Run Control Changes

Murrough Landon – 17 December 2001

http://www.hep.ph.qmw.ac.uk/~landon/talks

Overview

- Integration steps
- Run control changes
- Database
- Excel macros?

Miscellaneous

- Excel macros?
- Online software
- VME library API

Run Control and Module Services

First integration step

- Changes to run control (next slide)
- Development of L1CalDaq (Bruces extension of HDMC) to be the basis of our Module Services package
- Parts files for the modules used in CPROD tests modified for use by run control.
- Run controller successfully talked to real hardware (but not DSS or ROD: further work was required on handling their more complex parts files).
- Useful first step, but a lot more needs to be done!

Run Control Changes (1)

Recent

- (Previously reported) crate controller creates HDMC parts manager and VMEBus object.
- CtrlModules now create DaqModules from parts files.
- Crate database separated from run controller. Can be used by standalone programs to get list of CtrlModules and hence DaqModules.
- Consistent naming scheme: classes and database.
- Tried prerelease of next Online software version (crashes).

Run Control Changes (2)

Next Steps: Database

- Run types and run parameters. Define in database, updates via IS.
- IGUI panel to set (some) run parameters interactively (rather than editing database files)
- Extend default Online ConfdbModule class (requires release 0.0.16).
- Integrate (and further develop) existing work on calibration data.
- Ditto for trigger menu.

Database (1)

Types of database objects

- Static configuration of the system (crates, modules, slots, VME addresses, firmware configuration, etc): online configuration database (with some extensions). Constant for long periods.
- Calibration data: timing, BCID, lookup tables. May depend on run type (special settings). Also step through calibration phase space during calibration runs.
- Run parameters: run type dependent choices for module settings, choice of test vectors, etc. Very variable. Probably want defaults for each module type (for eac run type) with possibility to adjust some settings for each module interactively in the IGUI?
- Trigger menu. Selection from prepared set.
- Expected outputs from test vectors depend on choices of calibration and trigger settings.

Database (2)

How to organise it all?

- Module objects (CtrlModule and/or DaqModule) get passed a single ModuleDbInfo (subclass) object at initialisation.
- CtrlModule objects also get passed the RunType object at each run state transition (at least for those after the run type is definitely known).
- ModuleDbInfo could be composed of several other objects:
  - the online configuration database information (ConfdbModule)
  - a default calibration object for the module, which may be updated by the run controller behind the scenes
  - possibly also run parameters for the module?
- The RunType object contains the general run parameters.
- Maybe module specific run parameters come this way too (instead of the ModuleDbInfo object?) which would ensure they couldnt be used before the run type was fixed.

Database (3)

What exists?

- A document with suggested schema for calibration and trigger menu.
- Implementation of the schema in OKS: fairly complete for trigger menu, but only a partial CPM calibration.
- Implementation of data access libraries (DALs) to convert OKS trigger menu and (what is defined for) calibration data to C++ object structures.
- Beginnings of a GUI to create trigger menus: no further work envisaged now that CTP have started work on a more complete approach to this.
- Recently created OKS schema for run parameters for modules and overall “looper” type run parameters.
- Initial implementation of a DAL to create C++ classes.

Database (4)

What is still required?

- Define the way all the database objects relevant to a given module get passed to the corresponding module object.
- Extend the existing work to cover all types of module and all calibration data.
- Utilities to produce simple calibration databases.
- Allow modules to update calibration objects with values read from modules and store the results? Eg for saving the results of autocalibrations.

Excel Macros?

A few concerns...

- Norman has developed an Excel macro to convert a spreadsheet style register map into HDMC configuration and parts file for the CMM.
- A neat idea but I have a few concerns:
- Saving and sharing of source code: do we treat the Excel stuff as source code and store it in our CVS repository, so that others can use, and if necessary, modify it?
- If so, it introduces a non-Unix (and non-Makefile?) element into producing libraries and applications from source code.
- If not, the output from the Excel stuff is the stored source code and can them become out of step with the Excel input.
- Are there implications if we use this approach for some modules and not others? Eg multiple definitions of common registers (eg ModuleId)?

Miscellaneous

Online software news

- Current release (0.0.15) in use at QMW and RAL (and HD?).
- Imminent next release offers some improvements to the database (which could be useful to us) and first version of online histogramming package (this is just a transport mechanism for generic histograms and anyway needs further work).
- They are are not quite ready to make the official release with CMT. Their working model is described at:
  `http://akazarov.home.cern.ch/akazarov/cmt.html`

VME library API

- Do we have any comments on the VME library proposals?
  `http://atlas.web.cern.ch/Atlas/GROUPS/FRONTEND/document/API.html`