Published by IOP Publishing for Sissa Medialab

Received: *November 6, 2025*
Accepted: *January 28, 2026*
Published: *March 19, 2026*

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

Factory Acceptance Test for Serenity-S1, a CMS Phase-2 back-end card

H. Krause[a,*] **L.E. Ardila-Perez**[a] **M. Balzer**[a] **M. Fuchs**[a] **A. Howard**[b] **G. Iles**[b] **T. Mehner**[a] **D. Parker**[b] **A. Rose**[b] **F. Simon**[a] **and T. Williams**[c]

[a] *Karlsruhe Institute of Technology, Institute for Data Processing and Electronics, Hermann-von-Helmholtz-Platz 1, D-76344 Eggenstein-Leopoldshafen, Germany*
[b] *Imperial College London, Physics Department, Blackett Laboratory, Prince Consort Rd, London, SW7 2BW, U.K.*
[c] *STFC Rutherford Appleton Laboratory, Particle Physics Department, Harwell Campus, Didcot, OX11 0QX, U.K.*

E-mail: Hendrik.Krause@kit.edu

Abstract: The CMS Phase-2 upgrade for the High-Luminosity LHC requires the replacement of the back-end electronics in several sub-detectors. For this purpose, the Serenity consortium has developed Serenity-S1, a versatile ATCA-based FPGA processing card adopted by seven CMS sub-detector systems. With a total production of more than 700 boards, an efficient and reliable quality-assurance process is essential. This paper presents the Factory Acceptance Test (FAT), an automated test system that extends the assembly and electrical integrity checks performed by the PCB manufacturer and assembler, enabling comprehensive functional testing of each board directly at the production site in less than ten minutes. The FAT leverages on-board controllers and existing board-control frameworks, integrating them into a Python-based Pytest environment to structure and sequence test runs. A dedicated test PC runs the control software that orchestrates the test system, generates detailed reports with database upload, and offers two intuitive graphical interfaces for visualising the results. Experience from using the FAT during pre-series production demonstrates its capability to enable early fault detection, short repair cycles, and consistent board quality across multiple test sites.

Keywords: Detector control systems (detector and experiment monitoring and slow-control systems, architecture, hardware, algorithms, databases); Manufacturing

*Corresponding author.

https://doi.org/10.1088/1748-0221/21/03/C03035

Contents

2026 JINST 21 C03035

1 Introduction

For the CMS Phase-2 upgrade, the Serenity consortium has developed Serenity-S1, a versatile Advanced Telecommunications Computing Architecture (ATCA) -based FPGA processing card intended as a common back-end platform for multiple CMS sub-detector systems [1]. Seven sub-detector systems have adopted the board [2, 3], leading to a production of more than 700 boards across multiple manufacturers and assembly sites. Given this scale, an efficient and reliable quality assurance (QA) process is essential to ensure that all boards delivered to CERN are fully functional.

Standard quality-control procedures performed by the assembler — such as automated optical inspection and X-ray analysis — verify mechanical and soldering integrity but cannot detect electrical or functional defects. In addition, some components on the Serenity-S1 require an initial configuration before the card can be used. To cover both, a dedicated Factory Acceptance Test (FAT) has been developed to perform automated functional testing and initial component configuration directly at the production site.

The Serenity QA process is structured in multiple stages. First, the PCB manufacturer performs a series of bare-board electrical tests before assembly. Second, the assembler carries out visual and X-ray inspections, followed by the FAT described in sections 3 and 4. Once the boards are delivered to CERN, a User Acceptance Test (UAT) is performed to verify that no damage has occurred during shipping. Afterwards, mechanical components — such as heatsinks and support structures — are added. Finally, the sub-detector groups are customising the card with their choice of optical modules, which is followed by a burn-in test.

2 Serenity-S1 overview

As shown in figure 1, the Serenity-S1 [4] is partitioned into a service area, providing infrastructure/-management functionality, and a payload area, centred around an AMD VU13P FPGA for high speed processing of data transferred via Samtec FireFly electrooptical transceiver modules. The service

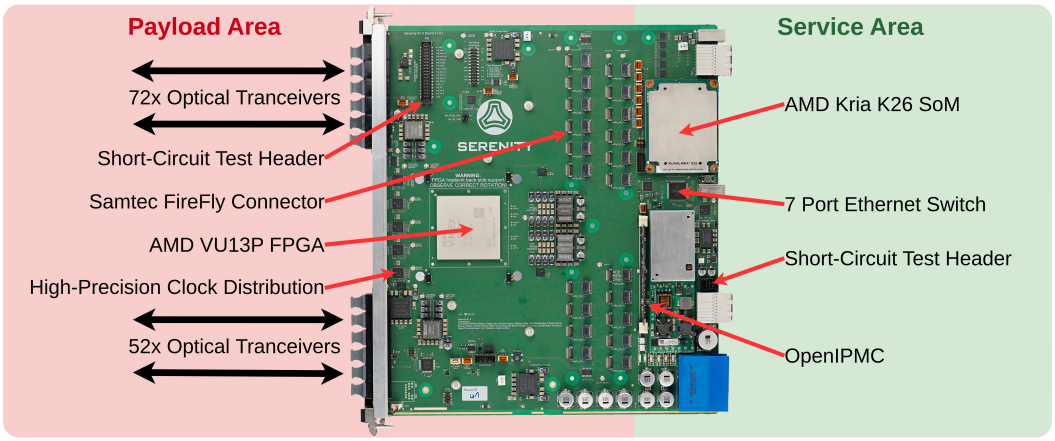

Figure 1. Overview of the Serenity-S1 ATCA back-end card, highlighting the main components and the partitioning into a service and payload area.

area hosts two controllers, an OpenIPMC module [5] for ATCA-related functionality and an AMD Kria K26 System-on-Module (SoM) [6] for board-level control.

The Kria SoM runs the Serenity MAnagement SHell (SMASH) [7] software which provides access to the peripheral components in the payload area via I^2C, PMBus and a JTAG connection to the FPGA. Low-level details are abstracted via a Python API, which provides functionality to verify, configure, and measure device properties.

The FPGA firmware is built using the Extensible Modular Processor (EMP) framework [8], which provides common infrastructure firmware functionality for multiple CMS Phase-2 systems. This includes functionality for multi-gigabit-transceiver (MGT) control and clock monitoring. Control register access is provided via an AXI Chip2Chip connection to the Kria SoM using the EMP Python bindings.

3 FAT concept

The FAT is designed to be used directly by the assembly companies to enable early fault detection and short repair cycles. It serves as an end-of-line test, performed after the assembler's standard quality-control measures. To meet the agreed requirements, the test must operate largely automatically, complete within ten minutes, and be easy to operate. An additional challenge is the initial configuration required for the clock chips, power supplies, and power sequencer within the payload area of the Serenity-S1.

Conventional functional testing typically requires custom fixtures providing test-point access to the device under test [9]. Since the Serenity-S1 is a high-speed design without such test points, this approach cannot be applied. Instead, a built-in self-test (BIST) concept was adopted from the outset, enabling the board to verify its own functionality through embedded control and monitoring features [10].

The FAT uses the on-board controllers of the Serenity-S1 to sequentially verify the service and payload area components. This is achieved by leveraging the existing board management frameworks SMASH and EMP, integrated into the Python-based test framework pytest [11], which coordinates and evaluates all test steps.

To ensure the Serenity-S1 can be safely powered for the BIST, a short-circuit test is performed beforehand.

2026 JINST 21 C03035

Figure 2. The FAT test stand at Karlsruhe Institute for Technology (KIT) showing the setup for the short-circuit test with the GW Instek DAQ-9600 Data Acquisition System (left) and for the functional test with the Serenity-S1 inserted in a crate and added OpenIPMC, Kria SoM and loopback cables (right).

4 FAT implementation

Two setups are used for the FAT, which are shown in figure 2. The first is used for the short-circuit test and connects the bare Serenity-S1 board to a data acquisition system for impedance measurements. The second setup, which is used for functional testing, includes an ATCA crate, an OpenIPMC, a Kria SoM and copper loopback cables. Both the OpenIPMC and the Kria SoM are preconfigured with a dedicated FAT image and used as "golden" modules; they are removed after each FAT run and reused for subsequent boards. A dedicated test PC runs the control software with a graphical user interface (GUI) to control the tests.

4.1 Short-circuit test

The short-circuit test is the first stage of the FAT. It ensures that no shorts exist in any of the main power rails. This is done by connecting a commercial data acquisition system (GW Instek DAQ-9600) to dedicated test headers on the Serenity-S1, which provide access to the power rails. The DAQ-9600 injects short current pulses in every rail and measures the resulting voltage to determine the impedance of each rail relative to ground. Parallel resistors have been added to the multiplexer card inside the DAQ-9600 to limit the voltage levels on high-impedance rails, preventing potential damage to components.

Measured values are compared to predefined thresholds, established from the statistics of testing 10 pre-series boards, to determine whether the board is safe to power ON or requires manual inspection. The test is executed without Kria SoM, OpenIPMC, or backplane connections, minimising external influences on the impedance. A Python-based GUI (figure 3, left) displays the measurement results, while the underlying test software automatically generates a formatted test report and uploads it to the Serenity production database.

4.2 Functional test

The functional test is the second stage of the FAT; it initially configures the components and tests the main functionality of the Serenity-S1. For this, the Serenity-S1 is installed into a single-slot ATCA crate with its OpenIPMC and Kria SoM attached. The FireFly connectors are looped using copper cables to simplify the test setups compared to installing and looping optical modules. Power is

2026 JINST 21 C03035

provided through the ATCA backplane by a custom power supply controlled from the test PC. After the OpenIPMC and Kria SoM have booted, the test PC automatically starts the tests.

Service area test. The service area is tested using the OpenIPMC, which exposes standard ATCA management functionality to the test PC via a telnet interface. The test PC communicates with the OpenIPMC to read temperature sensors, telemetry from the main power supplies and to access the EEPROMs. The telemetry includes input and output voltages, currents, and status registers information. All retrieved data is verified for successful communication and checked against expected limits. For the EEPROMs read and write access is verified.

Payload area test. Tests of the slow control components and FPGA in the payload area are performed via the SMASH framework running on the Kria SoM. For this, SMASH API calls are wrapped by pytest test functions, allowing the creation of parameterised test cases. Using the `pytest-dependency` plugin [12], the test flow ensures safe sequencing of dependent test case execution. I^2C communication to the power sequencer and supplies is verified, followed by their configuration. Afterwards, the power sequencer enables the rails, measures their voltages, and verifies that all values are within specification. Communication with I/O expanders and clock chips is then checked before the latter are configured as well. As a last step, the JTAG chain from the Kria SoM to the FPGA is validated by checking that the FPGA can be programmed successfully. JSON logfiles of the test are generated using the `pytest-json` plugin [13]. The test PC starts the pytest script on the Kria SoM and collects reports over an SSH connection.

FPGA test. The FPGA is programmed with an EMP firmware design, and the EMP Python bindings on the Kria SoM are used for register access and test control. As in the payload area tests, the pytest framework is employed. The test sequence begins by verifying the AXI Chip2Chip connection between the FPGA and Kria SoM through register access. Next, the clock distribution for the MGTs is validated by measuring the frequencies of all reference clocks. Finally, each MGT lane undergoes a bit error rate test using a PRBS31 pattern transmitted at 25.781 Gbit s^{-1} through the copper loopbacks, while monitoring for bit errors. Due to thermal constraints when operating the FPGA without a heatsink, only 16 MGT lanes are tested simultaneously for 5 seconds each. This corresponds to an estimated upper bit error rate limit of 3.8×10^{-11} at 95 % confidence [14].

Report generation and database upload. All test logs and JSON pytest result files are processed on the test PC to produce a detailed LaTeX report using the `pylatex` package [15] and a local TeX Live installation. The completed report is automatically uploaded to the central Serenity production database for traceability across multiple sites.

4.3 Graphical user interfaces

The control software on the test PC provides two dedicated graphical interfaces to operate the short-circuit and functional test (figure 3). Both interfaces are designed for simplicity; the only required user inputs are the board ID and the IP addresses or device file of the connected equipment (power supply, Kria SoM, OpenIPMC, and DAQ-9600). Once initiated, the complete test sequence runs automatically, with visual indicators showing the current stage and its status. The short-circuit GUI displays the measured impedances after each run, while the functional-test GUI presents a board layout view that highlights the status of all tested components (pass or fail).

2026 JINST 21 C03035

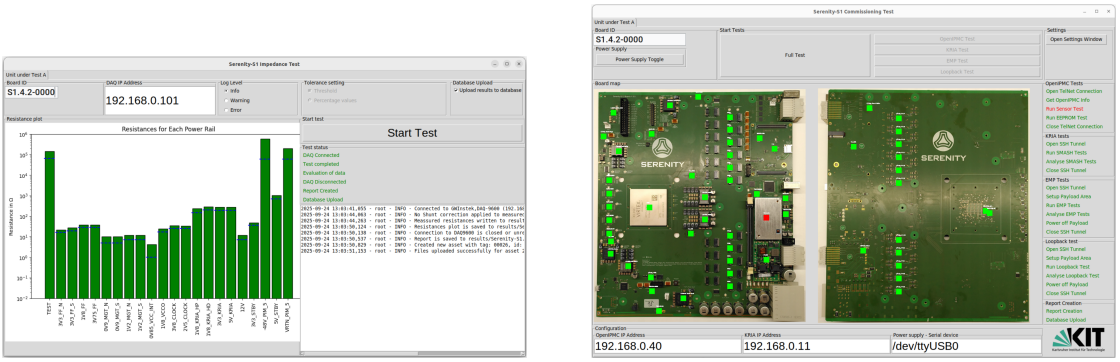

Figure 3. FAT GUIs for the short-circuit test (left) and the functional test (right), providing an intuitive way to control the test flow and get visual feedback.

5 Test results

During the pre-series production phase, the FAT was applied to a total of 34 boards at three sites: KIT (where 13 boards were also assembled), one assembly company (Assembler A), and CERN. The tests served both as initial verification at the assembler and as repeated validation after shipment to CERN, confirming the test results. A summary of the corresponding results is provided in table 1.

The usage of the FAT has demonstrated its capability to identify assembly and component issues at an early stage. A total of seven boards failed at least one FAT stage: three due to defective power-supply modules, two exhibiting single-link MGT failures, and two showing other single-component failures. Except for the MGT link issues, all boards were successfully repaired, retested, and passed on the second attempt.

Throughout its use, several improvements were introduced to the FAT based on feedback from Assembler A. These included the addition of the two dedicated short-circuit test header on the Serenity-S1, the division of the FAT into two stages — one for short-circuit testing and one for functional testing — and the replacement of a Jupyter notebook interface with a GUI, described in section 4.3, for more convenient test control.

Table 1. Test results from using the FAT at three different sites show that it is able to detect component failures and can prevent further damage to the board.

Test Location	Boards Tested	Passes	Failures
Assembler A	21	19	2
CERN		19	2
KIT	13	8	5

6 Conclusion and outlook

The FAT extends the Serenity-S1 QA process by introducing a fast, fully automated functional test directly at the production site. It leverages on-board controllers (the OpenIPMC & Kria SoM), together with the associated SMASH and EMP software/firmware frameworks, to perform comprehensive board verification with minimal external equipment. The use of Pytest ensures structured and

2026 JINST 21 C03035

reproducible test execution, while two Python-based GUIs simplify operations and make the system easy to use by the assembler.

Successful usage was demonstrated during the pre-series production, where it detected multiple hardware faults and verified that all boards handed to sub-detector groups are functional. With its proven reliability and efficiency, the FAT is now ready for the main series production, providing a consistent and traceable quality control method for all Serenity-S1 boards.

Further work involves the setup of the FAT at the second and third assembly companies and the preparation of a User Acceptance Test (UAT) setup at CERN. The UAT will extend the current FAT procedure with enhanced MGT link testing using optical Samtec FireFly modules. Together, these stages will ensure that every Serenity-S1 card for the CMS Phase-2 upgrade is fully functional and ready for integration.

Acknowledgments

Hendrik Krause, Marvin Fuchs and Torben Mehner acknowledge the support of the Doctoral School "Karlsruhe School of Elementary and Astroparticle Physics: Science and Technology" (KSETA).

References

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

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

[3] CMS collaboration, *The Phase-2 Upgrade of the CMS Beam Radiation Instrumentation and Luminosity Detectors*, CERN-LHCC-2021-008 (2021).

[4] T. Mehner et al., *Serenity-S1 — A Versatile ATCA Processing Card for the CMS Phase-2 Upgrade*, *TechRxiv* (2025).

[5] L. Calligaris et al., *A flexible and low-cost open-source IPMC mezzanine for ATCA boards based on OpenIPMC*, 2022 *JINST* **17** C03007 [arXiv:2112.12888].

[6] Advanced Micro Devices, Inc., *AMD Kria™ K26 SOM*, https://www.amd.com/en/products/system-on-modules/kria/k26.html (2025).

[7] Serenity collaboration, *All things smash*, https://serenity.web.cern.ch/serenity/smash/ (2019).

[8] EMP collaboration, *Emp framework guide*, https://serenity.web.cern.ch/serenity/emp-fwk/index.html (2019).

[9] C. Houdek, *Inspection and testing methods for PCBs: An overview*, Engineer/OwnerCaltronics Design & Assembly, White Paper #401 (2016) [PDF].

[10] E. McCluskey, *Built-In Self-Test Techniques*, *IEEE Des. Test. Comput.* **2** (1985) 21.

[11] H. Krekel et al., *pytest: The pytest framework makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries*, https://github.com/pytest-dev/pytest (2025).

[12] R. Krahl, *pytest-dependency — manage dependencies of tests*, https://github.com/RKrahl/pytest-dependency (2023).

2026 JINST 21 C03035

[13] M. Chun-Lum, *pytest-json: A plugin for py.test that generates json reports for test results*, https://github.com/mattcl/pytest-json (2016).

[14] A. Lebl, D. Mitić and V. Vasiljević, *Bit error rate estimation and confidence levels*, https://unitech-selectedpapers.tugab.bg/images/papers/2019/s3/s3_p202.pdf (2019).

[15] J. Fennema et al., *Pylatex: A python library for creating and compiling latex files or snippets*, https://github.com/JelteF/PyLaTeX (2025).

2026 JINST 21 C03035