Commissioning Procedures and Software for the CMS Silicon Strip Tracker

R. Bainbridge*, J. Fulcher, M. Wingham, *Imperial College London, UK*
G. Bruno, *Universite Catholique de Louvain, BE*
F. Drouhin, *Universite de Haute Alsace, FR*
L. Mirabito, *Institut de Physique Nucleaire de Lyon (IPNL), FR*
D. Vintache, *Universite Louis Pasteur, FR*

Abstract

The CMS silicon strip tracker (SST) requires sophisticated procedures to configure, synchronize and calibrate its control and readout systems. These procedures are implemented within the SST data acquisition software, that uses both the CMS online and offline software frameworks. This design ensures that both the local computing resources allocated to the tracker sub-detector and the global resources provided by the online computing farm can be used transparently. The former option will the default configuration during detector Start-Up and the latter offers significant improvements in detector readout speeds and CPU processing power, thus potentially reducing turn-around times between physics runs. We present an overview of the SST data acquisition system, the commissioning procedures and their software implementations within the online and offline frameworks.

DATA ACQUISITION SYSTEM

The CMS silicon strip tracker is unprecedented in terms of its size and complexity, providing a sensitive area of >200 m^2 and comprising 10^7 readout channels. Fig. 1 shows a schematic of the control and readout systems.

The SST control system [1] comprises 300 "control rings" that start and end at the off-detector Front-End Controller (FEC) boards and is responsible for distributing slow control commands, clock and Level-1 triggers to the front-end electronics. The signals are transmitted from the FECs to front-end digital opto-hybrids via digital optical links, and then electrically via 'token rings" of Communication and Control Units (CCUs) to the front-end electronics.

The SST readout system is based on the front-end APV25 chip readout chip [2], analogue optical links [3] and an off-detector Front-End Driver (FED) processing board [4]. The APV25 chip samples, amplifies, buffers and processes signals from 128 channels of a silicon strip sensor at the LHC collision frequency of 40 MHz. On receipt of a Level-1 trigger, pulse height and bunch-crossing information from pairs of APV25 chips are multiplexed onto a single line and the analogue data are converted to optical signals before being transmitted via optical fibres to the off-detector FED boards. The FEDs digitize, process and format the pulse height data from up to 96 pairs of APV25 chips, before forwarding zero-suppressed data to the DAQ online farm.

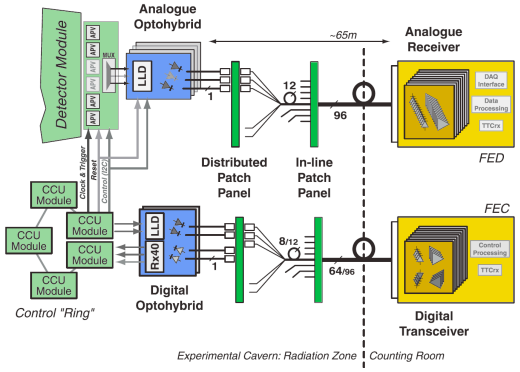

Figure 1: The SST control and readout systems.

COMMISSIONING THE SST

The complex control and readout systems require sophisticated procedures to bring the detector into an operational state that is suitable for physics. These commissioning procedures comprise several independent tasks that fall into one of the following categories: *optimization* of the hardware configurations; *synchronization* of the entire system, both internally and to LHC collisions; determination of low-level *calibration* constants that are used by the hardware and, in some cases, the CMS reconstruction software.

These procedures will be used to validate the operational functionality and performance of the detector during the Start-Up phase of the experiment and will also be performed (with varying frequencies) between fills to guarantee optimum detector performance during the subsequent period of data taking.

Several procedures have already been defined that concern either individual electronic components of the readout system or system-wide aspects. The most pertinent tasks are identified below:

- *Detector partitioning*: automated hardware scans that detect all front-end devices sharing a common trigger source; automated detection of the (38k) optical link fibre connections between the front-end modules and off-detector electronics (FEDs).

* robert.bainbridge@cern.ch

- *Front-End APV25 Chip*: tuning of the analogue pulse shape; tuning of various bias and gain settings.
- *Readout optical link system*: gain matching across the entire system; optimization of dynamic range usage.
- *Front-End Driver*: tuning of the ADC sampling time; determination of calibration constants for the zero-suppression algorithms (and reconstruction software).
- *Timing*: "internal" synchronization of the front-end, which effectively accounts for signal propagation delays in the control system [5]; "global" synchronization to LHC collisions and other sub-detectors, which is crucial as signal is attenuated by $\sim$4% per ns misalignment for nominal operating modes.

DATA ACQUISITION SOFTWARE

The SST data acquisition software is based on the CMS XDAQ online framework [6], which provides a core set of services and tools, including: a fast communication protocol providing peer-to-peer messaging between processes (known as *xdaqApplication*s) registered with the framework; a slower communication protocol and a finite-state machine schema for configuration of the framework processes; and standard event builder and memory management tools.

The SST data acquisition (DAQ) software [7] implements the procedures required by the various commissioning tasks. The implementation comprises dedicated DAQ loops that determine optimized configuration parameters and calibration constants from reconstructed calibration pulses, timing delay curves, dynamic range curves and other features of the APV25 data stream. These optimized configurations and calibrations are then uploaded to the SST online configuration database and provide the basis for subsequent commissioning tasks or physics run.

Each DAQ loop is defined by a run comprising several consecutive spills of events, separated by periods when the trigger is inhibited and the configuration of a device is changed via the control system. The configuration of several devices of the same type are usually tuned in parallel.

The DAQ loops require communication between various "Supervisor" processes that control the trigger system, hardware configuration, readout, event building and data analysis. The XDAQ framework provides this functionality and allows to automate the data acquisition loops, so reducing the need for repetitive run control sequences and complex book keeping. Consequently, this accelerates detector commissioning and start-up.

During *global* data acquisition, the FED data are read out via S-Link [8] and forwarded to the global online computing farm. *Local* data acquisition involves the data being read out via the VME backplane. A schematic of the software architecture for local data acquisition (readout stream only) is shown in Fig. 2. The data are accessible via the crate controller PCs and these PCs are also used to host the various processes that configure and control the FEDs, provide event building and data analysis. The schema is

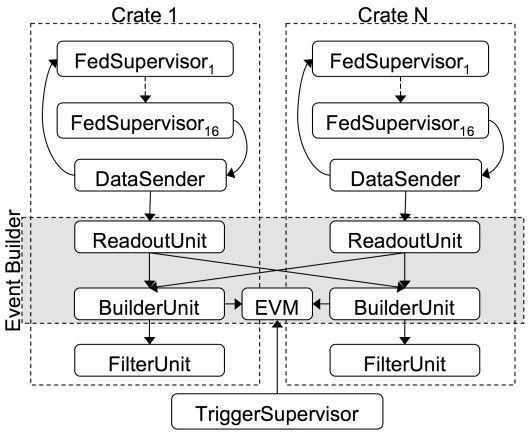

Figure 2: Software for local DAQ (readout stream only).

applied at the VME crate level and uses the following *xdaqApplications*:

- *FedSupervisor*: controls data capture, data processing and readout of the FEDs. There is one supervisor per FED (and therefore up to 16 per crate).
- *DataSender*: collects the FED data from the FedSupervisors and builds "super-fragments" that are forwarded a ReadoutUnit.
- *ReadoutUnit / BuilderUnit*: these are components of the standard event building toolkit provided by the XDAQ framework. The ReadoutUnit processes forward (on request) super-fragments from the same event to a single BuilderUnit process via a gigabit Ethernet switch, which builds the complete event.
- *FilterUnit*: requests events from the BuilderUnit.

Additional processes that are hosted on other local computing resources include:

- *FecSupervisor*: allows to configure all front-end devices, such as the APV25 readout chip.
- *TriggerSupervisor*: allows to define and control trigger patterns that are local to the tracker sub-detector.
- *Event Builder Manager*: receives triggers from the TriggerSupervisor and controls the event building.
- *TrackerSupervisor*: communicates with all "Supervisor" processes and "steers" the data acquisition loops.

HISTOGRAM-BASED DATA ANALYSIS

The DAQ software presently uses a Root-based histogram analysis known as *RootAnalyzer* [7] that provides optimized hardware configurations and calibration constants (that are inferred from the histograms). The RootAnalyzer module can also be registered with the XDAQ framework as a *xdaqApplication* (not shown in Fig. 2) and receives events directly from the FilterUnit process.

The RootAnalyzer module was originally developed for small beam test and laboratory set-ups that yielded small event data sizes, typically $\ll$1 MB. Thus, a single RootAnalyzer process was sufficient to process all events and

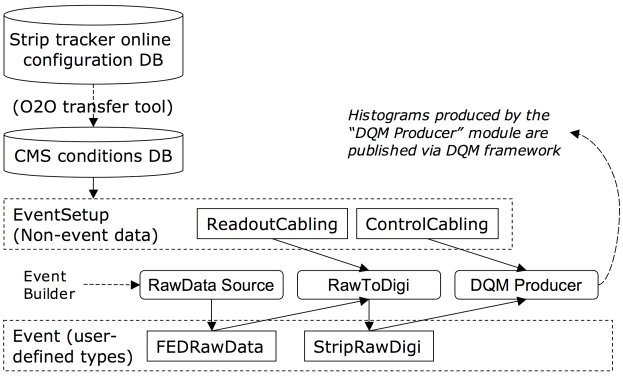

Figure 3: DQM "producer" components within CMSSW.

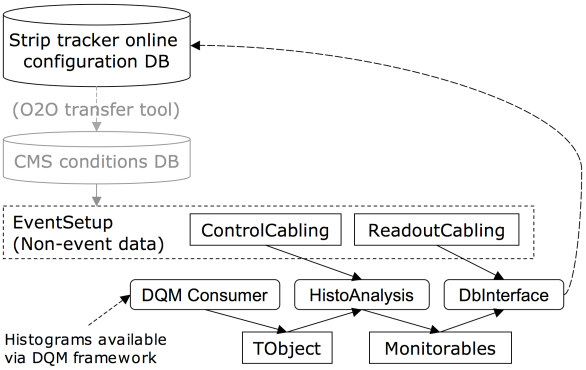

Figure 4: DQM "consumer" components within CMSSW.

the module was developed on the premise that the analysis received all events. Consequently, the event builder in the present system comprises multiple ReadoutUnits but only a single BuilderUnit, from which the RootAnalyzer module requests events (via the FilterUnit). This architecture does not allow parallel event processing and is therefore not scalable. The RootAnalyzer software has been used to commission readout systems comprising up to 50 FEDs, but experience with these systems suggest that a change in design is required to allow parallel event processing and the handling of larger systems, such as the final strip tracker readout system of 440 FEDs.

Much of the functionality and implementation provided by the RootAnalyzer software is currently being ported to the offline CMSSW software framework [9]. The offline software can be used in an online context within the online computing farm in order to provide a software-based high-level trigger and event filtering. This is achieved by implementing the FilterUnit *xdaqApplication* within CMSSW so that is can be used within the XDAQ-based online data acquisition system. This feature also allows the ported commissioning software to be used by the online data acquisition system.

The motivation to port the analysis software to CMSSW is the availability of many useful services and tools within the offline framework. One such service is the Data Quality Monitoring (DQM) framework [10] that allows remote *consumers* to subscribe to histograms that are defined and published by one or more *producers*. One important feature of the framework is the "collation" functionality, which allows a remote consumer to collate the contents of histograms from multiple producers.

The DQM framework therefore allows the commissioning software to define producers that can run on multiple FilterUnit nodes. The event builder distributes events between the different nodes and each of the producers publish and fill identical sets of histograms. The DQM framework provides the mechansim to collate the contents of histograms of each producer and provide the "complete" histograms on a consumer node, which hosts the histogram analysis code. Although the number of histogram bins per node can be large (as many as $\sim 10^7$), the data traffic between the producers and consumers is negligible as the histograms need only be updated at the end of a run. This architecture is easily scalable as the number of producers is configurable.

In addition, the CMSSW framework provides a conditions database infrastructure, access to the detector geometry and low-level reconstruction software, all of which are used by the commissioning analysis software.

The decision to port the commissioning analysis to CMSSW has important consequences. The chosen design means that either local or global computing resources (the online "filter" farm) may be used. The "global" DAQ configuration has two distinct advantages over local DAQ. Firstly, significantly higher trigger rates can be achieved via S-Link readout, O(kHz), with respect to O(Hz) for VME readout. Secondly, the online computing farm offers unparalleled processing power, with hundreds of CPU units avaiable with respect to tens for the local DAQ.

CMSSW is based around the Event Data Model [11], for which a basic premise is that all user-defined types are contained within a special container called the *Event*. Plug-in processing modules can only read from and, in some cases, write to the Event. Interaction between modules is prohibited. All non-event data are accessed via the *EventSetup*.

Fig. 3 shows the commissioning software components related to the DQM "producer". The *RawData Source* module requests events from the event builder (via the FilterUnit *xdaqApplication*). The *RawToDigi* module unpacks the FED raw data and creates Digis (basic hit information) that are used by the reconstruction software. These digis are then used to fill the histograms published by the *DQM Producer* module. The hardware configuration during a DAQ loop is propagated from the XDAQ framework processes to the analysis within the FED data streams, so that the histograms are filled appropriately. The histograms are arranged so as to reflect the logical structure of the control or readout systems using cabling information available via the *EventSetup*.

Fig. 4 depicts the software components that operate on the DQM "consumer" node. The *DQM Consumer* module

subscribes to the relevant histograms via the DQM framework. These histograms are then used as input to the *HistoAnalysis* modules that extract optimized hardware configurations and calibration constants, which are then appropriately formatted and uploaded to the strip tracker online configuration database via the *DbInterface* module.

UNPACKING THE RAW EVENT DATA

The commissioning analysis heavily uses the low-level reconstruction software within CMSSW and there have been significant developments within this area recently. The first stage in reconstruction is provided by the *RawToDigi* module, which creates Digis (basic hit information) using the raw data contained within the FED data buffers, and the assignment of Digis to the appropriate detector objects using readout cabling maps provided by the *EventSetup*.

A *DigiToRaw* module provides the reverse process and is a useful test facility. It generates FED data buffers that encode the signal information provided by an input sample of Digis (which is typically created from a simulated event). This module can be used to generate samples of FED buffers containing zero-suppressed data that reflect the event data from the SST during a physics run under nominal operation conditions.

This test facility has been used to measure both event data sizes and unpacking times as a function of detector occupancy. The event size is 344 kB for zero occupancy (due to various header information within the FED data buffer) and increases linearly at a rate of 175 kB per percent occupancy (averaged across the entire tracker). For a tracker-averaged occupancy of 1.2 %, the event size is 519 kB.

Fig. 5 plots the unpacking time of the raw event data as a function of the tracker-averaged occupancy, as measured within ORCA (top curve) and CMSSW (bottom curve), using a 2 GHz processor. The unpacking time of the RawToDigi module comprises: extracting the FED data buffers from the Event; extracting the hit information from the FED data buffers; creating the Digi collections; using the cabling maps; and writing the Digi collections to the Event. The unpacking time also increases linearly for non-zero occupancies, reaching 150 ms for a tracker-averaged occupancy of 1.2 %.

This measurement has consequences for the tracker-specific high level trigger processes that run on the online computing farm, which are required to provide event filtering (using tracker data) within 300 ms per event on a 1 GHz processor node [12]. Thus, unpacking the FED raw data is expected to take approximately the entire allocation, which provides a strong argument for regional reconstruction and "unpacking-on-demand".

SUMMARY AND CONCLUSIONS

The sophisticated commissioning procedures required by the CMS silicon strip tracker are well understood and have been implemented within the SST data acquisition software. The DAQ software, developed within the CMS XDAQ online framework, is a mature and stable project that has been used within the SST detector integration centres and will be used by the final experiment.

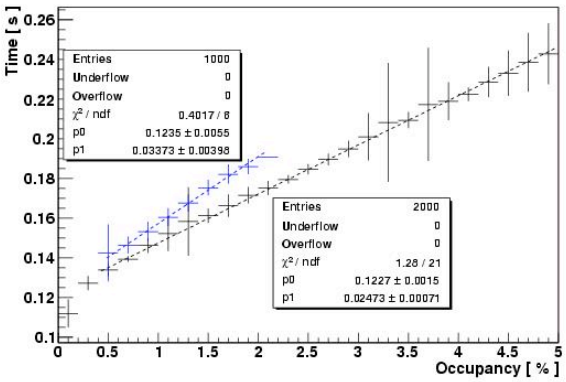

Figure 5: Time taken to unpack the raw event data as a function of detector occupancy, within ORCA (top curve) and CMSSW (bottom curve).

Recent developments include the port of the analysis modules within the DAQ software to the CMSSW offline framework. This allows distributed event processing and provides a scalable analysis framework that is suitable for the final detector and the large event sizes expected. These developments are expected to be completed in time for the CMS Cosmic Challenge and the 25 % commissioning tests in the Tracker Assembly Hall in the spring and summer of 2006, respectively.

REFERENCES

[1] F. Drouhin *et al*, CMS-CR 2004/032.

[2] M. French *et al*, Nucl. Instr. Meth. A534 (2001) 359-365.

[3] J. Troska *et al*, Nucl. Sci. Symp. 1 (2002) 233-237.

[4] J. Coughlan *et al*, CERN-LHCC-2002-034, p.296-300

[5] K. Gill *et al*, CERN-LHCC-2003-055, p.289-293

[6] `http://xdaqwiki.cern.ch/`

[7] L. Mirabito *et al*, CMS-IN 2003/021, CMS-IN 2004/019

[8] H.C. van der Bij *et al*., IEEE Trans. Nucl. Sci, 44/3, p.398

[9] `cmsdoc.cern.ch/cms/cpt/Software/html/General/`

[10] C. Leonidopoulos *et al*, these proceedings.

[11] `https://uimon.cern.ch/twiki/bin/view/CMS/EDM`

[12] CERN-LHCC-2005-023