Published by IOP Publishing for Sissa Medialab

Received: *November 6, 2025*
Revised: *January 10, 2026*
Accepted: *January 13, 2026*
Published: *February 17, 2026*

Topical Workshop on Electronics for Particle Physics
Rethymno, Crete, Greece
6–10 October 2025

EMP: a common infrastructure firmware framework for the CMS phase-2 upgrades

Kosmas Adamidis,[a] **Luis Ardila-Perez,**[b] **Ioannis Bestintzanos,**[c] **Serhii Cholak,**[d] **Jonathan Fulcher,**[d] **Kristian Hahn,**[e] **Alex Howard,**[f] **Gregory Iles,**[f] **Hendrik Krause,**[b] **Jacob Linacre,**[g] **Stavros Mallios,**[f] **Torben Mehner,**[b] **David Monk,**[e] **Michelangelo Pari,**[h] **Mark Pesaresi,**[f] **Giovanni Petrucciani,**[h] **Andrew Rose,**[f] **Ozgur Sahin,**[i] **Raghunandan Shukla,**[f] **Alessandro Thea,**[h] **Kirika Uchida,**[f] **Kate Whalen**[g] **and Tom Williams**[g,*]

[a] *Department of Physics and Astronomy, UCLA, 475 Portola Plaza, Los Angeles, CA 90095, U.S.A.*
[b] *Institute for Data Processing and Electronics (IPE), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, D-76344 Eggenstein-Leopoldshafen, Germany*
[c] *University of Ioannina, Ioannina, Greece*
[d] *Department of Physics, Boston University, Commonwealth Ave, Boston, MA, U.S.A.*
[e] *Department of Physics, Northwestern University, Sheridan Rd, Evanston, IL, U.S.A.*
[f] *Blackett Laboratory, Imperial College, Prince Consort Road, London, SW7 2BW, U.K.*
[g] *STFC Rutherford Appleton Laboratory, Harwell Oxford, Didcot, OX11 0QX, U.K.*
[h] *CERN, Geneva, Switzerland*
[i] *IRFU, CEA, Universite Paris-Saclay, Gif-sur-Yvette, France*

E-mail: tom.williams@cern.ch

Abstract: In its phase-2 upgrades, all of the CMS experiment's off-detector electronics systems will be replaced by ATCA boards featuring Xilinx UltraScale+ FPGAs and high-speed optical modules. The various off-detector electronics systems have different high-level purposes and I/O requirements, e.g. requiring different combinations of lpGBT links to/from detector modules, higher-speed data transfer links between off-detector boards, and readout links. The EMP framework provides common infrastructural firmware components, top-level designs and associated software for multiple CMS phase-2 boards and systems. It implements and integrates high-speed serial link engines, data capture and playback buffers, clocking components, and readout. In this paper, we present the EMP framework, along with its testing and validation, and associated build tooling.

Keywords: Control and monitor systems online; Detector control systems (detector and experiment monitoring and slow-control systems, architecture, hardware, algorithms, databases); Digital electronic circuits; Online farms and online filtering

*Corresponding author.

https://doi.org/10.1088/1748-0221/21/02/C02019

Contents

1 Introduction **1**

2 EMP firmware **2**
- 2.1 High-level structure 3
- 2.2 Major components 3
- 2.3 Build workflow 4

3 EMP software **5**

4 Testing & validation **5**

5 Conclusions **6**

2026 JINST 21 C02019

1 Introduction

All of the CMS experiment's detector systems [1, 2] will be completely replaced or significantly upgraded for the High-Luminosity LHC (HL-LHC) era, in order to cope with the increased luminosity of $7.5 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ [3]. As part of these upgrades, all of the off-detector electronics systems will be completely replaced by custom Advanced Telecommunications Computing Architecture (ATCA) boards, with eight different board designs adopted across the various systems. Each of these ATCA boards hosts: one or two Xilinx UltraScale+ FPGAs, which implement the relevant data-processing algorithm (e.g. readout logic, or particle reconstruction for the L1 trigger); high-speed optical modules, used for links with on-detector electronics and/or other ATCA boards; and a Xilinx UltraScale+ Zynq MPSoC, which runs a linux operating system and software that handles control, monitoring and management tasks.

The core data-processing algorithm implemented in the FPGAs on the off-detector boards differs across the various systems. However similar infrastructural components are required in all systems — e.g. to receive the LHC clock, communicate with on-detector electronics, and transfer L1 trigger data between boards — albeit with differing multiplicities and/or configurations. Adopting a common firmware and software implementation for this data-processing infrastructure across multiple boards and systems will reduce the total effort expended in developing and maintaining infrastructural components, allowing more developers to focus on system-specific functionality. Beyond the clear benefit of reducing code duplication during the development phase, sharing the infrastructural firmware and software will also improve productivity and system stability in the longer term, for example by eliminating duplicated effort in the investigation and resolution of bugs.

In this paper we present the EMP (Extensible Modular data Processor) framework [4], which provides common infrastructural firmware components, top-level designs and associated software for multiple CMS phase-2 backend boards (Apollo [5], BMTL1 [6], Serenity [7], DTH400 and DAQ800 [8]) and systems (tracker, HGCal, MTD, DT, RPC, BRIL and L1 trigger). The EMP firmware and its core components are presented in section 2, then the software is discussed in section 3 and finally the tests performed to validate the framework are described in section 4.

2 EMP firmware

The EMP firmware encompasses a control bus master; I/O data playback and capture buffers; and all high-speed serial link interfaces required in the relevant CMS phase-2 off-detector electronics systems. Figure 1 shows the major electronics systems and the interfaces between them. Each subdetector will be configured and read out by a system of off-detector boards over optical lpGBT links. The L1 trigger system [9] will receive coarse-granularity data for every bunch crossing from several subdetectors, and reconstruct particles from these inputs using fixed-latency algorithms implemented across several subsystems, with intermediate results sent between the subsystems. The final L1 subsystem sends the trigger's readout decision to the Trigger and Timing Control and Distribution System (TCDS), which forwards this decision to all other off-detector boards over the Trigger, Timing and Control (TTC) links, along with other fixed-latency commands (e.g. for bunch counter synchronisation). Subdetector and L1 trigger boards will send readout data for triggered events to centrally developed DAQ boards. In case of readout buffer overflows, readout boards can request that TCDS throttles triggers via the Trigger Throttling System (TTS) links. The EMP firmware supports each of these interfaces in the full range of required configurations.

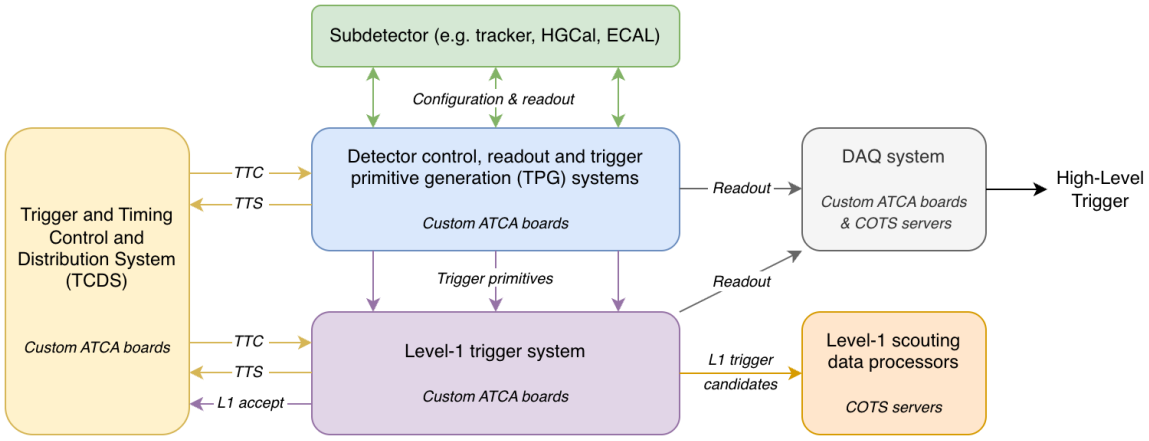

Figure 1. Diagram showing the major systems that will be involved in the configuration, control, and readout of the phase-2 CMS detector, and the dataflow between them. Each of the colours of arrows denotes links of a specific type of data transfer, using a unique protocol.

The EMP firmware components are complex entities with numerous input and output ports, which need to be interfaced with each other in the correct manner to ensure that they function correctly. To allow users to focus solely on system-specific functionality despite this infrastructural complexity, the EMP framework includes common top-level designs which instantiate all required components and connect them to the user-created *payload* firmware block — e.g. a particle/event reconstruction algorithm — that conforms to a specific interface, as shown in figure 2. This approach minimises code duplication, eases integration by providing users with the simplest possible interface, and centralises any changes required to update externally developed cores used by EMP.

2026 JINST 21 C02019

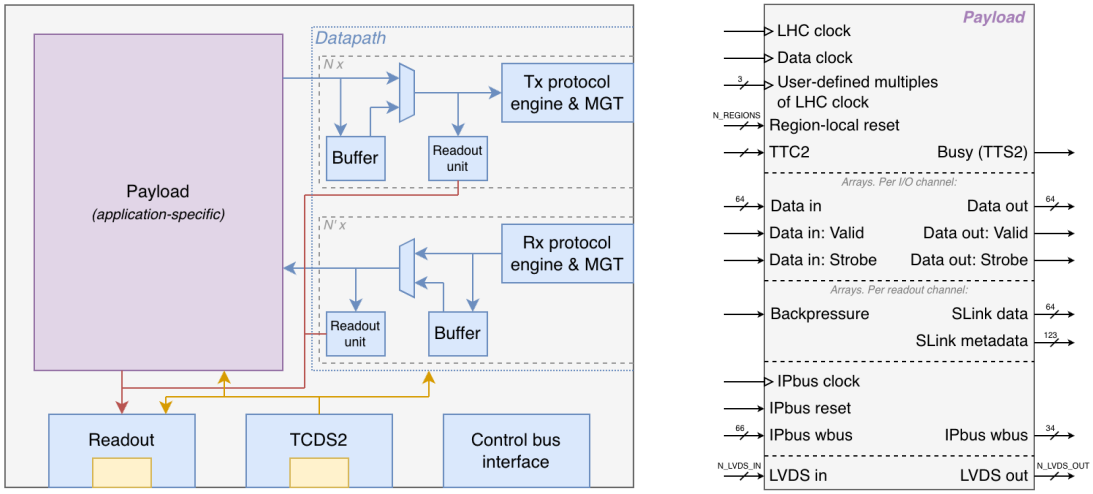

Figure 2. *Left:* diagram illustrating the main components in an EMP firmware design. The blue boxes are EMP components, yellow boxes are cores provided by the CMS DAQ group and the system-specific *payload* block is shown in purple. Orange arrows indicate TTC/TTS connections, the flow of lpGBT and trigger data is shown in blue and readout data is shown in red. *Right:* the standard port map for the `emp_payload` entity.

2.1 High-level structure

One of the key challenges in developing and maintaining the framework has been supporting a range of different FPGAs, boards and component configurations. For example, although all of the boards using EMP in the final system will feature one of two FPGAs (VU13P or VU35P), their prototypes have also used a wide range of other FPGAs. In order to simplify support of multiple boards and the migration of application firmware between them, the source code and build configuration files are organised into three groups, with clear interfaces for customisation:

1. *Common components:* these include the key building blocks such as link protocol engines and I/O buffers, other utilities such as wrappers of clocking primitives, and top-level entities (mainly 'glue logic' connecting the other components). These components are parameterised by build-time constants for settings which have board-/application-specific values.

2. *Board-specific settings:* several build-time settings vary between boards, but are independent of the application, e.g. the number of I/O regions. All board-specific constants are specified in the `emp_device_decl` VHDL package, which conforms to a standard specification across boards. Pin, area and clock constraints files are also provided for each board.

3. *Application-specific settings and algorithms:* the user-provided `emp_project_decl` VHDL package defines system-specific constants for RTL, e.g. specifying the link protocol and speed for each I/O channel. Users integrate application-specific logic by implementing an `emp_payload` entity that conforms to the standard port map shown in figure 2.

2.2 Major components

IPbus [10] is used as the control bus; the control bus slaves, fabric and master (the transactor) are all sourced from the IPbus firmware repository. Combined with use of the associated software library,

2026 JINST 21 C02019

uHAL, this has enabled the EMP software to transparently adapt to the different control links used across final boards (AXI Chip2Chip), prototype boards (PCIe) and simulation (Ethernet/UDP).

TCDS2 and readout links are implemented using centrally provided CMS cores, which implement the SLinkRocket, TTC and TTS protocols [11]. The surrounding firmware in the EMP TCDS and readout blocks handle common integration tasks, including the pattern generators, spy memories and clocking infrastructure required for system debugging and for standalone operation of EMP-based boards without CMS DAQ boards.

The payload's principal data ports are connected to the FPGA pins via the *datapath* components, that include buffer memories and the link engines. The datapath logic can be configured at runtime to synchronously play arbitrary user-defined data from the buffers into the payload or links, and to capture the output of the payload or links. The set of signals captured/driven by these buffers is identical to the payload-side interface of the link engines; this has allowed us to factorise tests of board-to-board data transfer from tests of payload firmware, streamlining system integration.

lpGBT links [12] are implemented using the standard core from the CERN ESE group, which is surrounded by EMP firmware that handles common tasks. EMP implements the CMS Standard Protocol (CSP) for sending LHC-synchronous L1 trigger data between boards with low latency (~ 0.1 µs) over 16 and 25 Gbps links. Due to latency and FPGA resource constraints CSP cannot rely upon retransmission in case of errors. Instead it includes dedicated fields and recovery mechanisms to detect bitflips and ensure that resulting disruptions to the data stream (if any) are confined to a short period of time (a few LHC clock cycles or orbits). Data is sent to the L1 scouting servers using the VHDL implementation of the TCP/IP protocol developed for the CMS DAQ system [8].

The EMP firmware repository also contains a *null algo* payload entity that simply passes data received on the main input ports to the output ports with a few clock cycles delay; this is instantiated in the example designs for application-independent development and testing of the framework. The resource usage of the EMP firmware is shown in table 1 for a few exemplar configurations.

Table 1. Resource usage of the EMP framework firmware with a few different exemplar configurations, in each case built for a Xilinx VU13P FPGA.

	Flip flops [10^3]	LUTs [10^3]	Block RAMs (36kb)
PCIe; 104 CSP links (bidir.)	301 (8.7 %)	198 (11 %)	717 (27 %)
Chip2Chip; 124 CSP links (bidir.)	342 (9.9 %)	218 (13 %)	765 (28 %)
Chip2Chip; 124 lpGBT links (bidir.)	405 (12 %)	259 (15 %)	693 (26 %)

2.3 Build workflow

In selecting a build tool for the EMP firmware, two key requirements were that it must provide simple command-line-based workflows for both synthesis and simulation, and use declarative-style build configuration files (listing source code files and relevant settings). Additionally given the range of FPGAs, boards and component configurations supported by EMP, the build tool's configuration files must naturally support modular designs and in particular support the inclusion of different files based on properties of the current build (e.g. FPGA series). The EMP firmware is built using IPBB (IPbus Builder), a command-line firmware project management and build tool that fulfills these criteria. In

2026 JINST 21 C02019

IPBB sources and settings are specified in *dependency files*; the EMP firmware repository includes one user-facing dependency file for each supported board.

3 EMP software

The EMP firmware is controlled and monitored using an object-oriented software library, implemented in C++. The functions and classes in this library encapsulate register names, the required sequencing of individual writes, and other similar low-level details of configuration and monitoring procedures. This simplifies the API used by EMP-based libraries and software applications, minimises code duplication across those applications, and helps to ensure that after successful standalone tests of EMP firmware the same outcome will always be observed when configured and monitored by users' EMP-based applications. The EMP software also includes Python bindings, along with a command-line interface (CLI) and test suite, both implemented in Python. The CLI provides a simple user interface for all procedures required to use the boards in laboratories and small tests stands, as well as all functionality required to enable developers and experts to diagnose the cause of any problems encountered.

4 Testing & validation

The EMP framework is validated by a test suite that uses the C++ library to repeatedly reset and configure the main components of a null algo EMP firmware build in each operational mode that is required for the supported high-level workflows. It reads status registers to check for erroneous values and inconsistencies, and whenever applicable, reads datapath buffers, TTC command history memories and other block memories to check that their contents also match expectations.

The EMP CSP engines are validated as part of the test suite by configuring the transceivers in an FPGA-internal loopback mode, including tests of recovery from bitflips, which are performed by injecting well-defined errors in firmware. The CSP engines have also been extensively tested in hardware, including operating links under nominal conditions for many thousands of link hours, and inducing random errors by deliberate attenuation of optical links. For example, a CSP link was operated for 48 hours with a bit-error rate of 1.3×10^{-5} arising from attentuation; the EMP CSP engine correctly recovered from all bitflips in control words, avoiding permanent disruption to the datastream. This test verified the robustness of the EMP CSP engine against errors that may occur intermittently in the final system, e.g. between the initial signs of degradation of optical modules and their replacement. Since comprehensive testing of the lpGBT links requires system-specific on-detector electronics, lpGBT workflows are not included in the EMP test suite; instead changes to lpGBT components are tested by system experts running the configuration, calibration and monitoring procedures of at least one system's on-detector electronics.

The test suite is implemented using `pytest`, which allows developers to easily run all tests, or specific subsets of them, with a single command. We have implemented GitLab Continuous Integration (CI) pipelines that provide comprehensive automated validation of code changes in both the firmware and software repositories. In addition to building the firmware and software for all supported boards and operating systems, these pipelines also run the test suite on hardware. Beyond simply speeding up the development workflow, this level of automated validation has proven critical for supporting EMP on several boards while minimising the effort required for maintenance.

2026 JINST 21 C02019

5 Conclusions

The CMS experiment's off-detector electronics systems will be completely replaced for the HL-LHC era. The EMP framework provides common infrastructural firmware components, top-level designs and associated software for multiple phase-2 boards and systems. All major components of the framework have now been developed; this includes a test suite and automated pipelines which streamline development, in particular minimising the effort required to support multiple FPGAs, component configurations and boards. The framework is being successfully used for a range of activities critical to the development and testing of the phase-2 upgrades, including test beams, and quality control tests of detector modules and ATCA boards. Its use across multiple systems has significantly reduced code duplication, allowed for pooling of infrastructure development effort, simplified system integration, and ultimately freed up more effort for system-specific work which will help to enhance the readiness of CMS for the HL-LHC era. With all major components now in place, over the next few years EMP development will be focused on refining the framework to support more complex, larger-scale system tests and commissioning of the final system.

References

[1] CMS collaboration, *The CMS Experiment at the CERN LHC*, 2008 *JINST* **3** S08004.

[2] CMS collaboration, *Development of the CMS detector for the CERN LHC Run 3*, 2024 *JINST* **19** P05064 [arXiv:2309.05466].

[3] D. Contardo et al., *Technical Proposal for the Phase-II Upgrade of the CMS Detector*, CERN-LHCC-2015-010, LHCC-P-008, CMS-TDR-15-02 (2015).

[4] EMP website, https://serenity.web.cern.ch/serenity/emp-fwk/ [Accessed: 2026-01-01].

[5] A. Akpinar et al., *Design, construction, and testing of the APOLLO ATCA blades for use at the HL-LHC*, 2025 *JINST* **20** C04001 [arXiv:2501.03702].

[6] I. Bestintzanos et al., *An ATCA processor for Level-1 trigger primitive generation and readout of the CMS barrel muon detectors*, 2023 *JINST* **18** C02039.

[7] T. Mehner et al., *Lessons learned while developing the Serenity-S1 ATCA card*, 2024 *JINST* **19** C02018 [arXiv:2311.02222].

[8] J. Alawieh et al., *Phase-2 CMS DAQ — growing from prototype boards to demonstrator systems*, 2025 *JINST* **20** C02005.

[9] CMS collaboration, *The Phase-2 Upgrade of the CMS Level-1 Trigger*, CERN-LHCC-2020-004, CMS-TDR-021 (2020).

[10] C. Ghabrous Larrea et al., *IPbus: a flexible Ethernet-based control system for xTCA hardware*, 2015 *JINST* **10** C02019.

[11] CMS collaboration, *The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger*, CERN-LHCC-2021-007, CMS-TDR-022 (2021).

[12] P. Moreira et al., *lpGBT documentation*, https://cds.cern.ch/record/2809058 (2024).

2026 JINST 21 C02019