ATLAS Level-1 Calorimeter Trigger: Control, Test and Monitoring Software

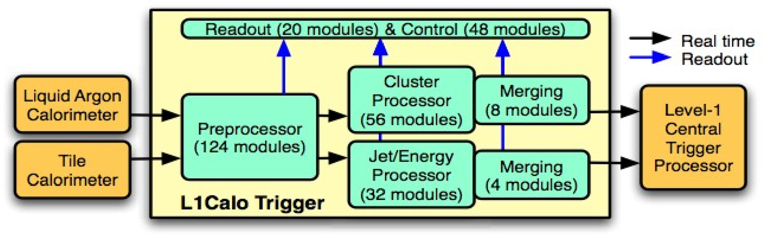
University of Birmingham – University of Heidelberg – University of Mainz – Queen Mary, University of London – STFC Rutherford Appleton Laboratory – Stockholm University

Presented by Murrough Landon (QMUL)

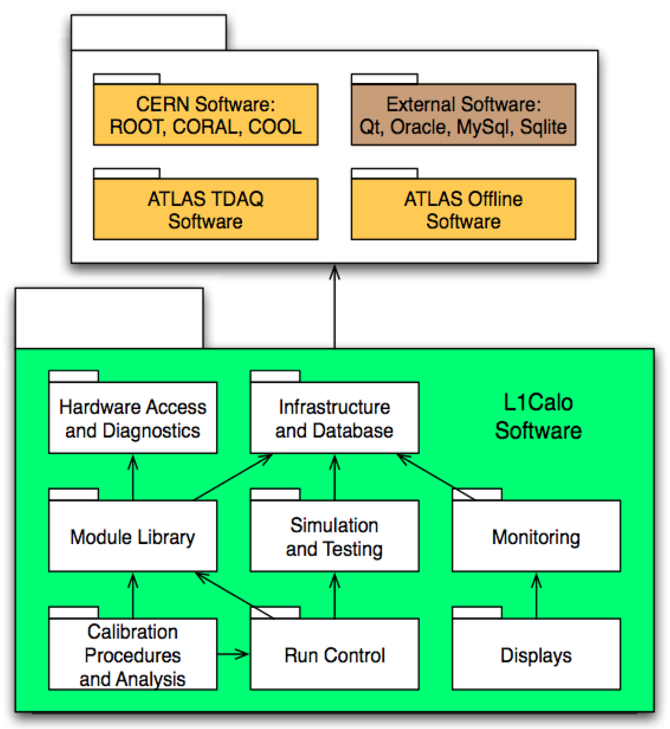
Level-1 Calorimeter Trigger (L1Calo) Overview

L1Calo installation near completion in the ATLAS electronics cavern

- The ATLAS level-1 calorimeter trigger (L1Calo) is a hardware based system comprising nearly 300 VME modules of about 10 different types each with its own programming model.
- The real time path, sketched in the diagram below, is divided into three processors, each of which is read out to the ATLAS data acquisition system:
  - The Preprocessor (PPr) digitizes analogue signals from the calorimeters
  - The Cluster Processor (CP) identifies electrons/photons and hadrons
  - The Jet/Energy Processor (JEP) finds jets and the total and missing Et
- The L1Calo trigger is housed in 17 crates each with its own CPU and timing infrastructure. This distributed computing system can be operated standalone for tests, together with the calorimeters for calibrations and as part of the whole ATLAS system for normal data taking.
- This large and complex system requires a substantial suite of software to control, configure, test and monitor it which is the subject of this poster.
- Other talks at the conference will cover the status of the L1Calo installation, performance and calibration.

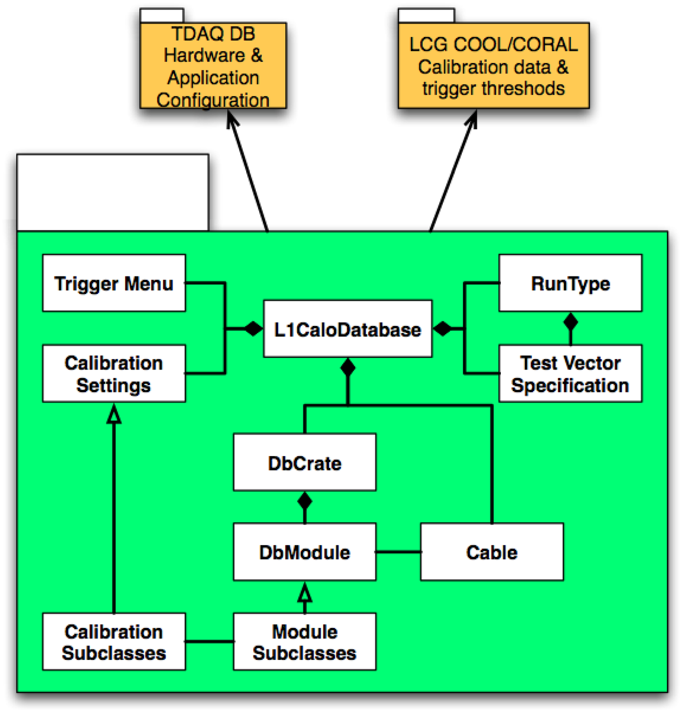


L1Calo Software Architecture



- The L1Calo online software is designed to control, test and monitor any configurable subset of the trigger system, from a single module under test to the complete installation at ATLAS.
- It consists of about 75 software packages in CVS, built together using the standard ATLAS code management tool (CMT).
- These packages are grouped into about eight main categories whose internal and external dependencies are shown in the diagram:
  - Infrastructure packages define basic classes, tools and interfaces to external software.
  - Database packages provide a uniform interface between various ATLAS databases and the higher layers of L1Calo software.
  - Hardware access packages encapsulate the detailed programming models of our modules and provide diagnostic tools.
  - Module libraries implement the higher level functions required to operate the modules under the ATLAS run control.
  - Each type of module also has a detailed hardware simulation package
  - Another group of packages handles the interaction with ATLAS run control and other distributed services.
  - The main module types in the three processors also have dedicated packages for calibrations and other tests.
  - Monitoring packages analyse event data and status information and publish histograms to the online histogramming service.
  - There are also packages for various graphical tools and displays.

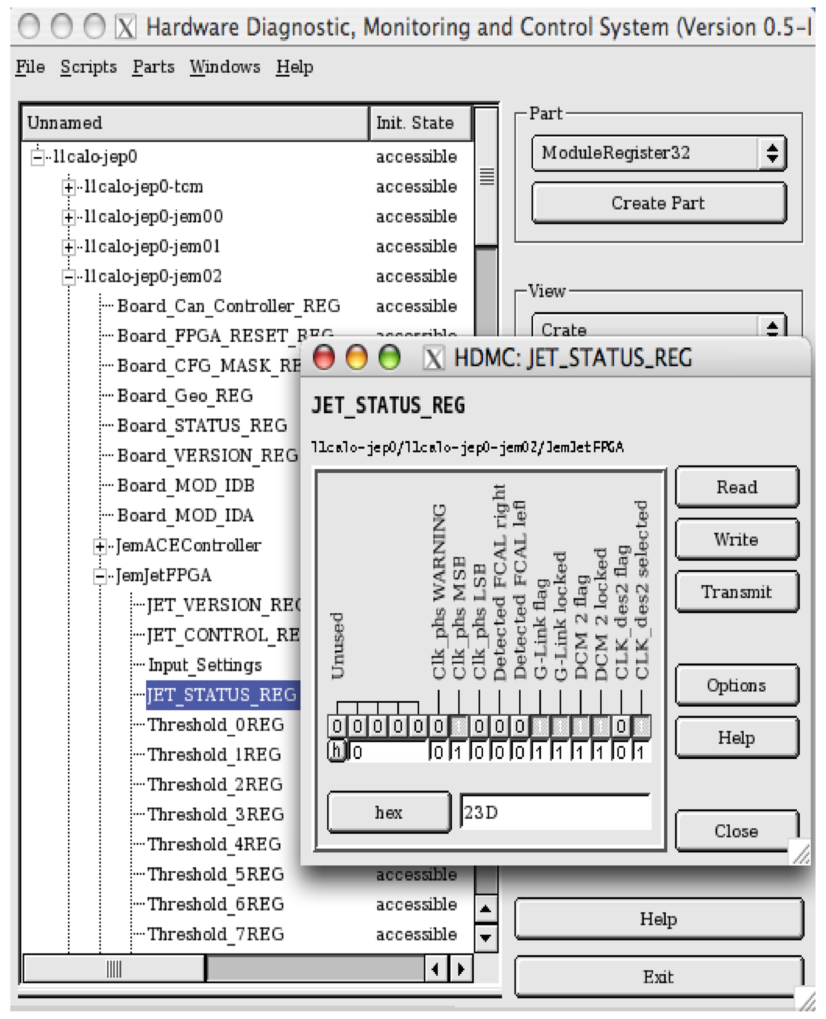
Databases



The main L1Calo database classes

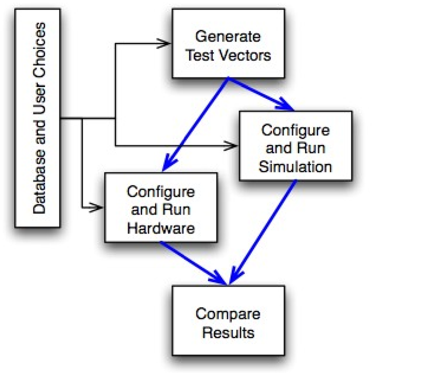
- The L1Calo software needs access to several different databases used in ATLAS:
  - The TDAQ database describes the hardware and software configuration that is available for data taking. The hardware configuration includes the crates, modules and cables connecting them. This database also contains sets of L1Calo "run types" with the specifications of the test vector configuration to be used for each type of test run.
  - All calibration values and most configuration settings used to load the L1Calo modules are stored in a COOL database which provides "interval of validity" history of the settings.
  - The trigger thresholds are taken from the ATLAS trigger configuration database, a purely relational database.
  - Volatile information for the current run is also read from the TDAQ distributed information service (IS).
- L1Calo database packages provide read (and where required, write) access to these databases.
- The details are encapsulated and each type of module in the system has its own DbModule subclass that provides it with the view of the data it requires.
- In addition to custom code for L1Calo, we have also developed a browser and "editor", ACE, for the LCG COOL database. This is now distributed as part of the LCG software.

Hardware Access and Diagnostics

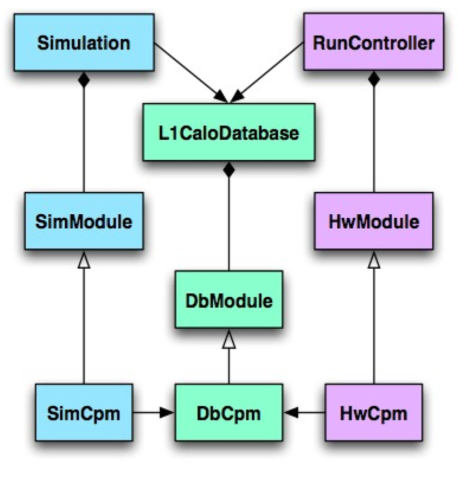


- The design philosophy of our hardware access packages is to simultaneously:
  - Provide complete access to our VME modules for debugging.
  - Implement a well defined access for higher level code.
- Each type of module in the system has its own programming model, ie the sets of registers and memories at the level of the module and its component submodules, some of which may have their own substructure.
- The hardware access packages provided complete descriptions of the VME address structure of each module together with bit field formats of each register and memory type.
- These descriptions, stored in configuration files, are used dynamically in a graphical diagnostic program HDMC (see figure) for debugging down to level of individual register bits.
- HDMC reads the hardware configuration from the TDAQ database so it can show a complete view of all the modules in one crate.
- A code generator uses the configuration files to create classes for use by higher level code.
- This layered approach means that
  - Some access checks can be policed by the compiler – only registers and bits declared in the configuration file can be accessed.
  - Also some common run time checks can be implemented at a low level.
  - Restrictions on higher level code are not imposed at the expert debugging level.

Simulation and Testing

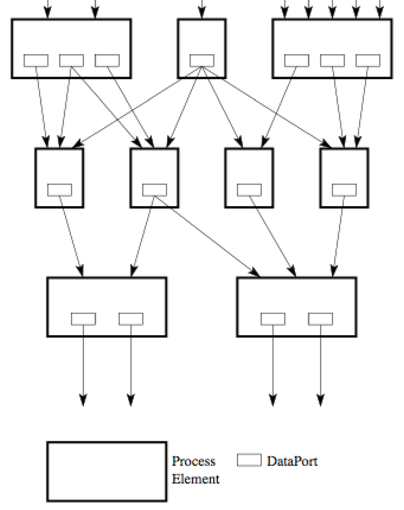


Generic test using simulation



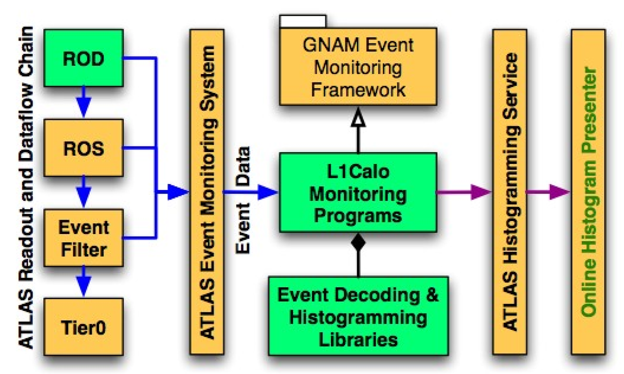
Simulation, Database & Hardware showing the set of subclasses for the Cluster Processor Module (CPM)

- The L1Calo project has evolved through phases of "demonstrator" and prototype development, preproduction and production testing to final installation and commissioning in the ATLAS cavern.
- All those phases require the ability to perform tests on any subset of the trigger system, using arbitrary test vectors.
- This requirement was met by providing hardware and software support:
- All modules in the real time path provide both playback and spy memories to feed generated data into the system and capture the results and any point in the digital pipeline
- A detailed simulation of the system, down to the bit level, was written using VHDL-like software components (processes and ports) that can be connected together to simulate any module or collection of modules.
- The diagram on the right shows how part of the system may be connected.
- Any "process" box may itself be a container for a complete set of lower level processes and port connections.
- The simulation, like the hardware, is configured from the TDAQ database.
- A generic test is performed by the user selecting suitable test vectors, where to load them and from where to capture the outputs.
- Bit by bit comparison of the results either verifies the correct operation of the system or else helps to pinpoint errors.
- The test vectors used can range from simple, complex or random patterns to events from offline simulation (or soon from real Physics data).



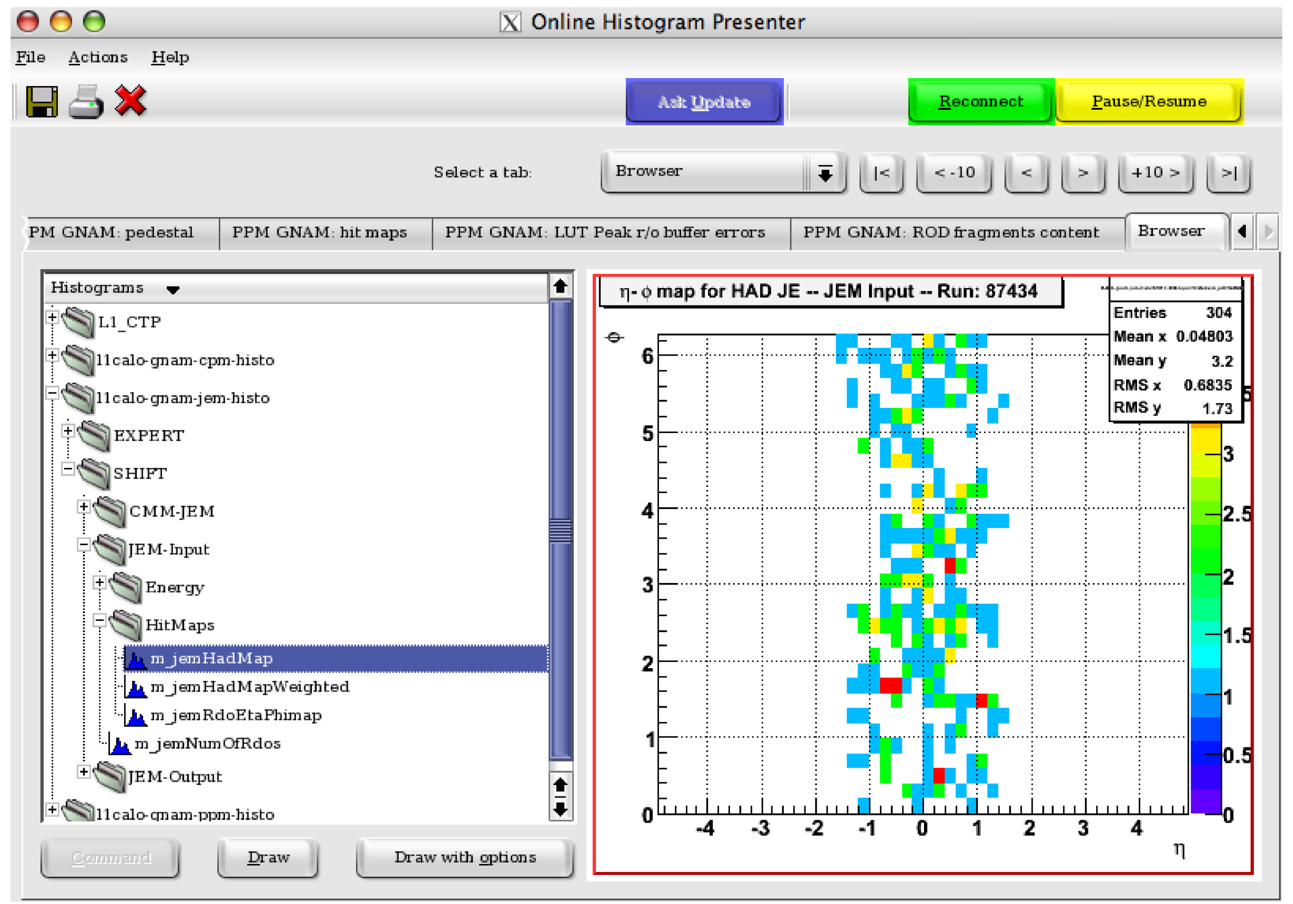
Simulation processes, ports and their interconnections

Monitoring



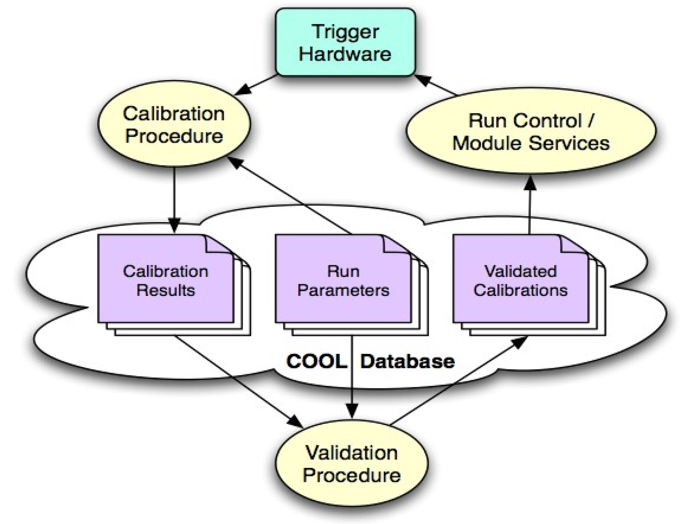
The ATLAS Monitoring Architecture

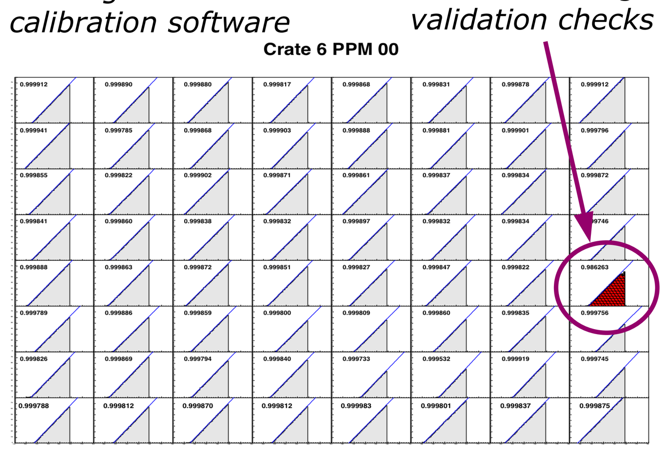
- The ATLAS TDAQ monitoring framework allows events or even fragments to be collected from any stage of the readout and dataflow chain.
- Monitoring clients can obtain events from the monitoring system, decode them and fill histograms that can be published on TDAQ histogramming servers.
- Any histogram published to any server may then be displayed to the user via a histogram browser, such as the Online Histogram Presenter (OHP) displayed here.



The online histogram presenter (OHP), a standard component of the ATLAS TDAQ software, was co-developed by a member of the L1Calo collaboration

Calibration Procedures and Analysis





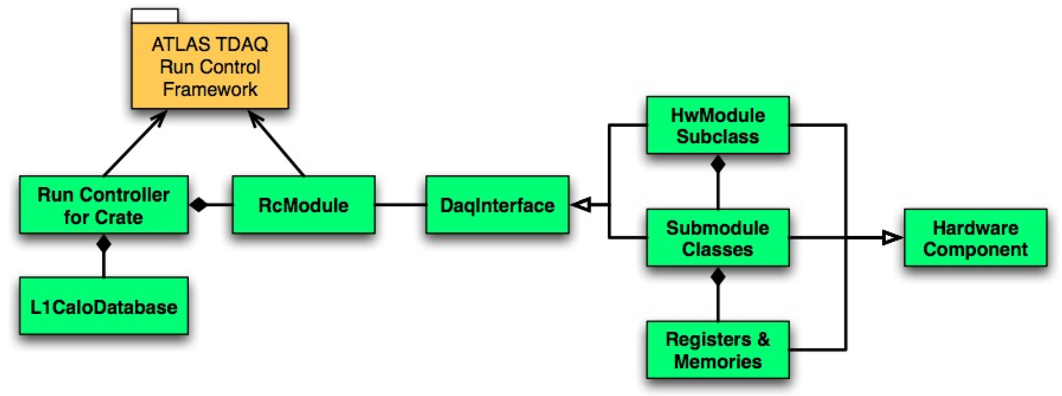
Histograms from calibration software

Channel failing validation checks

- The L1Calo trigger has about 50 configurable parameters for each of its 7200 channels. Many of these are configuration choices, but the majority must be determined from calibrations.
- The general procedure for performing a calibration is to configure the system as normal using the run control and module libraries, then execute a number of steps changing one selected parameter at each step.
- The operation mode ("run parameters") of each type calibration is defined in the COOL database.
- The timing parameters for the digital processors (CP and JEP) are determined by scanning clock phases and counting parity errors via VME.
- However the analogue parameters (pedestals, FADC strobe phase, filter coefficients, latency delays, noise cuts, etc) require data to be read out via the normal ATLAS DAQ path.
- The data is analysed and the results of each calibration are stored to the COOL database.
- A separate validation procedure checks the quality of the calibration.
- A calibration may be marked as "validated" for use in the next run if it passes the checks and if the new calibration constants are significantly different from the previous set.

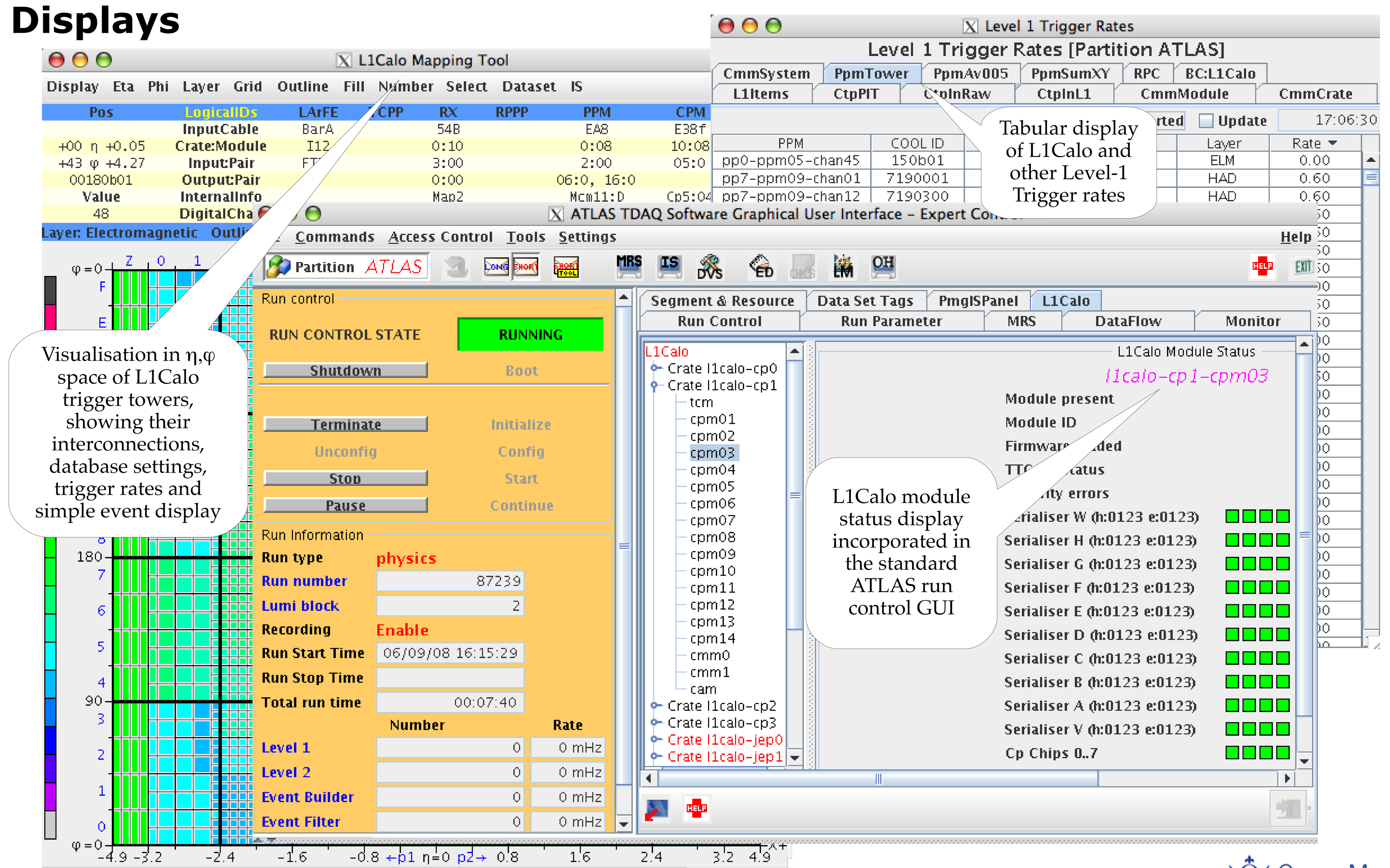
Run Control and Module Services

- The normal operation of the L1Calo trigger is handled under the ATLAS TDAQ run control framework.
- This provides initialisation of the distributed environment with numerous information, histogramming, monitoring and other services.
- Configuration and periodic status monitoring of L1Calo and other ATLAS subsystems is carried out via synchronous run control commands, most of which result in state transitions.
- Under the run control framework, each ATLAS component to be configured is controlled by a run control application. In the L1Calo system, there is one such application per VME crate. This is responsible for loading and monitoring all the modules in that crate.
- To insulate the module libraries from the TDAQ services, the functions required of each module are split into two distinct classes:
  - The RcModule class is completely generic and acts as a façade for any hardware module subclass. Together with its parent run controller object, it is responsible for accessing the database, responding to run control commands and publishing the status, trigger rates and (soon) onboard histograms to the corresonding TDAQ servers.
  - HwModule subclasses are responsible for configuring one type of module via VME and collecting data from it.
- This split has proved useful in hiding changes in the TDAQ API from the bulk of the hardware access libraries.
- The information published by the run control packages is available for display via run control panels and other tools (see next panel).



Overview of run control classes and aspects of module and hardware access libraries

Displays



Queen Mary
University of London