Control, Test and Monitoring Software Framework for the ATLAS Level-1 Calorimeter Trigger

R. Achenbach[b], P. Adragna[d], M. Aharrouche[c], V. Andrei[b], B. Åsman[f], B.M. Barnett[e], B. Bauss[c], M. Bendel[c], C. Bohm[f], J.R.A. Booth[a], J. Bracinik[a], I.P. Brawn[e], D.G. Charlton[a], J.T. Childers[b], N.J. Collins[a], C.J. Curtis[a], A.O. Davis[e], S. Eckweiler[c], E. Eisenhandler[d], P.J.W. Faulkner[a], J. Fleckner[c], F. Föhlisch[b], C.N.P. Gee[e], A.R. Gillman[e], C. Göringer[c], M. Groll[c], D.R. Hadley[a], P. Hanke[b], S. Hellman[f], A. Hidvégi[f], S.J. Hillier[a], M. Johansen[f], E.-E. Kluge[b], T. Kuhl[c], M. Landon[d], V. Lendermann[b], J.N. Lilley[a], K. Mahboubi[b], G. Mahout[a], K. Meier[b], R.P. Middleton[e], T. Moa[f], J.D. Morris[d], F. Müller[b], A. Neusiedl[c], C. Ohm[f], B. Oltmann[c], V.J.O. Perera[e], D.P.F. Prieur[e], W. Qian[e], S. Rieke[c], F. Rühr[b], D.P.C. Sankey[e], U. Schäfer[c], K. Schmitt[b], H.-C. Schultz-Coulon[b], S. Silverstein[f], J. Sjölin[f], R.J. Staley[a], R. Stamen[b], M.C. Stockton[a], C.L.A. Tan[a], S. Tapprogge[c], J.P. Thomas[a], P.D. Thompson[a], P.M. Watkins[a], A. Watson[a], P. Weber[b], M. Wessels[b], M. Wildt[c]

[a] School of Physics and Astronomy, University of Birmingham, Birmingham B15 2TT, UK
[b] Kirchhoff-Institut für Physik, University of Heidelberg, D-69120 Heidelberg, Germany
[c] Institut für Physik, University of Mainz, D-55099 Mainz, Germany
[d] Physics Department, Queen Mary, University of London, London E1 4NS, UK
[e] STFC Rutherford Appleton Laboratory, Harwell Science and Innovation Campus, Didcot, Oxon OX11 0QX, UK
[f] Fysikum, Stockholm University, SE-106 91 Stockholm, Sweden

m.p.j.landon@qmul.ac.uk

Abstract

The ATLAS first-level calorimeter trigger is a hardware-based system designed to identify high-p_T jets, electron/photon and tau candidates and to measure total and missing E_T in the ATLAS calorimeters. The complete trigger system consists of over 300 custom designed VME modules of varying complexity. These modules are based around FPGAs or ASICs with many configurable parameters, both to initialize the system with correct calibrations and timings and to allow flexibility in the trigger algorithms. The control, testing and monitoring of these modules requires a comprehensive, but well-designed and modular, software framework, which we will describe in this paper.

I. INTRODUCTION

The ATLAS[1] detector at the CERN Large Hadron Collider (LHC) consists of an inner tracker surrounded by electromagnetic (EM) and hadronic calorimeters enclosed by a muon spectrometer. The calorimeters provide the trigger with just over 7200 analogue signals.

The first-level calorimeter trigger[2] (L1Calo) is a hardware-based system with a high degree of adaptability provided by widespread use of FPGAs to implement the trigger algorithms that identify high-p_T jets, electron/photon and tau candidates and which measure total and missing E_T seen in the calorimeters.

The real-time path of the L1Calo trigger (see figure 1) is subdivided into a Preprocessor which digitizes the analogue signals from the calorimeters, followed by two digital processor systems working in parallel: the Jet/Energy-sum processor (JEP) and the Cluster Processor (CP). The outputs of the digital processor are sent to the ATLAS Central Trigger Processor. All stages of the L1Calo processor chain are read out to the data acquisition for monitoring the operation of the trigger.

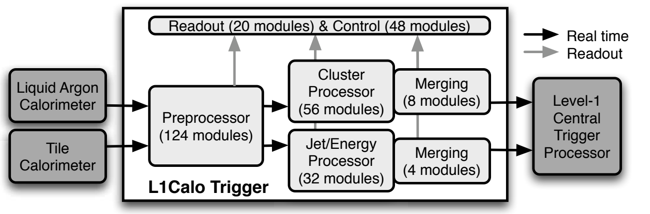

Figure 1: Overview of the L1Calo trigger architecture.

The system comprises over 300 VME modules of about 10 different types, each of which has a unique register and memory map. The most complex of these modules contains around 2000 individually programmable registers, as well as many kilobytes of look-up table memory.

It is clear that the software needed to control a system on this scale needs to be sophisticated enough to manage the different properties of each module, but also modular enough to be maintainable over the long period of commissioning and running of the ATLAS experiment.

There are several distinct areas of software that can be clearly separated, but which must have some means of interaction. For example, the configuration parameters must be stored in a common database framework which is independent of the rest of the software, but many of the other software components (e.g. configuration, monitoring) will need access to this information. The software framework must also fit into the existing ATLAS online software environment to successfully participate in a standard integrated run.

The following sections give an overview of the software architecture and provide details of the design choices and implementations of the main components.

II. L1Calo Online 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. The L1Calo software is primarily written in C++ with some Java libraries included. It consists of about 75 software packages in our code repository (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 figure 2.

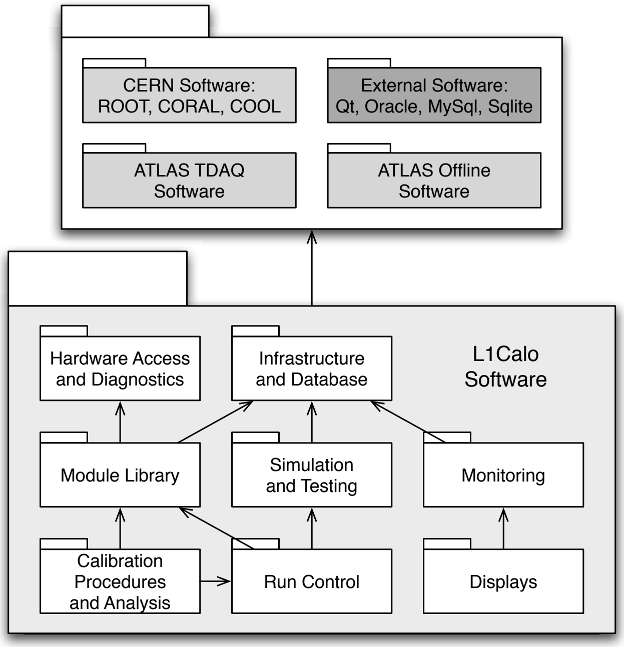

Figure 2: L1Calo online software architecture.

At the lowest level there is a collection of infrastructure packages that define basic classes, tools and interfaces to external software such as the ATLAS Trigger and Data Acquisition (TDAQ) software and the CERN LHC Computing Grid (LCG) packages. Several database related packages provide a uniform interface between the various ATLAS databases and the higher layers of L1Calo software.

One of the most fundamental areas is the set of underlying VME access libraries that encapsulate the detailed programming models of our modules. They were designed so that they could be used in several ways: under the ATLAS run control to configure modules at run start, from standalone programs or via an interactive GUI for expert intervention at the VME register level. Another large body of the software is dedicated to a detailed simulation of the hardware at the level of data that can be probed at each stage of the trigger processing.

A further 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 testing the system and for performing calibrations, both of the signal timing and the energy of the input signals. These are based around a common calibration strategy which extends the run concept to encompass multi-step runs, where parameters are adjusted between each step of the run.

Finally, and most recently, a set of libraries dedicated to monitoring and event-by-event analysis has been developed. These are used to ensure that the trigger is operating correctly during normal data taking, immediately flagging up errors, inconsistencies or merely unusual features to the shift crew. The monitoring area also includes packages for various graphical tools and displays.

III. Databases

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, which is 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 database 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.

IV. Hardware Access and Diagnostics

The design philosophy of our hardware access packages addresses a number of requirements. It needs to provide complete low level access to our VME modules for debugging and it should also implement a well defined access for higher level code. Each type of module in the system has its own programming model, i.e. 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, for debugging down to level of individual register bits. HDMC reads the hardware configuration from the ATLAS TDAQ database so that 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.

In addition to the completely generic HDMC display, there is also a dedicated debugging tool for the preprocessor which uses the higher level access code.

V. Simulation and Testing

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 both 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 at 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.

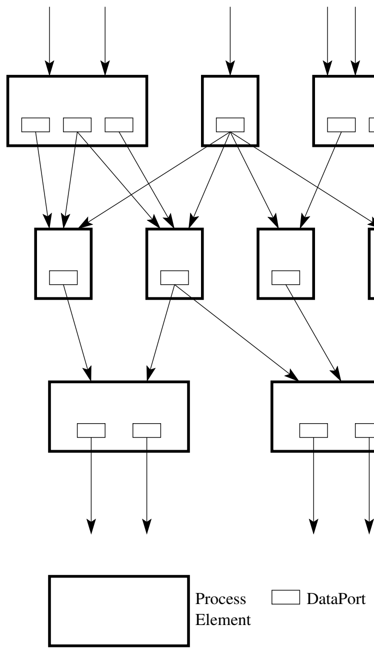

Figure 3: Simulation processes, ports and their interconnections.

Figure 3 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, eventually, from real Physics data.

VI. Monitoring

The ATLAS TDAQ monitoring framework allows whole events or event 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 ATLAS Online Histogram Presenter (OHP) which was co-developed by a member of the L1Calo collaboration.

L1Calo has developed a number of programs that use this framework to monitor the operation of each of the three L1Calo subsystems and of the trigger as a whole. These programs provided a large number of histograms for experts to debug problems and a set of summary plots for the shift crew to monitor the behaviour of the system from run to run.

VII. Calibration Procedures

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 of 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.

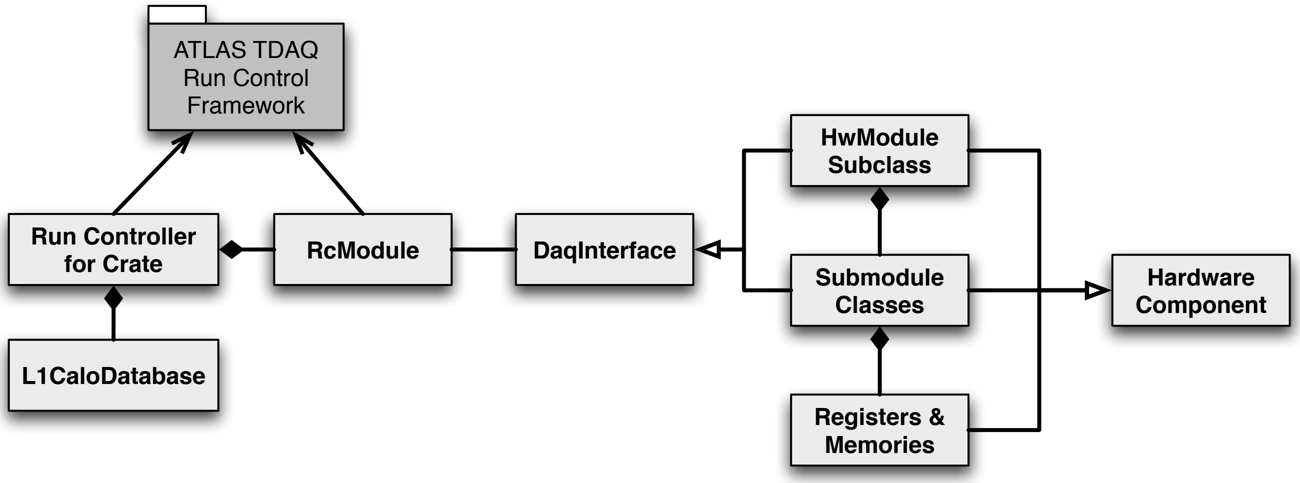

Figure 4: Overview of run control classes and aspects of module and hardware access libraries.

VIII. 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. An overview of the main classes involved is shown in figure 4. 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 corresponding 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.

IX. Displays

A number of graphical displays have been developed in addition to those, such as ACE, HDMC and OHP, which have already been mentioned.

The TDAQ run control GUI allows ATLAS subdetectors to add their own panels. The L1Calo panel in this GUI displays the detailed status of each module in the system in a hierarchical tree view with a colour code to propogate an error state up the tree.

Monitoring of trigger rates at the level of individual towers and for the whole system is crucial. The ATLAS TDAQ software includes a display for the final Level-1 trigger rates. In addition the L1Calo software provides a tabular display of many detailed rates from L1Calo and other parts of the Level-1 trigger.

Finally, there is a visualisation tool displaying the space of L1Calo trigger towers in pseudorapidity and azimuth. This shows the mapping of towers to hardware and cables throughout the system. It can also show database settings, status and trigger rate for each tower and can act as simple event display.

X. Summary

A large body of software has been written for configuring, testing, calibrating and monitoring the ATLAS level-1 calorimeter trigger. This has been successfully used at several stages of the L1Calo project. Initially for testing prototype and production modules, subsequently for the installation and commissioning the final trigger system in ATLAS and most recently for configuring it to trigger on events from the first beam in the LHC.

References

[1] The ATLAS Collaboration, G. Aad *et al.*, The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3 (2008) S08003.

[2] R. Achenbach *et al.*, The ATLAS Level-1 Calorimeter Trigger, JINST 3 (2008) P03001.