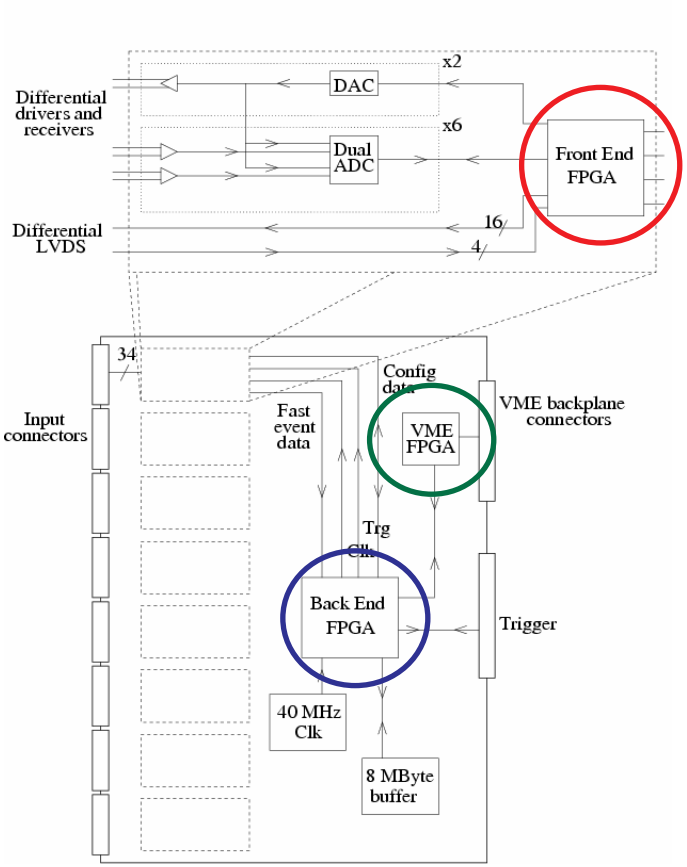
DAQ/Online Status

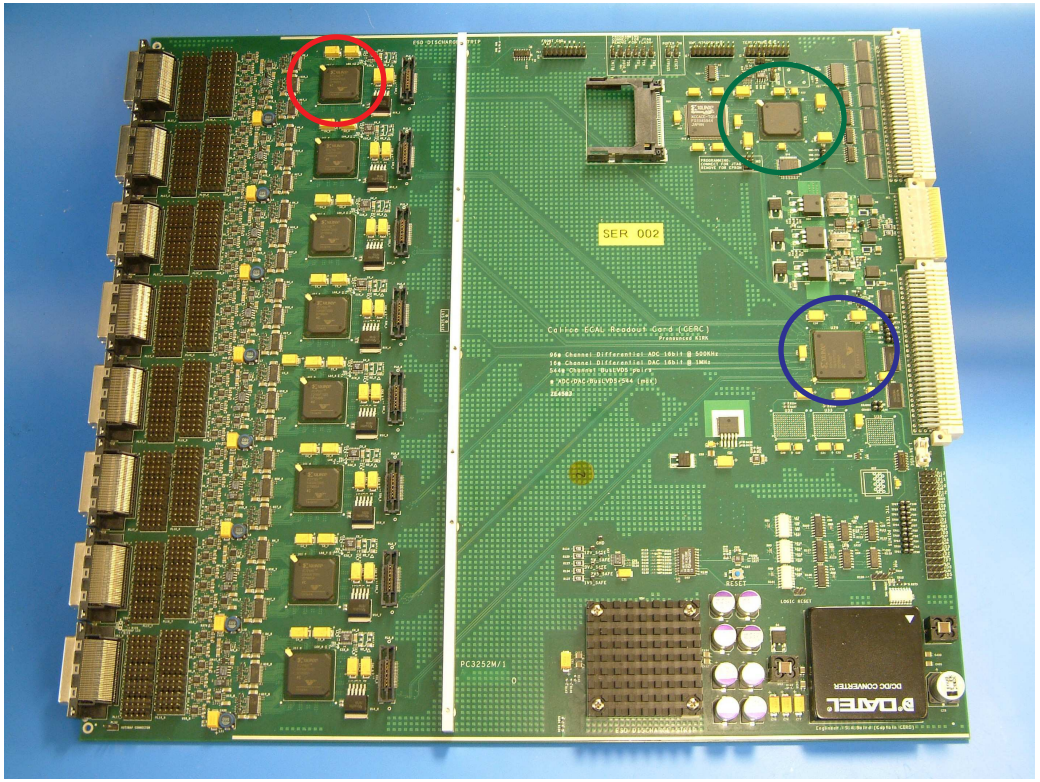
Paul Dauncey

Imperial College London

DAQ overview

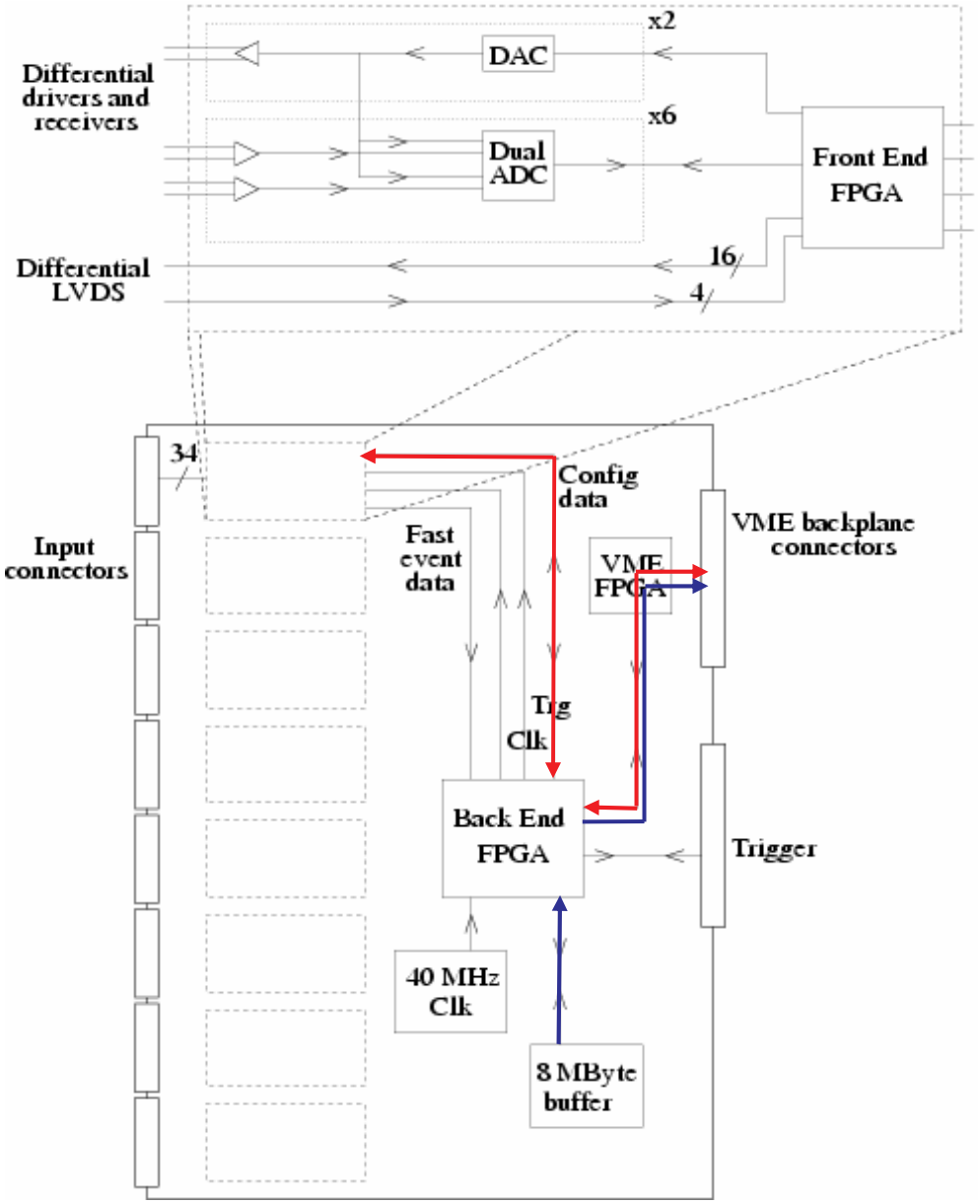
- Mainly based on Calice Readout Card (CRC) VME board
  - Modified from CMS silicon tracker readout board
  - Does very front end control, digitisation and data buffering



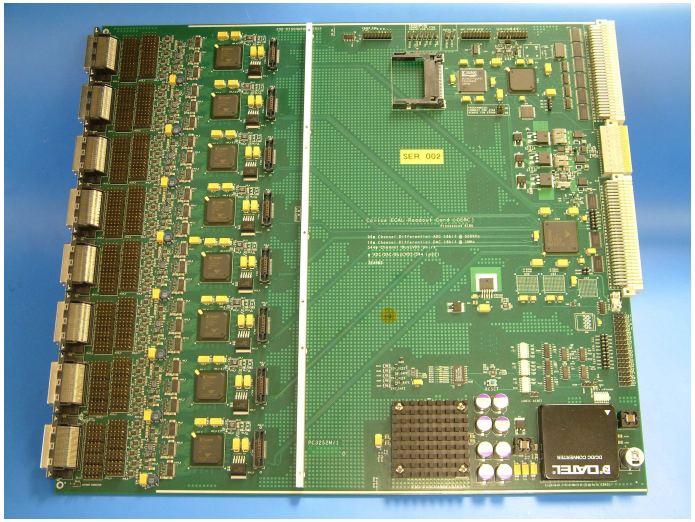


CRC readout

- Two VME data paths
  - Serial path: slow
    - Read and write directly to all FPGAs
    - Used for all configuration data loading and readback
    - Also used for temperature and power monitoring
  - Vlink path: fast (i.e. VME block transfer)
    - Read only from 8MByte memory (via BE)
    - Used for event data only



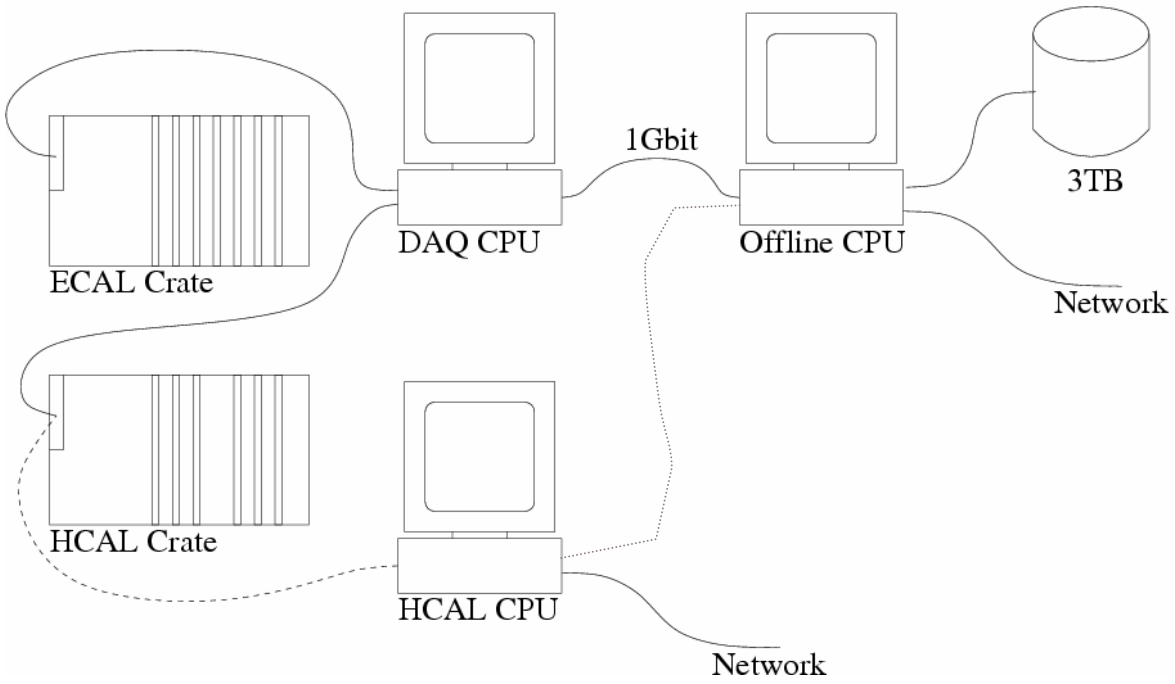
CRC hardware status



- Need 13 CRCs total
  - ECAL requires 6 CRCs
  - AHCAL requires 5 CRCs
  - Trigger (probably) requires 1 CRC
  - Tail catcher requires 1 CRC
- Status
  - 9 exist (2 preproduction, 7 production) and are tested
  - 7 are being manufactured via RAL, delivery in Nov
  - Should have 13 plus 3 spares tested by end of year
- DHCAL readout still uncertain
  - Complete conceptual design exists but is not yet funded
  - Could use CRCs to save money; would need 5 CRCs (like AHCAL) and so would use AHCAL ones, not make additional CRCs
  - No running with DHCAL planned before 2007; concentrate on 2006 here

DAQ hardware layout

- DAQ CPU
  - Trigger/spill handling
  - VME and slow access
  - Data formatting
  - Send data via dedicated link to offline CPU
  - Redundant copy to local disk?
- Offline CPU
  - Write to disk array
  - Send to permanent storage
  - Online monitoring
  - Book-keeping
- HCAL PC
  - Partitioning
  - Alternative route to offline PC

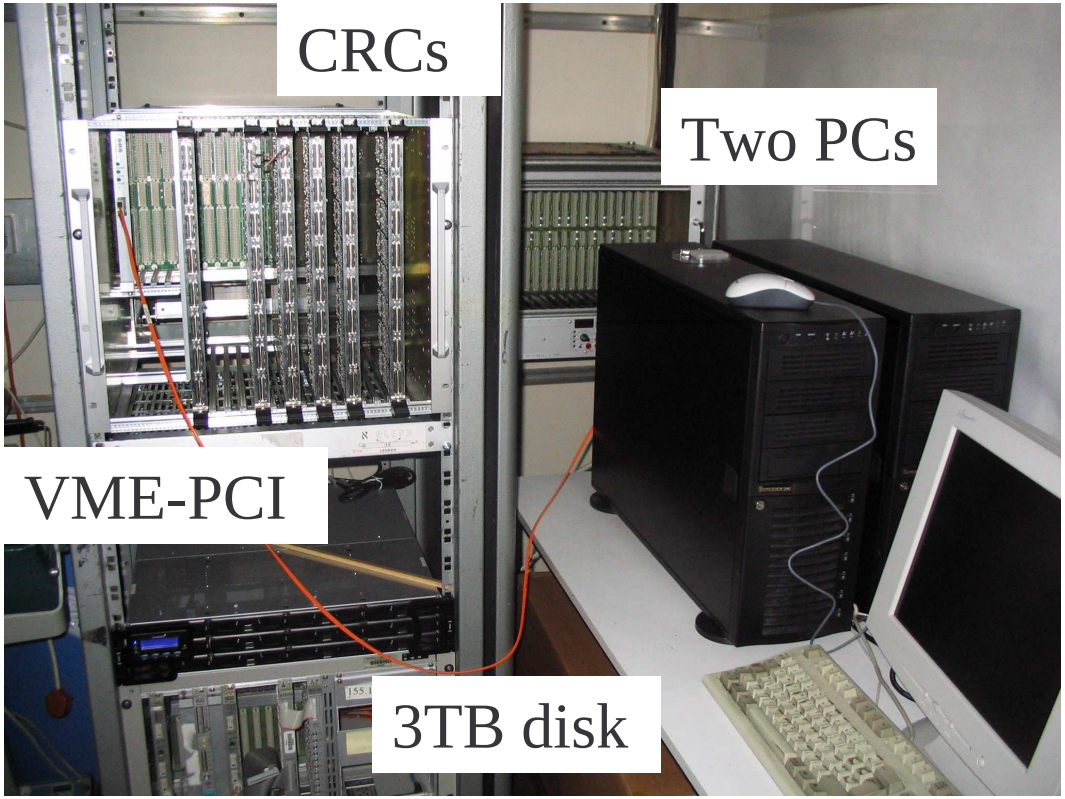


Status of non-CRC hardware

- Two 9U VME crates with custom backplanes needed
  - One for ECAL and trigger
  - One for AHCAL and tail catcher
  - Exist at DESY but no spares (for parallel testing, etc)

- Three VME-PCI bridges needed
  - All purchased and tested
- 150 mini-SCSI cables needed
  - Purchased 70 but not halogen free (needed at CERN)
  - Need to buy more in 2006
- Three PCs and disk
  - All purchased and tested

Test station at Imperial



Firmware status

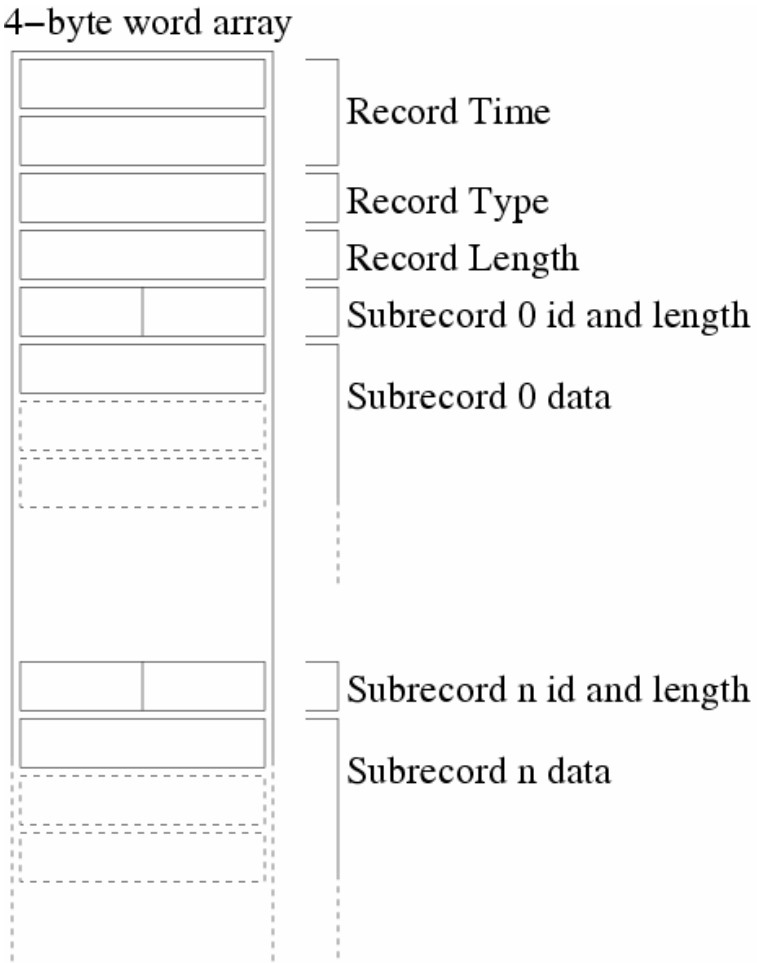
- Three different FPGA firmware designs needed
  - VME: can use CMS version directly; no work needed
  - FE: completely new, but effectively finished
  - BE: two parts to this
    - "Standard" BE: data handling on all CRCs
    - "Trigger" BE: specific for CRC being used for trigger control
- Standard BE firmware is most important issue; not complete
  - Can only buffer up to 500 events, but need 2000
  - Can only buffer in 2MBytes of memory, but need 8MBytes
  - Without both of these, data rate may be reduced by factor of four
- Trigger BE firmware needs work also
  - Trigger data (including detection of multi-particle events) can only be read via slow serial path: limits rate to ~40Hz (c.f. 1kHz, not 100Hz)
  - Need to route trigger data into 8MByte memory so can read via fast Vlink
  - Fallback is not to read these data

DAQ software status

- DAQ online software is based on a state machine
  - States have well-defined status where system is unchanging
    - CRCs configured, buffers full, etc.
  - Transitions between states cause changes to system
    - Download configuration data, take an event, etc.
  - DAQ pushes hardware round state machine by sending transition indicators
    - "Records"
  - System is never stationary once program is started
    - Always running at least slow monitoring
- Many changes since version used at DESY
  - Firmware changes
  - Change to use of LCIO rather than raw data for analysis
  - Experience from DESY run
  - Changes to (my understanding of) AHCAL requirements
  - Major rewrite currently in progress

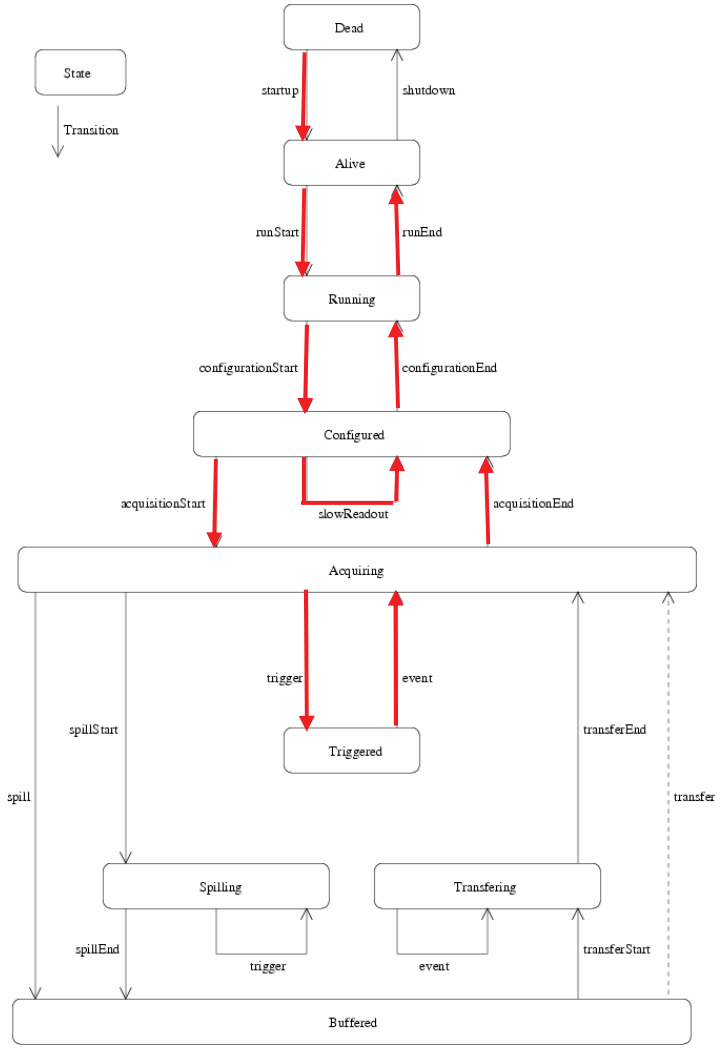
Records and subrecords

- Basic unit of online is a record: two uses
  - Data storage for transport of both upstream and downstream data
  - State machine transition indicator
- DAQ works by pushing records through system to cause transitions
  - Records contain data needed for transition
  - Any data generated by transition is appended to record
  - Final complete record is the raw data and is written to file
- Internal record data structure organised into subrecord objects
  - Map directly to C++ classes
  - Functions provided to access subrecord objects



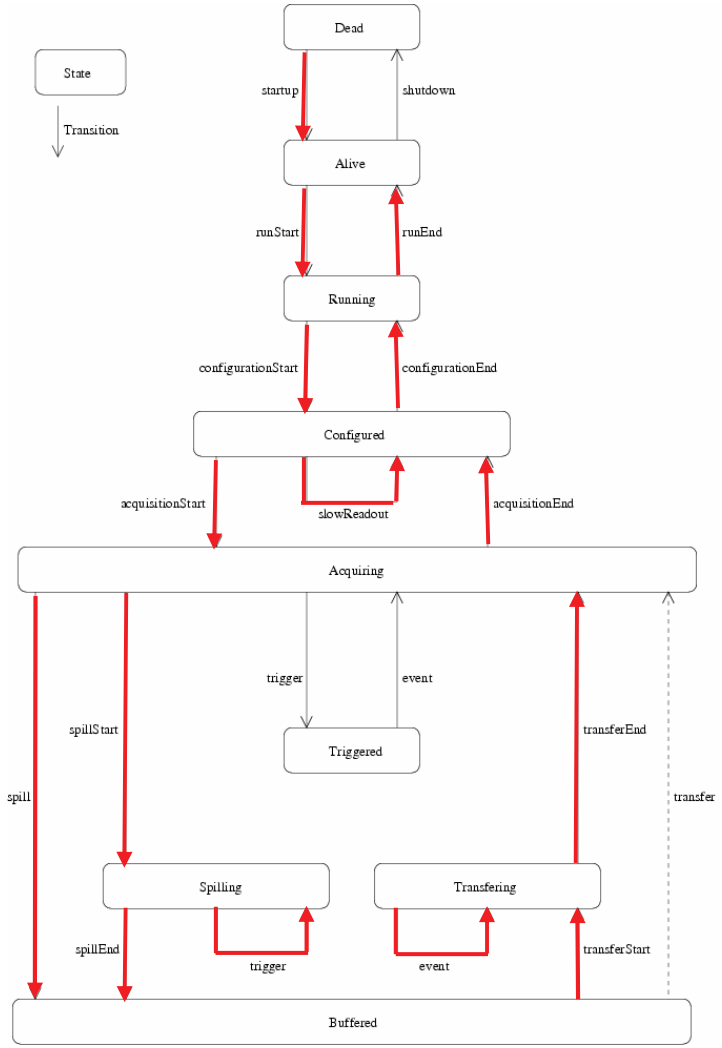
DAQ state machine

- Nested structure: arbitrary number of loops at each level
- Three main types of run
  - Slow monitor run
  - Non-spill run



DAQ state machine

- Nested structure: arbitrary number of loops at each level
- Three main types of run
  - Slow monitor run
  - Non-spill run
  - Spill run

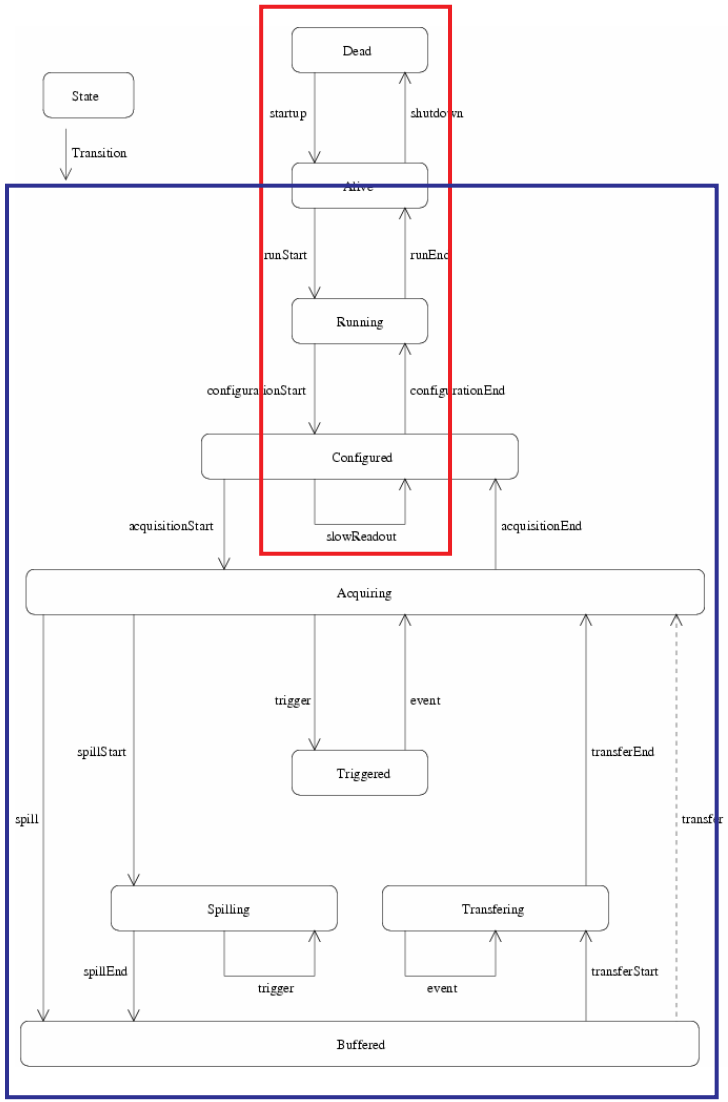


Slow controls/readout status

- Various slow controls and readout data are collected by DAQ
- CRC slow data
  - Temperatures: 22 different probes over surface of board
  - Power: 5 voltage level measurements of backplane inputs
  - Read out as standard during run: ~no work needed
- ECAL power and temperatures
  - Plan to read out via stand-alone PC (not yet existing, AFAIK)
  - Will need to interface to DAQ when it appears
- ECAL stage position
  - Stage controlled by stand-alone PC
  - Readout interface to DAQ tested and working
- AHCAL slow data and stage position
  - All centralised in stand-alone PC (running H1 slow control program)
  - Readout and control interface to DAQ tested; needs further work to be complete

Output native raw data files

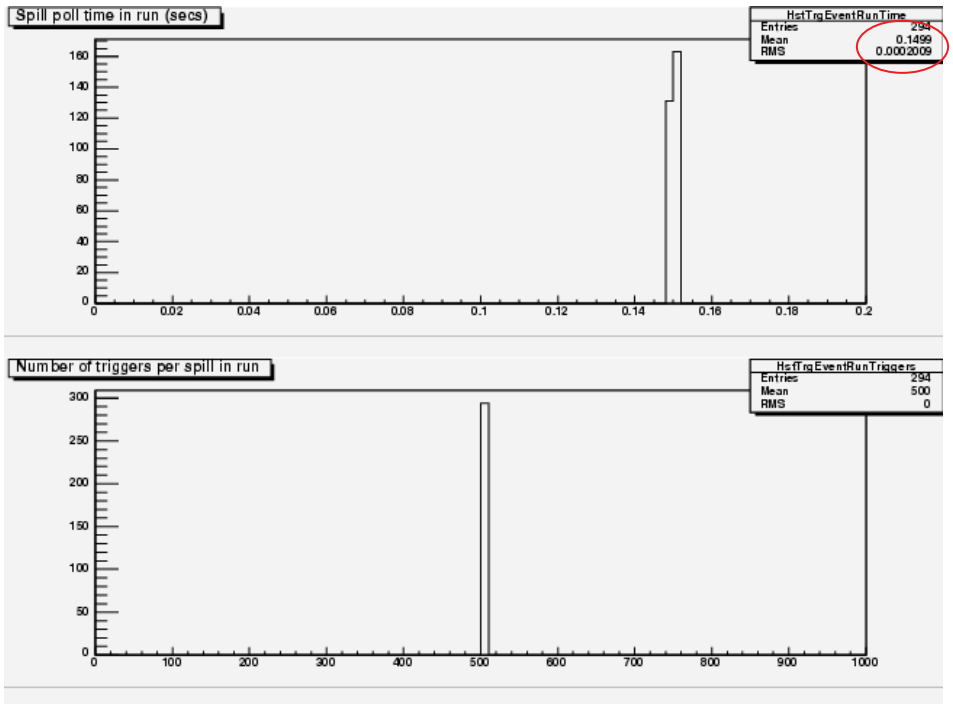
- Continuous running; always recording at least slow data
  - Red box records always written to slow data file
    - Slw<timestamp>.bin
    - All runs; slow monitoring or data taking, even if dummy write run
  - Blue box records written to standard data file
    - Run<runnumber>.bin
    - Only standard data runs if not dummy write
  - Run, configuration and slow readout records duplicated



DAQ trigger performance

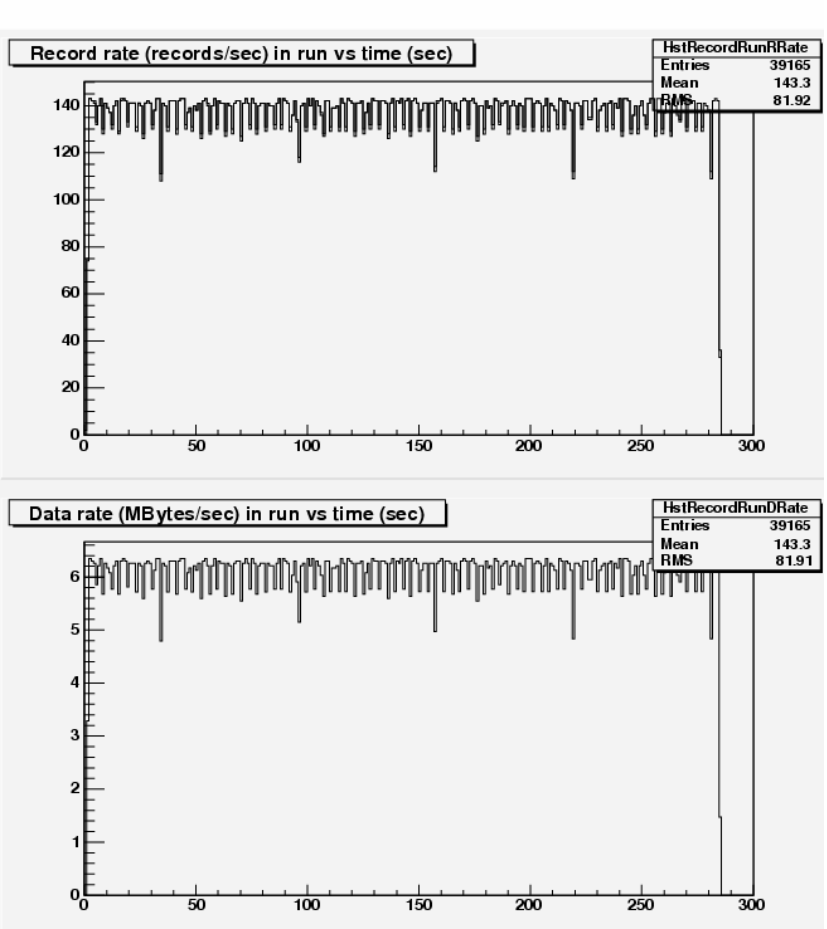
- Need at least 1kHz trigger rate during spill
  - Trigger BUSY is lowered in hardware; time delay set in configuration
  - No software intervention possible in time available
  - Cannot software access registers (e.g. trigger data, event counters)
- Fake “spills” of 500 events each
  - Using built in BE trigger oscillator
  - Average 0.15s; 3.3kHz
- Here limited by
  - ADC conversion cycle
  - Data transfer FE→BE → memory
- Currently take ~200µs
  - Could tune these further if needed

Avg = 0.15s



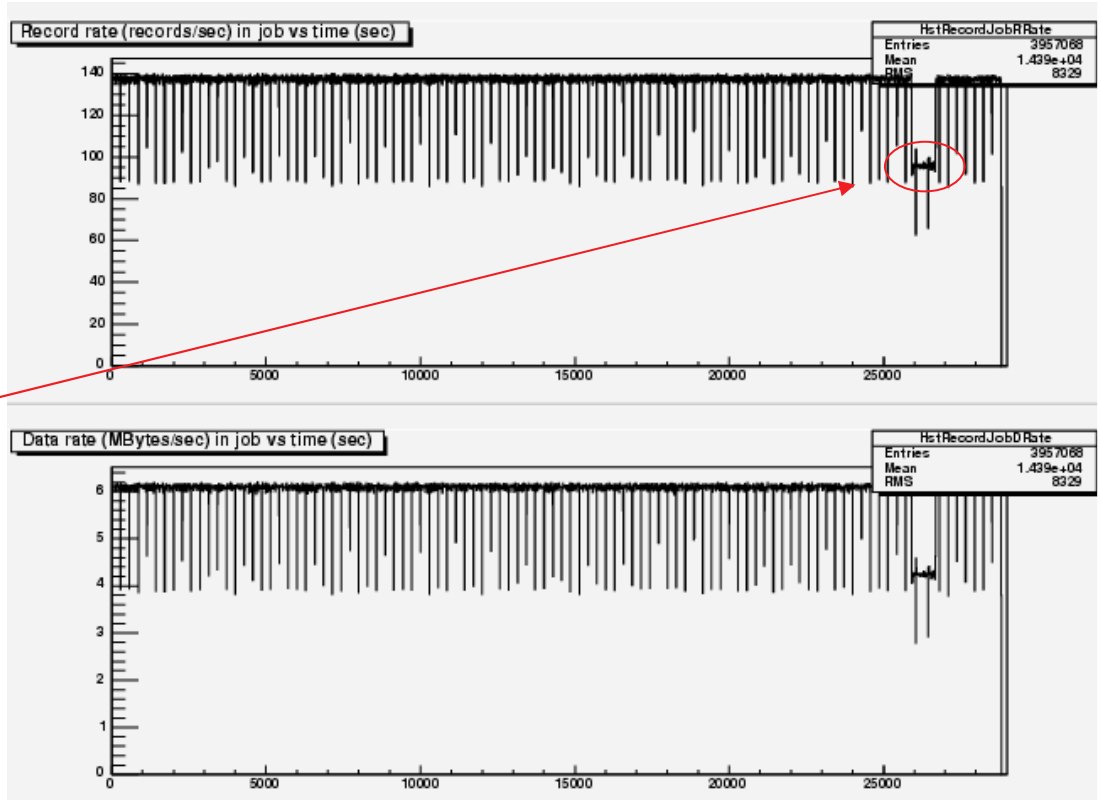
DAQ readout performance

- Need at least 100Hz readout rate after spill
  - With ~50kByte events, this is 5MBytes/s
  - Will read ECAL, AHCAL in parallel; PCs or PCI cards
- Use six real CRCs as for ECAL
  - Fake AHCAL and trigger contributions in software
  - Read in parallel using sockets
  - Software trigger immediately after last read complete
- Typical run history shows
  - Event rate ~130Hz
  - Data rate ~6MByte/s
- Run only lasts ~290s ~5mins
  - Hits file size limit (1.8GBytes)
  - ~39k events in run



DAQ readout performance (cont)

- Really WAS typical run
  - Next run starts automatically
- Left for 100 runs ~8 hours
  - Rates maintained (except when hammering the disk with an offline job!)
  - ~180GBytes, ~4M events written



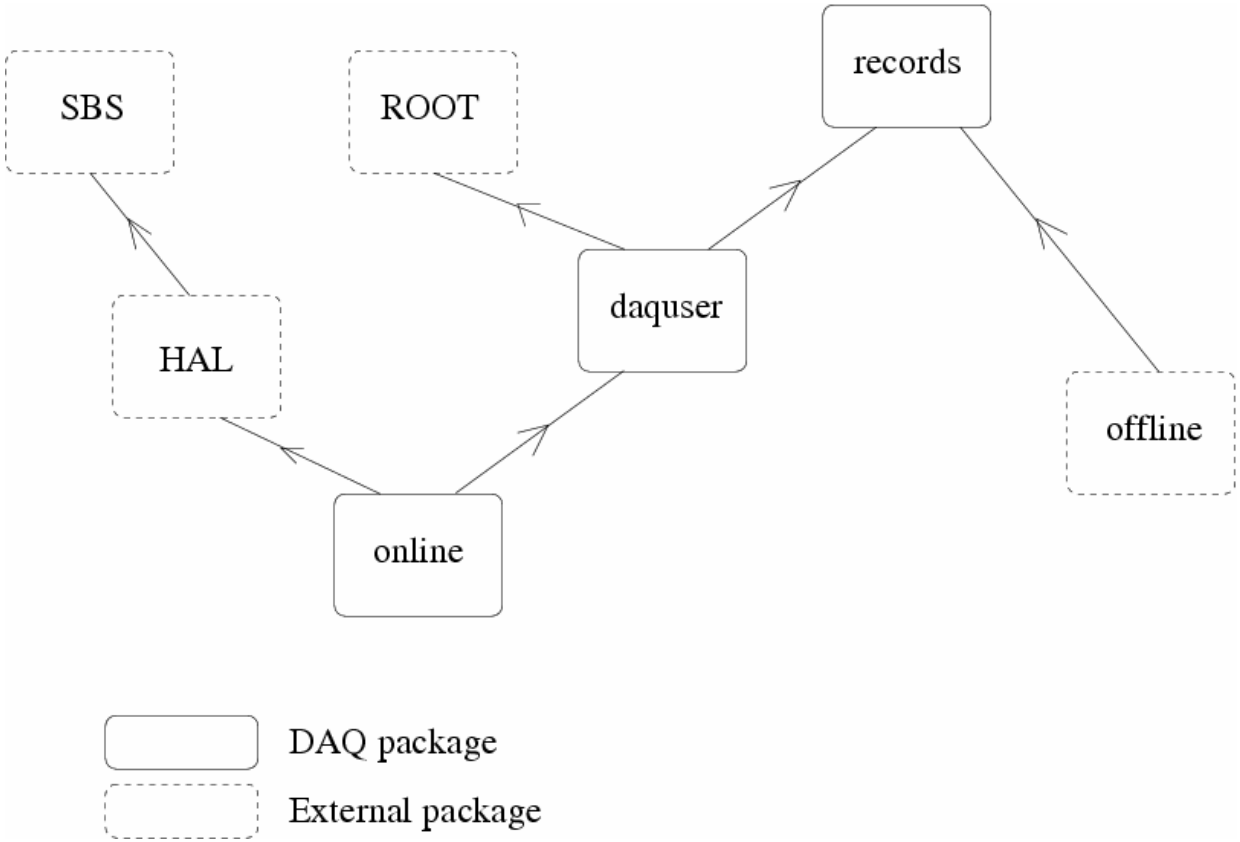
- Five days (Thu 6 Oct to Mon 10 Oct) left running for fun
  - 800 runs, 31M events, 1.4TBytes; all CRC noise
- N.B. Total sample we are aiming for is 100M events, 5TBytes
  - Would only take 16 days if able to continuously run!

Work to be done

- Debug future versions of BE firmware, test new CRCs
  - CRC testing should only be ~weeks; depends on number of problems found
  - Hope the firmware can be finished by early next year
- Complete the major rewrite of online software structure
  - The major task at present; target is end of year
- Test parallel access for two PCI cards in one PC
  - Must have Imperial equipment at DESY for this; need another VME card
  - PCI bus should not limit compared with two VME buses but need to check
- Test socket access for two PCI cards in two PCs
  - May hit network limitations unless we have direct connection
  - Each PCI card reads independently; need to merge records afterwards
- Integrate existing beam line equipment at CERN and FNAL
  - Big uncertainty at present

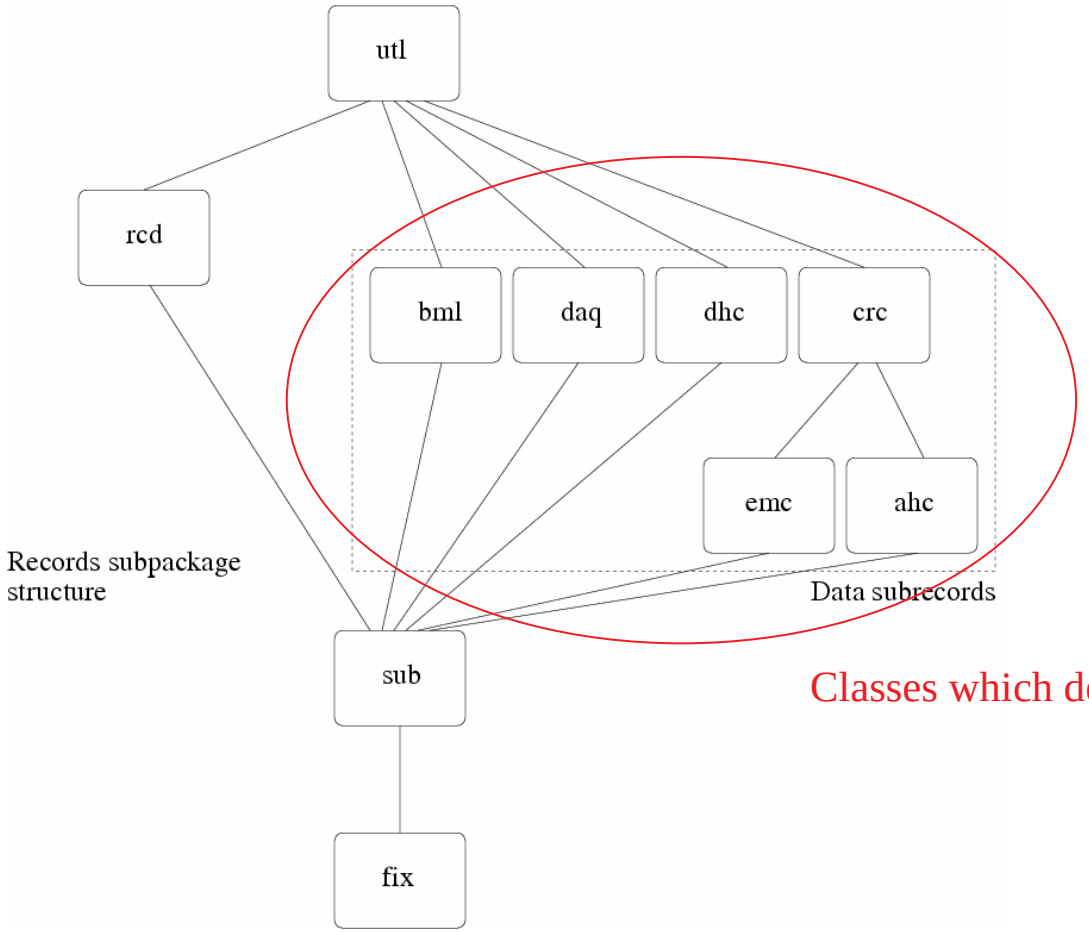
DAQ software packages

- Three major packages
  - records – code for handling records
  - daquser – code for “semi-offline” analysis/monitoring in DAQ system
  - online – true online code



Records software package

- C++ classes to read and access raw data records and subrecords
- Only package needed for “real” offline work; LCIO conversion



Classes which define subrecords

Offline use of records

- Straightforward to read raw data files; all code in records package
- Basic read

```
RcdArena arena;
RcdReaderBin reader;
assert(reader.open("myfile"));  // ".bin" is appended
while(reader.read(arena)) {
    // Use record
}
assert(reader.close());
```

- Accessing the subrecords, e.g.

```
SubAccessor accessor(arena);
std::vector<const DaqRunStart*> v(accessor.access<DaqRunStart>());
```

  gets a list of pointers to the DaqRunStart objects

- Need to check records package classes for data in each subrecord
  - XxxRunData, XxxConfigurationData, XxxEventData for each system
  - What each contains is very system-dependent