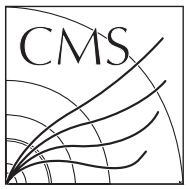
Available on CMS information server

CMS CR -2017/041



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

17 February 2017

SWATCH Common software for controlling and monitoring the upgraded CMS Level-1 trigger

Christos Lazaridis for the CMS Collaboration

Abstract

The Large Hadron Collider at CERN restarted in 2015 with a higher centre-of-mass energy of 13 TeV. The instantaneous luminosity is expected to increase significantly in the coming years. An upgraded Level-1 trigger system is being deployed in the CMS experiment in order to maintain the same efficiencies for searches and precision measurements as those achieved in the previous run. This system must be controlled and monitored coherently through software, with high operational efficiency.The legacy system is composed of approximately 4000 data processor boards, of several custom application-specific designs. These boards are organised into several subsystems; each subsystem receives data from different detector systems (calorimeters, barrel/endcap muon detectors), or with differing granularity. These boards have been controlled and monitored by a medium-sized distributed system of over 40 computers and 200 processes. Only a small fraction of the control and monitoring software was common between the different subsystems; the configuration data was stored in a database, with a different schema for each subsystem. This large proportion of subsystem-specific software resulted in high long-term maintenance costs, and a high risk of losing critical knowledge through the turnover of software developers in the Level-1 trigger project.The upgraded system is composed of a set of general purpose boards, that follow the MicroTCA specification, and transmit data over optical links, resulting in a more homogeneous system. This system will contain the order of 100 boards connected by 3000 optical links, which must be controlled and monitored coherently. The associated software is based on generic C++ classes corresponding to the firmware blocks that are shared across different cards, regardless of the role that the card plays in the system. A common database schema will also be used to describe the hardware composition and configuration data. Whilst providing a generic description of the upgrade hardware, its monitoring data, and control interface, this software framework (SWATCH) must also have the flexibility to allow each subsystem to specify different configuration sequences and monitoring data depending on its role. By increasing the proportion of common software, the upgrade systems software will require less manpower for development and maintenance. By defining a generic hardware description of significantly finer granularity, the SWATCH framework will be able to provide a more uniform graphical interface across the different subsystems compared with the legacy system, simplifying the training of the shift crew, on-call experts, and other operation personnel.We present here, the design of the control software for the upgrade Level-1 Trigger, and experience from using this software to commission the upgraded system.

Presented at *CHEP 2016 22nd International Conference on Computing in High Energy and Nuclear Physics*

SWATCH: Common software for controlling and monitoring the upgraded CMS Level-1 trigger

Simone Bologna[1], Karol Bunkowski[2], Giuseppe Codispoti[3], Glenn Dirkx[4], Carlos Ghabrous Larrea[4], Christos Lazaridis[5], Joschka Lingemann[4], Lukasz Kreczko[6], Alessandro Thea[7], Tom Williams[7]

[1]Universita & INFN - Milano-Bicocca, Italy.
[2]University of Warsaw, Poland.
[3]Universita & INFN - Bologna, Italy.
[4]CERN, Switzerland.
[5]University of Wisconsin-Madison, U.S.A.
[6]University of Bristol, U.K.
[7]STFC - Rutherford Appleton Lab., U.K.

E-mail: christos.lazaridis@cern.ch

Abstract. The Large Hadron Collider at CERN restarted in 2015 with a 13 TeV centre-of-mass energy. In addition, the instantaneous luminosity is expected to increase significantly in the coming years. In order to maintain the same efficiencies for searches and precision measurements as those achieved in the previous run, the CMS experiment upgraded the Level-1 trigger system. The new system consists of the order of 100 electronics boards connected by approximately 3000 optical links, which must be controlled and monitored coherently through software, with high operational efficiency. These proceedings present the design of the control software for the upgraded Level-1 Trigger, and the experience from using this software to commission and operate the upgraded system.

1. Introduction

After a two-year shutdown, the Large Hadron Collider (LHC) at CERN, restarted collisions in 2015, with a significantly higher proton-proton centre-of-mass energy, reaching 13 TeV. The Level-1 trigger system of the Compact Muon Solenoid (CMS) experiment selects 100 KHz of the most interesting collision events from the 40 MHz rate delivered by the LHC. This is achieved in a time window of 3.2 μs using coarse data from the calorimeter and muon detectors, while the full resolution data is held in pipeline memories in the front-end electronics.

The Level-1 trigger of the CMS experiment underwent through a major upgrade during 2015 and early 2016 [1] in order to cope with the increasingly demanding beam conditions delivered by the LHC. The VME-based system used in the Run-1 and the beginning of Run-2 of the LHC has been replaced by custom-designed processors based on the uTCA specification. The new design offers increased flexibility due to the use of high-bandwidth optical links between processor cards, modern FPGAs and larger memories for the trigger logic. In addition, the diversity of the hardware has been greatly reduced to a small number of general-purpose boards. This upgrade plan involved changing about 90% of the previous Level-1 trigger hardware. As

a natural consequence, a significant fraction of the firmware, low level drivers, control software and the databases that are used to configure, control and monitor the system had to be adapted to the new system.

The increased homogeneity of the hardware offered the possibility for a similar consolidation of the online software, identifying and taking advantage of common components wherever possible. In these proceedings, we present the design of the software that has been developed and used to control and monitor the upgraded Level-1 trigger system and the experience obtained from using this software to commission and operate the upgraded system.

2. The upgraded CMS Level-1 trigger

The CMS Level-1 trigger is composed of 9 subsystems (CPPF to be installed in 2017 during the Extended Technical Stop), connected as shown in Figure 1. Each subsystem comprises one or more processor boards, housed in uTCA crates capable of hosting up to 12 Advanced Mezzanine Card (AMC) modules. A common module, the AMC13 [2], provides the clock, data acquisition services and a feedback mechanism for the Trigger Throttling System, that monitors the status of the data buffers to avoid them becoming full. The data processing logic within each of the AMC cards follows a common model, shown in Figure 2. This is implemented on modern Xilinx Virtex-7 FPGAs and data is transported via high-speed serial optical links. The logic of each board follows a common pattern with each board consisting of:

- an *algorithm* block that performs particle reconstruction and processes the data
- a *Trigger Timing and Control* (TTC) block that receives the clock and fast (fixed latency) control commands
- zero or more *ports* and
- a *readout* block that sends the data from the input/output buffers to the Data Acquisition system of CMS. The recorded data is then used to validate the correct functionality of the firmware.

Three varieties of boards have been designed based on this common processor model, each optimized for a different task:

- the *Calorimeter Trigger Processor* cards have dedicated connections to the uTCA backplane for data sharing within the same crate
- the *Master Processor* cards are optimized for data sharing via a large number of optical inputs and outputs and
- the *Muon Track Finder* boards are capable of providing the large memory resources necessary for the CMS Muon Trigger.

3. The SWATCH Control Software

The similarities in the upgraded Level-1 trigger hardware lead naturally to a generic software design that can fully exploit them. The SWATCH (SoftWare for Automating the conTrol of Common Hardware) framework provides a set of interfaces for controlling and monitoring the hardware of the trigger system while remaining independent of the driver software, thus reducing code and effort duplication for the subsystems.

The architecture of each subsystem (processors, ports and interconnections) is stored in subsystem-agnostic data structures with each component represented by an abstract interface class. Subsystem-specific functionality is implemented in classes inheriting from the generic interface classes. The objects that represent a subsystem are built using the factory pattern, with the subsystem-specific implementations of the generic system, process and DAQ-TTC manager classes registered in subsystem-specific "plugin" libraries.

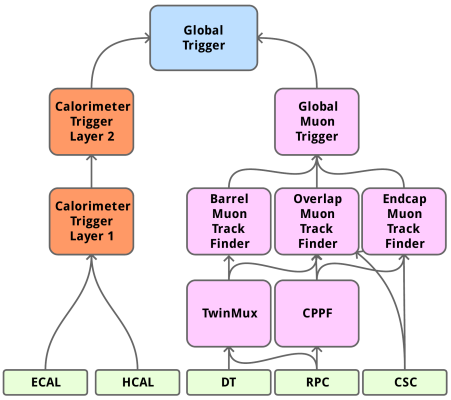


Figure 1. Illustration of the CMS Level-1 trigger system architecture. The detector back-end electronics are shown at the bottom of the diagram; on the left is shown the calorimeter processing chain and on the right the muon processing chain. The Global Trigger subsystem generates the final Level-1 accept decision. Each box corresponds to one or more crates of electronics boards.

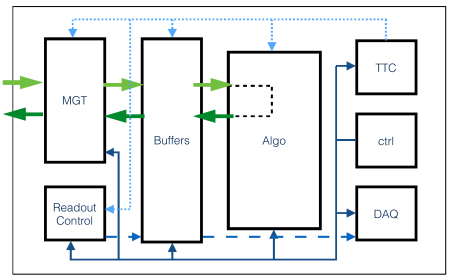


Figure 2. Diagram showing the common processor model for data processor boards.

3.1. Configuration

In order to take full advantage of the common architecture across subsystems as identified in the processor model, the SWATCH framework provides a generic interface for controlling each electronics board based on three concepts:

- *Commands*: Stateless actions and the basic building block of hardware control. They are represented by an abstract base class, customized by subtype polymorphism for each subsystem.
- *Command sequences*: A stateless action, composed of a series of commands that is executed in sequence.
- *Finite State Machine* (FSM): A stateful set of actions; each FSM consists of a set of named states. A change between two states is initiated and determined by a *transition*. Each transition is specified by a list of commands that are executed in sequence. The FSM that implements the set of transitions required for standard running in CMS is automatically defined by the framework and the transition actions are defined via subsystem-specific command sequences defined in the constructors of subsystem-specific classes.

The required parameters for each command are registered via a generic interface and their values are retrieved through the abstract *gatekeeper* interface, designed to provide a uniform communication protocol to SWATCH, independent of the source of the configuration parameter values. This was particularly useful during the initial commissioning and testing of the upgraded

Level-1 trigger hardware as it enabled quick testing using file-based configurations either at institutes outside of CERN or at the CMS Control Room. The backend-agnostic gatekeeper interface ensured a smooth transition when the control software was switched to reading the parameter values from the database when regular data taking commenced. A schematic overview of the SWATCH hardware configuration interfaces can be seen in Figure 3

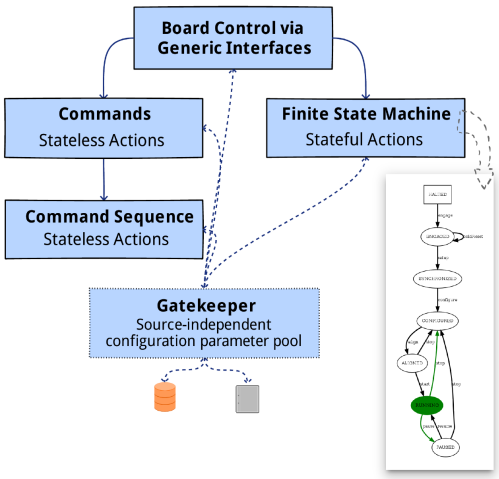


Figure 3. The SWATCH hardware configuration interfaces.

3.2. Monitoring

In order to efficiently monitor the status of the hardware, SWATCH offers two mechanisms inheriting from a generic monitoring interface (Figure 4):

- *Metrics* represent individual items of monitored data, read directly from the hardware. Each metric can have associated error/warning conditions which determine its state.
- *Monitorable objects* can contain metrics and/or other monitorable objects as child nodes. The overall state of a monitorable object is determined by the cumulative status of all its child metrics and monitorable objects. In SWATCH, each of the common firmware components within processors and DAQ-TTC managers are represented as a monitorable object.

The history of all monitored information is stored in an Oracle database, with common visualization tools under development. For the future, using a NoSQL database and Elastic Search is under consideration in order to take advantage of the greater flexibility offered in both data storage and retrieval.

3.3. Database

All SWATCH configuration data are stored using an Oracle relational database. A single, common database schema exists for all the trigger subsystems. This eases operations as the development and maintenance of the database structure is controlled by a small group of experts and in addition all information and data formats are standardized across trigger subsystems.

The values for the configuration parameters are split in four distinct XML-based modules, each containing a set of criteria belonging to a specific group:

- *Hardware* contains the full system architecture description (boards, links etc.)

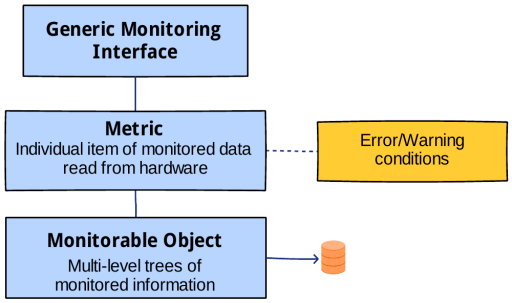


Figure 4. The SWATCH hardware monitoring interfaces.

- *Infra* includes board timing alignment points
- *Algo* includes calibration factors, physics-related thresholds
- *Run Settings* contains hardware masks, monitoring settings etc

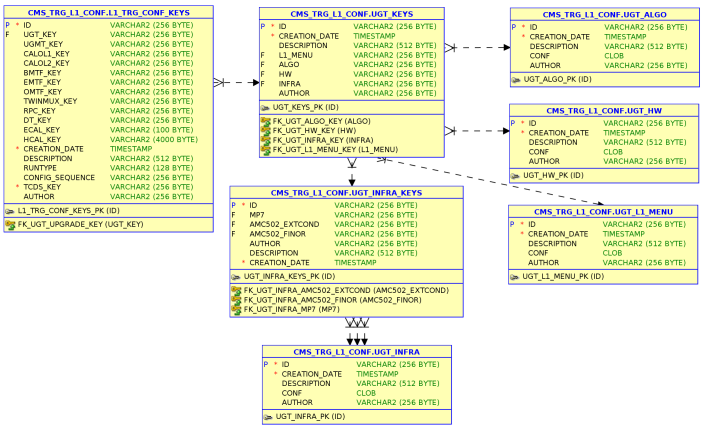


Figure 5. A part of the SWATCH configuration database schema showing a subset of the Global Trigger configuration database.

3.4. Distributed control

Besides hardware control and monitoring, the Level-1 trigger software has to provide the means to integrate the system into the CMS run control hierarchy so that it can be operated by the shift crew at the CMS control room.

The Trigger Supervisor [3] is a C++ framework which provides the required interfaces to create online applications that can be controlled via web graphical user interfaces and can communicate over the network by exchanging SOAP messages. Global running is coordinated by the Central Cell, a Trigger Supervisor application orchestrating the configuration of all Level-1 trigger components, enforcing the rules for their appropriate starting order and the relative configuration timing alignment between subsystems. All these components are using SWATCH for hardware control and the Trigger Supervisor to create the user interface and to handle network communications. The Central Cell is controlled by the Level-1 Trigger Function Manager, a Java application responsible for relaying messages from the central run control. All user interfaces are rendered using HTML5 technologies and Polymer [4].

3.5. Baseline SWATCH system

To further facilitate the development of subsystem code, a basic SWATCH sceleton code was developed and made available, through which monitoring and control features are exposed. Thanks to the common processor and system achitecture, a set of subsystem-agnostic common panels in order to control and monitor the hardware were developed (Figure 6). These allow subsystem experts and shift crew to:

- Control the hardware by executing commands, command sequences or FSM transitions and get immediate feedback on on the results of these actions
- View the monitoring status of a system and its individual processors, ports etc.
- Check monitored metric values and their warn/error conditions
- Plot metric values in real time for easy inspection.

The added flexibility from the subsystem-agnostic panels significantly reduced the necessary personpower needed to develop the online software during the commissioning period.

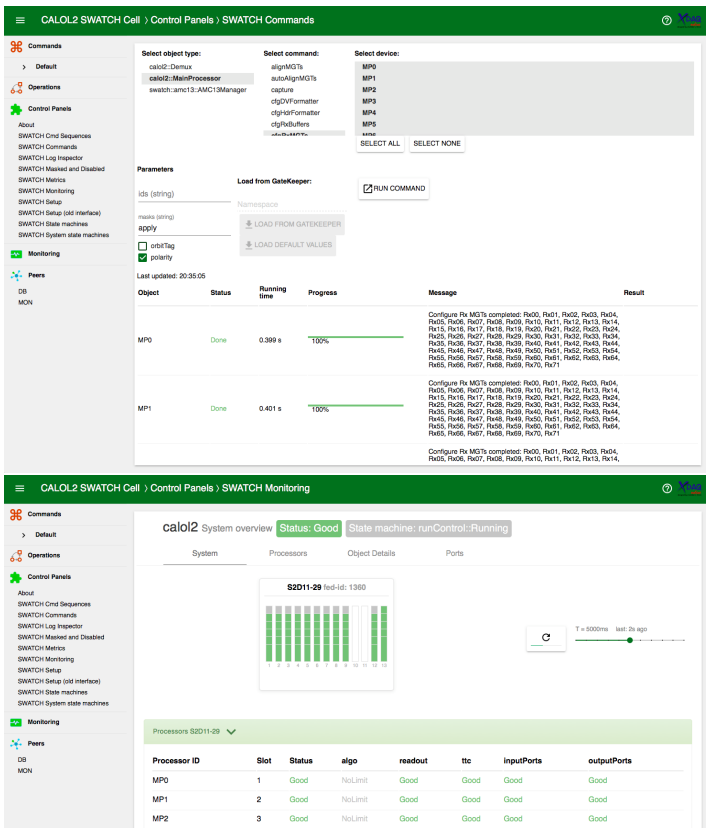


Figure 6. SWATCH generic control and monitoring panels.

4. Commissioning and operations

Thanks to the SWATCH architecture, a high operational efficiency was achieved in a short period of time, with only one month between the first integration of the online software in the central run control and the first commissioning with beam. This was largely due to its hardware-agnostic design which takes advantage of the commonalities between the electronics boards and maps them on the control software. The low-level hardware drivers were SWATCH-independent

and the bridge was realized via "plugin" libraries. However, due to the existence of common hardware between trigger subsystems, only one SWATCH "plugin" had to be developed per board type.

Extensive hardware-independent testing allowed many issues to be diagnosed and patched before software deployment. The framework was continuously tested by a comprehensive nightly build test suite and a parallel installation that mirrored the production system using "dummy" hardware was set up to test overall configuration.

5. Conclusions

The SWATCH software framework provides a generic way to control and monitor the upgraded CMS Level-1 trigger hardware. It facilitated to successfully replace a long-running and stable trigger system despite the short transition time. Its subsystem-agnostic approach led to significant advantages over the previous control and monitoring software, by reducing code duplication and allowing the developers to focus on subsystem-specific issues. The existence of a uniform graphical user interface accross the different subsystems simplified the training of the operations personnel at the CMS control room and experts alike. The use of a common database schema to store the hardware description and configuration parameters has been adopted. The introduction of the SWATCH framework has allowed the CMS Level-1 trigger to achieve high operational efficiency in a short amount of time with only minimal downtime attributed to trigger issues during the 2016 data taking period.

References

[1] The CMS collaboration, "*CMS Technical Design Report for the Level-1 Trigger Upgrade*", CERN, Geneva 2013. CMS-TDR-12

[2] E. Hazen et al, "*The AMC13XG: a new generation clock/timing/DAQ module for CMS MicroTCA*", 2013 JINST 8 C12036

[3] I. Magrans de Abril, C. E. Wulz, and J. Varela, "*Concept of the CMS trigger supervisor*", IEEE Trans. Nucl. Sci., vol. 53, pp. 474483, 2006

[4] Polymer authors, "*Polymer project*", `https://www.polymer-project.org/`