

FGC PRIVATE NETWORK: LOW-LATENCY COMMUNICATION STANDARD FOR THE FUTURE CONTROLS OF POWER CONVERTERS AT CERN

C. Zrounba*, V. Nappi, D. Zielinski, Q. King, R. Murillo-Garcia, CERN, Geneva, Switzerland

Abstract

The CERN Electrical Power Converters (EPC) group operates thousands of converters, most via the Function Generator/Controller (FGC) device family. Its new generation, FGC4, built on an AMD Zynq UltraScale+ MPSoC, will support future EPC applications. Some, like POPS+, require low-latency inter-FGC communication. We present the FGC Private Network (FPN) protocol stack and library enabling this private, real-time network.

FGC4 software runs on four Cortex-A53 cores: two of which are dedicated to real-time regulation on bare-metal, and the other two run background and frontend services under Linux. The Programmable Logic (PL) hosts two GbE endpoints able to fetch application data via cache-coherent DMA and transmit them over a switched Ethernet network, meeting a tight 10 μ s control budget.

The FPN implementation is split between software and gateway. To ensure consistency, protocol-related definitions in C++, VHDL, as well as debugging utilities, are all generated from YAML definitions acting as the single source of truth.

The C++ library minimizes CPU usage without compromising on clarity, and supports interchangeable transport layers (PL GbE, generic Ethernet, UDP, file replay...) for desktop development, regression testing, and Hardware-in-the-Loop validation. This model-driven, flexible and testable approach paves the way for POPS+ deployment on FGC4 and enables future EPC systems.

INTRODUCTION

The CERN Electrical Power Converters (EPC) group operates in excess of 5,000 power converters, of which nearly 4,000 are controlled with the Function Generator/Controller (FGC) device family. FGC4, the next generation controller, will support future EPC applications, such as the upgrade to the Power supply for Proton Synchrotron (POPS+). POPS is responsible for providing the electrical power and precise control required to generate the magnetic fields in all 101 main magnets of the Proton Synchrotron accelerator at CERN. These magnets require currents of up to 6 kA and DC voltages of up to ± 10 kV, corresponding to a maximal power of 60 MW [1]. One of the main objectives of POPS+ is to improve system availability through the introduction of redundancy: a third AC/DC will be added to the two existing units, along with an additional DC/DC converter to complement the current six. Each converter unit requires a dedicated FGC4 controller. Together with the

controllers for the crowbar system and the main control unit, this results in a total of 12 FGC4 devices, all coordinated to operate POPS+ as a single integrated system. In addition, POPS+ includes a Programmable Logic Controller (PLC) responsible for orchestrating all power converters and their FGC4-based controllers, managing the state machines, including startup, charging and stopping sequences, as well as interlocks and safe start conditions [2]. This distributed heterogeneous system requires real-time communication to enable the FGC4s to interact with each other and with the Multifunctional Platform (MFP), running Linux, of the PLC system.

In this context, communication must not only be fast, but also deterministic, robust, and maintainable over time. General-purpose networking solutions are not well suited to cyclic real-time exchanges when software overhead must remain minimal, and dedicated protocols such as EtherCAT introduce constraints and complexity that are not justified for our use-cases. EPC is therefore developing the FGC Private Network (FPN), a dedicated stack for low-latency real-time communication suitable for inter-FGC4 exchange.

This paper presents the main design choices of the protocol stack, its implementation on FGC4, and the model-driven approach used to keep protocol definitions consistent across the stack.

FGC4 PLATFORM AND COMMUNICATION CONTEXT

FGC4 is based on an AMD Zynq UltraScale+ MPSoC integrating a Processing System (PS) with four Arm Cortex-A53 cores and Programmable Logic fabric (PL). In the target architecture, two cores of the PS are dedicated to bare-metal (BM) real-time regulation, while the two others run Linux services for background and front-end functions. This separation keeps the regulation path isolated from less deterministic activities.

The PL hosts two Gigabit Ethernet endpoints. Within this environment, FPN provides a private communication channel between controllers. Regulation cores must spend as little time as possible on communication, since the target regulation iteration period may be as short as 10 μ s.

FGC PRIVATE NETWORK DESIGN

The FGC Private Network protocol stack was primarily designed for a narrow use case: cyclic exchange of structured data between known peers on a private network. The main priorities are deterministic behavior and low runtime overhead, while preserving ease of maintenance and evolution. It is therefore kept deliberately simple.

* clement.zrounba@cern.ch

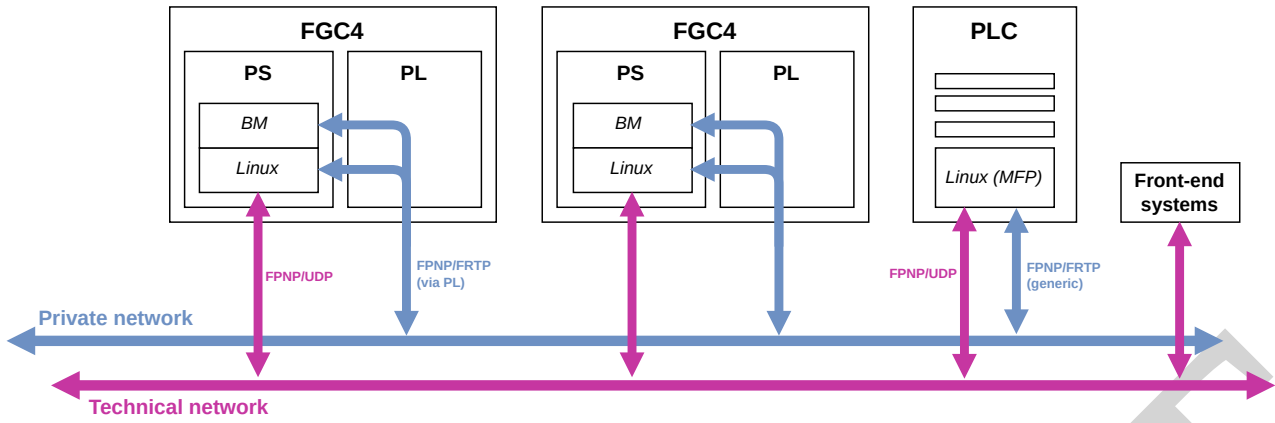


Figure 1: Communication paths in POPS+: several FGC4 devices and a PLC use the private network using FPNP/FRTTP to communicate real-time data; a non-real-time communication path using FPNP/UDP is also available for development (via the network interface managed by Linux).

The protocol stack comprises an application-layer protocol, the FGC Private Network Protocol (FPNP), and a transport-layer protocol, the FGC Real-Time Protocol (FRTTP). FPNP is implemented in software, while FRTTP may be implemented either in PL on the FGC4 platform, or in software on generic Linux hosts, such as the MFP PLC. Figure 1 depicts possible communication paths between devices, over a private network connected to the FGC4's PL interfaces, or over IP if needed.

Protocol-related definitions, such as offsets, bit masks, and enumerations, are maintained in YAML and used as a single source of truth. Required artifacts are then generated from those files using *cheby* [3], a tool developed at CERN for hardware/software interface definition. In particular, we can use *cheby* to generate the C++ memory mappings used by the software stack, the corresponding VHDL definitions, as well as documentation and debugging aids. Beyond protocol versioning, this approach also helps reduce mismatches between software and gateware implementations.

FGC Real-Time Protocol

At the transport layer, the FGC Real-Time Protocol introduces the concept of a *channel*, representing a logical communication path from a transmitter to one or more receivers. Channels support both uni-cast and multi-cast operation.

On MPSoC devices, FRTTP is implemented directly in the PL, ensuring that communication performance meets the latency and determinism requirements of POPS+, with margin for extendability to future CERN applications. In this implementation, represented in Fig. 2, a channel maps directly to a memory region, and a DMA engine transfers data to and from the L2 cache of the PS using the Accelerator Coherency Port (ACP). In constrained settings, the bare-metal regulation application will rely on periodic DMA transfers to send outgoing frames, in order to avoid PS-PL communications during control.

Thanks to system-wide time synchronization, both incoming and outgoing packets are timestamped in the PL. This enables the computation of precise, per-channel, latency

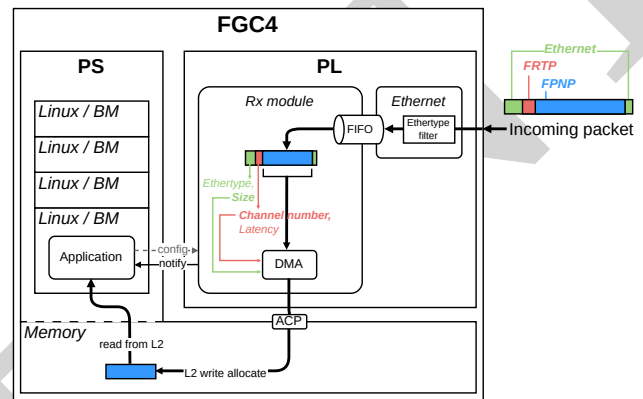


Figure 2: Illustration of the receiver data path for FRTTP implemented in the PL. The transmitter path is similar, with DMA transfers triggered by a timer in cycling operation.

statistics. Data integrity is ensured with the CRC of the 802.3 Ethernet L2 layer, which is verified in logic.

On generic Linux hosts, such as the MFP of the PLC, the protocol is implemented in software and integrated into a Linux user-space process. In this case, timing requirements are less stringent than those governing communication between FGC4 devices, although packets must still reach their destination within well-defined constraints.

FGC Private Network Protocol

At the application layer, the FGC Private Network Protocol allows communicating real-time data efficiently. As shown in Fig. 3, FPNP frames consist in a header, a variable-size payload and a footer.

The frame header provides key metadata for frame processing, including protocol version and sequence information. The frame footer is used to ensure data integrity, with an end-of-frame marker as well as a copy of the sequence number present in the header.

Application data is written to the payload and encapsulated into structured elements called Blocks. Each Block is a predefined, indivisible data unit comprising a header and an

associated payload, as illustrated in Fig. 3. The Block header contains metadata that describes the payload: its length, type (that uniquely defines the payload structure and interpretation), and a version number to ensure consistency between transmitter and receiver. A single FPNP frame may contain one or multiple Blocks, enabling efficient aggregation of heterogeneous data.

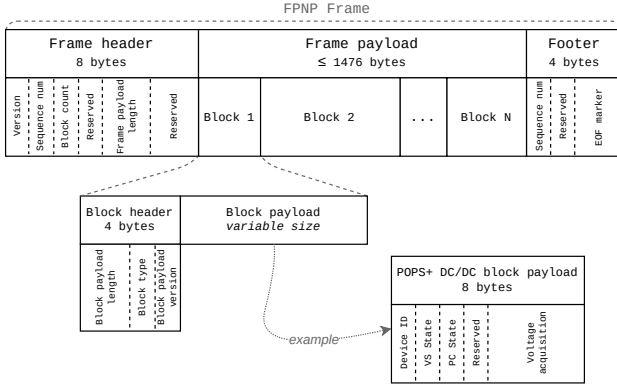


Figure 3: Specification of the FPNP frame and Block layout, and example of a payload block used by POPS+. Blocks are versioned independently from the protocol.

Figure 3 also provides an example of a Block transmitted by POPS+ DC/DC controllers to the PLC unit as well as to the Main Controller. This Block conveys information related to the converter state as well as an analogue voltage measurement. The PLC aggregates packets from multiple FGC4 devices to obtain a global view of the system state, enabling coordinated monitoring and control actions.

In constrained real-time scenarios, the frame layout is meant to be prepared during initialization and expected not to change during cycling. At runtime, the application only updates the values that vary from one iteration to the next, while transmission is handled by the PL using cache-coherent DMA during a pre-defined time window.

The header and footer metadata allows a receiver to detect corrupted data, and missing or duplicated frames without adding significant processing cost. It is the responsibility of the application to define how these use-cases are handled. For POPS+, we expect FGC4 devices connected to the GbE private network to utilize less than 10% of the available bandwidth, implying a very low probability of packet loss. Nevertheless, a policy must be established to define the system behavior under such conditions. For instance, failure of the DC/DC FGC4s to receive frames containing modulation indices from the Main Controller may lead to a power converter trip. Although this policy remains to be defined, the low network utilization suggests that such communication failures will be rare and unlikely to affect converter availability.

IMPLEMENTATION AND VALIDATION

The FPN software stack is implemented as a header-only C++ library aiming at low runtime cost without sacrificing

clarity. Structured access to frame memory is provided through semantic and typed C++ mappings generated by the protocol-related YAML definitions.

While the ultimate transport backend in FPN is the PL-based Ethernet path on FGC4, the same protocol logic can also be used with on Linux hosts over raw Ethernet, and additional transport backends such as UDP or file-replay are available for development and tests.

Supporting tools are being developed, including a Wire-shark dissector and Python bindings.

The C++ library is implemented and usable with the aforementioned transport backends. In particular, the PL communication path has been validated in the laboratory on FGC4 prototype boards, confirming the intended software/gateway split. Complete end-to-end timing characterization is still in progress and depends on a properly synchronized measurement setup.

This is sufficient to support the next integration steps for POPS+ and other EPC applications with similar low-latency communication requirements. As POPS+ validation based on Hardware-in-the-Loop testing continues [4], realistic latency statistics will soon be measured, once all the FGC4 devices are installed.

CONCLUSION

The FPN stack addresses the need for low-latency and deterministic inter-FGC communication on FGC4. Its main features are a hardware-assisted transmission path, a software design that minimizes work in the control loop, and model-driven protocol definitions shared across software and gateway.

This combination provides a practical basis for POPS+ deployment and for future CERN power converter control systems requiring private real-time exchanges.

The design trade-offs of the protocol stack also make it a good candidate beyond EPC group's activities, and it has been proposed to be contributed as a CERN-wide standard for low-latency multi-device communication, as part of the EPC group's effort to converge on established CERN technologies.

REFERENCES

- [1] F. Boattini, J.-P. Burnet, and G. Skawinski, "POPS: The 60 MW power converter for the PS accelerator: Control strategy and performances", in *Proc. EPE-ECCE Europe 2015*, Geneva, Switzerland, Sep. 2015. doi:10.1109/EPE.2015.7309267
- [2] M. Marin Rodriguez, O. Minaiev, W. Niinimaeki, and J. de Paco Soto, "A new PLC-based control system to orchestrate the PS main power converters", in *Proc. ICALEPCS'25*, Chicago, IL, USA, Sep. 2025, pp. 236-243. doi:10.18429/JACoW-ICALEPCS2025-TUBG002
- [3] T. Gingold, *Cheby*, GitLab repository, <https://gitlab.cern.ch/cohtdrivers/cheby.git>
- [4] D. Arominski, D. Zielinski, and R. Murillo-Garcia, "Validating VSlib, the voltage source control library used in the fourth

generation of function/generator controllers at CERN”, in *Proc. ICALEPCS'25*, Chicago, IL, USA, Sep. 2025, pp. 1112-1116.
[doi:10.18429/JACoW-ICALEPCS2025-WEPD036](https://doi.org/10.18429/JACoW-ICALEPCS2025-WEPD036)

PREPRINT