ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP063

THE EMBEDDED MONITORING PROCESSOR FOR HIGH LUMINOSITY LHC

P. Moschovakos*, V. Ryjov, S. Schlenker, CERN, Geneva, Switzerland
D. Ecker, University of Wuppertal, Wuppertal, Germany
J. B. Olesen, Aarhus University, Aarhus, Denmark

Abstract

The Embedded Monitoring Processor (EMP) is a versatile platform designed for High Luminosity LHC experiments, addressing the communication, processing, and monitoring needs of diverse applications in the ATLAS experiment, with a focus on supporting front-ends based on lpGBT (low power Giga-Bit Transceiver, a CERN-built radiation-hard ASIC). Built around a commercial System-on-Module (SoM), the EMP architecture emphasizes modularity, flexibility and the usage of standard interfaces, aiming to cover a wide range of applications and facilitating detector integrators to design and implement their specific solutions. The EMP software and firmware architecture comprises epos (the EMP operating system), quasar OPC UA servers, dedicated firmware IP cores and an ecosystem of different software libraries. This abstract outlines the software and firmware aspects of the EMP, detailing its integration with lpGBT optical interfaces, programmable logic development, and the role of the LpGbtSw library as a Hardware Abstraction Library for the LpGbt OPC UA server.

EMBEDDED MONITORING PROCESSOR CONTEXT

Introduction

In the context of high-energy physics experiments, especially with the High Luminosity upgrade of the Large Hadron Collider (LHC), the requirements for enhanced radiation-tolerant Detector Control Systems (DCS) are high. To address the emerging challenges, the Embedded Monitoring Processor (EMP) and the Embedded Monitoring and Control Interface (EMCI) CERN-made boards were introduced.

Designed to operate in harsh radiation environments near the detector, the EMCI [1] board serves as the frontend component, facilitating the flow of slow control data between multiple frontend electronic devices and with the experiment's control system.

The EMP [2], in contrast to the EMCI, serves as a baseboard for the TE0807 MPSoC Modules from Trenz Electronic [3] which feature the AMD Zynq [4] UltraScale+ MPSoC. This versatile platform is designed primarily for controlling and monitoring LHC experiments that utilize lpGBT-based frontends, such as EMCI. Residing in a low-radiation service area, the EMP functions as the backend that processes and exchanges frontend data with the DCS. Moreover, a single EMP can connect with up to 12 lpGBT-based frontend boards via optical links, interfacing them to the Supervisory Control and Data Acquisition (SCADA) system via Ethernet, as illustrated in Fig. 1.

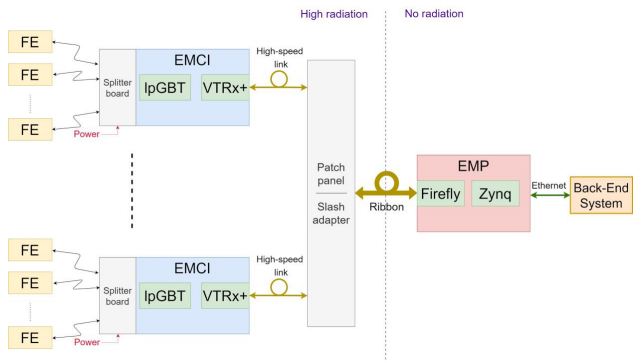

Figure 1: The EMP-EMCI system diagram.

The Modular Design of EMP

In its broader context, the EMP is more than a hardware entity. Its full functionality is realized with the integration of tailored software for the Zynq's Processing System, firmware for the Zynq's Programmable Logic, and hardware extensions. This approach allows the EMP to serve to a wide range of monitoring and control applications.

Furthermore, the platform enables a set of functional capabilities, organized into separate application paradigms. These paradigms, discussed in detail in the following section, can coexist and function concurrently within a single EMP unit, provided certain technical conditions, like the availability of logic resources is sufficient.

Interface Spectrum

In the context of ATLAS detector control systems, the low-power Gigabit Transceiver (lpGBT) [5] plays a crucial role, particularly its Internal Control (IC) communication channel. This channel is pivotal for the configuration, monitoring, and control of lpGBT. Elinks, electronic links that connect frontend electronics, are especially relevant in this context. These elinks are used in interfacing the frontends, as indicated in the preceding Fig. 1. Beyond its role in configuration, the lpGBT IC communication offers a wide array of I/O interfaces for slow control. In this context, the standard firmware IP core used for lpGBT communication, known as the lpGBT FPGA IP core, is used within the PL of the Zynq.

Moreover, the lpGBT facilitates data transmission using custom protocols, with employing the available elinks. To

* paris.moschovakos@cern.ch

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP063

elaborate on data transmission, the lpGBT transforms protocol data into specific line codes. An example is the 8b10b line code, which is prevalent in this domain. This transformation is realized through a procedure called custom data encapsulation. The line coding and its subsequent protocol are encoded and decoded by an EMP's programmable logic entity.

Transitioning to transmission aspects, the 'SelectIO' refers to a configurable I/O technology specific to certain Xilinx devices, including some Zynq models. This technology allows for flexible I/O pin configurations, essential for supporting standards such as LVDS or SLVS. When the SelectIO of the EMP's System-on-Module is set to operate as transceivers, these standards come into play. Concurrently, the EMP can adjust to specific configurations and IO layouts, with custom programmable logic entities taking a primary role, especially when handling line codecs.

Regarding universal interfaces, the functional mode is designed to connect with well-known protocols like CAN, UART, or SPI. Using programmable logic entities, a variety of industry-standard IP cores can be accessed, primarily via platforms such as AMD Vivado or OpenCores. Often, these interfaces are supported by Linux drivers, which require only basic configuration.

Finally, the System-on-Module's processing system is a key aspect, particularly when operating through Multiplexed-I/O (MIO). While its main function is to focus on EMP monitoring, additional functionalities, such as UARTs for console input/output, can also be added to enhance its capabilities.

Building upon the foundational capabilities of the Zynq, the EMP integrates custom protocols and tailored hardware solutions, inheriting and profiting from the best of both to meet its specific applications and needs.

Architectural Foundations

The EMP is designed to primarily support experiment controls in DCS backend systems such as SCADA. Given the standards adopted by DCS backend systems, we utilize the OPC UA (Open Platform Communications Unified Architecture) servers for relaying data from the hardware. These servers present data through an abstraction layer, emphasizing crucial metrics for users like equipment temperatures and voltages, rather than offering raw machine-specific data. The technological approach for the detector controls comprises: selecting a RHEL-based distribution for the operating system of the PS, using the *quasar* ecosystem for OPC UA interfaces towards DCS backend, incorporating the lpGBT FPGA IP core for MGT-based lpGBT connections in the PL, and employing the AXI bus to connect specific programmable logic entities to the EMP's processing system.

EMP HARDWARE DESIGN

The Embedded Monitoring Processor (EMP) is engineered as a robust hardware hub, optimized for lpGBT backend functionalities. Figure 2 depicts the EMP prototype board.

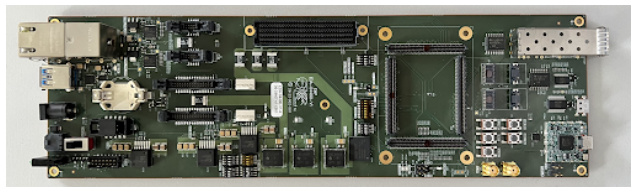

Figure 2: EMP prototype board.

Core Components At its core, the EMP houses a commercial Zynq Ultrascale+ SoM (Trenz TE0807) onto a custom-designed PCB baseboard (see Fig. 3). The SoM offers as minimum 4 GTH MGT quads and 269 IOs interfacing Zynq's capabilities. The module's small form factor and low power consumption aligns with the requirements of the project. The module comprises the required power supplies, manages the Zynq's power sequencing, and includes all essential peripherals for system operation.

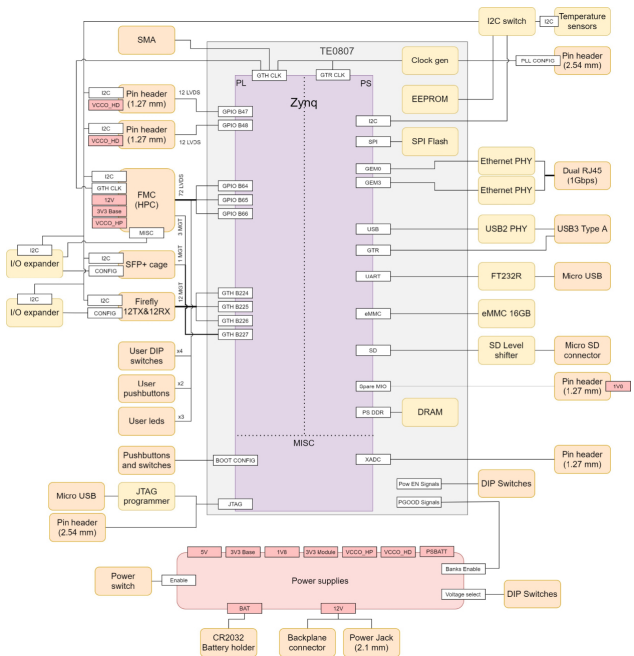

Figure 3: EMP baseboard diagram and its interfaces.

Peripheral Components The EMP's design interfaces with 12 Multi-Gigabit Transceivers (MGT) in accordance with the VL+ standards. This is achieved using Samtec UEC5/UCC8 connectors that are linked to transceivers located in designated banks. All configurations and status indicators are managed through the PS using an I2C interface. In addition to the main set of MGT transceivers, the EMP features an extra GTH transceiver that is connected to an SFP+ slot. Further expanding its connectivity capabilities, the device also integrates an FMC (HPC) connector that complies with the VITA 57 standard and provides 144 single-ended/72 differential lines. This connector not only facilitates diverse connectivity solutions but is also interfaced with the baseboard's power supplies, enabling power

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP063

provisioning for external components. Regarding the PS of the Zynq, it provides multiple interfaces. These include dual Ethernet ports, a USB3 interface, and UART communication. The Zynq PS employs specific pins (MIOs) to support diverse functions, and it's through these interfaces that remote access to the EMP is established. The Xilinx Analog to Digital Converter (XADC) provides dedicated pins for interfacing. Moreover, the EMP supports both remote reprogramming of the Zynq Programmable Logic and direct JTAG programming through dedicated connectors.

Memory The EMP is equipped with a comprehensive memory infrastructure, including a 256 kb EEPROM, 512 Mb SPI memory, 4 GB RAM, 16 GB eMMC storage, and an accessible MicroSD card slot, ensuring flexibility and capacity for extensive data handling.

Power Management The EMP operates on a single voltage input. Multiple DCDC converters are utilized to generate the various voltage levels needed for its components.The design specifics of the DCDC converters, as well as their power allocation, are currently being refined throughout the EMP development process The module implements a precise power-on sequence. To support the Zynq's real-time clock system, a CR2032 coin battery holder is included.

Further details on the hardware design and its components can be found in [2].

EMP SOFTWARE AND FIRMWARE ARCHITECTURE

epos: The EMP Operating System

Although the configuration of EMP hardware and software varies based on specific use-case requirements, the commonalities in hardware parts, such as the Ethernet or the on-board I2C devices, and software like the operating system distribution or development framework, result in a common need for a boot image, Linux kernel and file system with basic hardware periphery and software tooling support.

epos is the framework to build the boot image, the Linux image and the file system for the EMP [6].

- *epos-bsp* comprises the layer of software containing hardware-specific drivers and other routines that allow the chosen operating system to function in the particular EMP hardware environment. The software includes a PetaLinux BSP which contains the basic common hardware support with the corresponding device tree and the appropriate configuration to build and package the FSBL, U-Boot and Linux image for EMP.

- The *epos-rootfs* is a set of tools to create the file system that is deployed in EMP. The tools take into account the common standard OS distribution supported by CERN IT. As part of epos-rootfs a minimum set of packages commonly needed for EMP usage and development, including that for the development of OPC UA middleware is provided. The product of epos-rootfs is a fully functioning rootfs that can be loaded in a SD card or over NFS to EMP and paired with the products of epos-bsp.

- *etools*, a toolbox, that help to deploy the firmware from within the PS and the operating system. These tools can process a .xsa file to produce a .dtbo and a .bit.bin file, and then load given .dtbo and .bit.bin files to a Zynq device.

quasar OPC UA Servers for EMP

quasar is a generic framework for creation of OPC UA servers [7], corresponding OPC UA clients (in the C++ programming language), SCADA tooling for setting OPC UA [8] connectivity. Both ATLAS and JCOP (the collaborative controls project for LHC experiments) have chosen *quasar* as the framework to create OPC UA software components.

- *quasar* itself is used to model OPC UA server(s) running in the EMP. The C++ programming language is the native language for *quasar*-based server development and it is the preferred choice for embedded software development and production level applications.

- *Cacophony* is an extension module to facilitate integration of instances of *quasar*-based servers into WinCC OA [9], which is the SCADA product in which the ATLAS DCS backend is built.

- *UaObjects* is an extension module for creation of C++-based OPC UA clients for given quasar-based servers.

Every peripheral of the EMP that either handles frontend communication or monitors the EMP itself requires a DCS interface. This means they need an OPC UA server. The *quasar* ecosystem not only facilitates the creation of these servers but also aids in backend integration and long-term maintenance.

LpGbtSw Ecosystem

The LpGbtSw Library The LpGbtSw library [10] is a versatile C++ based library designed to provide comprehensive software support for the lpGBT chip, independent of the chosen communication medium. Its functional scope is wide-ranging. It offers a universally applicable polymorphic API for register-based interactions with the lpGBT. Additionally, there is an effective realization of this interface specific to the EMP lpGBT communication channel. This realization uses the GBT-SC IC adapter via the libuio library [11]. The library also provides an lpGBT simulator, which is ideal for testing and developmental phases, especially when physical access to EMP or the lpGBT-based frontend is not available. Furthermore, there's a specialized API designed for streamlined communication with the lpGBT's slow-control peripheral interfaces, such as ADC and GPIO. This API simplifies their register configurations. Lastly, LpGbtSw includes a collection of tools named "demonstrators" that

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP063

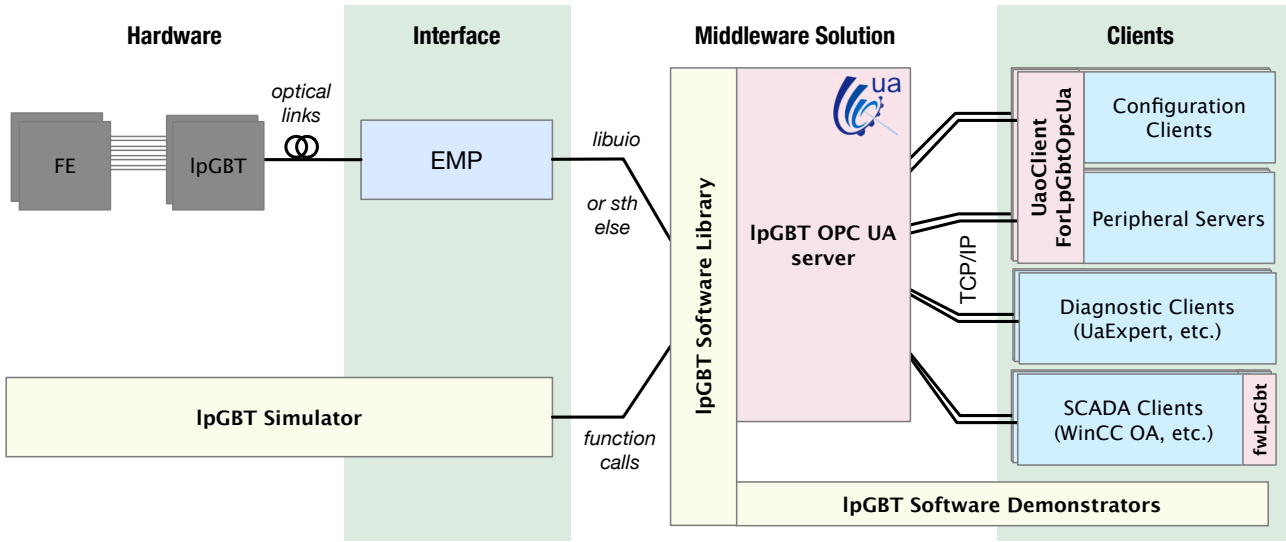

Figure 4: Global picture of the software suite. The *lpGbt Software package*, in light blue, comprises the *lpGbt Software API* to communicate with the lpGBT via different backends, the *lpGbt Simulator* to emulate lpGBT traffic for testing and development, and the *Demonstrator* tools which are used for standalone operations. The *LpGbt OPC UA* server and its ecosystem, in orange, is the middleware of choice to exchange data with the frontends. Finally, the *fwlpGBT* module automatizes the integration of the server data into WinCC OA.

allow users to engage with the library's varied functionalities through independent programs. Foremost, LpGbtSw's design purpose is to function as a Hardware Access Library (HAL) for an OPC UA server, ensuring users can efficiently engage with the lpGBT chip under diverse conditions and hardware configurations.

The LpGbt OPC UA Server The LpGbt OPC UA server provides access to the functionalities of the lpGBT chip for network clients, distinguishing it from the direct functionalities offered by LpgbtSw. This includes the chip's status, its configuration settings—including adjustments for the lpGBT's e-links—and its slow-control I/O operations. This server was created using the quasar framework for OPC UA, leveraging the LpGbtSw (as referenced above) and its high level API. The server allows for the usage of general purpose test clients for diagnostic purposes such as the UaExpert [12]. A global overview of the software ecosystem of lpGBT is illustrated in Fig. 4.

Dedicated Firmware Components

The lpGBT FPGA IP Core The *lpGBT FPGA IP core* is an interface between FPGA's MGTs (towards frontend) and the programmable logic that reads and writes particular elinks (towards backend). It provides clocking gearbox, multiplexing elinks into lpGBT frames, demultiplexing from lpGBT frames into elinks, dealing with the lpGBT frame structure. The IP core is used whenever the lpGBT is to be talked to using the EMP's optical interfaces. The lpGBT has a couple of configuration items (as VHDL generics). One instance of the IP core should be used per every MGT (optical fibre).

The GBT-SC Core The *GBT-SC IP core* was initially made for the GBT project [5] to provide means for EC and IC channel communication. However the same IC channel protocol is used to perform reads and writes of the lpGBT registers, thus, the same IP core is used. When composing the firmware of the EMP, an instance of this core should be placed between the IC elink exposed in an instance of the lpGBT FPGA IP core and the GBT-SC IC.

The AXI Adapter Core The EMP team developed the GBT-SC IC core AXI adapter IP core to facilitate efficient access to the GBT-SC IC core from the processing system through the AXI interconnect. This IP core is generated using the AirHdl tool [13] that is based on a register map as depicted in Fig. 5. The AXI4Lite is employed for linking custom components to the processing system, especially for peripherals that use a register map. These peripherals can benefit from the UserIO (uio) software support model. The Puzzled Lizard Wizard [14], a tool that is used to generate the register map in the software, were instrumental for tasks like modeling AXI4-Lite interfaces and the concurrent generation of software support libraries for components integrating both AirHdl and libuio.

Results and Demonstrations

A vertical slice demonstration was successfully carried out, showcasing the integration and functionality of the EMP software and firmware architecture. The LpGbt OPC UA server, when visualized via the UaExpert OPC UA diagnostic tool, is depicted in Fig. 6. This demonstration not only validated the theoretical concepts but also confirmed the practical application of the tools, libraries, and systems discussed in the preceding sections.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP063

Offset	Name	Description	Type	Access	Actions
0x0	magic	Magic identifier -- corresponds to empl	REG	READ_ONLY	
0x4	control	IC channel control commands	REG	READ_WRITE	
0x8	status	IC channel status flags	REG	READ_ONLY	
0xC	data_rx	Data received from the lpGBT	REG	READ_ONLY	
0x10	data_tx	Data to be sent to lpGBT	REG	READ_WRITE	
0x14	register_addr	The first address of the (consecutive) registers on the lpGBT of which we wish to read/write.	REG	READ_WRITE	
0x18	lpGBT_addr	The I2C address of the lpGBT device you are interacting with.	REG	READ_WRITE	
0x1C	interrupt_enable	Interrupt (PL->PS) enable flags ('1' to enable)	REG	READ_WRITE	
0x20	interrupt_flags	Interrupt (PL->PS) flags ('1' is a triggered interrupt)	REG	READ_ONLY	
0x24	interrupt_clear	Interrupt (PL->PS) clear commands ('1' to clear)	REG	READ_WRITE	
0x28	reset	Command for resetting the pipeline FW	REG	READ_WRITE	
↕ address gap: 212 bytes					
0x100	counter_lhc_clock	–	REG	READ_WRITE	

+ Register + Array + Memory Duplicate

Figure 5: The EMP AXI register map.

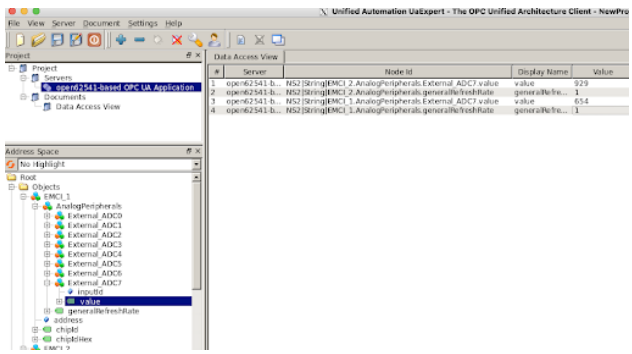

Figure 6: The LpGbt OPC UA server in UaExpert.

CONCLUSION

In conclusion, the Embedded Monitoring Processor effectively acts as a comprehensive control hub designed for the High Luminosity LHC experiments, offering versatility and functionality through its uniquely tailored software components. With the *epos* framework at its core, EMP facilitates the creation of boot, Zynq-specific Linux kernel, and filesystem images essential for various use-cases, providing a seamless interface with the underlying hardware environment. This foundational software layer ensures compatibility with CERN IT's standard OS distribution while supporting the development of OPC UA middleware, crucial for LHC experiment controls. The *quasar* framework, another essential software component, simplifies the development and integration of OPC UA servers and clients, with extensive tooling available for developers. It allows for efficient data management, ensuring that EMP can interact with backend systems such us SCADA smoothly, thereby reinforcing its role as an indispensable tool for control and monitoring applications in LHC experiments. This combination of hardware and software within the EMP not only highlights its potential as a universal control hub, but also shows the value of flexibility and modularity as key for efficient development and maintenance for different applications.

REFERENCES

[1] EMCI-EMP team, "EMCI Embedded Monitoring and Control Interface Specification", https://edms.cern.ch/ui/#!master/navigator/document?D:100643813:100643813:subDocs

[2] D. Blasco *et al.*, "Description and status of the EMCI-EMP interface", *J. Instrum.*, vol. 17, p. C06012, Jun. 2022. doi:10.1088/1748-0221/17/06/C06012

[3] Trenz Electronic, "Trenz Electronic TE0807 MPSoC Modules with Xilinx Zynq UltraScale+", https://shop.trenz-electronic.de/en/Products/Trenz-Electronic/TE08XX-Zynq-UltraScale/TE0807-Zynq-UltraScale/.

[4] Xilinx, "Zynq UltraScale+ MPSoC Data Sheet: Overview", https://www.mouser.com/pdfDocs/ds891-zynq-ultrascale-plus-overview.pdf

[5] GTB team, "lpGBT Manual", https://lpgbt.web.cern.ch/lpgbt/v1/

[6] P. Moschovakos, "epos gitlab repository", https://gitlab.cern.ch/emci-emp/epos

[7] P. P. Nikiel, P. Moschovakos, and S. Schlenker, "quasar : The Full-Stack Solution for Creation of OPC-UA Middleware", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 452. doi:10.18429/JACoW-ICALEPCS2019-MOPHA100

[8] W. Mahnke, S. H. Leitner, and M. Damm, "OPC unified architecture", Springer, Berlin, Germany, 2009.

[9] "Modern, efficient and flexible simatic wincc open architecture v3.15", ETM professional control GmbH, a Siemens Company, Tech. Rep. 6ZB5370-1EG02-0BA0-V2, Jan. 2017, https://siemens.com/wincc-open-architecture

[10] P. Moschovakos, "LpGbtSoftware gitlab repository", https://gitlab.cern.ch/atlas-dcs-emp/LpGbtSw

[11] Benedikt Spranger, "libuio - UserspaceIO helper library", https://github.com/missinglinkelectronics/libuio/

[12] UaExpert, "Unified Automation", https://www.unified-automation.com/products/development-tools/uaexpert.html

[13] AirHdl: register management done right, "AirHdl webpage", http://www.airhdl.com

[14] P. Moschovakos, "fwLpGbt gitlab repository", https://gitlab.cern.ch/atlas-dcs-emp/fwlpgbt