Level 1 Calorimeter Trigger Database Experience

Murrough Landon – 26 March 2003

(L1Calo group)

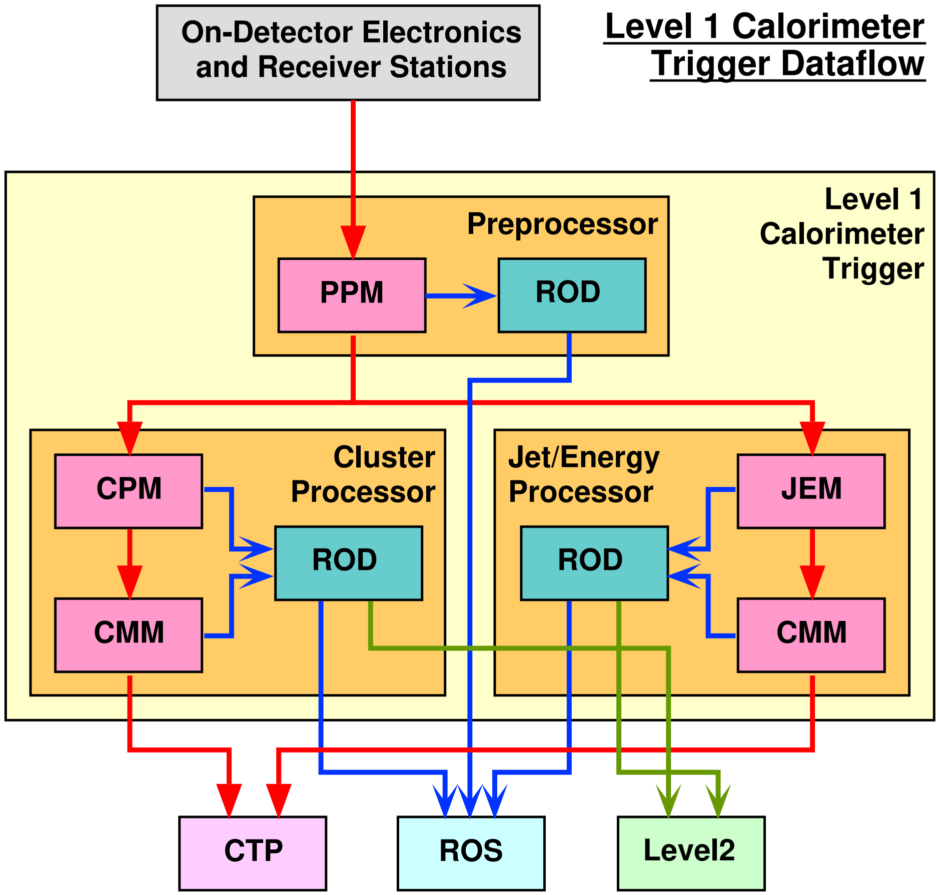
Overview

- L1Calo overview
- L1Calo database requirements
- What we have done (so far)
- What we have not done
- Comments on the Online ConfDB
- Databases in Dataflow
- What about ROD Crate DAQ?

Overview of the Calorimeter Trigger (1)

Hardware

- Three main subcomponents: preprocessor, cluster processor and jet/energy processor – plus RODs, TTC infrastructure, cables, etc
- About 15–20 crates, some with custom backplanes (not quite VME)
- Six types of custom 9U module
- Real time data to CTP. RoIs and readout to DAQ via RODs
- Crate CPUs will run a variant of ROD crate DAQ – but not all our crates are ROD crates



Overview of the Calorimeter Trigger (2)

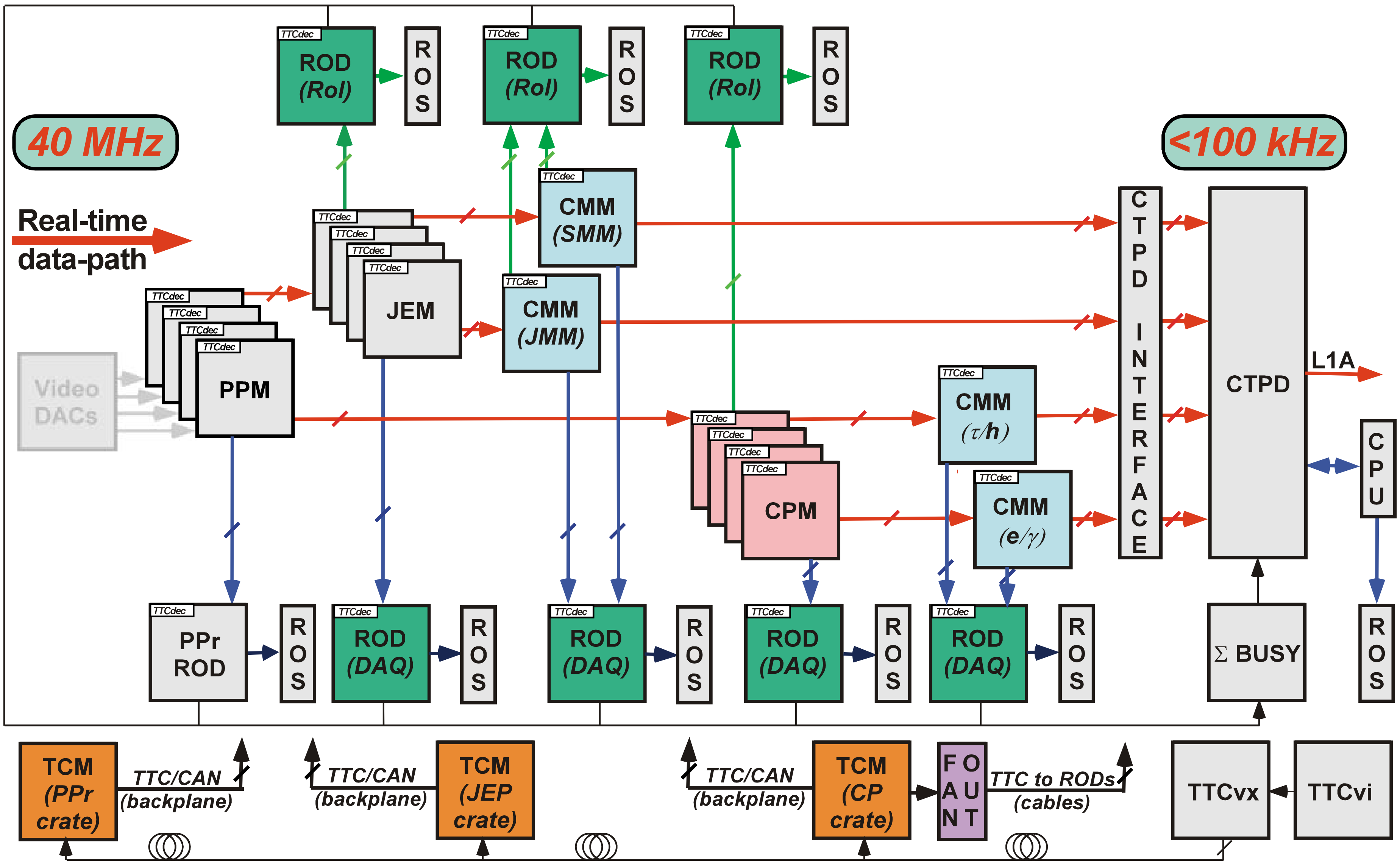
Slice Test

- 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

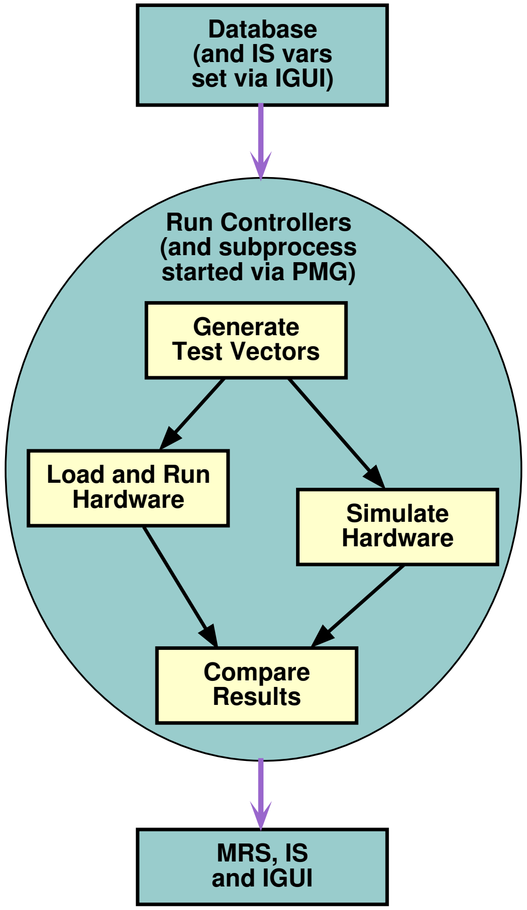
Slice Test Setup



Overview of L1Calo Software

Slice Test Procedure

- Choose (one of several) hardware configurations and (one of many) tests 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
- Make rigorous check of operating modes, speeds, error handling, etc



L1Calo Database Requirements (1)

Aims and Intentions

- Use Online configuration database: gain experience with OO databases and provide feedback and requests for missing features
- Parametrise our tests as far as possible via the database (or data files) and not in code
- Pass to “hardware” module classes integrated database objects which contain all the information they need

L1Calo Database Requirements (2)

Categories of Data (by type)

- Typical Online configuration data: modules, crates, programs, etc
- Customised extensions and additional configuration information (cables, firmware)
- Calibration data
- Trigger menus
- Run parameters, groups of options selected by run type
- Test vectors to verify the behaviour of the system

Categories of Data (by how it changes)

- “Static” configuration: changes very slowly
- Time varying data, such as calibrations
- Dynamically user chosen data, such as trigger menu, run parameters

L1Calo Database Implementation (1)

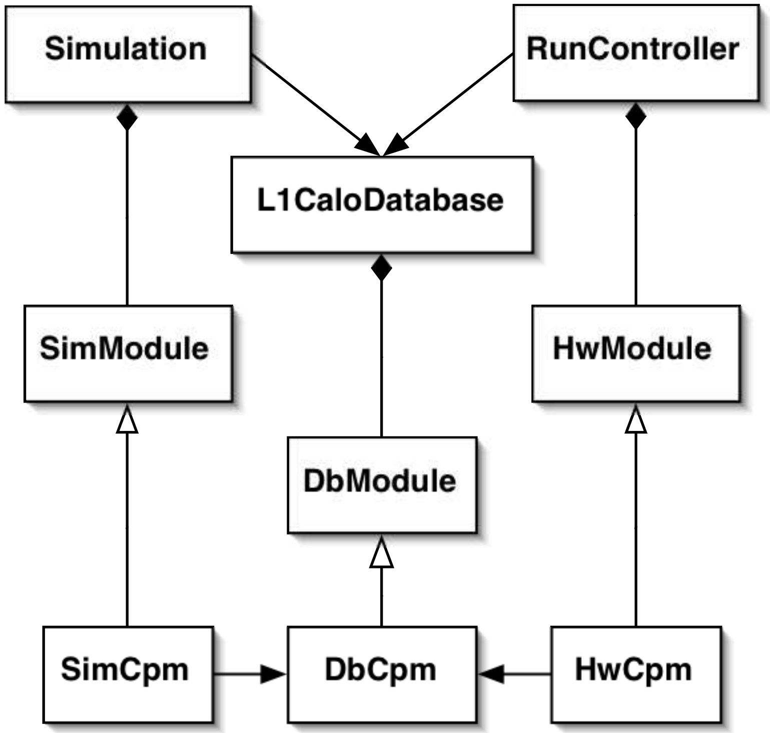
Overview

- Extended Online Data Access Library (DAL)
- Use the DAL idea (hide underlying DB implementations such as OKS from run time classes) for other areas like calibration and trigger menus
- Additional layer to integrate all data for each module

L1Calo Database Implementation (2)

Overview of DB usage

- Both crate controllers and the detailed hardware simulation devolve most of the work to subclasses for modules (and their subcomponents)
- Each hardware and simulation module object is passed one database object containing its complete configuration



L1Calo Database Implementation (3)

Writing/extending a DAL via ConfDB

- Create a separate OKS schema file containing subclasses of Online database classes and/or new classes
- Write C++ run time classes inheriting from ConfdbObject class
- Write subclass of C++ ConfdbConfiguration class to instantiate these from OKS objects
- Edit confdb_gui configuration files to add new classes (and icons) to menus and windows

L1Calo Database Implementation (4)

Extensions of Online ConfDB DAL classes

- Subclass of ConfdbPartition to add new categories of information (trigger menus, calibration datasets, run type options)
- Subclasses of ConfdbModule with additional attributes for our various module types

Additional classes

- Classes to describe cable connections between modules and bundles of cables between modules
- Classes to describe the firmware configuration of each module in terms of the collection of firmware programs to be downloaded into each one (with appropriate checks!)
- These classes are linked to our ConfdbModule subclass

L1Calo Database Implementation (5)

Additional separate DALs

- Decided to stick with a single underlying database technology
- So used OKS and created simple DALs for calibration data and also for trigger menus
- (NB in future the trigger menu will have to migrate to new non-OKS XML based implementation from Thomas Schoerner-Sadenius)
- Also added set of classes (instead of simple run type strings) to describe groups of options selected by run type

Small amount of code generation

- Online ConfDB uses strings for enumerations
- Convert these to automatically generated “wrapped enum” classes (so typos get picked up at compile time)

L1Calo Database Implementation (6)

Integrated layer over Online ConfDB

- Online ConfDB is static and readonly (latter now changed?)
- We need to be able to change trigger menus, update calibrations and to vary run parameters
- Decided to create "integrated" database class to hold all the data for one module: this holds the Online ConfdbModule and calibration, trigger menu, run type options etc for that module
- This "integrated" layer insulates our other software from any ConfdbModule changes (eg if Online DAL is automatically generated in future)

L1Calo Database: Missing Areas (1)

Module substructure: the problem

- Online ConfDB knows about Crates and Modules, but nothing smaller (except CPUs)
- VME modules present detailed substructure, eg registers, memories and larger components such as FPGAs which have their own substructure
- Some aspects of the substructure are common to all instances of the module (eg address offsets of registers, their format, read/write nature of bit fields, etc)
- These aspects are akin to a schema or template (change of one instance should be propogated to all)
- Other data is specific to each instance, eg the values to be loaded into registers
- For different registers, memories, etc the source of the data may be very different: run parameters, calibrations

L1Calo Database: Missing Areas (2)

Module substructure: L1Calo (non) solution

- We have detailed hierarchical description module and submodule substructure – but in a completely separate legacy “database” developed for our interactive hardware diagnostic toolkit (HDMC)
- This is (unhappily) integrated with the run control
- Substructure of our other module data is implemented separately in parallel
- We have long intended to integrate these better, replacing the HDMC database with a common ConfDB style solution...
- ...but other priorities and some uncertainty as to the future of the present OKS/ConfDB have delayed this
- We would be interested in looking at anybody elses solution. (the Level1 TGC group were considering this – but may just use MySQL)

Comments on Online ConfDB (1)

Summary of our experience

- DAL classes relating to software used "as is"
- We decided to extend DAL for hardware classes: not hard but a little tedious
- Some of our extensions proposed for inclusion in Online ConfDB (with some changes)
- "const"-ness and static nature of the DAL seen as a problem (especially in our slice test environment)

Comments on Online ConfDB (2)

Wishlist

- Automatic DAL generation from schema would save a lot of work...
- ...but probably still want hand coded layer (for some classes) for additional algorithms to manipulate data and to apply detailed checks that are not easily automated (or just not possible at the moment)
- Would like (somehow) to keep some information static, other data selectable, still other data editable within IGUI and saved back to single database. Not easy to combine all these with the present system
- Existing database editing tools may not be adequate for large systems?

Configuration Database Future?

Online/Dataflow Divergence?

- Dataflow software now uses privately generated DAL
- This is still based on OKS so the Online GUI editors can be used but the run time API is different
- Existing Online DAL is hand written: mainly data containers but also some algorithms for convenient data reorganisation
- Online group proposed automatic DAL generation in future
- Can use Dataflow automatic DAL generation to extend Online ConfDB classes and for separate classes – but algorithms would need to be added again
- If ROD crate DAQ is based on ROS (ie Dataflow) software and subdetectors also need other data from Online ConfDB we may be using both DALs at once
- Online may adopt and extend the Dataflow scheme later this year (autumn?) – but this implies significant API changes for existing ConfDB users?

Databases for ROD Crate DAQ (1)

Mystery of the disappearing ROD crate class

- In the past the Online ConfDB DALs included a RODC class to be used for ROD crates (to distinguish them from readout crates?)
- The latest Online software (online-00-19-00) released two weeks before this workshop has removed the RODC class from Online and Dataflow schemas!
- This is not a problem (RODC had no extra attributes compared to the Crate base class and I think the new ROS software doesnt use the Crate class any more)
- ...but it seems a little symptomatic of ROD crate DAQ getting lost among other developments such as divergence of Online and Dataflow DALs

Databases for ROD Crate DAQ (2)

What would we like?

- Existing configuration DAL already meets many requirements
- Additional classes for modules and hardware configuration could be added
- Detailed description of module substructure?
- DAL for general ROD crate DAQ and subdetector specific extensions to be generated automatically

What should we do *right now*?

- ROD crate DAQ aims to reuse ROS software
- Seems logical to use Dataflow generated DAL for ROS-like aspects?
- Subdetectors may already be using Online ConfDB (and some of us have invested in extending it)
- Im not sure what the short term direction should be

Summary

- L1Calo have used and extended the Online ConfDB in a number of ways
- Some extensions (firmware, cables) may be incorporated – in a more general way – in future ConfDB releases
- Online ConfDB schema could be extended to provide common ROD crate DAQ features and to meet subdetector specific requirements
- Automatic generation of DALs would ease this process considerably – but are currently only available via the Dataflow DAL
- Dataflow DAL may have to be used by ROD crate DAQ software which reuses ROS software
- Worry about recent divergence of Online and Dataflow database approaches