face HEX1 && plus-face HE autoranging disabled
/* this is the case for the current failure, but it requires a configuration we have not used */
/* if LEX1 is best range and saturated, fix it */
/* if plus-face first range == LEX1 && plus-face > 4050 (i.e. saturated),
        calculate E using the opposite face and the externally provided longitudinal position
        clear the bad energy bit
        set the externally provided longitudinal position bit
        set the failure-mitigation energy bit
*/


case == bad plus-face LEX8 && bad plus-face LEX1 && bad plus-face HEX8 && bad plus-face HEX1
/* this is the case we may get to soon if this GCFE continues to degrade and the LE ranges fail */
/* in all cases,
        calculate E using the opposite face and the externally provided longitudinal position
        clear the bad energy bit
        set the externally provided longitudinal position bit
        set the failure-mitigation energy bit
*/
```

Algorithms for correcting for saturated xtals

This is the rough outline that we (Eric G., Philippe, Tracy and myself) agreed on during the F2F pass8 meeting in Pisa (Feb 2011).

1. Perform the start xtal rec step.
    - We'll put in all the new bits to keep track of what's happening, according to the scheme outlined above. Chul and Tracy will follow up on this and get it done.
2. If there's one or more saturated xtal(s), the first thing we'll do in CalRecon is a fit using the transverse information only with the **full** xtal collection (i.e. before the clustering) to be used to fix the position for saturated xtals. Luca B. and Philippe will take care of this.
    - This will require moving the current fitting routine (now living in CalRecon/src/Clustering/MomentsClusterInfo.cxx) in some common area (Tracy suggests CalRecon/src/Utilities/) to be used in more than one place. Luca B. will take care of this.
    - As a side project, the possibility of **not** using Minuit for the fit will be investigated, in order to gain in speed and avoid a dependency on ROOT. This is low-priority project that Luca B. will take a look into.
3. At this point (if there are saturated xtals) from CalRecon we will call a dedicated tool in CalXtalResponse to set the longitudinal position to our best value, based on the transverse fit (will need some coordination between all of us).
    - After this point the CalXtalRecData objects in the TDS will have a reasonable longitudinal position information even for the saturated xtals, and the correct information will propagate trasparently to the downstream code (with the proper status bit set).
4. The next step is the calorimeter clustering (i.e. the splitting of the xtal collection in clusters).
5. And after that we'll go through the standard transverse fit/moments analysis/classification on a cluster by cluster basis.
    - We decided to keep the CalMomentsData as the basic object the CAL moments analysis acts on. Luca B. will take care of propagating all the new information (i.e. the status bits) to the object.
    - Further possible adjustments to the xtals that need transverse fit at the cluster level will be done by calling the proper tool at this point.