ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

RENOVATION OF THE BEAM-BASED FEEDBACK CONTROLLER IN THE LHC

L. Grech*, G. Valentino, University of Malta, MSD 2080 Msida, Malta
D. Alves, A.Calia, M. Hostettler, J. Wenninger, S. Jackson, CERN, 1211 Geneva 23, Switzerland

Abstract

This work presents an extensive overview of the design choices and implementation of the Beam-Based Feedback System (BBFS) used in operation until the LHC Run 2. The main limitations of the BBFS are listed and a new design called BFCLHC, which uses the CERN Front-End Software Architecture (FESA), framework is proposed. The main implementation details and new features which improve upon the usability of the new design are then emphasised. Finally, a hardware agnostic testing framework developed by the LHC operations section is introduced.

INTRODUCTION

The Large Hadron Collider (LHC) was designed to handle particle momenta between one and two orders of magnitude higher than previous accelerators [1]. The highly energetic beams inside the vacuum chambers are stripped off of halo particles by a beam cleaning and machine protection system, also known as the collimation system. The collimation system required the machine tolerances to be tightened to work effectively. As a result, the LHC was the first proton accelerator to require automatic feedback control systems on various beam and machine parameters [2].

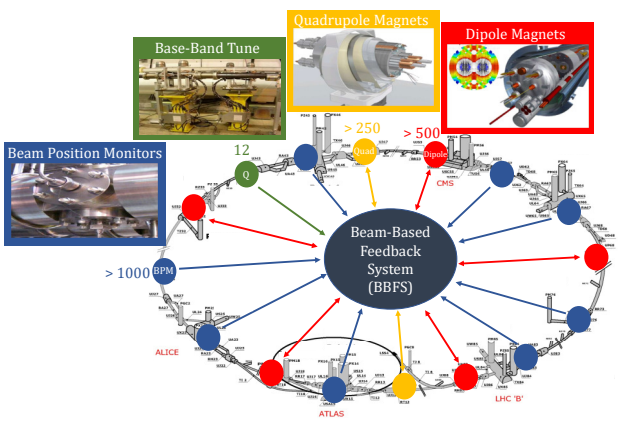

Figure 1: Schematic view of the BBFS in the LHC complex.

In this work, Beam-Based Feedback System (BBFS) denotes the program which was originally responsible for implementing the feedback control of the Radio Frequency (RF) systems, orbit and tune [3]. Figure 1 shows the BBFS within the LHC complex. The systematic beam energy offset from the ideal trajectory can be inferred from the orbit and in turn, the orbit is obtained from the measurements of 1088 Beam Position Monitors (BPMs) placed at different locations in the LHC [2, 4]. The Base-Band Tune (BBQ) systems are responsible for estimating the horizontal and vertical tunes of both beams in the LHC [5]. It is well known that the tune estimates from the BBQ were unstable due to 50 Hz noise harmonics present in the BBQ spectra [6, 7]. This instability also caused the Tune Feedback (QFB) to frequently switch off. As a consequence, the QFB was used intermittently and only when necessary, e.g. at the start of the ramp.

In its original design, the BBFS was foreseen to automatically control the coupling and chromaticity as well. Both these quantities are derived estimates from the BBQ system measurements. Considering that the tune estimates were often unstable, the control of coupling and chromaticity was deemed impractical to be used in operation, despite being implemented in the code.

The BBFS was made up of two components, which were historically named the Orbit Feedback Controller (OFC) and the Orbit Feedback Service Unit (OFSU). The OFC comprised the main program written in C++ and was primarily responsible for communicating with real-time Front-End Computers (FECs) to obtain beam measurements and applying magnetic corrections. The OFSU was implemented using the Front-End Software Architecture (FESA) framework used at CERN [8].

This work will provide a summary of the design and the limitations of the BBFS which was used in operation until the end of Run 2 in 2018. The BBFS underwent renovation during the LHC Long Shutdown 2 (LS2) and the upgraded version is called the Beam Feedback Controller LHC (BFCLHC). The BFCLHC is a FESA-based application which incorporates all the useful functionality of the BBFS, along with new features requested by the LHC operators. During LS2, our colleagues from the LHC operations section also developed a testing framework for the BFCLHC which for the first time allows closed loop tests to be performed on the feedbacks offline.

DESIGN UNTIL LHC RUN 2

Figure 2 illustrates a more detailed view of the BBFS architecture which is comprised of the OFSU and the OFC. The OFSU is connected to the OFC via a private Ethernet connection where Transmission Control Protocol (TCP) is used for lossless communication of the OFC settings and User Datagram Protocol (UDP) is used to stream real-time acquisition BPM, BBQ and magnet data from the OFC to the OFSU. Measurement data coming from the BPM and the BBQ systems is received by the OFC in the form of User

* leander.grech.14@um.edu.mt

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

Datagram Protocol (UDP) packets. The BPM packets arrive from 67 FECs, which can interface up to 18 BPMs each. In addition, longitudinally consecutive BPMs are interleaved amongst at least two, geographically nearby FECs, so that if a FEC fails, a contiguous section worth of measurements is not entirely lost. Similarly, the calculated current corrections for the magnets are sent to the Power Converter (PC) gateways via UDP packets. These gateways are connected to the Function Generator/Controllers (FGCs) which control the current flowing in the magnets [9].

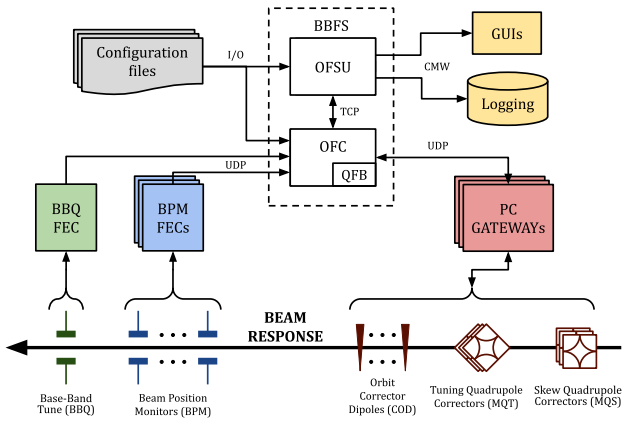

Figure 2: Schematic view of the data paths and a more detailed view of the BBFS architecture.

The principle design objective of the BBFS was to obtain a deterministic behaviour of the feedbacks on the hardware available at the time. The modular nature of the BBFS allowed the OFC to collect information about the tune and orbit in real-time, and calculate the required magnet corrections at a rate of 25 Hz. By keeping the program structure of the OFC simple, the designers could ensure a deterministic execution [2].

The OFC was written in object-oriented C++ and relied heavily on ROOT libraries. ROOT is written and maintained by CERN and was originally developed for efficient calculations on very large data accumulated by high-energy physics experiments [10]. The programming paradigm of ROOT is object-oriented where all objects derive from what is called a TObject. All TObjects can use the various functionalities implemented in ROOT, such as I/O handling, memory management and error handling. The OFC consists of several TObject-derived classes, which implement all the required functionality. The core of the OFC was a loop-based program, where every iteration was one simultaneous pass through the control loops of the OFC and the QFB. The latest version of the OFC used ROOT Release 5.34/20 [11].

Figure 2 also shows that the OFSU served as an interface for the OFC and was used by the LHC operators to monitor and control the OFC [3]. The OFSU was also responsible for loading some of the configuration files, which contained the settings and beam optics required by the OFC. This information was communicated to the OFC upon initialisation as well as by request of the operators. The OFSU also served as a hub for the data collected by the OFC and as a result a myriad of Graphical User Interfaces (GUIs) used by the LHC operators depended on the data relayed by the OFSU. One example is the relay of BPM orbit data, which: (a) underwent serialisation through ROOT TInterlink objects; (b) sent via UDP to the OFSU; (c) and finally set to the appropriate OFSU FESA property for orbit data. The OFSU was responsible for logging the data collected by the OFC along with the settings of the BBFS itself. The OFSU was developed using the FESA real-time framework and the sheer number of tasks described above made the OFSU the largest FESA-based application at CERN until LHC Run 2 [8, 12].

In its initial implementation the BBFS comprised a series of nested feedback loops, each controlling a specific LHC parameter. First and foremost was the orbit feedback loop, which calculated the change in Closed Orbit Dipole (COD) deflections necessary to steer the beam towards the reference orbit. Second was the RF frequency feedback loop, which calculated the change in frequency required in the RF cavities to counteract any systematic momentum offset introduced by the CODs during orbit correction. The other three nested feedback loops controlled the tune, coupling and chromaticity simultaneously. It is important to note that in the latest implementation of the OFC, only the QFB remained. Due to the instabilities observed in the tune estimates, subsequently derived measurements, namely coupling and chromaticity, were not reliable enough to allow for feedback loops of their own. When adjustments on the coupling and chromaticity were required, operator-calculated corrections had to be sent manually to the skew quadrupoles and sextupoles [13].

The impact of a change of currents in the CODs on the beam orbit is estimated using a Response Matrix (RM). The main principle behind the orbit correction performed by the OFC is to: (a) measure the average beam positions using the BPMs; (b) calculate its difference with respect to the reference orbit; (c) use a Proportional-Integral (PI) controller as a feedback loop; (d) calculate the change in current needed in each COD to correct the beam position; (e) scale down all the currents by the same factor to accommodate the slowest acting magnet and finally; (f) send the current corrections to the CODs. A similar procedure is used by the QFB to correct the tunes in both planes for both beams using quadrupole correctors.

Figure 3 shows the flowchart of the OFC main program. `GLOBAL_RUN` was the variable which controlled the control loop. The `INIT` block is expanded in Fig. 4 where it can be seen that several initialisation files were required to set up the OFC. The configuration file set up the thread parameters such as their priority and affinity and also contained two important parameters related to testing the OFC; `SIM_MODE` and `BASE_PORT`. `SIM_MODE` were used extensively throughout all the OFC code to change its behaviour during testing. Functionalities such as sending the UDP packets to the Power Converters (PCs) and RF cavities were suppressed when `SIM_MODE` was set in order to confine the output of the OFC to the testing framework. `BASE_PORT` served as an offset port number for all the network sockets used by the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

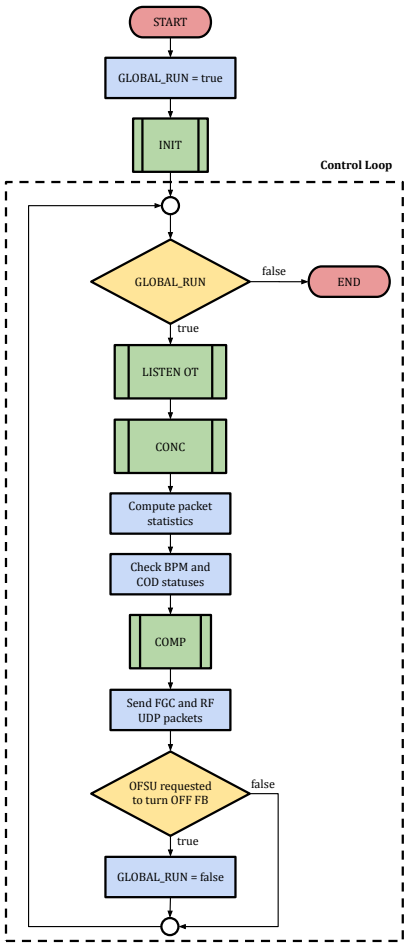

Figure 3: Flowchart of the main program of the OFC.

OFC. It was only changed during testing, when data was supplied by the testing framework. The configuration file contained other parameters, however, their use had become deprecated throughout the OFC lifetime.

The OFC also required a ROOT file upon initialisation which contained default optics stored in a ROOT object. This file existed since the first version of the OFC and due to its age it proved difficult to verify its contents. Its removal was also deemed dangerous due to its ubiquitous use throughout the initialisation code and therefore it was left up to the next major renovation to be removed. Several other text files were also required upon initialisation which provided: a) the hostnames of all the FECs that the OFC communicated with; b) the parameters of the magnets, e.g. maximum current rate; c) network routing information for all the BPMs, CODs, and quadrupoles required to create the network packets.

The OFSU was the interface that the operators used during real operation in order to monitor the status of the OFC, and changing parameters needed for operation. Two of the main problems with the OFSU were that it was bloated with unused functionality as well as complicated and less intuitive procedures which were required to change certain settings, e.g. fetching optics. The early design of the BBFS allowed access to more dedicated settings in the OFC, however, operator experience throughout the years has dictated which functionality to keep and improve, and which is not relevant for operations. On a final note, to avoid changing the major release of the BBFS during Run 2, any code renovations were delayed until LS2.

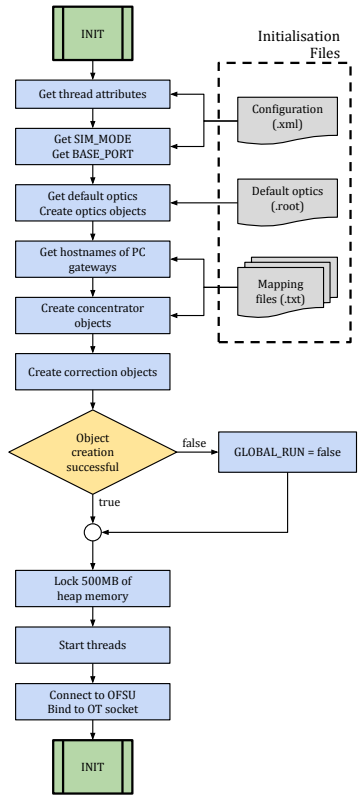

Figure 4: Flowchart of the `INIT` block in Fig. 3.

CODE RENOVATION

During LS2 it was decided to condense all the code in the BBFS, into one FESA-based application called BFCLHC. The hardware requirements were no longer an issue following a hardware upgrade of the OFC and OFSU machines. The hardware upgrade was from a 24-core, 32 GB RAM, 15 MB cache machine to a 64-core, 200 GB RAM and 22MB cache machine [14]. This upgrade meant that the conservative measures taken in the initial design of the OFC could be relaxed. During LHC operation, the OFC never crashed due to insufficient hardware requirements, therefore the hardware upgrade meant that it was safe to implement more advanced code and retain a deterministic execution.

At the same time, the FESA framework was undergoing development. With the introduction of new features, most of the basic operations done by the BBFS until Run 2 could be replaced by simple FESA commands. Some of the code used in the OFC and QFB was left intact: a) concentration of the COD, BPM and BBQ data; b) computation of the corrections required on the CODs and the tuning quadrupoles; c) Sending of the corrections to the respective gateways. The main program shown in Fig. 3 and the OFSU FESA class,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

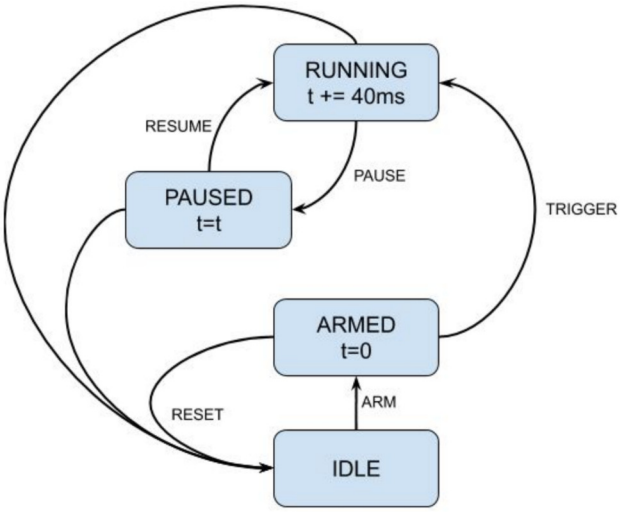

Figure 5: State machine of the FP as implemented in the BFCLHC.

contained the majority of the code which was replaced. The FESA framework was used to re-implement the logic of the main program.

To improve maintainability, the BFCLHC was designed to have a minimal amount of user code by relying more on the FESA framework features to implement the basic logic of the BFCLHC. For example, the data concentration is performed in a Real-Time Action (RTA), which updates the shared memory of the device with the latest BPM orbit data. A standard FESA interface property is then set up for providing the most recent BPM orbit measurements directly from shared memory. Following this design approach, trivial tasks (e.g. relaying data from real-time devices to users) were easier to implement and to debug. Considering that back-compatibility is ensured by FESA, future versions of BFCLHC will be easier to maintain [15]. It is important to note that the design of the BFCLHC Application Programming Interface (API) was based on the API of the OFSU, however, it was adapted significantly to better fit the needs of the operators.

Function Players (FPs) was a new feature that was added to the BFCLHC, at the request of the LHC operators. Using FPs for automated settings control was introduced in [16]. In the BFCLHC, the FPs are used to automate the change of: a) reference values, e.g. reference orbit; b) control loop gains; and c) optics models. Presently, the BFCLHC can accept a list of reference values along with a list of time offsets when the references should be applied. Linear interpolation of reference values was also implemented, e.g. varying the reference orbit linearly. Each FP is used depending on its current state. Figure 5 shows the general state machine of the FPs as implemented in the BFCLHC. The state is stored in the FESA shared memory, and can be changed by a FESA command. To change the state of the FP, the operators must either manually send events, or synchronise them with LHC timing. Some events are only available in specific states and a FESA exception is thrown when the user violates the state machine rules shown in Fig. 5; e.g. when the FP is in the `ARMED` state, only a `TRIGGER` or `RESET` event can change its state. A function can only be set when the respective FP is in `IDLE` state.

The optics calculation procedures are now delegated to the BFCLHC class. The approach to initialising and loading new optics to the feedbacks has therefore been changed. Any optics model must now be added to the BFCLHC memory via a FESA set command. Subsequently, a FP must be used to instruct the feedbacks when to change to new optics. The operators rely on their own sub-routines which can obtain a set of optic models from LHC Software Architecture (LSA) settings database [17], and forward it to the BFCLHC.

Following the Hardware Acceleration (HA) feasibility study in [18], we concluded that the use of GPUs did not improve computation times when calculating the Pseudo-Inverse (PInv) of the RMs when compared to a multi-core approach. In the new design, after an optics model is added, a new thread is spawned which calculates the Response Matrix (RM) as well as the corresponding PInv. In the event of a sensor or magnet malfunction, a thread per optic model can be spawned to re-calculate the adjusted models without blocking the control loop. The time to perform an optics recalculation took in the order of minutes to complete on the BBFS. The BFCLHC reduces this time to the order of seconds, which makes it possible to attempt in real-time in the LHC Run 3.

TESTING FRAMEWORK

The first formal testing of the OFC occurred at the start of the LHC Run 2 in 2014/15, during a code renovation that saw to the porting of the OFC and OFSU code from a 32-bit to a 64-bit architecture [19]. Considering the maintenance and correct porting of more than 90,000 lines of code, our colleagues from the operations group have built a separate testing framework during LS2 which mimics the input signals of the BBFS, by sending UDP packets disguised as coming from the BPM FECs.

The original design of the OFC made this operation complex due to its global control loop architecture. A series of code changes had to be made to the OFC and OFSU themselves simply to be able to test them. In particular, a global variable controlled key mechanisms within the OFC which determined whether the code was being used operationally or within a simulation. In addition, the output network ports were also tweaked in order to allow for a different base port to be used during testing. This was done as a protective measure against UDP packets being sent to the magnets during testing.

One of the main goals of the first testing framework was to verify that the renovated code respected all operational boundaries of the OFC. An identical hardware setup which was set up as a back-up to the operational hardware, had to be used to run the tests. The tests were written in a Java-embedded Domain Specific Language (eDSL) and the JUnit

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

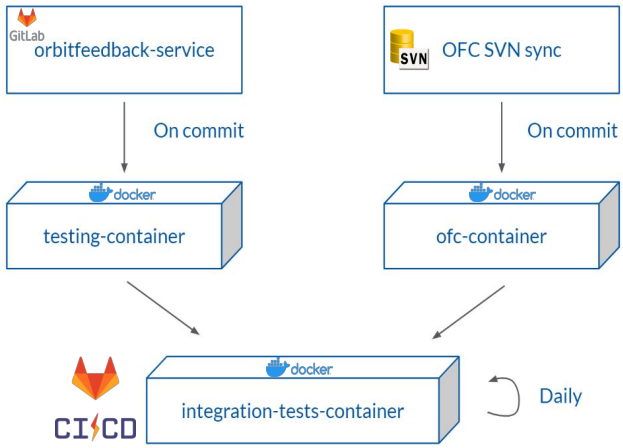

Figure 6: Testing framework schematic, showing the use of Docker containers in the GitLab Continuous Integration (CI) framework.

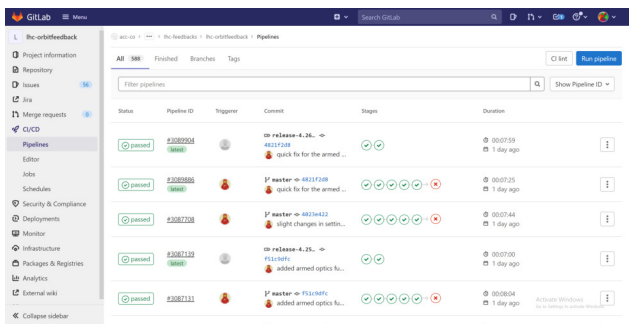

Figure 7: Screenshot of the BFCLHC testing framework test results using CI on GitLab.

framework was used to create the tests and generate reports. Due to the inherent complexity of this testing framework it was quickly realised that it was difficult to maintain and to use efficiently. Despite these limitations, this testing framework was successful in detecting bugs which aided the code renovation prior to the LHC Run 2.

The new BFCLHC testing framework is still undergoing development, and will continue at least until the start of Run 3. Figure 6 shows the basic structure of the latest design. It was decided to make the testing framework hardware agnostic with the use of Docker containers [20]. This new design allowed tests to be performed on the BFCLHC during active development through the use of Continuous Integration (CI) tools in GitLab [21]. As an example, algorithm 1 shows the pseudo-code for a test that checks the limit on the number of optics that can be loaded to the BFCLHC. Figure 7 is a screenshot from GitLab CI which shows the results of the tests that were performed automatically after successful compilation and Docker image creation.

CONCLUSION

The BBFS implemented the automatic feedbacks on the RF frequency, beam orbit and the tunes until LHC Run 2 in 2018. The renovation of the BBFS during LS2 saw the upgrade of the hardware, software and operational sides of the feedbacks. The experience gained when operating the feedbacks in the past, was taken into consideration during the renovation of the BBFS. Ultimately, the renovated BBFS is a FESA-based application called the BFCLHC, which includes all the required functionality of the BBFS, along with the addition of new features. The three main upgrades are: a) Introduction of FPs for the automatic changing of settings during LHC operation; b) More efficient optics computations and; c) a simpler and more user-friendly interface. A hardware agnostic testing framework was also developed by the LHC operations section to help with the development of the BFCLHC.

Algorithm 1 Test which asserts that only 30 optics can be loaded to BFCLHC.

```
optics_set                          ▷ Contains set of 30 optics
bfc                                 ▷ Running instance of BFCLHC
extra_optics                        ▷ 1 extra optics
res                                 ▷ Test result
for optics in optics_set do
    bfc.loadOptics(optics)
    if Allowable time expired then
        res ← Fail
        Exit
    else if not bfc.opticsLoadedSuccessfully() then
        res ← Fail
        Exit
    end if
end for
bfc.loadOptics(extra_optics)
if bfc.opticsLoadedSuccessfully() then
    res ← Fail
else
    res ← Success
end if
```

REFERENCES

[1] European Organization for Nuclear Research, O. Bruning, and European Council for Nuclear Research, *LHC Design Report, Volume I: The LHC Main Ring*, en. CERN, 2004. http://books.google.com/books/about/LHC_Design_Report_Volume_I.html?hl=&id=n-BmNQAACAAJ

[2] R. J. Steinhagen, "LHC beam stability and feedback Control-Orbit and energy," Ph.D. dissertation, RWTH Aachen U., Sep. 2007. http://cds.cern.ch/record/1054849

[3] L. K. Jensen *et al.*, "Software architecture for the LHC Beam-Based feedback system at CERN," CERN, San Francisco, USA, Tech. Rep. CERN-ACC-2013-0257, Nov. 2013. https://cds.cern.ch/record/1628552/files/CERN-ACC-2013-0257.pdf

[4] P. Forck, D. Liakin, and P. Kowina, "Beam position monitors," in *CERN Accelerator School: Beam Diagnostics*, D. Brandt, Ed., Dourdan, France: CERN, 2009, pp. 187–228. https://cds.cern.ch/record/1213277/files/p187.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

[5] M. Gasior and R. Jones, "High sensitivity tune measurement by direct diode detection," in *Proceedings of 7th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC 2005)*, Lyon, France, 2005, p. 4. https://cds.cern.ch/record/895142/files/ab-2005-060.pdf

[6] S. Kostoglou, G. Arduini, L. Intelisano, Y. Papaphilippou, and G. Sterbini, "Impact of the 50 hz harmonics on the beam evolution of the large hadron collider," Feb. 2020. arXiv: 2003.00140 [physics.acc-ph]. http://arxiv.org/abs/2003.00140

[7] L. Grech *et al.*, "An alternative processing algorithm for the tune measurement system in the LHC," in *Proceedings of the 9th International Beam Instrumentation Conference (IBIC 2020)*, Virtual, 2020.

[8] M. Arruat *et al.*, "Front-end software architecture," in *ICALEPCS07*, vol. 7, Knoxville, Tennessee, USA, 2007, pp. 310–312. https://accelconf.web.cern.ch/accelconf/ica07/PAPERS/WOPA04.PDF

[9] R. Murillo-Garcia, Q. King, and M. M. De Abril, "Control of fast-pulsed power converters at CERN using a function generator controller," CERN, Tech. Rep. CERN-ACC-2015-0128, Oct. 2015. https://cds.cern.ch/record/2059133/files/CERN-ACC-2015-0128.pdf

[10] R. Brun and F. Rademakers, "ROOT: An object oriented data analysis framework," *Nucl. Instrum. Meth. A*, vol. 389, M. Werlen and D. Perret-Gallix, Eds., pp. 81–86, 1997. DOI: 10.1016/S0168-9002(97)00048-X.

[11] *Release 5.34/20 - 2014-08-13 | ROOT a data analysis framework*, https://root.cern.ch/content/release-53420, Accessed: 2019-7-31. https://root.cern.ch/content/release-53420

[12] J. Wenninger and R. Steinhagen, "LHC orbit feedback control requirements," CERN AB-OP, Tech. Rep., Mar. 2007.

[13] J. Wenninger, L. Grech, Ed., Personal communication, CERN, Jun. 2018.

[14] R. Jones, *BE-BI 2019: The year in review*, BI Day 2019, La Villa du Lac, Divonne-les-Bains, France, 2019. https://indico.cern.ch/event/857941/contributions/3612395/attachments/1960206/3257519/BI_Group_End_of_year_2019_-_compressed.pdf

[15] F. W. Hoguin, *FESA quality assurance*, 1st Developers@CERN Forum, CERN, Geneva, Switzerland, Sep. 2015. https://cds.cern.ch/record/2056256

[16] D. Alves, K. Fuchsberger, S. Jackson, and J. Wenninger, "Test-driven software upgrade of the LHC beam-based feedback systems," in *2016 IEEE-NPSS Real Time Conference (RT)*, Padova, Italy: IEEE, Jun. 2016. https://indico.cern.ch/event/390748/contributions/1825184/attachments/1283146/1927869/CRX_PosterSession2_64.pdf

[17] C. Roderick and R. Billen, "The LSA database to drive the accelerator settings," Tech. Rep. CERN-ATS-2009-100, Nov. 2009. https://cds.cern.ch/record/1215575?ln=en

[18] L. Grech, D. Alves, S. Jackson, G. Valentino, and J. Wenninger, "Feasibility of hardware acceleration in the LHC orbit feedback controller," en, in *17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, 05-11 October 2019*, JACOW Publishing, Geneva, Switzerland, Aug. 2020, pp. 584–588. https://accelconf.web.cern.ch/icalepcs2019/doi/JACoW-ICALEPCS2019-MOPHA151.html

[19] S. Jackson, D. Alves, L. Di Giulio, K. Fuchsberger, B. Kolad, and J. Pedersen, "Testing framework for the LHC beam-based feedback system," in *Proceedings of the 15th International Conference on Accelerator and Large Experimental Physics Control Systems*, L. Corvetti, K. Riches, and V. Schaa, Eds., Melbourne, Australia, Jan. 2016, pp. 140–144. https://accelconf.web.cern.ch/ICALEPCS2015/papers/mopgf024.pdf

[20] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, p. 2, Mar. 2014. https://dl.acm.org/doi/10.5555/2600239.2600241

[21] *GitLab CI/CD*, https://docs.gitlab.com/ee/ci/, Accessed: 2021-8-31. https://docs.gitlab.com/ee/ci/