PROCEEDINGS OF SCIENCE

Implementation of a CAN bus interface for the Detector Control System in the ALICE ITS Upgrade

S.V. Nesbo*[a], J. Alme[b], M. Bonora[e], M. Rentsch Ersdal[b], P. Giubilato[c], H. Helstrup[a], M. Lupi[e], G. Aglieri Rinella[e], D. Röhrich[b], J. Schambach[d], A. Velure[e] and S. Yuan[b]

[a] *Western Norway University of Applied Sciences, Norway*
[b] *University of Bergen, Norway*
[c] *Università e INFN, Italy*
[d] *The University of Texas at Austin, United States of America*
[e] *European Organization for Nuclear Research (CERN), Switzerland*
E-mail: svn@hvl.no, johan.alme@uib.no, matthias.bonora@cern.ch, magnus.ersdal@uib.no, Piero.Giubilato@cern.ch, Havard.Helstrup@hvl.no, matteo.lupi@cern.ch, Gianluca.Aglieri.Rinella@cern.ch, Dieter.Rohrich@uib.no, jschamba@physics.utexas.edu, arild.velure@cern.ch, shiming.yuan@uib.no

For the Upgrade of the ALICE Experiment in Long Shutdown 2, a new Inner Tracking System (ITS) is being commissioned, based on the ALPIDE Monolitihic Active Pixel Sensor (MAPS) chip.

Data readout from the ALPIDE chips is performed by 192 Readout Units (RU), which are also responsible for trigger distribution, detector monitoring and configuration, and control of the sensor chips. The monitoring and control of the experiment is performed by the Detector Control System (DCS), which communicates to each RU through the same optical GBT-Versatile link used for slow-control operations. A Controller Area Network (CAN bus) interface is provided as a supplementary control path, and this paper will discuss the implementation of this interface.

Topical Workshop on Electronics for Particle Physics TWEPP2019
2-6 September 2019
Santiago de Compostela - Spain

*Speaker.

PoS(TWEPP2019)083

1. Introduction

The upgraded ITS employs 24 120 ALPIDE [1] pixel sensor chips arranged in seven cylindrical barrels, as shown in figure 1, and 192 Readout Unit (RU) boards responsible for:

- Configuration, monitoring, and control of the sensor chips
- Timing and trigger distribution
- Readout and formatting of sensor data
- Power distribution system control

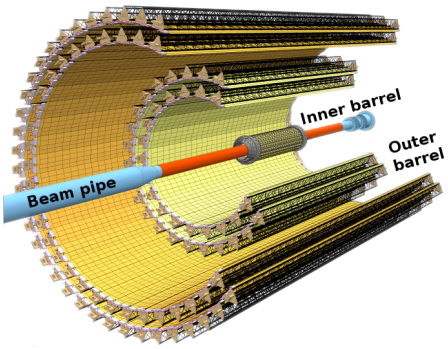


Figure 1: Upgraded ITS detector [2].

The RU boards communicate on optical links with the Common Readout Unit (CRU) boards hosted in the First Level Processor (FLP) computers. The data travel from the ALPIDE chips to the RUs, and continue to the CRUs in the FLPs.

1.1 Detector Control System (DCS)

DCS is a large scale SCADA control system, implemented with WinCC OA [3]. Figure 2 shows the communication with the Front End Electronics (FEE), which is designed as a client/server architecture. The FEE client (ALF) subcribes to the server (FRED) using the Distributed Information Management (DIM) protocol.

During normal operation, the main communication path between DCS and the main FPGA in the RU is via the FLP using the GBT-Versatile links. A custom protocol allow for direct access to internal registers in the FPGA design, which enables configuration, control, and monitoring of the RU, ALPIDE sensor chips, and external Power Boards. For redundancy, the RUs can be accessed via Controller Area Network (CAN bus) as well. A dedicated ALF client for CAN bus will be implemented, allowing for seamless transition between GBT and CAN, and making DCS completely agnostic as to which communication path is used.

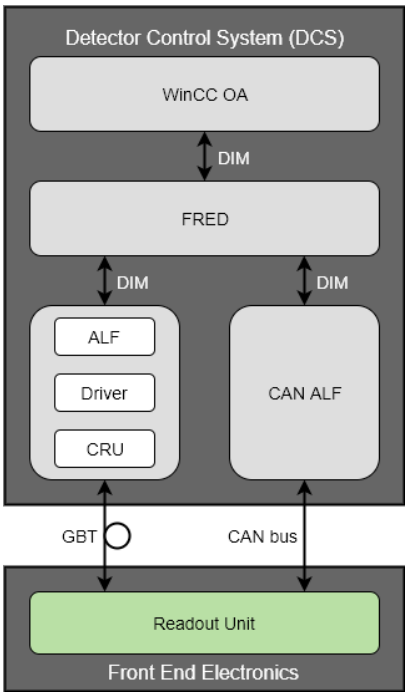


Figure 2: Block diagram illustrating the DCS communication paths via GBT and CAN.

2. ITS Readout Unit

The main component of the RU (figure 3) is an SRAM-based Xilinx UltraScale FPGA, which is responsible for most of the RU board's functionalities. Figure 4a illustrates a simplified structure of the FPGA design for the main FPGA, highlighting the main functional blocks and their connections to the internal Wishbone (WB) bus. The bus uses 15-bit addressing and has a data

width of 16 bits. There are currently three WB masters and a round-robin arbiter (not illustrated) that allows the masters to share access to the bus. The WB masters have two FIFO interfaces, one for WB transaction *requests*, and one for WB transaction *results*. The requests would be to either read or write a register, and the results would be the register data that was read, or indication of write success in the case of a write. The FIFO interfaces allows for queuing and asynchronous sequencing of WB transactions. The three WB masters connect to external interfaces (via some protocol logic): GBT, CAN bus, and USB (the USB is a debug feature and will be disabled for use in the cavern during the data taking). This makes it possible to initiate WB transactions and control the RU via these interfaces.

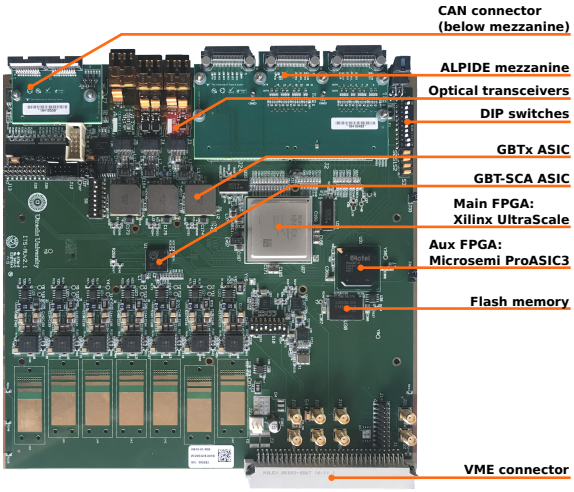
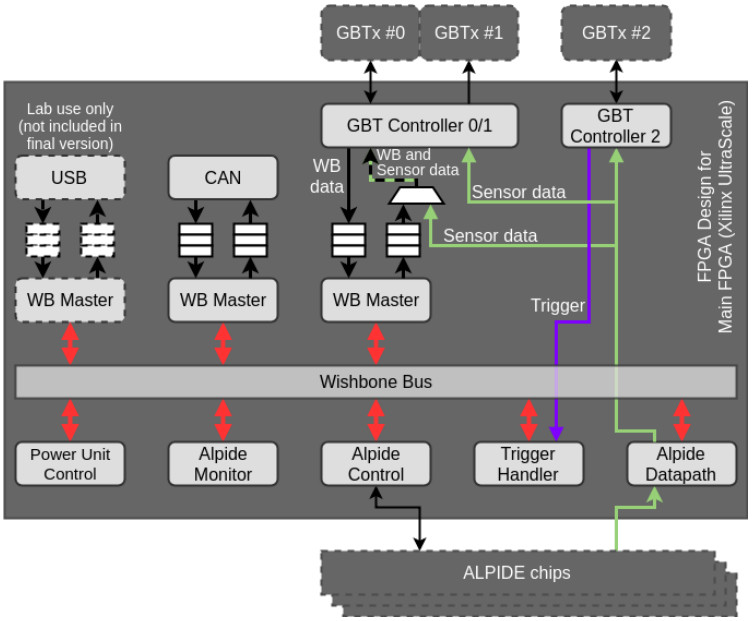
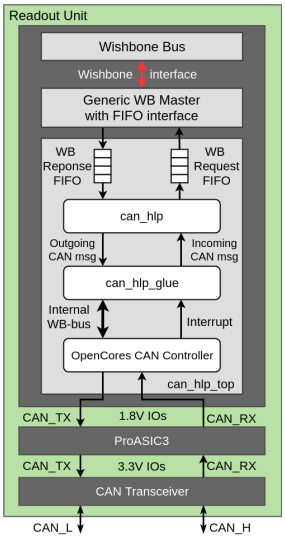


Figure 3: The ITS Readout Unit version 2.1.



(a) Simplified block diagram of the FPGA design for the main FPGA in the RU, with emphasis on connections to the internal Wishbone Bus.



(b) FPGA implementation of CAN bus and HLP protocol.

Figure 4: Block diagrams of top-level FPGA design and HLP implementation for the main FPGA.

3. CAN bus interface to the Readout Units

The GBT links that the DCS normally relies on may not be available under some circumstances, such as during maintenance periods or due to technical problems in the control chain toward a specific RU. To address this, the specification for the DCS calls for an additional CAN bus

interface to the RU. That provides the DCS with the redundancy to ensure that critical parameters, such as temperatures and voltages for the sensor chips, are monitored under all conditions.

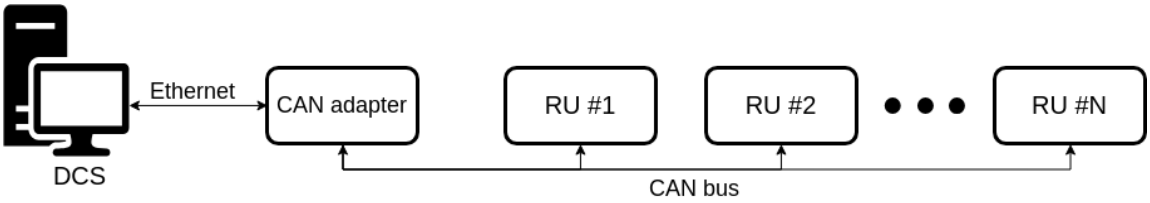


Figure 5: Simplified illustration of CAN bus connectivity for DCS.

3.1 High Level Protocol (HLP)

A custom High Level Protocol (HLP) for DCS is implemented on top of the CAN frames, and is based on a protocol developed for the TOF detector in the STAR experiment [4]. The protocol was chosen for its simplicity, although there are standard protocols such as CANopen. Table 1 shows the implementation of our HLP protocol. In this protocol the DCS acts as a master on the bus and the RUs are slaves, as indicated in figure 5. The DCS initiates all requests, and the RUs are only allowed to respond. Our implementation provides access to the WB bus in the RUs, and uses standard CAN frames with 11-bit ID. Eight of those bits are used for node ID, and the remaining three bits are used to indicate the type of command. Register address and data are sent as payload in the CAN frames.

4. Implementation of CAN bus and HLP in Readout Unit

The current implementation of CAN bus and HLP in the RU FPGA design is illustrated in figure 4b. It is based on an open source CAN controller from the OpenCores website [5], and the necessary logic for the aforementioned HLP protocol is implemented in the *can_hlp* block in the figure. In addition, a layer of glue logic is implemented to interface between the protocol logic and the CAN controller. The CAN controller has a WB interface to which the glue logic interfaces with directly and that is not connected to the main WB interface in the FPGA design.

Table 1: High Level Protocol (HLP) commands and payload.

Command	Byte 0	Byte 1	Byte 2	Byte 3
Write Request (0x2)	Addr[14:8]	Addr[7:0]	Data[15:8]	Data[7:0]
Write Response (0x3)	Addr[14:8]	Addr[7:0]	Data[15:8]	Data[7:0]
Read Request (0x4)	Addr[14:8]	Addr[7:0]	N/A	N/A
Read Response (0x5)	Addr[14:8]	Addr[7:0]	Data[15:8]	Data[7:0]

5. Results and outlook

The current version of the system was tested using one RU in a setup as shown in figure 6, with the exact type of cable that will be used in the experiment, 118 meters in length. The cable

has 0.5 mm^2 twisted pairs with 120 Ω impedance. In this setup the system performed reliably with a data rate of 250 kbps. Although the recommended cable length limit for 500 kbps is 100 meters [6], such a high bitrate with 118 meter cables may be achievable by adjusting the bit timing values in the CAN controller.

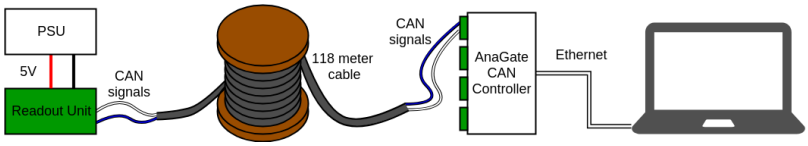


Figure 6: Test setup for CAN bus HLP.

Random data was written and read back from a test register in the RU, and counter registers read back and verified. The test script that performed the transactions ran for 18 hours without errors, performing around 17 million HLP read transactions and 6 million HLP write transactions.

The average turnaround was of 83 HLP transactions per second, which corresponds to around 3 ms per transaction on average. Tests with two RUs have also been run to verify that nodes can be individually addressed and that broadcasting of HLP commands works as intended. This test was not performed with the long cable. Tests of a full setup with up to 10 RUs per cable will be performed in the near future.

The current version of CAN bus and HLP in the RU is not protected against radiation effects. In particular, the CAN controller that was used proved difficult to protect using the techniques employed in the FPGA design for the RU [7]. A new CAN controller is currently being designed, which along with the HLP logic should be relatively easy to protect using Triple Modular Redundancy (TMR) with the aforementioned techniques. The protocol logic will also be able to interface with the new CAN controller directly, which removes the need for the glue logic in the current design, and should give the design a smaller footprint in terms of resource utilization on the FPGA.

References

[1] G. Aglieri Rinella, *The ALPIDE pixel sensor chip for the upgrade of the ALICE Inner Tracking System*, *Nucl. Instrum. Meth.* **A845** (2017) 583.

[2] The ALICE Collaboration, *Technical Design Report for the Upgrade of the ALICE Inner Tracking System*, *Journal of Physics G: Nuclear and Particle Physics* **41** (2014) .

[3] The ALICE Collaboration, *Technical Design Report for the Upgrade of the Online-Offline Computing System*, Tech. Rep. CERN-LHCC-2015-006. ALICE-TDR-019, Apr, 2015.

[4] J. Schambach, L. Bridges, W. Burton, G. Eppley, K. Kajimoto and T. Nussbaum, *CANbus protocol and applications for STAR TOF control*, *Journal of Physics: Conference Series* **331** (2011) 022038.

[5] I. Mohor, "CAN Protocol Controller." `https://opencores.org/projects/can`, 2003.

[6] S. Corrigan, "Controller Area Network Physical Layer Requirements, Application Report." `http://www.ti.com/lit/an/slla270/slla270.pdf`, 2008.

[7] K. M. Sielewicz, G. A. Rinella, M. Bonora, P. Giubilato, M. Lupi, M. J. Rossewij et al., *Experimental Methods and Results for the Evaluation of Triple Modular Redundancy SEU Mitigation Techniques with the Xilinx Kintex-7 FPGA*, in *2017 IEEE Radiation Effects Data Workshop (REDW)*, pp. 1–7, July, 2017, DOI.