

OPEN ACCESS

ATLAS ITk-Pixel DAQ system

To cite this article: Wael Alkakhi on behalf of the ATLAS ITk collaboration 2024 *JINST* **19** C11013

View the [article online](#) for updates and enhancements.

You may also like

- [Memory-assisted measurement-device-independent quantum key distribution](#)
Christian Panayi, Mohsen Razavi, Xiong Ma et al.
- [Physically unclonable functions based on small delay defects in QCA](#)
Mohammad Hadi Valavi and Ghassem Jaberipur
- [Entanglement distribution in multi-platform buffered-router-assisted frequency-multiplexed automated repeater chains](#)
Mohsen Falamarzi Askarani, Kaushik Chakraborty and Gustavo Castro do Amaral



UNITED THROUGH SCIENCE & TECHNOLOGY

ECS The Electrochemical Society
Advancing solid state & electrochemical science & technology

**248th
ECS Meeting**
Chicago, IL
October 12-16, 2025
Hilton Chicago

**Science +
Technology +
YOU!**

**SUBMIT
ABSTRACTS by
March 28, 2025**

SUBMIT NOW

25TH INTERNATIONAL WORKSHOP ON RADIATION IMAGING DETECTORS
LISBON, PORTUGAL
30 JUNE – 4 JULY 2024

ATLAS ITk-Pixel DAQ system

Wael Alkakhil¹ on behalf of the ATLAS ITk collaboration

*II. Physikalisches Institut, Georg-August-Universität Göttingen,
Friedrich-Hund-Platz 1, 37077 Göttingen, Germany*

E-mail: wael.alkakhi@uni-goettingen.de

ABSTRACT: During the ATLAS High-Luminosity Large Hadron Collider (HL-LHC) upgrade, the current inner detector is going to be replaced by an all-silicon Inner Tracker (ITk). The pixel detector, located in the innermost part of the ITk, comprises 9716 modules arranged in 5 cylindrical layers around the beam line. The ITk-Pixel Data Acquisition (DAQ) system basic read-out chain includes the YARR software, communicating with the FELIX PCIe board acting as an interface connected through lpGBT transceivers to the on-detector front-end (FE) chips ITkPix. The FEs are grouped in triplet (3-FE) and mostly in 4-FE Quad Modules (QM), that are installed on local supports, which are integral parts of the ITk structure. The FELIX system is also used for performing Quality Control (QC) tests during integration. The work describes the development steps and corresponding testing and read-out chain validation results. A representative read-out sub-system will be used to develop and validate the different aspects of the read-out chain. This subsystem can be a Loaded Local Support (LLS) comprising few tens of ITkPix QMs with serial powering (SP) and opto-box connection. In order to have the readout chain validated, developments on trigger and command sending and data reading of YARR FelixClient controller were consequently required, working first on a lab setup with a couple of ITkPix single chip cards (SCCs) and QMs. This step has been carried out successfully, paving the road to the next LLS sub-system readout test, once more QMs are available.

KEYWORDS: Data acquisition circuits; Data acquisition concepts; Hybrid detectors; Particle tracking detectors (Solid-state detectors)



Contents

1	Introduction	1
2	ITk read-out DAQ	2
3	Development and validation results	3
4	Conclusion	6

1 Introduction

After Run 3, the LHC will undergo a significant upgrade, entering the HL-LHC era for Run 4. During this period, the luminosity will increase by about 5–7 times, reaching $\approx 7 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$. At a bunch-crossing rate of 40 MHz, the ATLAS trigger-DAQ system during HL-LHC will support a level-0 trigger accept rate of 1 MHz, while around 200 pile-up events are expected.

The current ATLAS inner detector cannot meet the HL-LHC requirements, motivating an upgrade of the ATLAS inner detector. It will be replaced by the all-silicon ITk. The ITk consists of a 5-layer pixel detector surrounded by a 4-layer strip detector, as shown in figure 1.

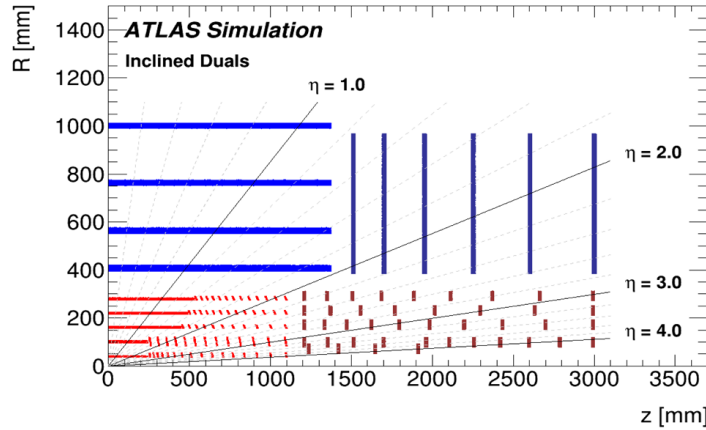


Figure 1. Schematics of a quarter of the Inner Tracker, showing its two sub-detectors: the ITk Pixel detector (red) and the ITk Strips detector (blue). The x-axis represents the beam line, and the point (0,0) represents the interaction point (midpoint) of the ATLAS experiment. Reproduced with permission from [1]. CC BY-NC-ND 4.0.

The ITk Pixel detector [2] consists of 5 cylindrical layers, arranged in an Inner System (IS), an Outer Barrel (OB), and in addition an Outer Endcap (EC). The IS is formed by the innermost 2 layers of the ITk. The OB and EC are formed by the last 3 layers of the ITk. The OB consists of flat staves and inclined rings, while the EC has vertical rings of modules.

The ITk Pixel detector covers an area of 13 m^2 and spans a high pseudorapidity of $|\eta| < 4$. The pixels used have a high resolution, with a pixel size of $25 \mu\text{m} \times 100 \mu\text{m}$ in some part of the innermost layer and a pixel size of $50 \mu\text{m} \times 50 \mu\text{m}$ for the rest of the ITk pixels.

OB hosts around 4772 QMs, each with 4 front-end (FE) chips. These QMs are mounted to LLS, of which there are 2 types, as shown in figure 2:

- A Longeron hosting 36 of the flat QMs.
- An Inclined Half Ring hosting between 16 and 28 of the inclined QMs.

The structure of the OB system of the ITk presents significant challenges in reading out a large number of QMs. In addition a database called ITk production database (ITk PD) is used to store the information about the LLS during the production stage. The main target is to have a valid data acquisition readout system capable of reading out all QMs, starting with one FE and then scaling up to read out all QMs.

In the following, the readout DAQ of the ITk is described in section 2. The results of the ITk DAQ software are presented in section 3, and finally, a conclusion about the project status and its outlook is presented.

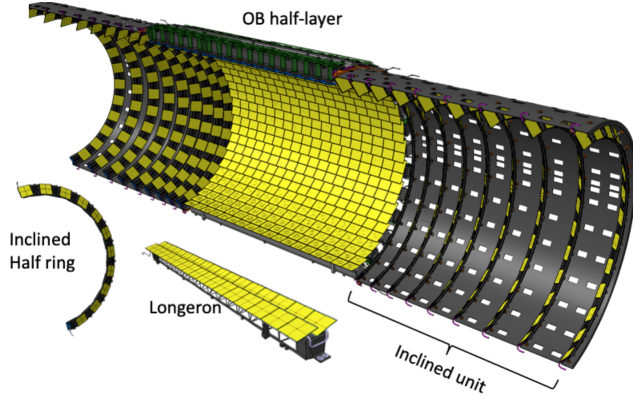


Figure 2. Schematics of OB loaded local supports. Reproduced with permission from [3]. CC BY-NC-ND 4.0.

2 ITk read-out DAQ

During the operation of the ITk, about 10,000 modules of FE chips have to run simultaneously. Commands have to be sent to about 5 billion pixels, and the ITk DAQ system has to be able to cope with the correspondingly huge amount of data received from these modules.

The ITk DAQ system is shown in figure 3. It consists of the modules of ITkPix FE chips (triplets or QMs). Each QM is connected to one optobox [4], which is a unit that hosts up to 8 optoboards. Each optoboard has 3 ASICs: GBCR, lpGBT, and VTRx+. The GBCR enhances the electrical signal integrity for commands before they are sent to QM and also for incoming data from the QMs. The lpGBT distributes the commands to the QMs and aggregates the data received from the QMs. The VTRx+ converts commands from optical to electrical and converts the received readout data from electrical to optical signal, the optobox is connected to the ITk DAQ PC via optical cables, with command transmission at 2.56 Gb/s and data reception at 10.24 Gb/s. The DAQ PC consists of ITk DAQ hardware represented by a FELIX Card, a PCIe FPGA board which interfaces between the optoboard and the ITk DAQ software. The DAQ PC also includes ITk DAQ Software, which is YARR (Yet another Rapid Readout) [5, 6]. YARR creates the commands and sends them to QMs through FELIX and also receives the event data in order to process and stores it in the database for later analysis.

The FELIX is operated via the Felix-Star application, which is the central process of the FELIX system. It is a multithreaded operation where many FELIX cards can be operated simultaneously. Felix-Star consists of 3 sub-applications: felix-tohost (from FELIX Card to DAQ software), which is responsible for dealing with the readout data coming from the QM; felix-tofx (from DAQ software to FELIX Card), which is responsible for timing, trigger, and control commands sent to the QM; and felix-register, which provides the ability to access the FELIX register for writing and reading purposes.

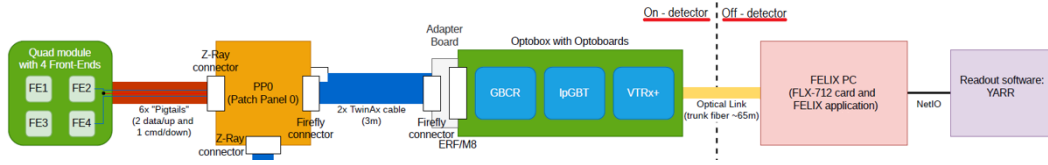


Figure 3. Schematics of ITk Data Acquisition system. Reproduced from [7]. CC BY 4.0.

In addition, a protocol named Netio-Next is used, which is a fast communication protocol for data messaging between Felix-Star and FELIX clients. The FELIX software provides the user with an application named Felix-Client Interface that interacts with Netio-Next and hides the complexity of Netio-Next.

YARR is a C++ software that includes two types of libraries: the Chip libraries and the Controller libraries. In our case the YARR software uses the libRD53b chip library, which is responsible for creating the trigger commands for the ITkPixV1¹ frontend and processing data coming from the frontend. It also uses the libFelixClient controller library allowing to interact with the Felix-Client Interface. LibFelixClient has a TX-Core for command transmission preparation and an RXCore for data preparation for further processing in YARR Chip libraries. The TXCore and RXCore were developed in a small lab setup to first read out a single ITkPixV1 frontend chip, then validated by scaling up to a larger lab setup with multiple ITkPixV1 frontend chips. Figure 4 shows the schematic of the YARR software.

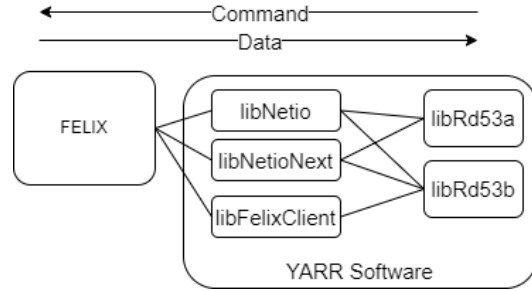


Figure 4. Schematics of the YARR software showing the used chip libraries and the FELIX related controllers.

3 Development and validation results

One of the requirements for the LLS production phase is that it undergoes quality control (QC) tests after mounting all the QMs to ensure that they are in good condition after loading them.

This requires to have an ITk DAQ system that is able to read out all the QMs in the LLS, as mentioned in section 1. The Felix-Star and Netio-Next are validated by doing a YARR digital scan to read out the prototype readout Chip RD53A single chip card. This was the starting point to move to the ITkPix case, during which some software blocs were adapted or newly introduced as libFelixClient, libRD53b, and later libITkPixV2. The chain validation was first achieved on a lab setup at the University of Goettingen, as shown in figure 5. The setup consists of a digital QM with 4 ITkPixV1 frontend chips and an optoboard connected to a FELIX card in a DAQ PC that hosts the YARR software.

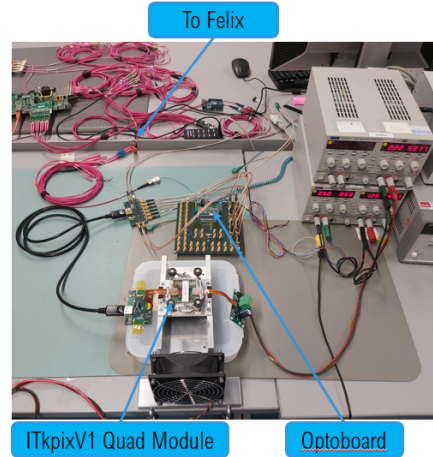


Figure 5. Lab setup at university of Goettingen to develop YARR to read out ITkPixV1 QM.

¹ITkPixV1 frontend chip is the pre-production readout chip of the ITk.

Debugging procedure was necessary to get rid of communication and processing errors to get scans with no failing pixels. This was done first for 1 FE at a time then for 2, 3 and 4 FEs simultaneously, as shown in figure 6.

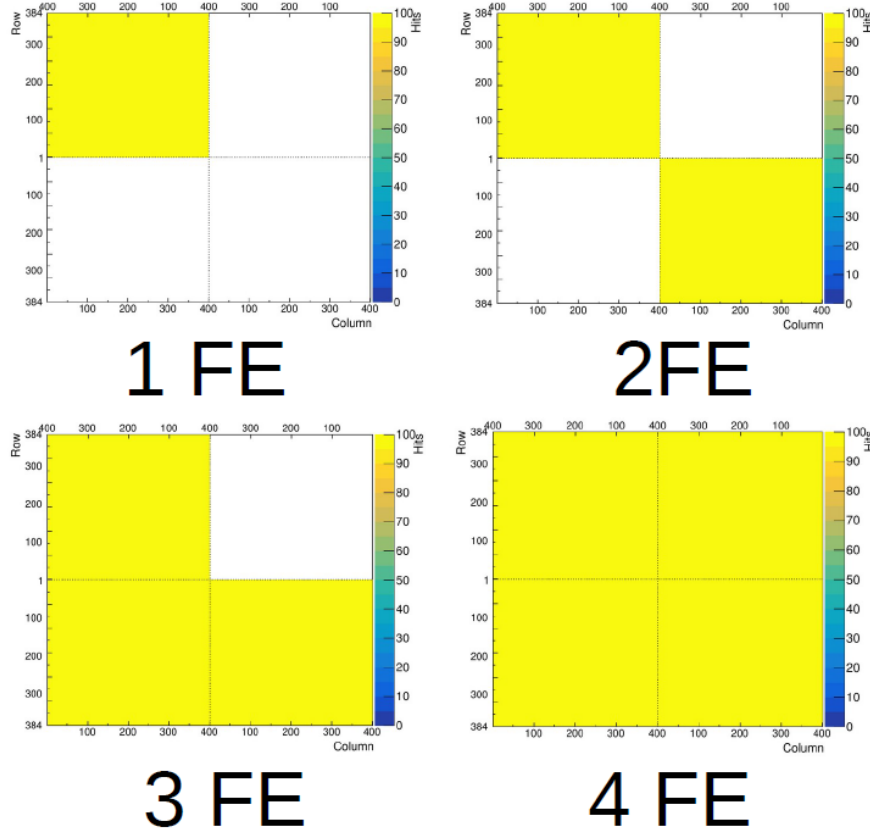


Figure 6. ITkPixV1 QM hit map examples obtained during validation phase.

To scale up the testing procedure, and due to the unavailability of additional QMs, another lab setup at the University of Bonn was used. This setup uses serial powering and targets the LLS DAQ test as shown in figure 7. Since more QMs were not yet available, the setup was restricted to only three ITkPixV1 QMs connected to optoboard via electrical connections which is connected via optical connection to FELIX server hosting a FELIX card with installed optoboard software, FELIX software, and YARR software.

The validation procedure started with the validation of two frontends simultaneously by performing a successful YARR digital scan. It was then scaled up by adding more frontends one by one and validating the YARR software each time a frontend is added. Finally, the validation of the ITk DAQ readout for 10 ITkPixV1 frontends with no data transmission errors was achieved as shown in figure 8.

During the scaling-up procedure, the time consumed by the YARR processes during the scan is recorded. The YARR processes include Configuration, Scan, Processing, and Analysis. Configuration is the first stage where YARR sends commands to configure the chips. Scan is when YARR sends the scan command. After the scan, YARR checks the incoming data in the Processing stage and then analyzes it in the Analysis process.



Figure 7. Lab setup at University of Bonn.

```
[12:59:48:854][ info ][ ScanConsole ][46207]: Processor done, waiting for histogrammer ...
[12:59:49:084][ info ][HistogramAlgorithm][46297]: Histogrammer done!
[12:59:49:092][ info ][ StdAnalysis ][46296]: [6][0x154c8] Total number of failing pixels: 0
[12:59:49:093][ info ][HistogramAlgorithm][46294]: Histogrammer done!
[12:59:49:102][ info ][ StdAnalysis ][46293]: [5][0x15478] Total number of failing pixels: 0
[12:59:49:105][ info ][ StdAnalysis ][46296]: [6][0x154c8][0] ToT Mean = 7 +- 0
[12:59:49:114][ info ][ StdAnalysis ][46293]: [5][0x15478][0] ToT Mean = 7 +- 0
[12:59:49:177][ info ][HistogramAlgorithm][46306]: Histogrammer done!
[12:59:49:182][ info ][HistogramAlgorithm][46279]: Histogrammer done!
[12:59:49:183][ info ][ StdAnalysis ][46305]: [9][0x14724] Total number of failing pixels: 0
[12:59:49:187][ info ][ StdAnalysis ][46278]: [0][0x1467c] Total number of failing pixels: 0
[12:59:49:195][ info ][HistogramAlgorithm][46291]: Histogrammer done!
[12:59:49:195][ info ][ StdAnalysis ][46305]: [9][0x14724][0] ToT Mean = 7 +- 0
[12:59:49:197][ info ][HistogramAlgorithm][46282]: Histogrammer done!
[12:59:49:198][ info ][ StdAnalysis ][46290]: [4][0x154b8] Total number of failing pixels: 0
[12:59:49:200][ info ][ StdAnalysis ][46278]: [0][0x1467c][0] ToT Mean = 7 +- 0
[12:59:49:201][ info ][ StdAnalysis ][46281]: [1][0x14647] Total number of failing pixels: 0
[12:59:49:203][ info ][HistogramAlgorithm][46303]: Histogrammer done!
[12:59:49:203][ info ][HistogramAlgorithm][46285]: Histogrammer done!
[12:59:49:205][ info ][HistogramAlgorithm][46300]: Histogrammer done!
[12:59:49:208][ info ][ StdAnalysis ][46302]: [8][0x14735] Total number of failing pixels: 2
[12:59:49:208][ info ][ StdAnalysis ][46284]: [2][0x14627] Total number of failing pixels: 0
[12:59:49:209][ info ][ StdAnalysis ][46299]: [7][0x154d8] Total number of failing pixels: 0
[12:59:49:211][ info ][ StdAnalysis ][46290]: [4][0x154b8][0] ToT Mean = 7 +- 0
[12:59:49:214][ info ][ StdAnalysis ][46281]: [1][0x14647][0] ToT Mean = 7 +- 0
[12:59:49:220][ info ][ StdAnalysis ][46302]: [8][0x14735][0] ToT Mean = 7.000078125 +- 0.021949139614520392
[12:59:49:221][ info ][ StdAnalysis ][46284]: [2][0x14627][0] ToT Mean = 7 +- 0
[12:59:49:222][ info ][ StdAnalysis ][46299]: [7][0x154d8][0] ToT Mean = 7 +- 0
[12:59:49:226][ info ][HistogramAlgorithm][46288]: Histogrammer done!
[12:59:49:226][ info ][ ScanConsole ][46207]: Processor done, waiting for analysis ...
[12:59:49:230][ info ][ StdAnalysis ][46287]: [3][0x14636] Total number of failing pixels: 0
[12:59:49:242][ info ][ StdAnalysis ][46287]: [3][0x14636][0] ToT Mean = 7 +- 0
[12:59:49:445][ info ][AnalysisAlgorithm][46287]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46299]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46302]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46293]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46284]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46278]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46281]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46290]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46296]: Analysis done!
[12:59:49:626][ info ][AnalysisAlgorithm][46305]: Analysis done!
[12:59:49:627][ info ][ ScanConsole ][46207]: All done!
[12:59:49:627][ info ][ ScanConsole ][46207]: #####
[12:59:49:627][ info ][ ScanConsole ][46207]: ## Timing ##
[12:59:49:627][ info ][ ScanConsole ][46207]: #####
[12:59:49:627][ info ][ ScanConsole ][46207]: -> Configuration: 18417 ms
[12:59:49:627][ info ][ ScanConsole ][46207]: -> Scan: 12727 ms
[12:59:49:627][ info ][ ScanConsole ][46207]: -> Processing: 1 ms
[12:59:49:627][ info ][ ScanConsole ][46207]: -> Analysis: 772 ms
```

Figure 8. The output of the YARR digital scan shows a successful scan of 10 used FEs simultaneously, with no data processing errors.

The time consumed by each YARR operation and the total time consumed by all the processes are presented in figure 9. As the number of scanned chips increases, the time consumed by YARR configuration increases since YARR configures the read-out chips one by one. The scan time consumed by YARR shows a slight increase as the number of chips increases. This is due to buffering and command distribution across all FEs, as the scan commands are sent to all FEs simultaneously, not FE by FE. The next step will be to extend these tests to a full featured LLS readout, once more QMs are available.

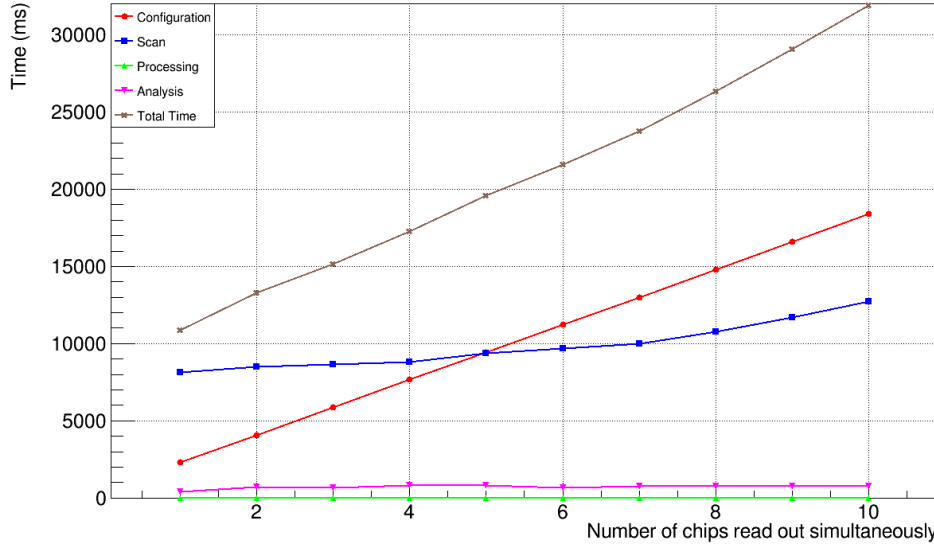


Figure 9. Time consumed by YARR processes as function of the number of FEs under scan.

In addition some instabilities have been observed when sending commands during calibrations with more than 2–3 digital quad modules, where this issue seems related to an older, unresolved problem with the netio buffering and the non-blocking method used to flush the netio buffers while sending data. This issue was solved by introducing small delays.

4 Conclusion

The preparation of the ITk DAQ readout system, for the next ATLAS upgrade is very challenging in terms of higher performance and increasing size. The ITk-Pixel readout part should be able to handle simultaneously about 10k triplet and quad modules. The work described here shows the first steps of evaluating the ITk-Pixel DAQ system, starting by validating the readout chain on a lab setup, then extending the setup targeting a full-featured LLS, with tens of QMs connected as in the final ITk layout. Scans operating simultaneously up to 10 FEs, using 3 digital ITkPixV1 QMs were successfully carried out. The overall execution timing analysis pointed out some indications for the scalability of the ITk-Pixel DAQ system. In fact, the found linear increase of the readout time should clearly be optimized to allow scaling up the DAQ system, for the LLS readout and beyond. Different architectural options for the DAQ software are being investigated. Further development efforts will continue once more QMs, currently being assembled, are available for the LLS sub-system testing.

References

- [1] V. Izzo, *ATLAS upgrades*, *PoS LHCP2020* (2021) 094.
- [2] ATLAS collaboration, *Technical Design Report for the ATLAS Inner Tracker Pixel Detector*, *CERN-LHCC-2017-021* (2017).
- [3] F. Munoz Sanchez, *Carbon Based Local Supports for the ATLAS ITk Pixel Detector*, *PoS Pixel2022* (2023) 077.
- [4] S. Möbius, *The Optosystem: validation and testing of the high-speed electro-optical conversion system for the readout of the ATLAS ITk Pixel upgrade*, *2024 JINST* **19** C04015 [[arXiv:2310.19637](#)].
- [5] T. Heim, *YARR — A PCIe based readout concept for current and future ATLAS Pixel modules*, *J. Phys. Conf. Ser.* **898** (2017) 032053.
- [6] N.L. Whallon et al., *Upgrade of the YARR DAQ system for the ATLAS Phase-II pixel detector readout chip*, *PoS TWEPP-17* (2018) 076.
- [7] J. Hoya, *FELIX: First operational experience with the new ATLAS readout system and perspectives for HL-LHC*, *EPJ Web Conf.* **295** (2024) 02012.