Calibration constants tables

- Calibration constants tables are defined in the /calibration directory of the CLAS12 ccdb database.
- Tables of the CLAS12 detector should be created in a subdirectory of /calibration
- The structure of calibration constant tables is defined by detector groups using the following convention:
    - tables will always contain three columns, namely "sector", "layer", "component" to identify detector elements.
    - Sector and layer numbers should start from 1; for example for forward, sector-based detectors (DC, LTCC, FTOF, EC) sector will go from 1 to 6; for FTOF layer will go from 1 to 3. For non-sector-based detectors, e.g. CTOF, sector will be 1 and layer will be 1.
    - for calibration constants that do not refer to an individual detector element, for instance the overall timing offset of the detector, sector, layer and component should be set to 0; in more detail:
        - for constants that are related to individual counters, we would have:

sector	layer	component	constant
1	1	3	0.4
1	1	4	0.2
1	1	5	0.1
1	1	6	0.7

        - for layer related constants:

sector	layer	component	constant
1	1	0	1.4
1	2	0	1.2
1	3	0	1.1
1	4	0	1.7

        - for sector related:

sector	layer	component	constant
1	0	0	5.4
2	0	0	5.2
3	0	0	5.1
4	0	0	5.7

    - in defining the table structure, one should remember that:
        - once created, the table structure cannot be altered for example adding columns or rows;
        - when filling tables with values, the whole table needs to be filled at once.

DAQ parameters tables

- Information such as translation tables and fADC parameters should be stored in the directory /daq of the CLAS12 ccdb database.
- Translation tables should be created under /daq/tt (see for example /daq/tt/ec) with the following format:

Columns info

N.	(type)	: (name)
0	int	: crate
1	int	: slot
2	int	: channel
3	int	: sector
4	int	: layer
5	int	: component
6	int	: order

(see http://clasweb.jlab.org/clas12offline/docs/software/html/io/readingRawEvioFiles.html for more information about the translation table format)

- FADC parameters should be created under /daq/fadc (see for example /daq/fadc/ec) with the following format:

Columns info

N.	(type)	: (name)
0	int	: crate
1	int	: slot
2	int	: chan
3	double	: pedestal
4	int	: nsb
5	int	: nsa
6	int	: tet

Creating and filling tables:

- Once the table structure is defined, detector groups can create and fill their tables using either the "test" directory of the database or an sqlite copy of the database to avoid affecting other people work while testing codes. Instructions on how to use the database are available at https://clasweb.jlab.org/wiki/index.php/CLAS12_Constants_Database
- To insert tables in the main calibration database, detector groups should get in contact with Harut and Bryan. Since the main calibration database is what is used by simulations and reconstruction, relevant changes to existing tables should ALWAYS be coordinated with Harut and Bryan.

Use of variations and local sqlite copies:

- Database "variations" will be used equivalently to CLAS RunIndexes.
- The "default" variation of the calibration database is the default used by reconstruction.
- "Locked" variations, which can be modified only by authorized people, will be used equivalently to CLAS run group RunIndexes.
- Other variations can be created for specific tests or studies:
    - KPP studies and “calibration run”: for Monte Carlo studies with “realistic” (mimicking the level of accuracy we will have at the beginning of data taking) or “distorted” calibrations, specific variations will be created in coordination with Harut and Bryan;
- To allow using different variations for different detectors when running simulations, future versions of gemc should allow selecting both run number and variation via gcards.
- The database can be "cloned" into a local SQLITE file:
    - for test and debugging of specific detector software, calibrators are encouraged to use sqlite copies of the database rather then variations in the main database.
    - future releases of gemc should include an sqlite copy of the database to allow running simulations when no network is available or the database is not accessible.

Run ranges

- Calibration constants should be defined from run numbers from 1 to inf.
- First run numbers will be used in the CCDB for specific Monte-Carlo studies, as was done in CLAS.
    - Different run numbers from 1 to 100 will be used to store constants to simulate different detector responses, from the ideal detector (run 1) to the realistic one.
    - Geometry constants for run 1 to 10 should be selected to have no misalignments; misalignments should be inserted, if necessary, should be inserted for runs after 10.
- The proposed guidelines for usage of CCDB run/variation for Monte Carlo studies are detailed in the next paragraph.
- Detector groups should define constant values for Run 1 to X as described and enter them in the database.
- Final MC simulations for physics analysis should be done using the run number range of the data set under analysis.

Proposed usage of CCDB run/variation for Monte-Carlo runs

Presumably physics runs will not begin with Run 1 but (for example) Run 1000. This leaves space in the CCDB for run numbers which can be assigned to Monte-Carlo runs used to evaluate reconstruction and calibration suites, and to anticipate quality of data needed to meet KPP and physics running performance goals. This may require a variety of runs to simulate the CLAS 12 detector in various states of readiness. Here is proposed a simple set of guidelines in organizing the CCDB constants used for these studies:

- Runs define different sets of conditions.
- Variations define different parameter values within a specified condition.

The general idea is a gradual transition from perfect to realistic detector in steps defined by the run number. Variations within a run number are constrained by the condition defined by the run number, but otherwise up to the individual detector groups.

Example of run condition assignments:

Run 1: All parameters equivalent to perfect detector. No TDC smearing, no Poisson fluctuation of ADCs, no light attenuation, no missing channels, no misalignments, etc.
Run 2: Parameters set to design goals for best resolution detectors. No sector, layer, component variation of parameters.
Run 3: Introduce only sector dependence of parameters.
Run 4: Introduce only layer dependence of parameters.
Run 5: Introduce only component dependence of parameters (smooth parameterization).
Run 6: Sector-layer variation only.
Run 7: Layer-component variation only.
Run 8: Sector-component variation only.
Run 9: Sector-layer-component variation.
Run 10: Use realistic calibration-derived parameters intended for physics data replay.

Within each run condition different variations would correspond to different choices of what combination of parameters to vary and by how much. For example, for EC in Run 2 there may be variations with different photoelectron yields (p.e./MeV) and other variations with different attenuation lengths, as each affects the global performance differently. Under Run 5 we may want to check the systematic effect of poor calibration. Under Run 8 we may want one sector with good performance and another sector with poor performance to test the effect on 2 -photon reconstruction for pizeros.

Organizing Monte-Carlo runs this way attempts to separate and hopefully isolate the calibration conditions likely to influence the sort of systematic errors sometimes encountered in CLAS 6.