Online Software Intro

Murrough Landon

7 January 2016

- Aims
- TDAQ Run Control
- L1Calo Introduction
- L1Topo Experience
- Tools
- Summary

Aims of the Meeting

- Launch regular online software discussions
- Reminder of (or introduction to) existing L1Calo SW
- Find out what we have done separately so far
- Start to integrate developments
  - In as common a way as possible
- Discuss next steps…

Common Project Lifecycle

- Early days
  - Many experts
  - Lots of time and enthusiasm
  - "Our component is special and we need something different"
- Years later…
  - Few experts
  - Less time, less knowledge
  - "Why is everything done in completely different ways and where is the documentation?!"

TDAQ Run Control (1)

- Run Control Task
  - Get ATLAS into data taking state
    - From any other condition, eg immediately after a run or after power cut
- Run Control State Model
  - INITIALIZE: start processes (1000s), no HW access
  - CONFIGURE: get individual components to working state
  - CONNECT: link configured components together
  - START: final setup for new data taking run
  - STOP etc: reverse the process (actually needs more steps)
- Run Control Tree
  - Hierarchy of controllers from RootControllers to segments for each detector and down to individual applications
    - Eg one L1Calo run control application per crate
  - Parent controllers propagate state transitions to children

TDAQ Run Control (2)

- CONFIGURE: for each L1Calo module
  - Ensure correct firmware is (re)loaded
  - Setup clocks => output links stable
  - Configure inputs (but cannot yet rely on them)
  - Load other settings: menu, calibration, etc
  - Load test vectors if required
- CONNECT: for each L1Calo module
  - Try to lock input links, clear errors
  - Anything else that depends on other modules being configured
  - Start regular status publication
- START (prepareForRun) step: for each L1Calo module
  - RODs: set new run number
  - All: clear error counters

L1Calo Introduction

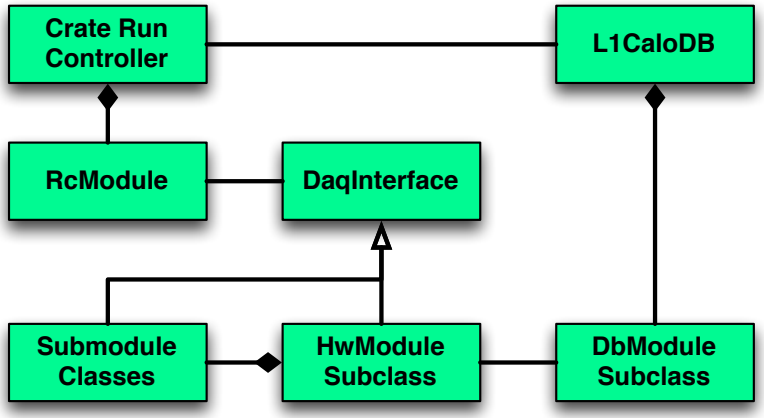
- L1Calo Online Software Functionality (Runs 1 & 2)
  - Configure VME (and IPbus) modules for data taking
    - Under the TDAQ run control framework: one run controller per crate
  - Detailed hardware simulation of each module
    - Including generation of test vectors and L1A patterns
  - Common database access for hardware and simulation
    - TDAQ "OKS" database, Calibration/Conditions (COOL), Trigger Menu
  - Diagnostic and basic monitoring tools
    - NB sophisticated event monitoring done via Athena (P1/Tier0)
  - Test and calibration programs
- Documentation
  - Online Software section on main L1Calo twiki page
  - Various notes, incomplete and often extremely out dated: could do better!
  - Overview paper
    - indico.cern.ch/getFile.py/access?contribId=41&sessionId=32&resId=0&materialId=paper&confId=21985

L1Calo Packages

- For each module type XXX
  - xxxControl: IPbus module hardware configuration
    - NB for VME modules we used xxxServices
  - xxxSim: detailed hardware simulation
  - xxxTests: test/calibrations programs (if needed)
- Large number of other base/support/etc packages

L1Calo Hardware Classes

- One hardware class per module
  - Implementing a common interface (used by run controller)
  - Classes for subcomponents (also using the same interface)
  - If I understand what Giordan has done for the Zynq this model might also be OK for gFEX - at least the run ctrl side?
- Instance of DbModule subclass per HW module
- One run ctrl application handles all modules in a crate
  - Not required with IPbus, but probably still sensible?

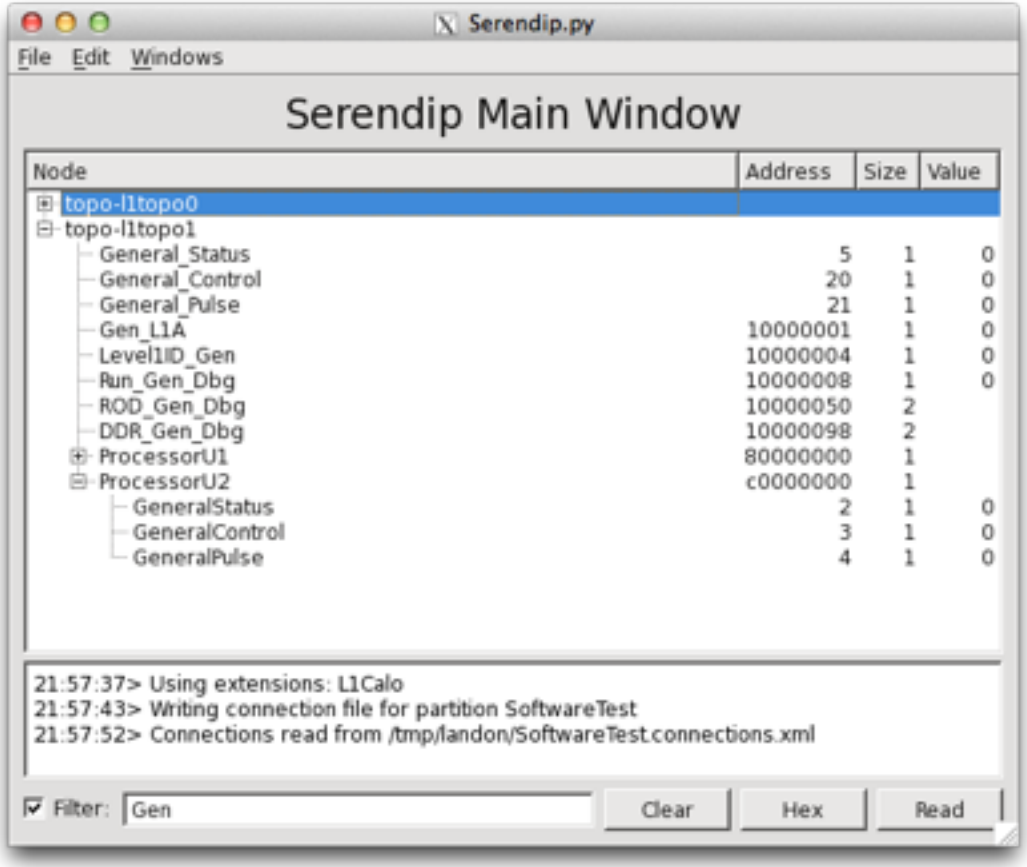


L1Topo Experience

- L1Calo SW framework developed with VME modules
- Run 2: added L1Topo as first ATCA/IPbus module
- Easiest to just add it to the existing framework
  - Crate control application sees both VME and IPbus modules as conforming to a common interface
    - Can imagine separate evolution in future if needed
- Run 1: each module had dedicated simulation package
  - For L1Topo wrapped Athena-free offline package in online simulation framework
  - But we still havent recovered the nice simulation framework we found extremely useful before and during Run 1
  - Not yet sure of the best strategy for the FEXes

Tools: Serendip

- Homegrown GUI for IPbus
  - I wanted something similar to what we had for VME (hdmc)
  - Original CMS GUI used widget set not on lxplus/LCG
    - And with a somewhat different philosophy
  - Main window and windows for registers, memories, also module windows for custom actions per module, etc



- See https://indico.cern.ch/event/368312/contribution/0/attachments/732332/1004773/l1calo_0127_serendip.pdf

Summary So Far

- Whats been done in software (that I know of)
  - Minimal support for all new modules in OKS database
  - eFEX, jFEX, FTM: basic packages in SVN
  - eFEX, jFEX: started implementing standard package
  - FTM: mainly python test scripts
  - HUB, ROD: not much yet?
  - gFEX: package to bridge Zynq to external IPbus
- Whats not been done (that I know of)
  - Common response to Weimings suggestion of common parts of IPbus address space?
  - All modules have minipods and MGTs - can we view these from software in a common way on all modules?

Questions?

- How do we plan to use the Zynq?
  - Eg Run Ctrl running remotely via IPbus bridge but also other applications? Or something completely different?
- Integration of Hub & Rod?
  - Standard model: Hub module with Rod subcomponent?
  - Or something a bit different?
- Approach for test vectors and simulation?
  - Internally and with LAr