EPJ Web of Conferences **214**, 01032 (2019) https://doi.org/10.1051/epjconf/201921401032
CHEP 2018

The Continuously Running iFDAQ of the COMPASS Experiment

Ondřej Šubrt[2,4,*], *Martin* Bodlák[3], *Vladimir* Frolov[5], *Stefan* Huber[1], *Matouš* Jandek[4], *Vladimír* Jarý[4], *Igor* Konorov[1], *Antonín* Květoň[4], *Dmytro* Levit[1], *Josef* Nový[4], *Dominik* Steffen[1], *Jan* Tomsa[4], and *Miroslav* Virius[4]

[1]Technical University of Munich, Germany
[2]CERN, Geneva, Switzerland
[3]Charles University, Prague, Czech Republic
[4]Czech Technical University in Prague, Czech Republic
[5]Joint Institute for Nuclear Research, Dubna, Russia

Abstract. Recently, a stability of Data Acquisition System (DAQ) has become a vital precondition for a successful data taking in high energy physics experiments. The intelligent, FPGA-based Data Acquisition System (iFDAQ) of the COMPASS experiment at CERN is designed to be able to readout data at the maximum rate of the experiment and runs in a mode without any stops. DAQ systems fulfilling such requirements reach the efficiency up to 99%. The newly introduced continuously running mode enables to collect data without a necessity of any other user intervention. Such mode affects all processes of the iFDAQ with high emphasis on timing and precise synchronization. However, every undesirable interruption of data taking can potentially result in a possible loss of physics data. Running 24/7 puts stress on reliability and robustness of the system. Therefore, the improvement of the iFDAQ stability had to come first. The continuously running mode and the improved iFDAQ stability helped to collect more physics data in the Run 2017. In the paper, we present the continuously running mode in more detail and discuss the overall iFDAQ stability.

1 Introduction

COMPASS (Common Muon and Proton Apparatus for Structure and Spectroscopy) [1, 2] is a fixed-target experiment at the Super Proton Synchrotron (SPS) accelerator at CERN near Geneva, Switzerland. It was proposed and approved in 1996, commissioned in 2001, and started taking physics data in 2002. The main objective of the experiment is the study of hadron structure and spectroscopy using high intensity muon and hadron beams.

Due to increased requirements in 2014 in terms of data acquisition scalability, reliability and data throughput, the COMPASS experiment developed the intelligent, FPGA-based Data Acquisition System (iFDAQ) using a novel approach to the event building network. In contrast to traditional event builders which are based on distributed online computers interconnected via an Ethernet Gigabit network, the event building task is completely executed in hardware.

*the corresponding author, e-mail: ondrej.subrt@cern.ch

The paper is organized as follows. In Section 2, an overview of the iFDAQ is presented from a hardware and software point of view. The iFDAQ stability over the last few years is shown in Section 3.

The improved stability opened up the possibility to keep the system continuously running without interruption of data flow. This feature reduces time consuming synchronization phases at each start and stop of a run. Section 4 describes the continuous operation mode of the iFDAQ, the necessity of proper timing and synchronization, and the logic in the affected processes.

2 iFDAQ Architecture

The current COMPASS data acquisition system (iFDAQ) [3–6] has been commissioned in 2014, replacing the previous obsolete DAQ, which had been in use since the conception of the experiment. Its objective is to read out raw data of physics events from detectors and store them on hard drives. These data are subsequently sent to CASTOR (CERN Advanced STORage manager) [7].

In the COMPASS spectrometer, there are in total more than 300,000 detector channels of which the iFDAQ collects hit, time, and amplitude information. The data streams emerging from the detector front-ends are multiplexed, using three layers of acquisition modules:

1. HGeSiCA [8], CATCH [9] and Gandalf [10] modules (data concentrators – at this stage latest, zero suppression and feature extraction algorithms are applied),
2. Slink multiplexers, TIGER VXS modules, first level of DHCmx (optional intermediate multiplexing stage to make more efficient use of link bandwidth),
3. second level of DHCmx modules [3, 4] (currently six cards, up to eight in full setup).

After the multiplexing process is finished, the data are sent to the FPGA switch (DHCsw), which receives all subevent information and assembles a full event from these fragments. The recombination and merging of event information during the event building process follows a strict pattern whose consistency is verified on each stage.

Finally, the data are read out by four readout engine computers (up to eight in full setup) and stored on hard drives. Data transfer from the S-Link [11] receiver which collects the fully assembled events to the RAM of the computer is handled via a dedicated PCIe card with an FPGA. In conclusion, Figure 1 shows the topology of the iFDAQ setup in 2018.

The software side of the iFDAQ [6, 12] is used for control and monitoring of the final three layers of the iFDAQ hardware (FPGA multiplexers, FPGA switch, and readout engines) and for read out of physics events from readout engines. It consists of several processes:

- **Master process** — A process which runs on a dedicated computer and acts as the communication mediator between the iFDAQ processes. It incorporates supervision of states of the iFDAQ control system and the error handling.
- **Slave-control** — A process which runs on the readout engine computers and handles monitoring and configuration of the FPGA cards (including the PCIe cards in the readout computers), using direct communication through IPbus [13].
- **Slave-readout** -– A process which runs on the readout engine computers and handles decoding and verification of physics data.
- **Runcontrol GUI** — A graphical user interface which provides control of the iFDAQ.
- **MessageLogger** -– A process which collects messages from the slave and master processes and stores them in a database.

- **MessageBrowser** –– A graphical user interface which allows the user to view messages from the slave and master processes in real time or retroactively browse the messages stored in a database.

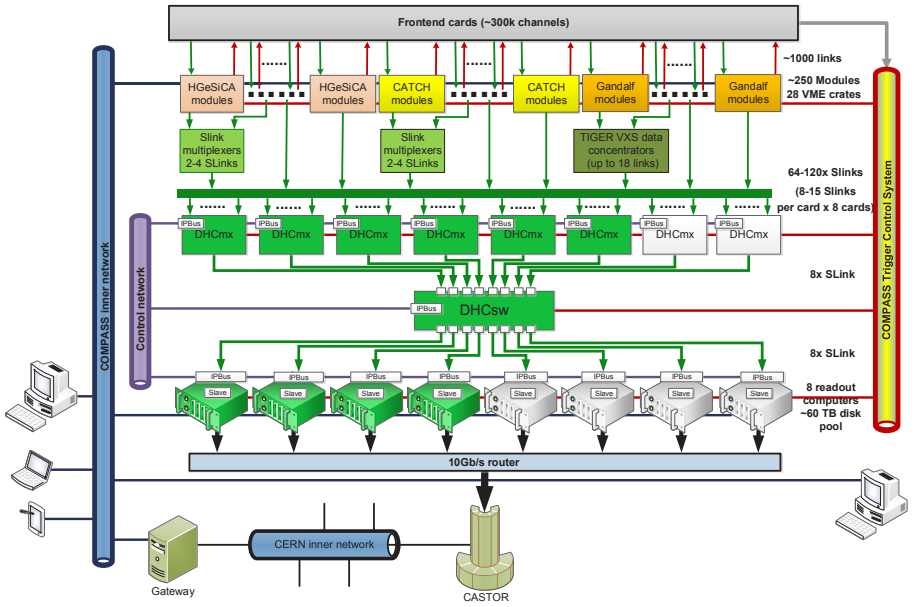

Figure 1. Topology of the iFDAQ setup in 2018.

3 iFDAQ Stability

Since beam time is highly valuable and it is usually provided 24/7 for most of the calendar year, high reliability of the DAQ system is of major importance to high-energy physics experiments. This section discusses the iFDAQ stability over the last few years.

There are three main sources of instabilities leading to time periods when no physics data can be taken. The overall time period when no physics data can be taken is called the iFDAQ downtime. On the other hand, the iFDAQ uptime denotes the overall time period when the iFDAQ is stable and ready for a proper data taking.

The first source of instability is a memory access error (PCI/DMA) caused by scrambled data being transferred to the RAM of the readout engines. It is the most time-consuming failure, since it requires to reboot all readout engine computers and the recovery procedure takes approximately 10 minutes on average. The second one is an unrecoverable loss of synchronization in the hardware event builder leading to a safe stop of a run. The safe stop of a run might be considered as one of the intelligent elements of the iFDAQ since a safe stop prevents more serious problems which would lead to the higher downtime. The contribution of these two problems to the overall system downtime decreased during the course of one year due to better commissioning and calibration of detectors and consequently higher data quality.

The third source of the downtime is based on unknown software crashes being not fully understood. The DIALOG library [14] and, especially, the DAQ Debugger [15] have been

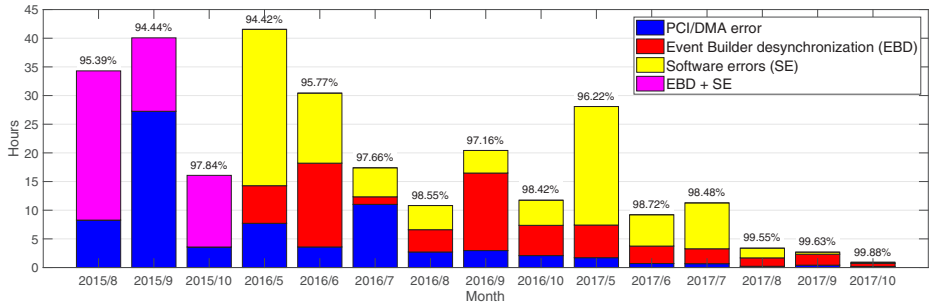

Figure 2. Absolute downtime of the iFDAQ per month and corresponding relative uptime.

developed in order to eliminate software errors in the iFDAQ and thus, to improve the iFDAQ software stability.

Figure 2 shows the iFDAQ stability in the last few years. The values are calculated by parsing messages that have been stored by the MessageLogger in a database.

The plot shows data from August 2015 to October 2017. Each bar shows the number of hours per month representing the total downtime for this period and, moreover, it is split into three components – PCI/DMA errors (blue), event builder desynchronization (red) and software errors (yellow). In addition, the relative uptime per month is stated in the top of each bar.

In August, September and October 2015, messages that were stored do not include information about the kind of error. Thus, it can not be distinguished between event builder desynchronization and software errors (purple).

From the plot, it can be seen that in 2015, most of the 40 hours downtime per month were caused by software errors.

The DIALOG library integration to the iFDAQ is the first big milestone in the iFDAQ stability improvement. The DIALOG library was introduced to the iFDAQ in June and July 2016. The bar plot in Figure 2 shows a significant drop in the software errors component of each bar from that time.

The second milestone is located in June 2017 when the DAQ Debugger has been deployed and helped to identify all remaining software errors in the following months.

Finally, the iFDAQ has reached the state when no software errors are present anymore. The last software error was observed at the end of September 2017.

Presently, the iFDAQ reaches a system availability of 99.88% and only PCI/DMA errors and event builder desynchronization appear from time to time. In absolute numbers, the downtime of the iFDAQ has decreased from around 40 hours per month in 2015 to only 1 hour per month in October 2017.

The increase of downtime in May of each year can be explained by the commissioning phase that takes place in the beginning of each year and in which all detectors, the frontend electronics and the iFDAQ do not operate in stable conditions.

4 Continuously Running iFDAQ

Another significant contribution to the loss of beam time originates from the time that is needed to initiate a synchronized data flow through the event builder. Establishing synchronous processing of data by all involved hardware nodes is achieved by distributing trigger

and spill cycle information to all nodes via the Trigger Control System (TCS) [16] and applying reset commands and timeouts. Following section describes how the SPS spill cycle information is used initiate and maintain proper synchronization.

4.1 Proper Timing and Synchronization

Before the continuously running mode is presented in a deeper way, it is worth of mentioning how to deal with accurate timing and synchronization being absolutely essential for such mode. To synchronize the iFDAQ processes correctly, the iFDAQ needs to take advantage of some proper timing mechanism. The SPS super cycle is a good candidate offering and ensuring the proper timing.

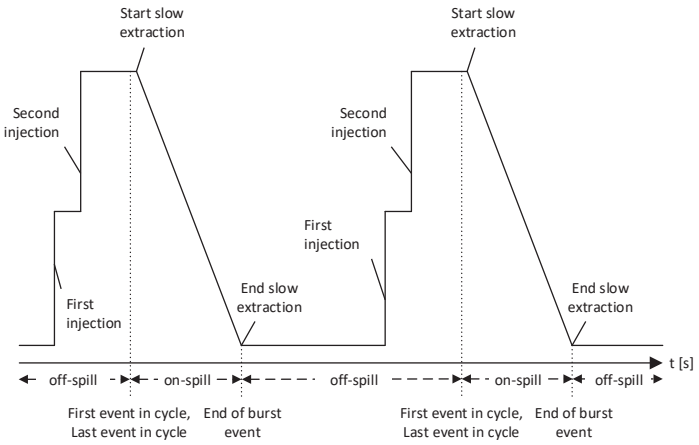

Figure 3. Particle intensity in the SPS for an exemplary SPS cycle.

The beam for COMPASS is provided by the SPS. As a circular particle accelerator, the SPS provides beam to experiments in bursts, called spills. Before being able to deliver a particle spill to COMPASS, the SPS has to be filled with proton bunches, the bunches have to be accelerated to the desired energy, and the particle load inside the SPS has to be debunched. Only when the particles circulate homogeneously distributed in the SPS, a spill of stable particle intensity can be delivered. After one filling is completely extracted from the machine, the cycle starts over. Figure 3 shows the proton intensity in the SPS during two cycles which together form an exemplary SPS super-cycle.

For COMPASS, the load in the SPS is slowly extracted over the course of 4.8 seconds. After the spill, there are at least 5 seconds without spill: filling the SPS with two injections from the PS takes approximately 2 seconds, and ramping up the energy and debunching (flat top in Figure 3) takes another 3 seconds. However, the off-spill period can take significantly longer -– depending on the SPS super-cycle -– when the next charge of the SPS is not conducted to COMPASS but to other recipients (e.g. the LHC).

For a successful synchronization of the iFDAQ to the SPS cycle, dedicated types of triggers have been established. In the iFDAQ, following types of events are distinguished:

- **Start of run event** is first event in a run after it starts for the first time.
- **End of run event** is last event in a run before it terminates.

- **First event in cycle/Start of burst event** is first event in burst (start of beam extraction) and hence in the on-spill period. At the same time, it marks the beginning of a cycle.
- **Last event in cycle** is last event in the off-spill period, it is followed by a reset signal for buffer memory in the frontend electronics ensuring proper synchronization for the next spill.
- **End of burst event** is last event in burst (end of beam extraction and on-spill period).
- **Physics event** contains real physics data and is collected during the on-spill period.
- **Calibration event** can be collected in both the on-spill and off-spill period. It contains calibration data for calorimeters.

Timing and synchronization are based on the artificial events and their correct order. To initiate data flow through the event builder and establish correct timing, three cycles are required – and thus lost – before first physics triggers can be sent to the frontend electronics.

4.2 The Continuously Running Mode

To safe time-consuming stop and restart of the data flow between two runs, the continuously running mode was introduced. It ensures a smooth transition between two consecutive runs without intervention of the shift crew and without stop of data flow through the event builder. The transition between two runs requires several important things to do. Data files from the previous run must safely be closed and new files for a new run have to be opened in the same time on all machines. The run number has to be increased in the correct time so that data belonging to the previous run and data belonging to the new one are properly distinguished. The records in the electronic logbook concerning the previous run have to be filled in and new records concerning the following run have to be created. All these functionalities require proper synchronization.

Moreover, the idea of maintaining data flow in the event builder has been extended to periods when no data taking is requested. To do so, a new state, the so called Dry Run was introduced. It maintains the data flow through the whole acquisition chain and provides a monitoring data stream to the online monitoring tools but the acquired events are not written to hard drives. Thus, it serves as verification, monitoring, and diagnostic stage, even in periods when the experiment is not ready for data taking. The consecutive step relevant to real data taking is called Run and its start is possible on the next delivered spill since synchronization is already established. Using the Dry Run state and a smooth transition between runs, the data flow in the iFDAQ is only stopped in case of serious errors (see Section 3) or in case of interventions on detectors that require a stop of trigger distribution to the frontends.

At this point, basic functionality requirements are specified. Using the continuously running mode, the Dry Run state is designed in order to meet the following requirements:

- It checks the incoming data stream from the detectors on consistency.
- The transition from Dry Run to Run and vice versa must be fast and smooth.
- It uses the artificial run number in a range from 999,990 to 999,999.
- Starting procedure – The Master process sends a command to initiate a run and waits for the slaves to enter the Dry Run state.
- Terminating procedure – The Master process sends a command to slaves to stop a run.
- It provides physics data to online monitoring tools.
- No data are stored in files.
- No records in the electronic logbook concerning current run are created.

- The spill number is iterated. After the last spill, the spill counter is reset to 1 in the following spill and the artificial run number is changed.

Similar characteristics can be summarized also for the Run state. Using the continuously running mode, the Run state must meet the following requirements:

- It is started from the active Dry Run state.
- It always uses the last real run number incremented by one.
- Starting procedure – It sends a command to reset the spill counter to 1 and set the real run number after the end of current spill.
- Terminating procedure – A run stops in the Run state and moves to the Dry Run state, where a new run starts.
- It provides physics data to online monitoring tools.
- The data are stored in files.
- Records in the electronic logbook concerning current run are created.
- The continuously running mode can be switch on/off in Runcontrol GUI.
- If a run is stopped manually or automatically (by reaching the maximum number of spills set in Runcontrol GUI) in the Run state, the iFDAQ moves to the Dry Run state.

To sum it up, the iFDAQ using the continuously running mode is a self-running data-taking system where decisions related to the continuously running mode are taken based on delivered events. Timing critical actions that have to be taken in transitions from one run to another or from the Dry Run state to the Run state and vice versa are executed on reception of artificial events and depend on their type of trigger:

- **First event in cycle in first spill of a run** opens/closes files and create the electronic logbook records, if necessary.
- **End of burst event in last spill of a run** resets the spill counter and set the next run number.
- **Last event in cycle in last spill of a run** fills in the electronic logbook records and moves to Dry Run/Run, if necessary.

4.3 Contribution of the Continuously Running Mode

Before the incorporation of the continuously running mode, the procedure for starting a new run after another run was successfully finished was always connected to a loss of beam time. Firstly, there is loss due to the already mentioned necessary synchronization phases. The start of run procedure requires three spills for the synchronization of the TCS with the SPS cycle. The terminating procedure takes one spill to end up data taking. Secondly, there is beam time loss due to necessary human intervention. Before the incorporation of the continuously running mode, a run had to be started manually. An inattentive shift crew could hence add a significant part to the beam time loss by not starting the next run right after the previous run is stopped. Eliminating both factors, the continuously running mode contributes to the efficiency of data taking.

In Table 1, the contribution of the continuously running mode is demonstrated. In order to eliminate problems unrelated to the iFDAQ, one day with smooth data taking is selected for the Run 2016 (without the continuously running mode) and the Run 2017 (with the continuously running mode). The loss unrelated to the iFDAQ on 2017-07-30 refers to the beam polarity change procedure. In case of smooth data taking, the percentage of collected spills reaches almost 100% in case of the continuously running mode. Without the continuously running mode, the loss of spills is around 3.5%.

Table 1. The contribution of the continuously running mode.

	2016-10-10		**2017-07-30**	
	# spills	**%**	**# spills**	**%**
Delivered from SPS	4,648	—	4,569	—
Recorded	4,487	96.54	4,488	98.23
Loss related to TCS synchronization	112	2.41	4	0.09
Loss related to inattentive shift crew	49	1.05	0	0
Loss unrelated to the iFDAQ	0	0	77	1.69

5 Conclusion

The iFDAQ was successfully deployed and commissioned in 2014. The improved iFDAQ stability over the last few years gave an opportunity to introduce the continuously running mode. Without any stops, the iFDAQ runs 24/7 regardless of nights, weekends or holidays. It helped to collect more physics data in the Run 2017 and 2018.

6 Acknowledgment

This research has been supported by OP VVV, Research Center for Informatics, CZ.02.1.01/0.0/0.0/16_019/0000765 and by the Maier-Leibnitz-Labor, Garching and the DFG cluster of excellence Origin and Structure of the Universe.

References

[1] P. Abbon et al., Nucl. Instrum. Methods Phys. Res. **A577**, 455 (2007)
[2] V.Y. Alexakhin et al., *COMPASS-II Proposal*, The COMPASS Collaboration (2010)
[3] D. Steffen et al., Proceedings of Science **TWEPP-17**, 127 (2018)
[4] Y. Bai et al., Journal of Instrumentation **11**, C02025 (2016)
[5] M. Bodlak et al., Journal of Physics: Conference Series **513**, 012029 (2014)
[6] M. Bodlak et al., Journal of Instrumentation **8**, C02009 (2013)
[7] *CASTOR — CERN Advanced Storage manager*, http://castor.web.cern.ch, accessed 2018-09-09
[8] *iMUX/HGESICA module*, https://twiki.cern.ch/twiki/pub/Compass/Detectors/FrontEndElectronics/imux_manual.pdf, accessed 2019-03-16
[9] *Electronic developments for COMPASS at Freiburg*, https://twiki.cern.ch/twiki/pub/Compass/Detectors/FrontEndElectronics/catch-userguide.ps, accessed 2019-03-16
[10] *The GANDALF Module*, https://gandalf-framework.web.cern.ch/, accessed 2019-03-16
[11] *S-Link — High Speed Interconnect*, http://hsi.web.cern.ch/HSI/s-link/, accessed 2018-09-09
[12] M. Bodlak et al., Nuclear and Particle Physics Proceedings **273**, 976 (2016)
[13] C.G. Larrea, K. Harder, D. Newbold, D. Sankey, A. Rose, A. Thea, T. Williams, Journal of Instrumentation **10**, C02019 (2015)
[14] M. Bodlak et al., International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering **11**, 372 (2017)
[15] M. Bodlak et al., International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering **11**, 448 (2017)
[16] B. Grube, Ph.D. thesis, Technical University Munich (2006)