Software Meeting

Murrough Landon – 5 March 2003

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

Overview

- Database developments
- Run control and module services

Database Developments (1)

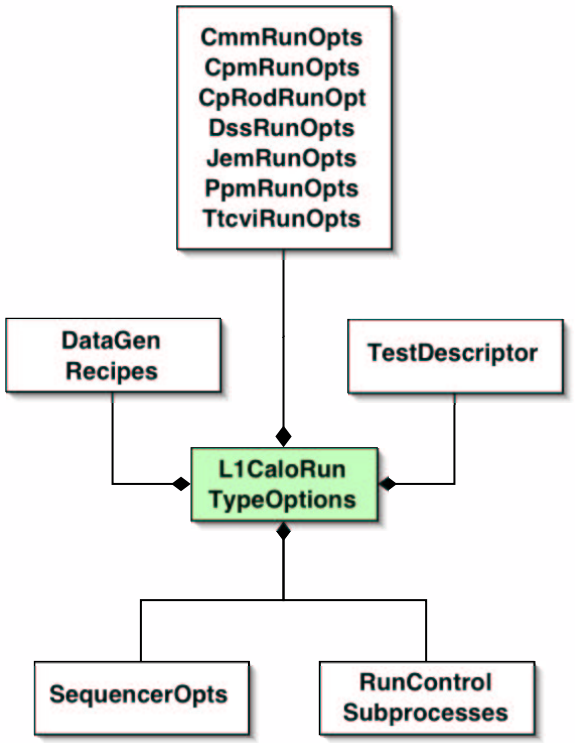
Overview

- New run type objects (see later)
- Calibration scans: calibration objects implement a new interface which accepts an object defining start point and increment
- More internal consistency: calibrations, trigger menu, IS parameters, run types, etc are now all implementations of a new interface which gets passed, Visitor style, to all DbModules

Database Developments (2)

L1CaloRunTypeOptions

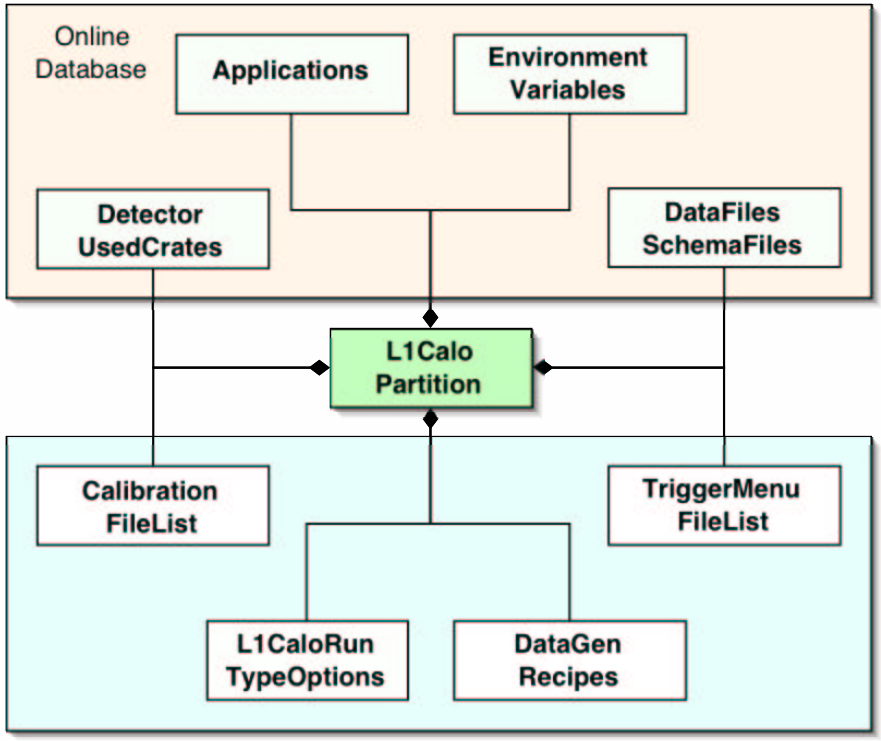
- Named object defining a run type
- Various module options (by module type or specific module)
- Options for multistep sequence runs
- TestDescriptor for this run type
- DataGenRecipes specific to this run type (NB there can also be global ones)



Database Developments (3)

L1CaloPartition

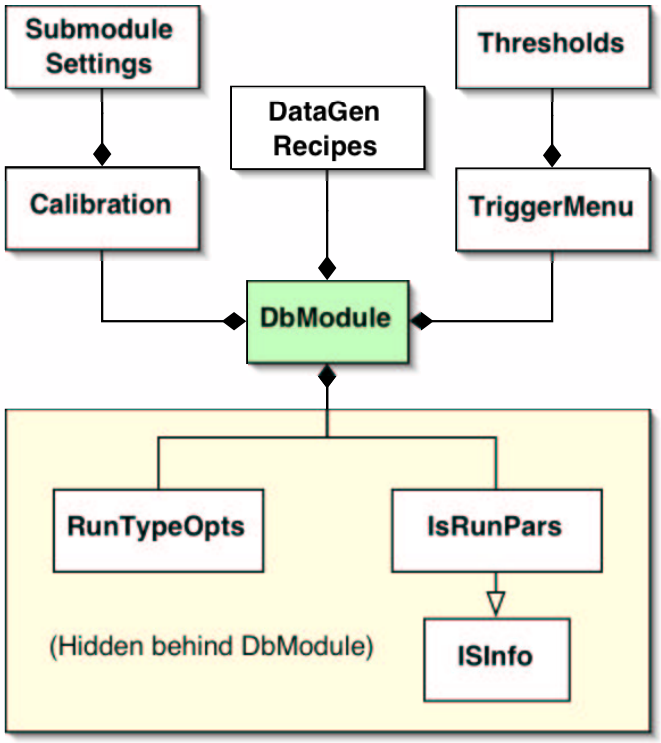
- Partition now contains list of available run type objects (NB different from Online list of run type strings)
- Also now contains list of default DataGenRecipes (NB must now be linked explicitly to be found)
- Links to lists of calibration and trigger menu files as before
- Plus all the stuff inherited from the parent Partition class



Database Developments (4)

L1CaloPartition

- DbModule returns XxxCalibration objects as before (for modules with calibration data) and a pointer to the trigger menu
- New RunTypeOptions objects remain hidden (values returned via DbModule subclass APIs – as with the IS run parameters
- DataGenRecipes are returned as before, but selecting new run type may change which ones are used (ie dont cache DataGenRecipes)



Run Control and Module Services (1)

What happens at Boot, Load

- RC Boot: basic initialisation of run controller
- RC Load:
  - initialise HDMC, open database
  - create CtrlModule for each module: read its parts file and find its DaqModule (NB some modules, eg DSS, make VME accesses at this point)
  - DaqModule::initModule – basic initialisation (no DB)
  - DaqModule::setDbModule – pass DbModule
  - DaqModule::setModuleStatusPointer – pass L1CaloModuleStatus object
  - DaqModule::disable() – inhibit input/output
  - DaqModule::load() – perform load actions (finally)
    (assumed to be only reset, firmware check and clock setting)

Run Control and Module Services (2)

What happens at Configure, Start

- RC Configure:
  - RC (re)reads IS parameters, (re)sets run type
  - DaqModule::configure() – perform configure actions
    (download most settings: calibration, trigger menu, test vectors)
- RC Start:
  - RC (re)reads IS parameters, (re)sets run type
  - If calibration, initialise calibration parameter sequence
  - DaqModule::start() – perform start actions
    (reload calibration parameter being scanned?)
  - DaqModule::enable() – inhibit input/output
  - DaqModule::updateStatus() – get status and save to IS
  - RC starts any required processes

Run Control and Module Services (3)

What happens at Pause, Resume

- RC Pause:
  - RC (re)reads IS parameters
  - DaqModule::disable() – inhibit input/output
  - DaqModule::pause() – perform pause actions
    (reload calibration parameter being scanned?)
  - DaqModule::updateStatus() – get status and save to IS
- RC Resume:
  - RC (re)reads IS parameters
  - DaqModule::resume() – perform resume actions
  - DaqModule::enable() – inhibit input/output
  - DaqModule::updateStatus() – get status and save to IS

Run Control and Module Services (4)

What happens at Stop, Unconfigure, Unload, Idle

- RC Stop:
  - RC (re)reads IS parameters, (re)sets run type
  - RC stops any processes it started
  - DaqModule::disable() – inhibit input/output
  - DaqModule::stop() – perform stop actions
  - DaqModule::updateStatus() – get status and save to IS
- RC Unconfigure:
  - DaqModule::deconfigure() – perform unconfigure actions (if any)
- RC Unload:
  - DaqModule::unload() – perform unload actions (if any)
- RC Idle: (called automatically every few seconds)
  - DaqModule::updateStatus() – get status and save to IS (if running)