SVT Conditions System

Omar Moreno
Santa Cruz Institute for Particle Physics
University of California, Santa Cruz
omoreno1@ucsc.edu

November 6, 2014
Heavy Photon Search Software Meeting

Conditions Database Overview

- Currently, all detector conditions are being stored in a MySQL development database at SLAC (on Jeremy's linux box)
- A series of converters are used load conditions records from the database and convert them to Java objects (e.g. `SvtCalibration`)
  - This results in a collection for each of the conditions sets (e.g `SvtCalibrationCollection` has an object for every SVT channel)
  - These conditions collections are further encapsulated by a sub detector conditions object (e.g. `SvtConditions`) in order to simplify access.
- Which versions (`collection_id`) of each of the conditions sets are loaded are determined by a conditions record
  - A conditions record

id	run_start	run_end	updated	created	tag	created_by	notes	name	table_name	collection_id
1	0	0	2014-09-10 17:20:12	2013-09-20 13:19:55	dev	jeremym		svt_calibrations	svt_calibrations	1

    relates a run range and, soon, a run tag to a table name and a collection ID
  - A conditions record per table and conditions set is required except for bad channels

All code can be found in the package `org.hps.conditions`

What SVT Conditions are Currently in the Database

- The following conditions sets for engineering (test) run are currently in the database *contained in both sets
- The SVT DAQ Map
  - Contains FEB ID (FPGA ID), FEB hybrid ID (hybrid ID), SVT half*, layer and side
- SVT "calibrations" i.e. channel baseline and noise
  - Contains pedestal and noise for each of the six samples read out per channel
- Channel gains and offsets
- Shaper fit parameters
  - Amplitude, t_0, t_p and chi^2 (not used) of fit to calibration pulse for each channel
- t_0 shifts
- Alignment constants (may not be used)
- Bad channels
- SVT DAQ configuration
- There is also a table containing a list of channel ID's which denote a specific FEB ID (FPGA ID), FEB hybrid ID (Hybrid ID) and channel

These are not set in stone and will change if necessary. Feedback is appreciated …

Accessing SVT Conditions

- All SVT conditions collections are encapsulated by an `SvtConditions` object, however, this is not the recommended way to access all conditions
- Instead, a setup class (`SvtDetectorSetup/TestRunSvtDetectorSetup`) is used to load all conditions, except for alignment constants and configurations, into sensor objects (`HpsSiSensor/TestRunSiSensor`) which are part of the geometry
  - `HpsTestRunSiSensor` is a subclass of `HpsSiSensor` which is a subclass of `SiSensor` -- Done for backwards compatibility with the test run
- The sensor objects also give access to the sensor layer, orientation (axial/stereo), detector volume (top/bottom) and it's identifier in the actual system i.e. FEB ID (FPGA ID), FEB hybrid ID (hybrid ID)
- The details of these classes can be found in the code by looking in the lcsim package

  `org.lcsim.detector.tracker.silicon`

- Adding the drivers `ConditionsDriver` and `TestRunConditionsDriver` to the beginning of your steering file will load the conditions and will setup all sensor objects (See the readout/recon steering files for some examples)
- Note: The driver `SvtSensorSetup` also needs to be added to the chain of drivers, otherwise, things will crash.

Typical Use Case

```
public class SvtDummyAnalysis extends Driver {

    private List<HpsSiSensor> sensors = null;
    private String rawHitCollectionName = "SVTRawTrackerHits";
    private static final String SVT_SUBDETECTOR_NAME = "Tracker";

    @Override
    protected void detectorChanged(Detector detector) {
        // Get the collection of all sensors from the geometry
        sensors = detector.getSubdetector(SVT_SUBDETECTOR_NAME).getDetectorElement().findDescendants(HpsSiSensor.class);
    }

    @Override
    public void process(EventHeader event) {

        // Get the RawTrackerHits from the event
        List<RawTrackerHit> rawHits = event.get(RawTrackerHit.class, rawHitCollectionName);

        for(HpsSiSensor sensor : sensors){
            sensor.getT0Shift();
            for(int channelN = 0; channelN < 640; channelN++){
                for(int sampleN = 0; sampleN < 6; sampleN++){
                    sensor.getNoise(channel, sampleN);
                    // Do Something …
                }
            }
        }

        // Loop over all fitted hits
        for(RawTrackerHit rawHit : rawHits) {

            HpsSiSensor sensor = (HpsSiSensor) rawHit.getDetectorElement(); // Or cast ot HpsTestRunSiSensor
            int channel = rawHit.getIdentifierFieldValue("strip");

           for(int sampleN = 0; sampleN < 6; sampleN++){
               double pedestal = sensor.getPedestal(channel, sampleN);
               double noise = sensor.getNoise(channel, sampleN);
               // Do something ...
           }
        }
    }
```

Writing Conditions to the Database

- Currently, all SVT calibration code has already been written in C++ with no plans to port to Java.
- The plan is then to dump all extracted calibrations (baseline, noise, gain and offset) to a file in XML format which would then be parsed and loaded into the conditions database using the existing Java API (`SvtCalibrationsLoader`)
  - Work on the loader is in progress
- Before conditions are loaded, a user will need to verify that all conditions look reasonable e.g. check baseline and noise vs golden baseline and noise
  - Still debating whether the monitoring app will be used to do this

```
<?xml version="1.0"?>
    <system id=”SVT”>
        <FEB id=”0”>
           <Hybrid id=”0”>
               <channel id=”0”>
                   <baseline sample=”0”>3000</baseline>
                   <noise sample=”0”>50</noise>
                   <baseline sample=”1”>3000</baseline>
                   <noise sample=”1”>50</noise>
                   <baseline sample=”2”>3000</baseline>
                   <noise sample=”2”>50</noise>
                  <baseline sample=”3”>3000</baseline>
                   <noise sample=”3”>50</noise>
                  <baseline sample=”4”>3000</baseline>
                   <noise sample=”4”>50</noise>
                  <baseline sample=”5”>3000</baseline>
                   <noise sample=”5”>50</noise>
               </channel>
           </Hybrid>
        </FEB>
    </system>
```

To Do …

- Currently, running the readout and recon require the use of an internet connection. A user can connect to a local copy of the database, but this requires a user to clone the database. Can this be simplified? (Jeremy)
- The use of tags to distinguish multiple conditions sets with the same run number needs to be verified to work (Jeremy/Omar)
- The EVIO to LCIO conversion of SVT data needs to be updated (Omar)
  - The new data format is already known, so it's just a matter of putting in the work
- All test run conditions need to be loaded into the database (Omar)

Warning! The flat file conditions drivers will be deleted on Monday, 11/10/2014. If you haven't tried using the new conditions system by then, be aware that your analysis drivers may break.