**Integration with Detector Software**

**Implementation (Ideal):**

* Supply a CBF to the detector "device driver" prior to taking an image. CBF would be what is normally thought of as "header" information with possible thumbnails.

* Supply a CBF to the detector "device driver" after taking an image. Again, "header", including information only obtainable during or after an image, plus possible thumbnails.

* Merge the above two CBFs, add detector device driver supplied information, and add the output image(s), and output the entire CBF to disk.

**CBF Inputs to Detector Control Software**

**Integration with Detector Software**

**Implementation (Practical):**

* CBFs supplied to the detector device driver should contain all experimental parameters, including goniostat information, beam center, and detector setting information which is required to compute fundamental header information.

* Detector device driver should not need to know much if any of the information passed to it via the input CBF (header). Device driver will add a small number of header items to input CBF.

* It would be sufficient to just pass the "second" CBF, after the image has been taken, to the detector device driver.

**Who Has to Create the CBF Header Information?**

**Ideally: Site Scientific Staff**

* Needed: REALLY well documented tools for creating, modifying, and verifying CBF headers. Release 0.7.7 is a big step forward.

* Needed: Real examples encompassing "standard" beam line configurations. I need to do two or three of these.

**Practically: Detector Vendors**

* Needed: Some two items as above. We should be able to take examples of similar beam line configurations and quickly come up with proper headers.

* Note: The more "standard" examples which can be shown, the

**Programming Issues**

**CBFlib_0.7.7:**

* Includes many more programming examples. Needs a few descriptive paragraphs for some of the examples.

* Next release: Contain some site specific examples which have had data images converted, processed, and verified for correctness.

**Using CBFlib code:**

* As much as possible use common routines such as cbf_simple.c

* Error reporting is much improved. I think the current "local_exit" scheme will actually work best with cbf_simple.
**Programming Issues**

**Beam Center:**

* It is easy to get this wrong, and difficult to decide when it is right, or so it seems.

* Conversion examples in CBFLib depend on input beam centers which themselves are subject to “interpretation”.

* Verification of beam center via data processing depends on processing program’s interpretation; “double errors” can lead one into a false sense that your CBF beam center is right.

* What is the status of additional data processing programs taking CBF files as input?

**Verification**

CBF Distribution should always contain:

* Verification programs for CBF file integrity.

* Comparison with standard dictionary entries. We don’t want to preclude additions to headers which are useful, but there should be an automated way to highlight the non-standard ones found in CBFs being produced. This is an excellent way to discover which dictionary items are being reproduced in a different way (for example, two definitions of “distance”).