Level 1 Calorimeter Trigger Software

Murrough Landon – 13 November 2002

Overview

- Introduction
- Overview of L1Calo software
- Database
- Run Control
- IGUI Panels
- Development environment
- Software Process
- Plans
- Wish list!

Contributors

Past and Present

Many people have been involved in developing software for the calorimeter trigger.

The main contributions so far have come from:

Bruce Barnett, Norman Gee, Steve Hillier, Murrough Landon, Gilles Mahout, Cornelius Schumacher.

Future

Other colleagues are now starting to contribute to our joint efforts.

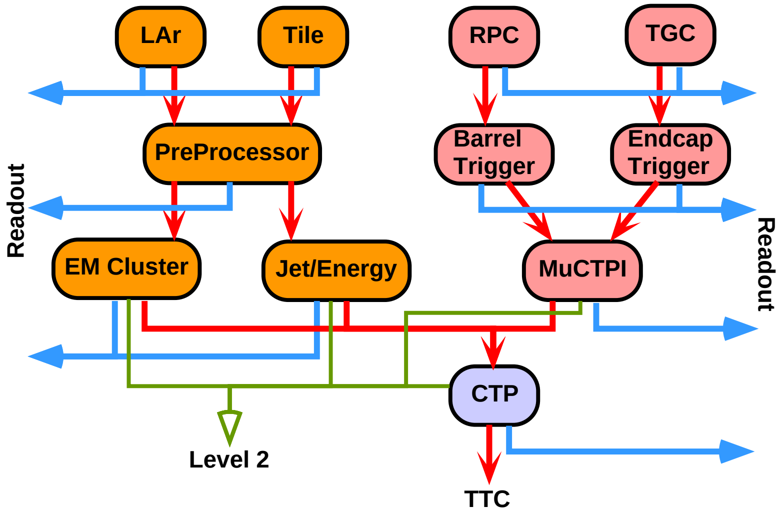
Introduction

Level 1 Trigger

- "Level 1" comprises the calorimeter trigger, the barrel (RPC) and endcap (TGC) muons triggers, the Muon CTP interface and the CTP
- The different subprojects mostly work independently
- The way one "Level 1" subproject uses the Online software is likely to be very different from another subproject (and we probably wont know about it)

- This talks only covers the Level 1 Calorimeter Trigger

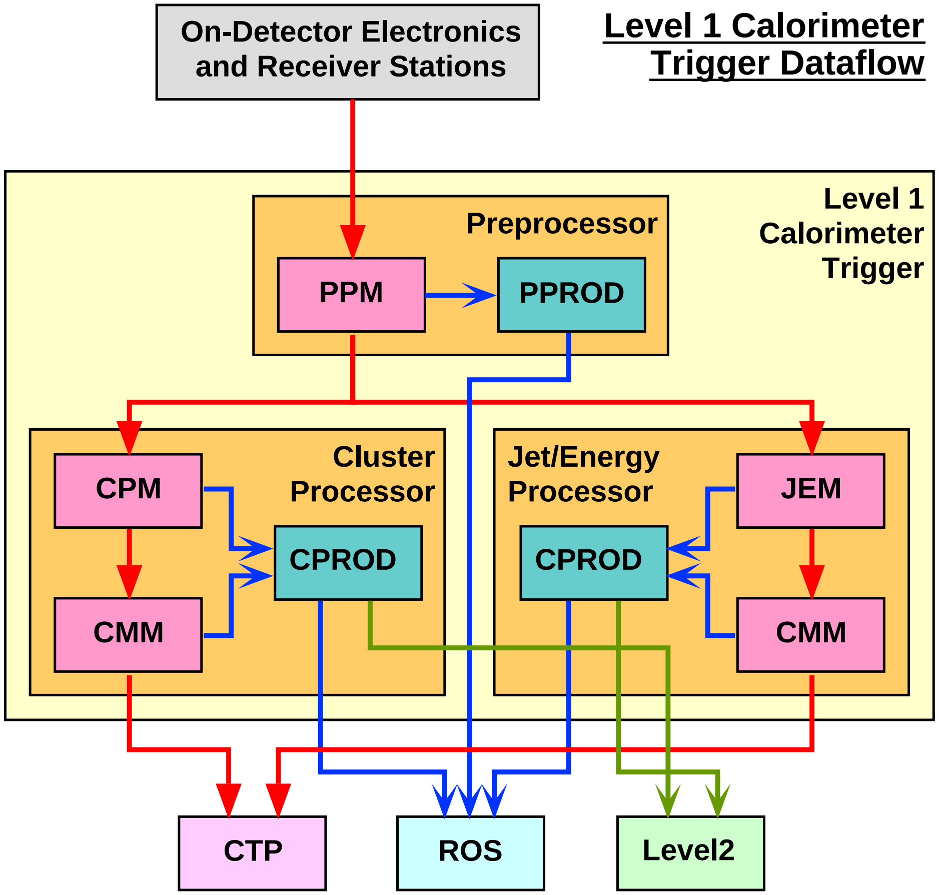
L1 Trigger: Realtime, ROIs, Readout



Overview of the Calorimeter Trigger (1)

Hardware

- Three main subcomponents: preprocessor, cluster processor and jet/energy processor
- Each processor is implemented in custom electronics in a number of 9U crates, some with custom backplanes (not quite VME)
- Each crate is controlled by a CPU and will run a variant of ROD crate DAQ – but not all our crates are ROD crates


On-Detector Electronics and Receiver Stations
Level 1 Calorimeter Trigger Dataflow
Level 1 Calorimeter Trigger
Preprocessor
PPM
PPROD
Cluster Processor
CPM
CPROD
CMM
Jet/Energy Processor
JEM
CPROD
CMM
CTP
ROS
Level2

Overview of the Calorimeter Trigger (2)

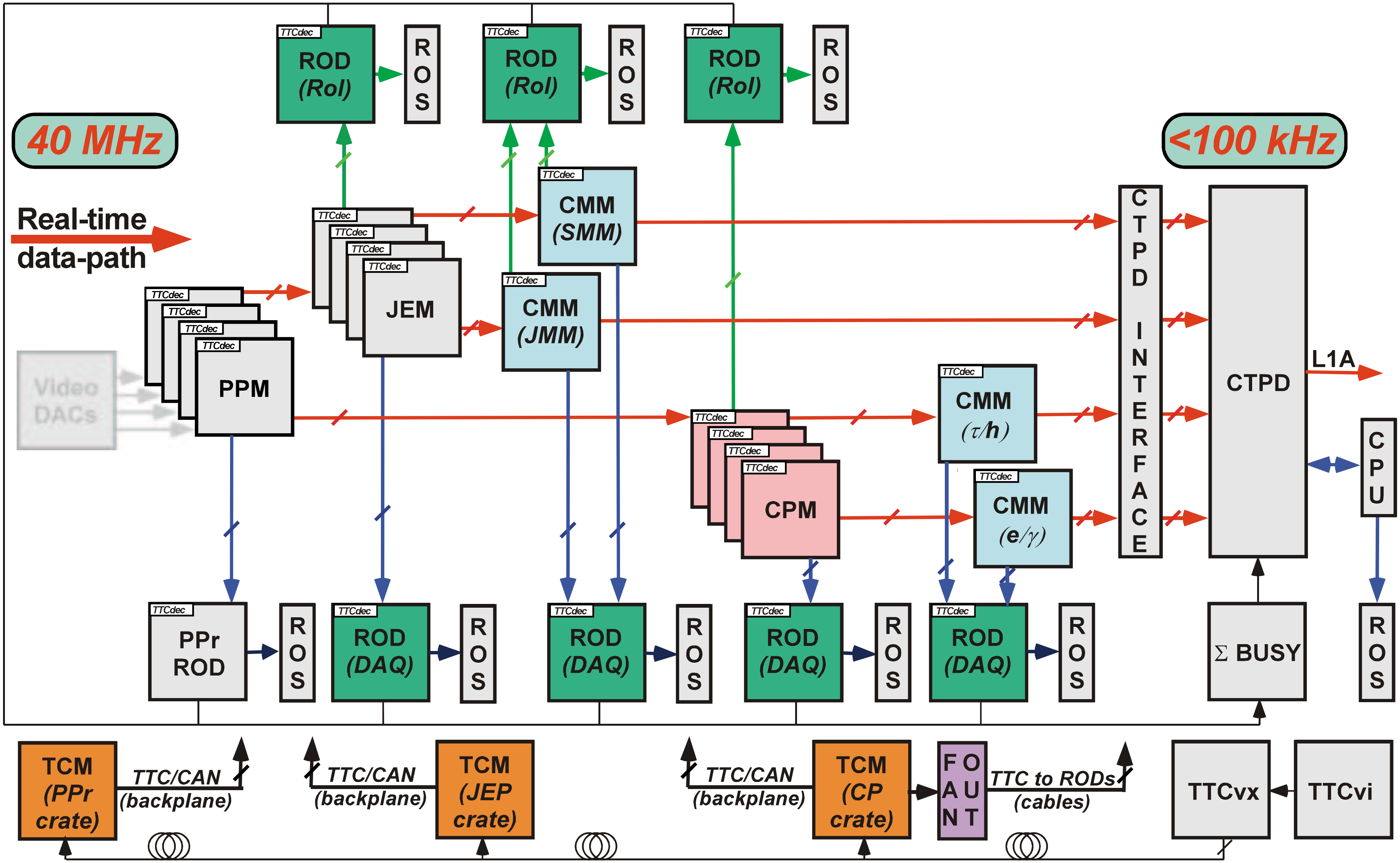
Slice Test

- Aim to test a full slice through the final system
- Also test interfaces with CTP, ROS, RoIB and calorimeters
- The slice test system will have about five or six crates for the calorimeter trigger alone

Software

- Distributed multicrate system is an obvious candidate for Online run control environment
- Aim to develop a prototype of the software required for final ATLAS
- Although the focus is on testing the hardware

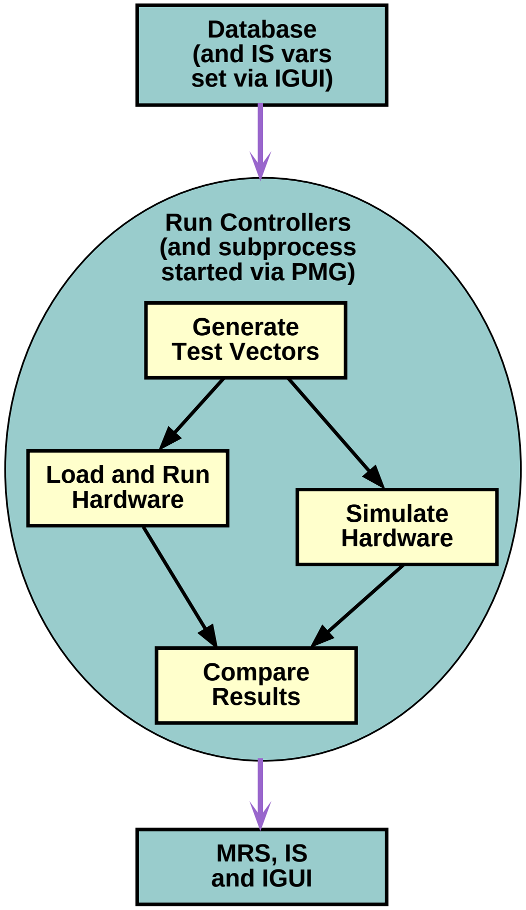
Slice Test Setup



Overview of L1Calo Software (1)

Slice Test Procedure

- Choose a hardware configuration and a test to run
- Generate test vectors (if necessary)
- Load the hardware with test vectors
- Simulate expected output of the selected configuration
- Run the system, collect data, compare and report
- Use this to make rigorous check of operating modes, speeds, error handling, etc

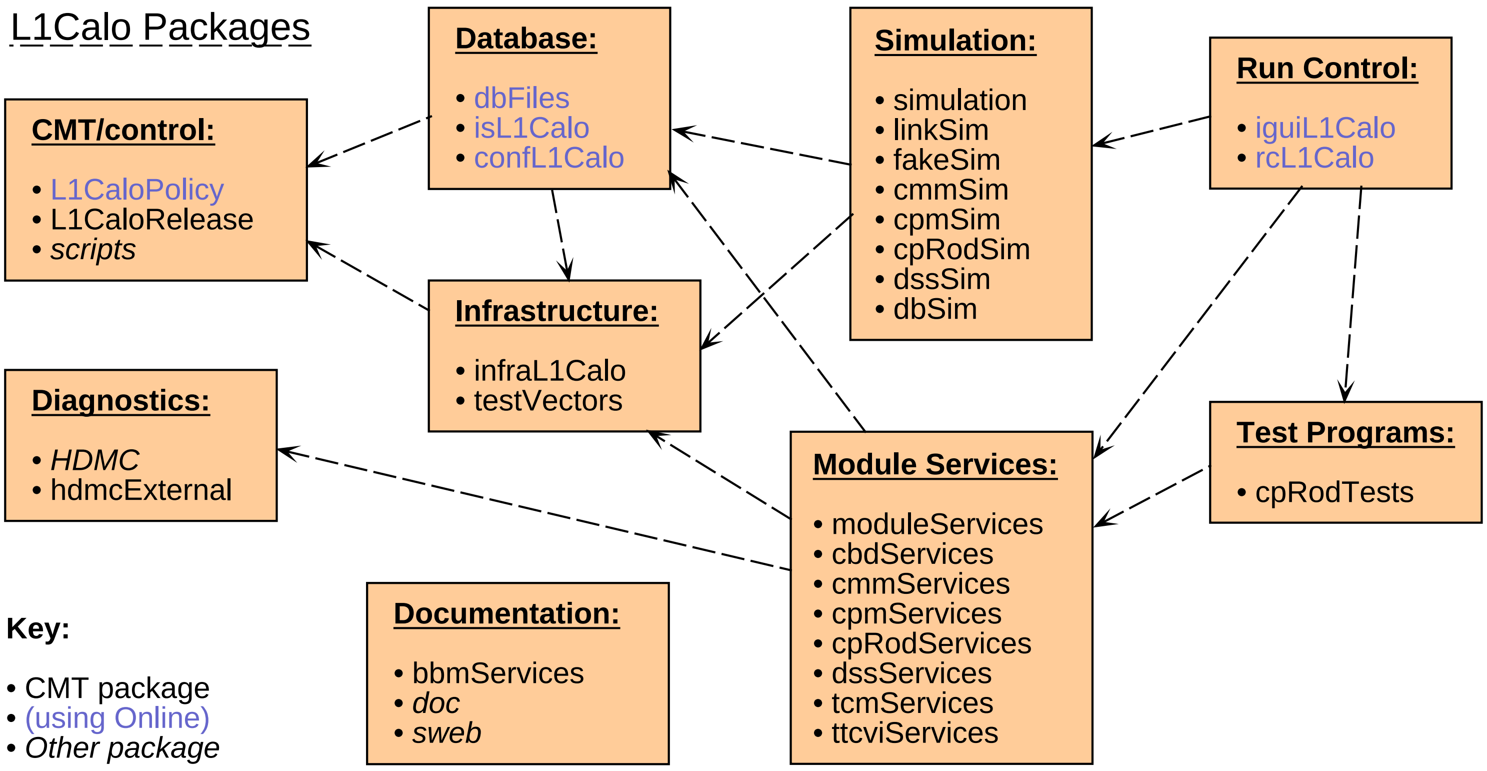


Overview of L1Calo Software (2)

Major Components

- Interactive diagnostics (HDMC): graphic view of details of the hardware, module registers etc
- Higher level “module services”: configure modules and their submodules using database objects
- Simulation and test vector generation
- Databases: extension of standard DAL, other DALs for calibration data, trigger menu and an layer integrating all of that
- Run control: our run controllers and IGUI panels etc
- Standalone test programs
- Common utilities, CMT infrastructure
- NB the majority of our effort (by my colleagues!) is not *directly* using the Online Software

L1Calo Components and Packages



L1Calo Packages

CMT/control:

- L1CaloPolicy
- L1CaloRelease
- *scripts*

Database:

- dbFiles
- isL1Calo
- confL1Calo

Simulation:

- simulation
- linkSim
- fakeSim
- cmmSim
- cpmSim
- cpRodSim
- dssSim
- dbSim

Run Control:

- iguiL1Calo
- rcL1Calo

Infrastructure:

- infraL1Calo
- testVectors

Diagnostics:

- *HDMC*
- hdmcExternal

Test Programs:

- cpRodTests

Module Services:

- moduleServices
- cbdServices
- cmmServices
- cpmServices
- cpRodServices
- dssServices
- tcmServices
- ttcviServices

Key:

- CMT package
- (using Online)
- *Other package*

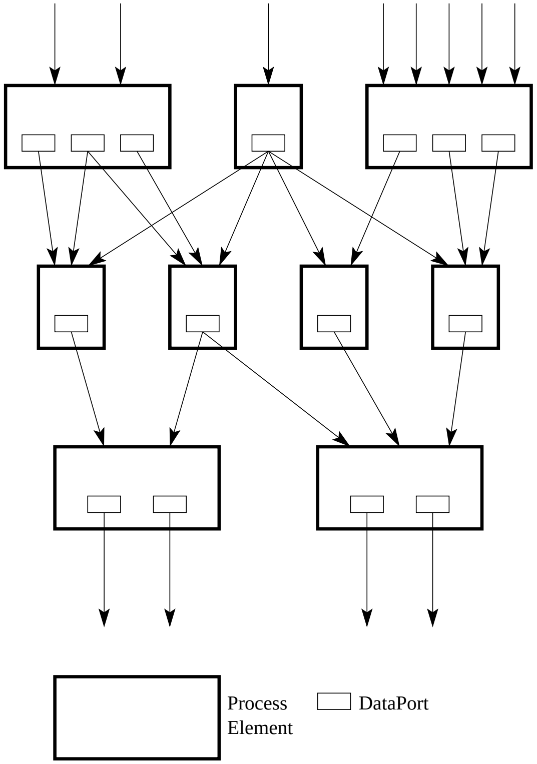
Documentation:

- bbmServices
- *doc*
- *sweb*

Simulation

General framework

- Generic VHDL-inspired framework with LHC 25ns clock
- Simulation to arbitrary level of detail
- ProcessElements may implement algorithms or contain groups of other ProcessElements
- File input/output at any point
- Simulate single chip (for firmware tests), single module, subsystems or the whole system



Module Services

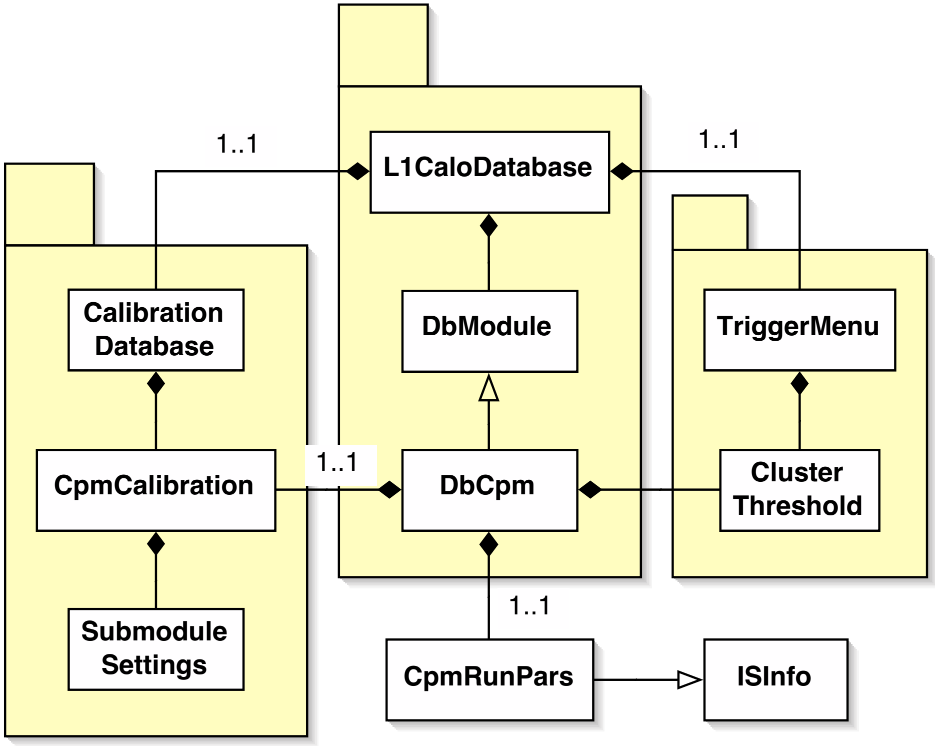
DAQ view of modules

- Layered on and extends existing generic HDMC package
- Adds L1Calo specific modules and submodules via common interface
- Implements actions required at each run state transition
- Create complex module and submodule object structures from a template
- Still uses separate HDMC database and configuration (but we would like to change this someday)

Database (1)

Integrated Database

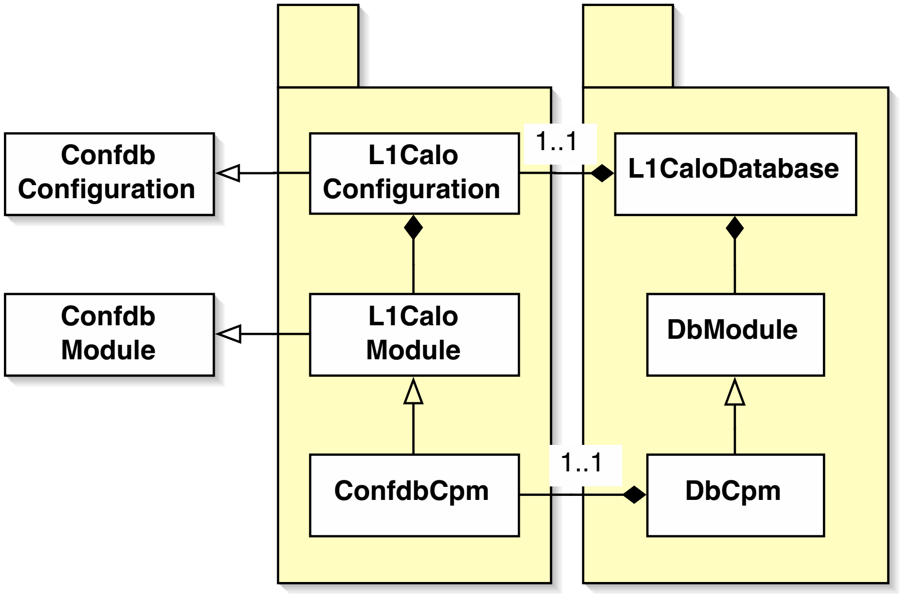
- Combines the static configuration, calibration, trigger menu and IS run parameters (for now)
- Selected static configuration data can be overridden by IS run parameters (if IS server is running)
- Separate DALs for calibration and trigger menu. Successive runs may use different trigger menus
- Integrated database object for a module returns all current data relevant to that module



Database (2)

Standard DAL Extensions

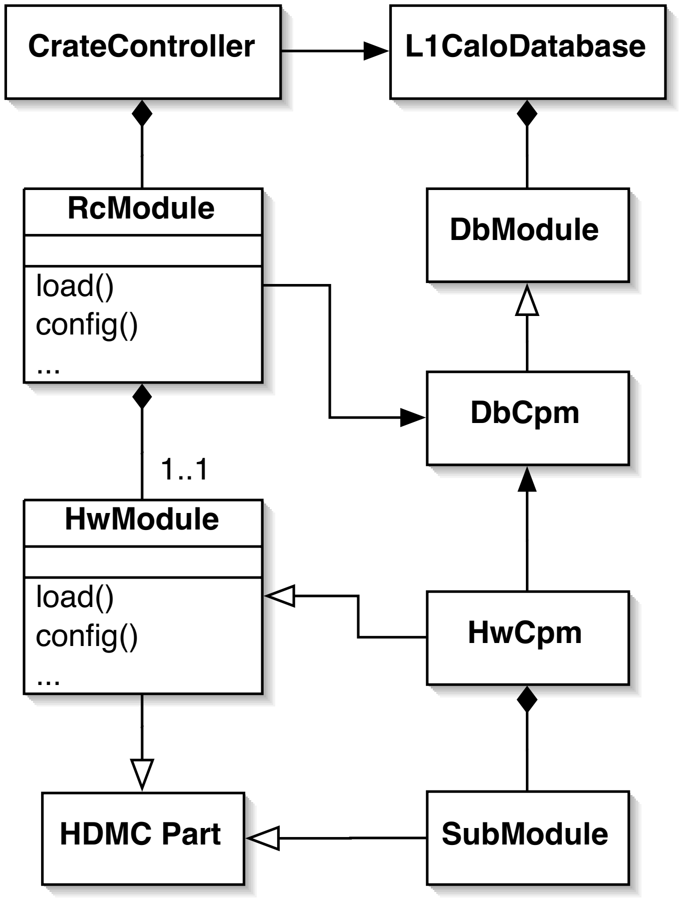
- Subclasses for our partition, modules and crates
- New classes to describe cabling between modules
- New classes to specify lists of files which may be selected by the user
- New classes to describe firmware configuration (under discussion)
- Convert database string enumerations to wrapped enum classes



Run Control (1)

Devolution to modules

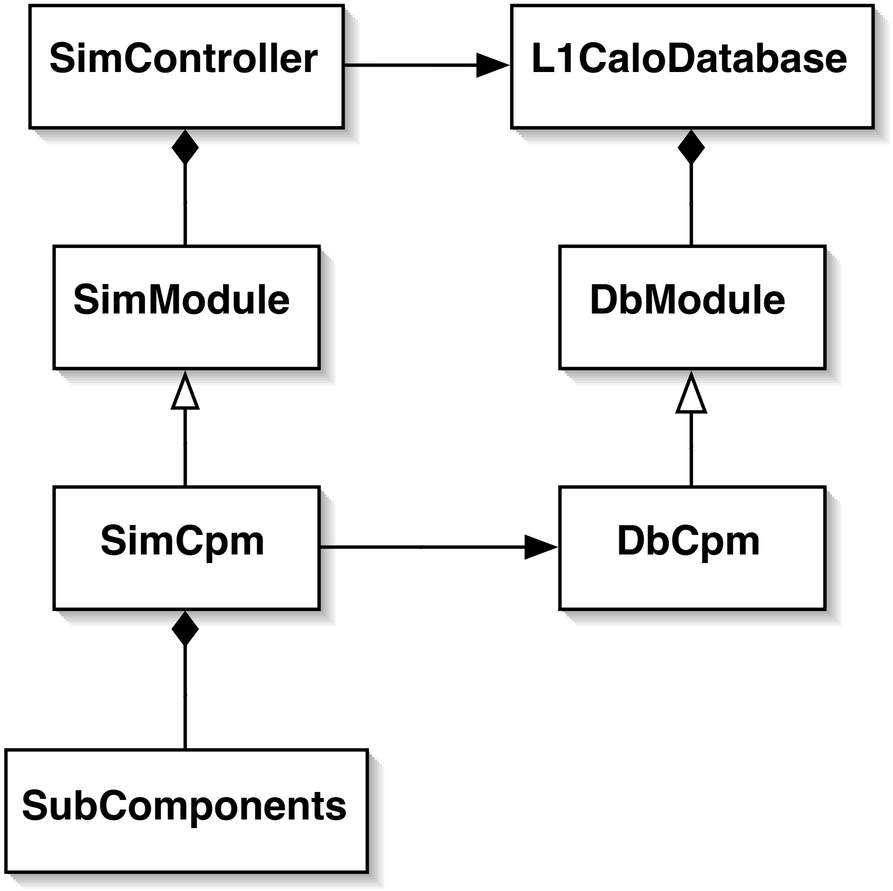
- Generic controller for all our crates
- Actions for each transition handled by the modules
- RcModule handles interaction with Online (MRS, IS)
- HwModule does all the work and can also be used in standalone test programs
- Readout process started by PMG
- Also envisage separate hardware status monitoring program



Run Control (2)

Simulation controller

- Also use run controller for the simulation
- Synchronises activity and reports any errors to MRS
- Generates test vectors from descriptor file when required
- Simulation configured from the integrated database (using IS run parameters if present)
- Can also run simulation standalone

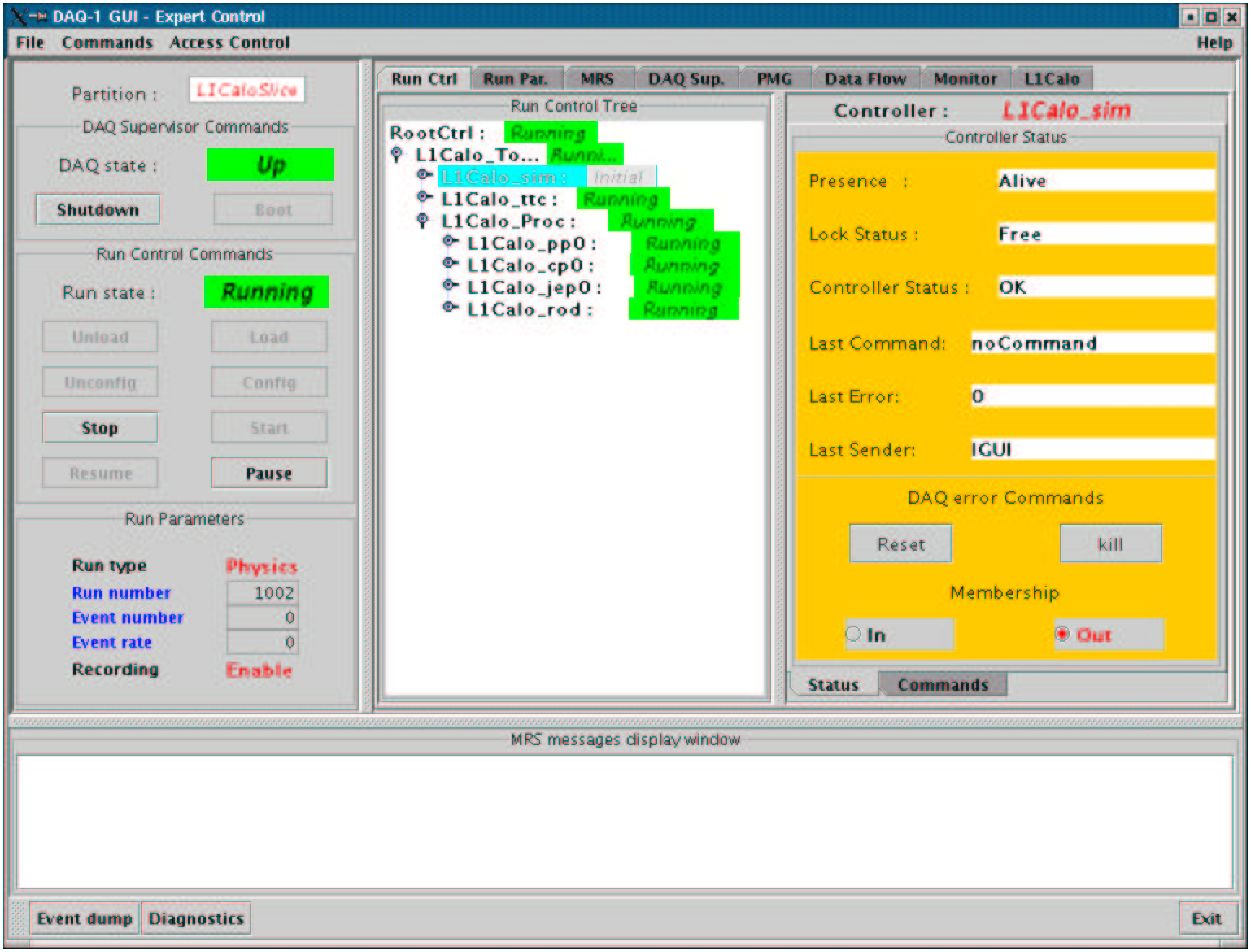


IGUI Panels

L1Calo subpanels

- Single L1Calo panel with subpanels (how many top level panels are expected for final ATLAS?)
- Subpanels conform to common interface (now adopted for top level panels)
- Expect frequent changes in slice test environment...
- ...so construct run parameter and status panels dynamically from IS schema (with help from Sergei)
- Simple layout – could be customised later by subclassing

Run control tree for slice tests?



Partition : L1CaloSlice

DAQ state : Up

Run state : Running

Run Parameters

Run type	Physics
Run number	1002
Event number	0
Event rate	0
Recording	Enable

Run Control Tree

- RootCtrl : Running
  - L1Calo_To... Runni...
    - L1Calo_sim : Initial
    - L1Calo_ttc : Running
    - L1Calo_Proc : Running
      - L1Calo_pp0 : Running
      - L1Calo_cp0 : Running
      - L1Calo_jep0 : Running
      - L1Calo_rod : Running

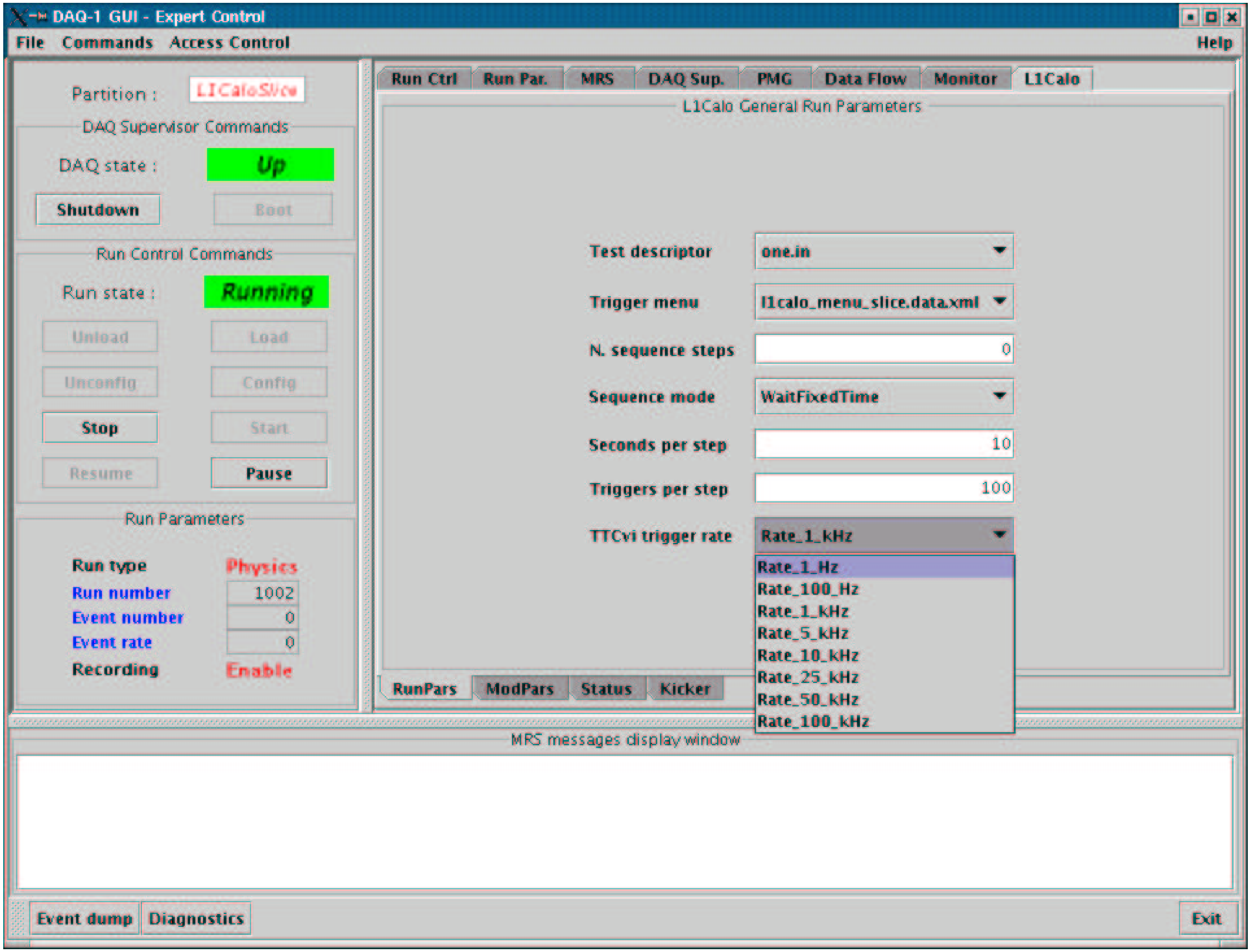
Controller : L1Calo_sim

Presence :	Alive
Lock Status :	Free
Controller Status :	OK
Last Command:	noCommand
Last Error:	0
Last Sender:	IGUI

Membership

○ In ● Out

L1Calo general run parameters



Partition : L1CaloSlice

DAQ state : Up

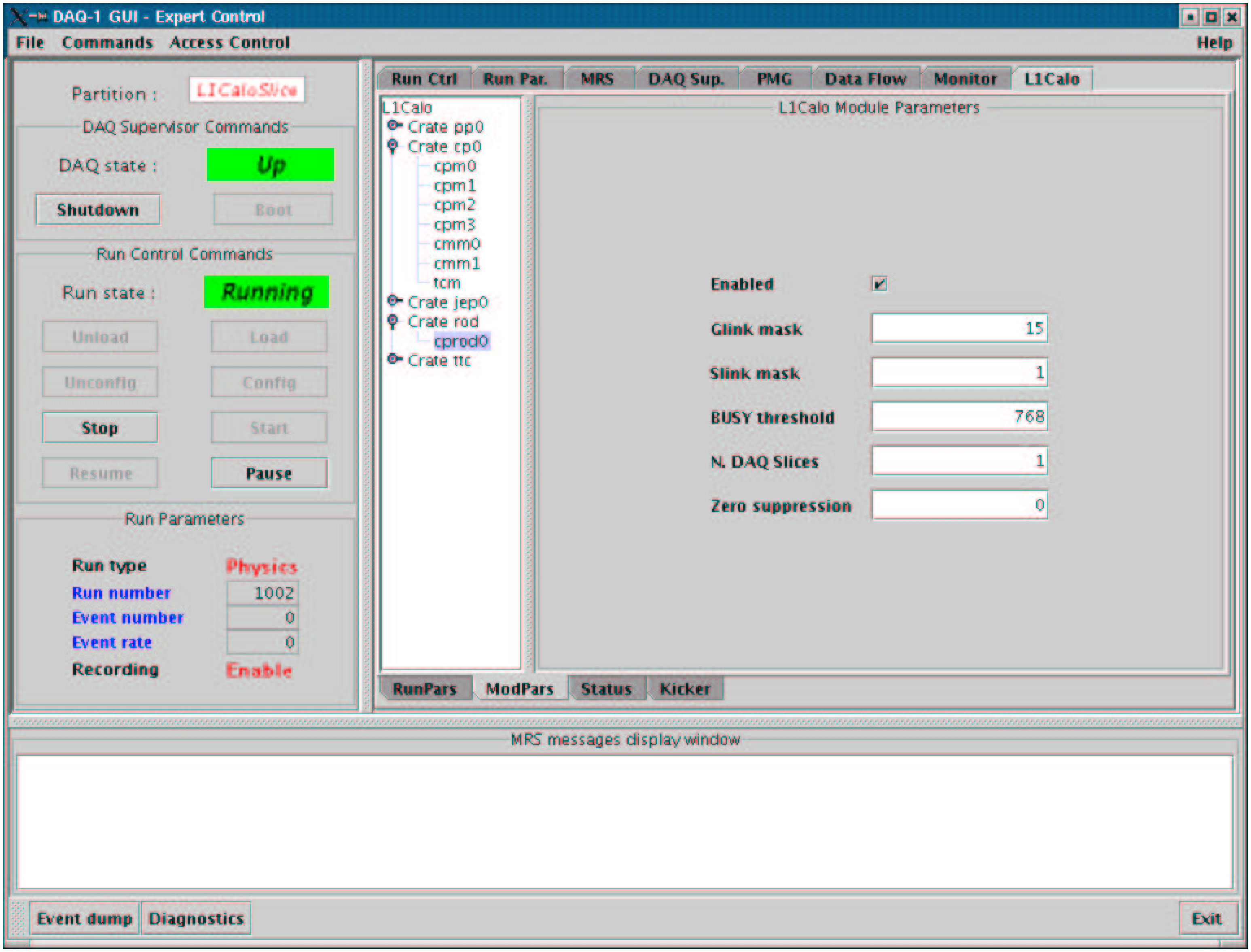
Run state : Running

Run type	Physics
Run number	1002
Event number	0
Event rate	0
Recording	Enable

Test descriptor	one.in
Trigger menu	l1calo_menu_slice.data.xml
N. sequence steps	0
Sequence mode	WaitFixedTime
Seconds per step	10
Triggers per step	100
TTCvi trigger rate	Rate_1_kHz

Rate_1_Hz
Rate_100_Hz
Rate_1_kHz
Rate_5_kHz
Rate_10_kHz
Rate_25_kHz
Rate_50_kHz
Rate_100_kHz

L1Calo module run parameters



Partition : L1CaloSlice

DAQ state : Up

Run state : Running

Run Parameters

Run type	Physics
Run number	1002
Event number	0
Event rate	0
Recording	Enable

L1Calo Module Parameters

Enabled	☑
Glink mask	15
Slink mask	1
BUSY threshold	768
N. DAQ Slices	1
Zero suppression	0

Development Environment

Mostly following the Online

- Using CMT and OnlinePolicy
- Added a few CMT fragments of our own in L1CaloPolicy package
- Adopted scripts from DataCollection and added a few of our own
- Nightly builds
- Doxygen (or Javadoc) for each package, linked by Doxygen tag files and custom HTML header
- Website and mailing list for software developers (in addition to the main L1Calo website)

Software Process

Fairly informal

- Overall requirements document: internally reviewed (but still a draft)
- Uses cases for calibration (ditto)
- Various discussion documents, requirements summary and user guides for individual packages
- Some use of Together for design and documentation
- Following ATLAS coding standards
- Starting to implement check targets for packages
- First (internal) release: about a week of testing
- Feedback at (minuted) monthly meetings rather than reviews and reports

Calibration

Big picture

- At the end of an LHC fill, remove L1Calo, TileCal and LAr from ATLAS partition
- Run a calibration sequence using combined L1Calo and TileCal partition
- Run a calibration sequence using combined L1Calo and LAr partition
- Possibly run combined calibration with L1Calo, TileCal and LAr together?
- Return L1Calo, TileCal and LAr to ATLAS partition
- L1Calo, LAr and TileCal will also perform calibrations at other times

Procedure

- Use run control to configure all elements of the combined partition
- Iterate over a sequence of calibration steps (synchronously stop triggers and change calibration parameter across the whole system)
- We need a fast lightweight way of doing this

Plans (Online SW related)

Short term

- Use ROS for collecting and building events
- Use Monitoring framework for analysing events
- Use Online Histogramming

Longer term

- Rewrite HDMC internal database (using OKS?) and integrate with Confdb (would like to be able to create object structures from templates)
- Try OBK for recording which tests were done with what configuration
- Package simple standalone module tests for use in the Test Manager? (Realistic tests need the run control)

Wishlist

General Online

- Combined release of Online, ROS (and DataCollection)
- Customisable event dump (long standing request!)
- Same DAL API in Java as C++, also eformat in Java?
- Create object structures from templates in OKS/Confdb
- Faster IGUI (for remote access to systems behind firewalls)

Run States and Run Control

- Intermediate states could make life easier
- Define what a checkpoint actually is
- How phases within a run happen. Can a low level controller declare the end of one run step (eg from local trigger processor crate?)
- Kind of test manager to drive sets of tests via run control

PS

- The Online software seems in good shape
- We have had many helpful interactions with the Online group (thanks!)
- We hope to continue to provide input for future improvements...