AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI
INSTYTUT INFORMATYKI

Projekt dyplomowy

Offline reconstruction software for diamond sensors based on SAMPIC readout in the CMS-PPS detector at CERN laboratory.

Oprogramowanie na potrzeby rekonstrukcji danych z detektorów diamentowych używanych w projekcie CMS-PPS w CERN.

Autor: Krzysztof Misan
Kierunek studiów: Informatyka
Opiekun pracy: dr Leszek Grzanka

Kraków, 2022

Contents

1. Project goals and vision

The primary purpose of this section is to provide a brief overview of the proposed project alongside with an introduction into CERN's and CMS-PPS systems and related nomenclature. It contains a basic description of hardware and software-based solutions which were chosen for the project's development.

1.1. Introduction

CERN

The European Organization for Nuclear Research (CERN [1]), located in Geneva, is the world's largest particle physics laboratory and one of the most respected research organisations. It's primary goal is to further extend our knowledge about high-energy particles by gathering and analyzing data from particle accelerators. The Large Hadron Collider accelerator [2] operated by CERN is currently considered the largest machine ever built. LHC collides beams of accelerated particles and measures the properties of the particles generated during collisions in various detectors specifically designed and constructed for a given task. Data is then initially filtered, digitalized, and processed by specialized software. During data acquisition, millions of collisions are recorded and stored for further processing. In 2019, CERN was reporting the production of over 100 petabytes of data per year with over three billion files saved across EOS storage system [10].

Among the numerous achievements in the field of experimental physics, the grand discovery was made in 2012 by CMS [6] and ATLAS detectors, which observed a new particle, later confirmed to be the postulated Higgs boson. CERN is also considered a father of the World Wide Web and a pioneer in early distributed computing solutions. Multiple projects in the field of Computer Science are being continuously developed and maintained by the engineers working at CERN. Among others, many technological breakthroughs were made in the fields of distributed storage and computing systems, data compression, and parallel data processing.

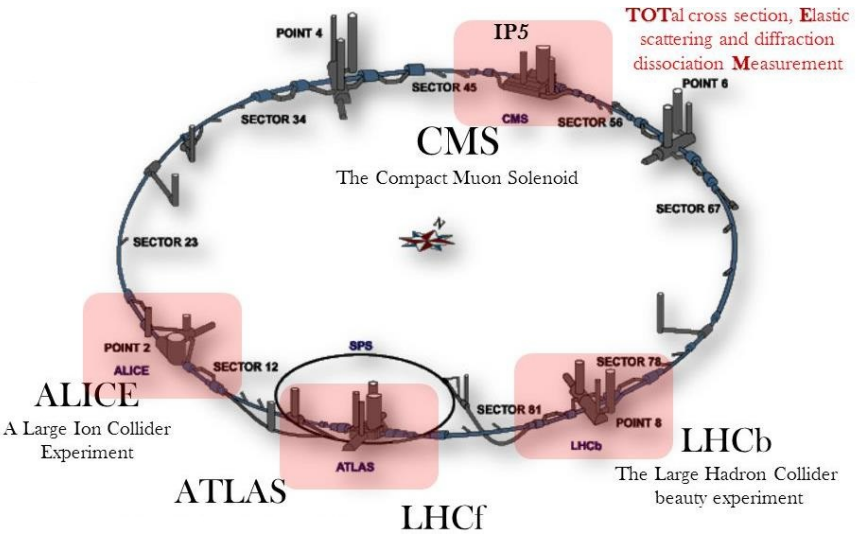


Figure 1.1: Scheme of the LHC with TOTEM and CMS experiments at IP5

LHC particle accelerator

CERN is a home to multiple particle accelerators, the Large Hadron Collider [2] is the biggest and most recognized. It measures 27 km in length and is placed between 75-100 m underground to minimize the external noise effect on the detectors - such as cosmic rays and traffic vibrations.

Two counter rotating beams are injected into the LHC from the SPS accelerator (the Super Proton Synchrotron). Proton beams are injected at 450 GeV and then accelerated to 7 TeV, circulating around the LHC ring inside a continuous vacuum which pass through a large number of magnets.

LHC ring is divided into 8 arc sections, each 2.45 km long host dipoles (1232 across all sections) which operate in a superconducting state at temperature 1.9 K and a magnetic field of over 8T. Different kinds of particles might be accelerated - mainly protons but also heavy ions. Each proton beam at full intensity consists of $\sim$2800 bunches separated by 25 ns, and each bunch contains $\sim 10^{11}$ protons per bunch. After reaching the nominal energy in LHC, the particle beam is collided at four locations (IPs, Intersection Points), where two beams traveling in opposite directions cross each other. The angle of such crossing and many other parameters, depend on the desired luminosity and is adjusted throughout the data-taking to maintain the optimal collision rate for the experiments. When the bunch intensity is too low, the particles are steered away from the LHC ring and dumped into graphite blocks. Four main experiments are located at the corresponding intersection points, these are: ATLAS (IP1) [4], ALICE (IP2) [3], CMS (IP5) [6], LHCb (IP8) [11]. Aside from the experiments, energy of the particles is being increased at Point 4 using radiofrequency cavities. Collimators are located at Point 3 and Point 7, they are responsible for filtering the beam from strayed particles. When there are too few collisions in LHC or malfunction occurs, the particle beam is dumped at Point 6.

TOTEM and PPS experiments

PPS project (Precision Proton Spectrometer) has started as a joint collaboration of CMS and TOTEM [15] experiments, later evolved into the current CMS subsystem (fig. 1.1). TOTEM experiment is focused on measuring and analyzing properties of proton-proton interactions including elastic scattering and total cross-section. The Totem experiment is located near IP5 (Intersection Point 5), it is equipped with multiple detector stations with sensors placed inside Roman Pots (RPs) [21]. Roman Pots are vacuum vessels which were designed to host different types of sensors. Detector stations, which are composed of multiple RPs, are located symmetrically on both sides of IP5 (fig. 1.2), spreading across half-kilometer distance the TOTEM is known as the CERN's 'longest' experiment. The TOTEM experiment currently operates 26 RP detectors and four particle telescopes with the involvement of over 100 scientists from all around the world.

Among the achievements of TOTEM is the discovery of the odderon particle, whose existence was postulated in 1973, with the final results presented at CERN in 2021. The discovery was only possible due to the unique setup of the TOTEM detectors, specialized in measuring elastic-proton collisions where particles remain intact, with a small deviation in their paths, instead of being broken into quarks and gluons.

The TOTEM RP stations are now part of the PPS layout. RP units are equipped with sensors with different functions, providing time and tracking measurements of arriving protons. Each arm consists of a tracker, which measures the particle trajectory, and a timing detector to determine the particle's time of flight from the interaction point. These measurements allow to establish the kinematical properties of at the interaction point (scattering angle, energy loss) as well as the interaction position along the beam line.

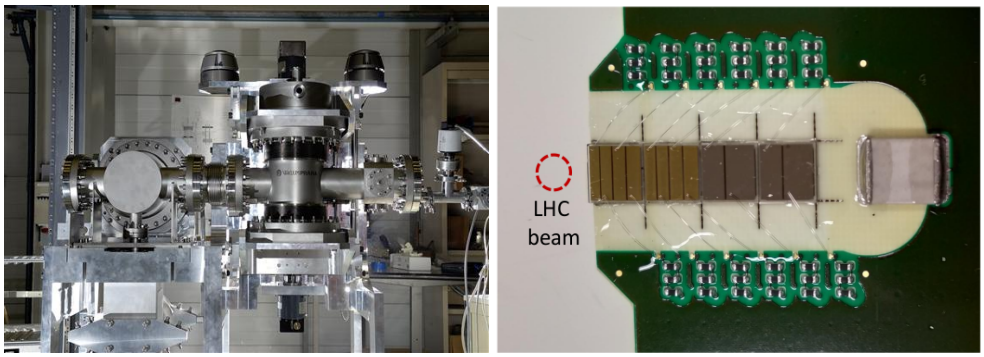
The silicon strip detectors used in TOTEM could only register the hit along one-dimensional pads, hence it was necessary to use at least two strip sensors angled in different directions to measure the hit in the detector plane. These sensors are now used only for calibration data. Currently the RPs are equipped with silicon pixel detectors which can register the hit location in 2D space. To obtain the particle trajectory, multiple sensor boards are stacked together in layers. Those layers provide discrete points in space which can be then interpolated to acquire a particle path within the detector.

Timing is a vital component of PPS [22] system which allows to match the tracks of scattered protons from the two arms and determine the common collision vertex. Resolution less than 100 ps was achieved in 2018 for the 500 m span distance between individual timing stations. This work is focused on a specific detector type - Diamond timing detector, which is being used in the experiment for precise particle arrival time measurements. The primary readout is based on a a fast discriminator, the NINO chip, followed by HPTDC (High Precision Time to Digital Converter) readout system [17], which is supporting up to 32 channels, but when used with maximum resolution, the number of channels is reduced to 8. The NINO is an 8-channel ultra-fast discriminator, designed to operate with the HPTDC. In addition to recording the signals' leading edge, the chip is capable of encoding the input charge collected from the detector as the output duration, which could be used for corrections based on Time Over Threshold (TOT) technique [19]. HPTDC is able to correctly measure both the leading and trailing edges of the signal and it can operate at rates above 1 Mhz per channel but is unable to fully recover the time precision (as it delivers only two digital time measurements), which, in case of PPS sensor, caused results degradation of up to 30%.

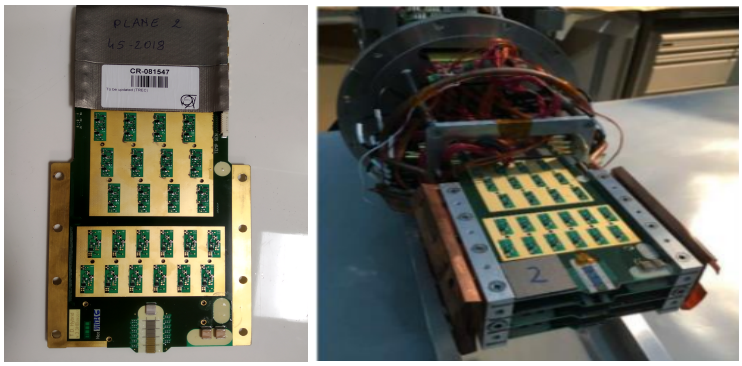
Given the limitations of HPTDC, another readout system was considered. The SAMPIC (SAMpler for PICosecond time pick-off) [17] is a fast sampler ASIC which supports up to 16 channels (64 samples each) and acts as a waveform digital converter (similar to an oscilloscope) by sampling the analog input. Each channel is digitized independently and the readout order is random, which implies that the data have to be sorted elsewhere. Even though SAMPIC readout is capable of conducting more precise measurements than HPTDC, it is operating much slower due to the amount of samples it has to process. This limitation excludes SAMPIC from the possibility of being used as single-readout system during data-taking. Instead, both systems are going to operate in parallel, with SAMPIC readout connected only to a fraction of available channels. It will provide a high quality sample which could be then used to further tune the HPTDC calibration.

The SAMPIC chip is placed on the mezzanine board which is connected to RMB (Readout MotherBoard). Three fully connected RMBs can support up to 2 RPs, each hosting 4 diamond detector planes.

The other type of detector, based on UFSD (Ultra-Fast Silicon Detector) [18], was previously used for particle arrival time measurements. The full system was consisted of 4 RPs with 16 UFSD sensors and 192 unique channels. This system was used in 2018 with SAMPIC readout, allowing for an acquisition window of 3.1 ns with a channel dead-time of 250 ns. The full rising edge of the signal was recorded and the signal maximum was extracted with no timing resolution loss due to the digitization. Limitations and advantages of the SAMPIC and HPTDC systems were precisely understood, which led to a proposed joint HPTDC-SAMPIC readout for the Run3.



(a) Detector station with 3 RPs (b) Pads of the Diamond sensor



(c) Diamond plane (d) Diamond detector package

Figure 1.2: Detectors hardware

1.2. Project overview

The goal of this project is to adapt and add new features to the existing software to introduce the SAMPIC readout for data acquisition of Diamond timing detectors, which is the acquisition method to be used in LHC-Run3 in parallel with the currently used HPTDC system. Anticipated product of this work is an offline reconstruction software capable of parallel processing HPTDC and SAMPIC acquired data. This involves modifications to multiple stages of the current workflow (fig. 2.3), as well as creating additional modules which extend the capabilities of already existing software. The product was commissioned by the TOTEM project and the only intended users of the resulting software are the CERN researchers. This is implied by the specifics of the project which has to be designed and built according to unique hardware requirements and linked to CMS software framework (CMSSW) [7]. Although, due to those restrictions, the resulting software can not be deployed outside of the CMS experiment, it is certainly possible to reuse its components and solutions in other fields. This include many applications involving very precise signal processing, such as measurements taken in medical cyclotrones and radioisotope laboratories.

Scope of the project could be divided into the following steps:

- Create a waveform simulator acting as data input
- Adapt existing software to support SAMPIC readout
- Validate the results
- Produce new calibration procedures
- Create Data Quality Monitor for SAMPIC acquired data

Furthermore, this project has to fulfill the requirements posed by the scientists working at CERN:

- Validity of provided solutions
- Parallelization of HPTDC and SAMPIC readouts
- Capability to process large amount data
- Compatibility with existing software
- Consistency with existing deployment procedures

The first phase of the project was focused on generating input data for the reconstruction stage. It is a necessary step as it is not possible to use existing data due to the lack of viable data - there was no LHC data-taking for Diamonds operating under SAMPIC. Therefore, already existing SAMPIC data (gathered from the testbeam data-taking which conditions are different to LHC) had to be mapped to the correct Diamond detectors and placed in the corresponding LHC-compatible structures.

Generated data is going to be forwarded to the reconstruction software which is responsible for extracting proton tracks. This software has to be adapted to be used with Diamond detectors. Event reconstruction is divided into several substages. Firstly, the detector-specific process unpacks and decodes the input data, which are then used to derive the object hit. The second step involves reconstructing the entire tracks from previously extracted hit, which is also the most computation heavy task. The next step rebuilds the primary and secondary vertex candidates and ultimately the PPS and central detector tracks will create the candidate event. Modifications to this part of the software include creating new configuration procedures and changes to already existing reconstruction algorithm. The output is then going to be validated against DQM [16] plots already available in database.

Parallelization of the process can be achieved by ensuring consistency with CMSSW, which is based on EDM [20] and capable of processing thousands of event containers simultaneously and storing them in event collections for further analysis.

Due to numerous dependencies, the project has to be conducted in full compatibility with existing CMSSW software and follow EDM conventions. This imposes a modular software architecture with the starting module acting as a data source and other modules performing the role of data producers which process and add new information to considered Event.

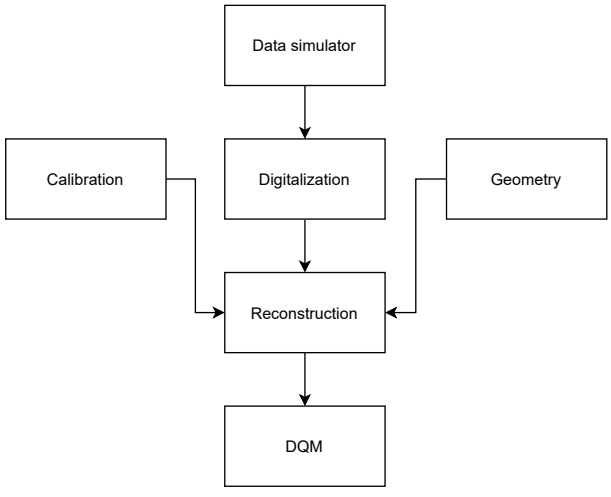


Figure 1.3: Simplified workflow of SAMPIC readout software

1.3. Data processing at CERN

The project has been conducted within CMSSW (CMS Software) [7], which is implied by characteristics of the work focused around expanding existing software components. CMSSW is a C++ based, multi-component, modular software built around EDM (Event Data Model) framework and used in CMS event processing. The software is heavily utilising C++ efficiency and parallel computation to process large amounts of LHC data. When all data related computations are written in C++, the configuration management is based on script files written in Python.

Additionally, CMSSW depends on ROOT Data Analysis Framework [14] which is C++ based, fast environment, featuring tools for data processing and data analysis. It also allows for C++ data structure serialization to *.root* binary files which is standard, self describing, input/output data format for entire CMSSW. This framework is also a basic tool used for large volume data visualization, providing options to present results as histograms and scatter plots. Every module within CMSSW is able to store processed events in *.root* files. There are also multiple existing ROOT validation plots which allow for fast results evaluation within CMSSW.

The Event Data Model (EDM) [20] describes the workflow of all existing software including simulation, calibration, alignment, reconstruction, and analysis modules. The framework is built around Event data structure which can contain any C++ class. Attached modules perform actions on data stored in Event structure by modifying or adding the content. Processing by modules is sequential within one Event but is frequently performed in parallel within the whole set of Events. The information from each step of simulation and reconstruction is divided into different data tiers described below.

CMS Data Hierarchy

Data in CMS experiment is organized into tiers according to the level of details stored in the dataset:

- RAW - contains the full event description from Tier-0 [section 2.2], including detector oriented data.
- RECO - output from the Tier-0 reconstruction, contains objects created during the event reconstruction stage, such as reconstructed hits or higher level tracks and vertices. The structure also maintains links to the RAW data.
- AODs (Analysis Object Data) are constructed from RECO and contain lightweight, compact information which could be distributed to others and used by them for physical analysis.

From the project perspective, the most important tiers are RECO and AOD.

1.4. Project risks

The described project was exposed to numerous potential risks. The most critical of them were technical in nature:

1. The software created for this project had to be run in CERN computing environment. Moreover, it required access to a large amount of data stored on EOS [10]. Therefore, the access to CERN systems and their stability were vital points, constantly exposed to threats such as: temporary service suspension, system overload, access revocation or CERN lockdown due to the pandemic. In order to decrease this risk, the project was developed locally

and remotely synchronized with CERN infrastructure. Additionally, existing issues were regularly addressed by specialists from the TOTEM project.

2. The new readout system heavily relied on existing parts of reconstruction software. If, for some reason, it had been impossible to use Diamond detectors with already implemented SAMPIC data structures and processing algorithms, it could've posed a serious threat to entire project. To minimize this risk, the existing source was carefully reviewed beforehand and alternative solutions were discussed with TOTEM experts.

3. Due to lack of real data, all input had to be derived from different data structures than the ones used in LHC data-taking, which would have posed considerable risk to the project if data formats were incompatible. This would imply that project validation could only take place after measurements were taken from particle beam by physically connected Diamond detectors. Exposition to this threat was decreased by discussing it with CERN experts and securing existing SAMPIC data.

Taking those risks into account, it is important to consider the probability of its occurrence. CERN infrastructure was designed and built to sustain extensive stress and overload, which makes the chances of DOS or prolonged instability very low. The more probable risks are the ones related to the compatibility and lack of necessary validation data. These are considered the main risks of the project and were carefully evaluated and acted on in advance.

1.5. Feasibility study

This section is aimed to describe technical, operational, and scheduling challenges by providing an objective assessment of the practicality of the project.

Technical

Minimal technical requirements of the project include access to existing source code and archived data. CMSSW is an open source project published using popular VCS tools which results in free and unrestricted access to relevant files. In turn, accessing the necessary measurements requires a connection to EOS [10] storage system behind CERN authorization firewall. Those data are available for project participants, students, and researchers without any limitations.

Development and deployment phases face additional constraints because of the customized operating system CC (based on Linux CentOS distribution) integrated with CERN computing environment. Configuring and deploying the said system in a local environment is difficult and prone to numerous issues including but not limited to version and architecture incompatibilities. To prevent that from occurring, a different possibility was taken into account. Local environment with Ubuntu Linux distribution ROOT framework and Visual Studio Code with Python3 and C++ support were deployed and bidirectionally synced with the corresponding files stored on EOS by utilising CERNBox service. This allowed to access CERN based services using SSH protocol while developing code and analyzing results of the computations locally. It also ensures that at least two copies of the currently developed software are stored at physically different locations, as opposed to VCS, in which case the files are not synced automatically, which in turn could result in a loss of the latest changes. Creating an efficient and secure local environment was vital for the development of this project, which was, for the most part, conducted remotely and only partially on site.

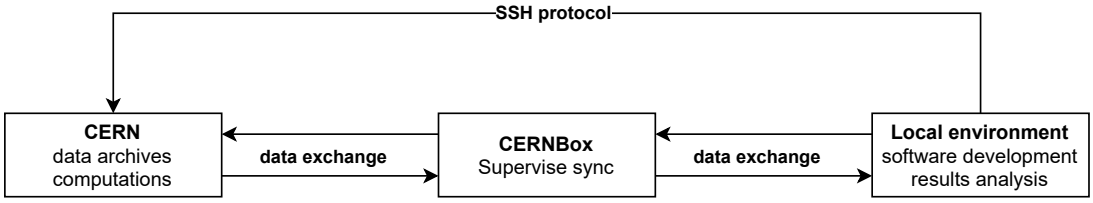


Figure 1.4: Remote communication schema

Another technical challenge, which was promptly discussed in 1.4, concerns the validation and comparison of the developed software with a previous readout system. The input data, which should have been gathered from physical detectors, didn't exist at the time this project was conducted and therefore replacement data constructed from testbeam measurements had to be considered. This implies additional difficulties caused by the format incompatibilities which have to be manually addressed and changed to correct geolocalization, effectively deceiving the software that delivered data come from the correct Diamond detectors.

Operational

Operational requirements for the project include deploying the software as part of CMSSW, either by creating independent CMSSW modules or modifying the existing files. Accessibility of the service will be limited to the CERN computation environment due to the necessary data and computation complexity. Produced source code is going to be published as part of CMSSW open source framework. Making the software work with other analysis tools used at CERN does not require additional overhead beyond deploying it to CMSSW, thanks to the unified data format. Considering the backward compatibility of the modified part of the software, two mechanisms were deployed:

- Duplication - which presumes duplication and encapsulation of the code which has to be modified but could not be generalized
- Generalization - which presumes the generalization of the modified source code by utilizing polymorphism and code generation mechanisms

Those guidelines not only ensure backward compatibility of the software, but also make it easier to maintain by providing clear namespace division for the code with significant differences while avoiding the creation of unnecessary files for similar workflows.

Summary

Conclusion of this assessment is that the considered project has all the means to overcome the described technical and operational challenges, including having access to the necessary data, tools, and solutions. It has also a rich resource base, including tools, expert opinions, source repositories, and guidelines, which could help when dealing with unforeseen difficulties. Software deployment procedures and maintenance difficulties were discussed with experts and taken into account by creating suitable guidelines and mechanisms in advance.

1.6. Glossary

- *CERN* - European Organisation for Nuclear Research

- *LHC* - Large Hadron Collider, particle accelerator under CERN management
- *TOTEM* - TOTal Elastic and diffractive cross section Measurement, detector experiment at CERN
- *RP* - Roman Pot, cylindrical vessel which hosts different detectors
- *UFSD* - Ultra Fast Silicon Detector, timing detector used in 2018 experiment
- *Diamond* - Timing detector currently operating in TOTEM experiment
- *HPTDC* - High Precision Time to Digital Converter, current readout system of Diamond detectors
- *SAMPIC* - SAM-pler for PICosecond time pick-of, alternative readout system used previously with UFSD detectors
- *CMSSW* - CMS SoftWare, framework of CMS and TOTEM experiments
- *DQM* - Data Quality Monitor, part of the CMSSW which is responsible for data consistency and correctness
- *ROOT* - data analysis framework
- *EDM* - Event Data Model
- *EOS* - Eos Open Storage, CERN data storage system
- *RECO* - RECOnstructed data format
- *AOD* - Analysis Object Data
- *VCS* - Version Control System
- *CC* - CERN CentOS operating system
- *Conditions Database* - Database of CMS which stores running conditions of various workflows
- *PCL* - Prompt Calibration Loop - system which runs calibration workflows
- *Tier-0* - First layer of CERNs' computing grid which includes on site computing and storage centers

2. Functional scope

This chapter's intent is to provide a detailed explanation on the potential users of the produced software, its relationship with other systems in CERN computing environment, and the functional and nonfunctional requirements of the project.

2.1. Users context

Primary users of the system are physicists and data analysts at CERN who can use the reconstructed particle objects and particle tracks in their own workflow. Those specialists do not need to know about the technical details within the provided implementation and should be able to analyze and process the output using custom CERN developed tools without any additional knowledge on how the software operates.

2.2. Associated systems

Software components are going to be run using top level distributed systems. The reconstruction and calibration will be primarily used for LHC-run3 data-taking and therefore used as part of a Tier-0 system (on site). Within 24h of data acquisition, the Express system within Tier-0 is going to run the calibration and reconstruction on the partial data set. The second system - Prompt is going to run the reconstruction with a full data set and with more precise calibration returned from the Express. The last domain, Offline is going to run re-reconstruction with the goal to apply fixes to incorrectly reconstructed data and deliver high precision results. Both reconstruction and calibration software are interacting with Condition Database of CMS. For the reconstruction - this is to establish a working environment, ensure that correct conditions are used, and deliver correct data records, whereas for channel alignment and calibration it is to deliver the input records and update the existing calibration with the created output.

CERN Computing Grid

CERN Computing Centers are divided into 3 tiers. The Tier-0 are the centers located directly on site and are responsible for immediate data processing. Tier-1 and Tier-2 centers are distributed across the world and are used for tasks such as physics analysis or re-reconstruction. Tier-1 nodes are big, national data centers which fulfill the extensive need for storage capacity, whereas Tier-2 nodes are usually small, university-located computing centers without storage capabilities but with substantial computing capabilities. As of 2018, there were 55 T2 centers connected to the CERN infrastructure.

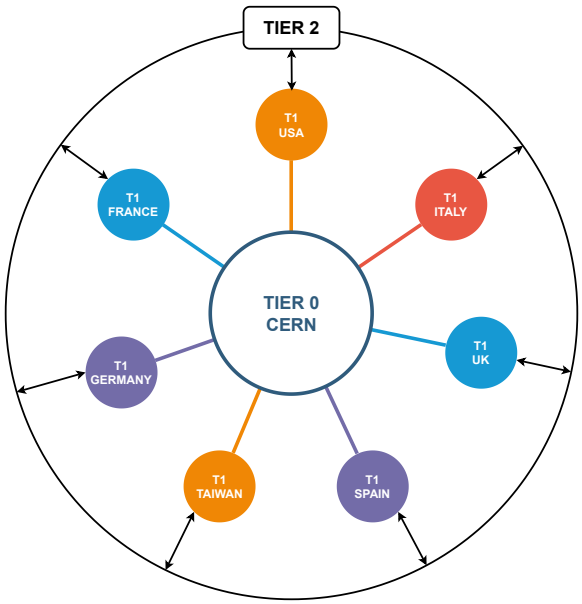


Figure 2.1: CERN computing grid structure

Tier-0

CERN's computational infrastructure is hierarchically divided into three different layers called Tiers. The software presented in this work is intended to run in so-called Tier-0, which is the first layer in the model, operating on a CERN site. Tier-0 system has few objectives and could be considered an entry point for the RAW data processing acquired from the Online Data Acquisition. It repacks, archives and distributes those data to the distributed environment of Tier-1. Aside from this, it also performs the prompt calibration and runs the prompt reconstruction.

Tier-0 system is often being associated with online processing, as it's performing RAW data computations directly on site. Being the system closest to the acquisition hardware, it provides fast and reliable feedback as to detectors performance. It is also important to mention the limitations which come with fast data processing in Tier-0. Limited resources have to be assigned to the most important tasks, which means that many calculations have to be performed on a statistical sample rather than the whole dataset and often resort to the use of approximations.

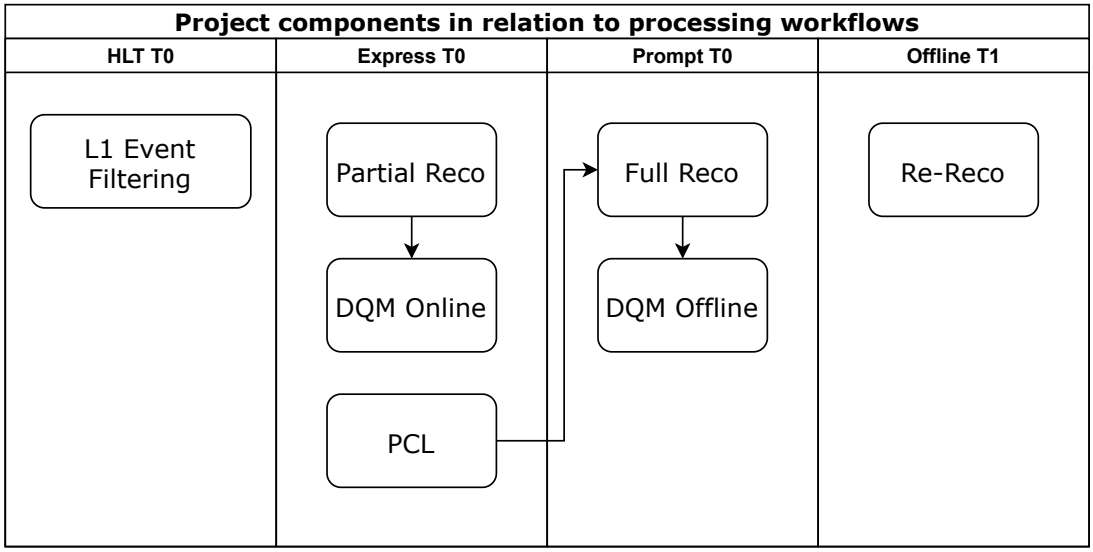


Figure 2.2: CMSSW processing domains

Prompt Calibration Loop

Channel alignment calibration implemented for the SAMPIC readout [17] as part of this project is intended to run in Prompt Calibration Loop environment. PCL takes as an input a statistical sample of the whole dataset, performs alignment and calibration by taking advantage of parallelization where it's possible, and saves the results to Condition Database for immediate use in further T0 processing. Thanks to the PCL, the best calibrations are available as soon as it's possible, which greatly improves results of the reconstruction.

Data Quality Monitor

Data Quality Monitor (DQM) is a mechanism which processes serialized C++ data classes and outputs user-defined histograms in *.root* file format. This procedure is inherited by software components called DQM modules, which run within CMSSW workflows. Within those modules, the input data is aggregated and processed following the provided algorithm. Output structure is also defined within the module, alongside with the histogram's parameters and descriptions. It is also possible to store the outputs in structures other than histograms, but it's

usually not advised due to additional memory requirements. Within CMSSW we can distinguish two DQM processes: Online and Offline. Online DQM is an automatic process which runs DQM modules in T0 Express (hence Online), whereas Offline runs without strict time constraints. Following this separation, Online modules have to be lightweight in terms of memory and processing power requirements and usually provide output to ensure the hardware is operating correctly, whether Offline modules can perform more sophisticated data processing and focus more on physics driven analysis.

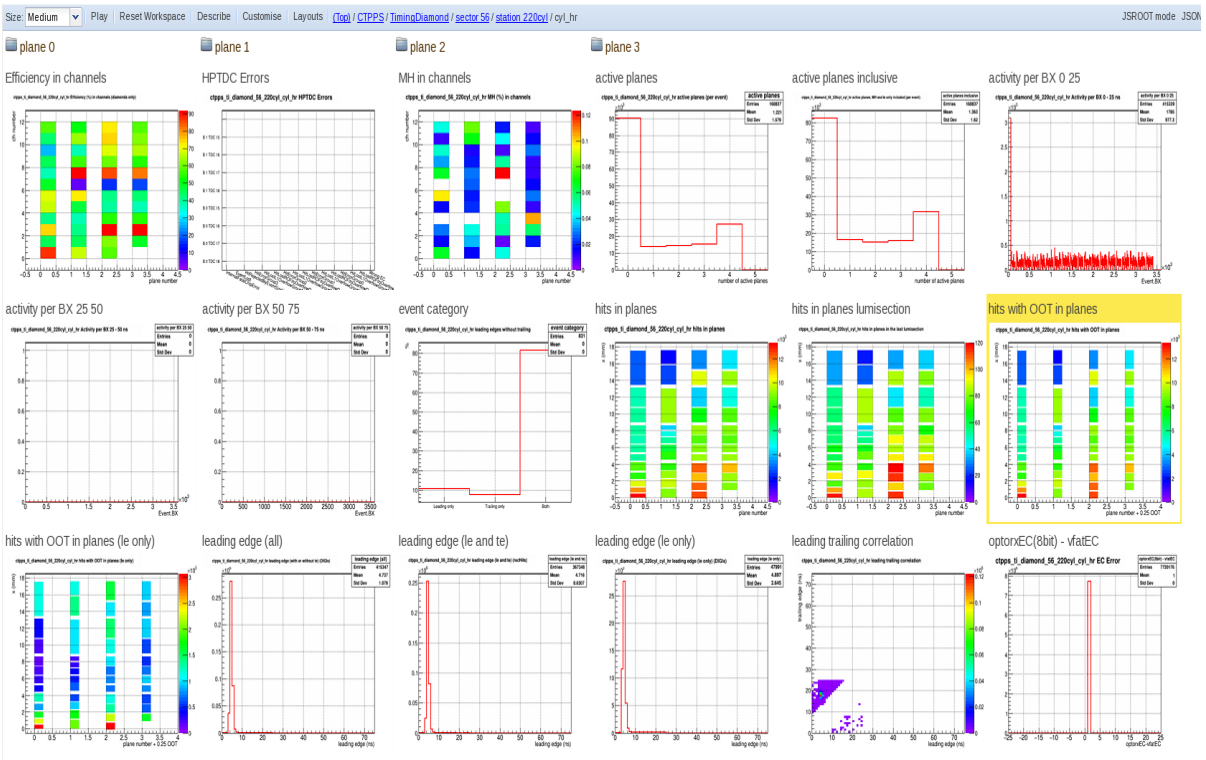


Figure 2.3: Example of DQM Online results presented in DQM GUI

DQM GUI is a graphical interface which takes the DQM root file as input and displays the plots in an organized, predefined, interactive layout for user convenience.

2.3. Functional requirements

Given the modular nature of the developed software, each module has to work under constantly changing conditions and often run in various environments.

The reconstruction is the module whose objective is to produce reconstructed physical objects and tracks within the detectors. Input data for this module is the output of the previous digitization stage. The structures used to transport the data have to match the previously used for UFSD-SAMPIC flow. The developed modules have to be able to process both UFSD [18] and Diamond detector [19] measurements, which requires the usage of code generalization techniques, such as class templates, class specialization, and inheritance. Software should be able to accept the payload from both local file systems or the specified database object (condition database payload).

Channel alignment calibration has to run in PCL workflow which is a part of Tier-0 Express software. Calibration software takes the digi and ADC calibrated reco data (ADC calibration is produced offline) as the input alongside with the initial calibration values. The goal of the

calibration software is to analyze the data characteristics and produce a channel alignment offset which is going to be then uploaded to the database object and used in Prompt reconstruction of the Tier-0. The software is to be implemented as PCL, with multiple workers running in parallel and producing partial histograms, which are then combined by the harvester to derive the correct values of the channel offsets.

The DQM (Data Quality Monitor) is a stand-alone module which runs after the reconstruction stage and is producing plots to visualise the reconstructed objects characteristics. The produced DQM module is intended to run offline and online analysis.

2.4. Non-functional requirements

The project has to fulfill multiple additional requirements concerning the efficiency, security, and reliability of its components. First and foremost, the developed software has to be efficient enough to handle large volume data processed at LHC. This is partly done by the CMSSW framework itself (parallel execution of certain modules), but additional work towards efficient implementation is necessary to ensure that the software is running within the allowed time limits. Time constraints are vital for any code executed within Tier-0 as any delay could result in a forced termination. This is because Tier-0 is used to validate the sensor operation and is most important to detect any issues as soon as it is possible.

Another constraint is the security and reliability of the software. It is required not to cause any issues to the ongoing workflow, which is partly ensured by the modular design of the software. Nevertheless, the code has to be designed in a way which allows to run it under exceptional conditions without causing exceptions or, in more severe case, crashes. Edge cases have to be closely reviewed and put under consideration to guarantee that the software will always successfully run within a given amount of time.

The scalability of the project has to be considered in the context of its ability to run in a distributed environment and utilise its resources. Some of the algorithms could run and produce output in parallel. Therefore, it is of most importance to design the modules in a way that supports parallel execution and changes in flow control.

All parts of the developed software have to be validated either against the tests already available in the CERN database or explicitly designed validation routines. Most of the validation in CERN is done using the DQM which produces the plots compared against the reference. Additionally, the execution test on Tier-0 is needed for the software which is to be run during data-taking.

Although the software under consideration is to be used exclusively by CERN and more precisely, the TOTEM experiment, it is an important part of a project to design and deploy a suitable configuration mechanism which will allow users to use it locally or in a different distributed environment. CMSSW delivers such a mechanism through a top level Python configuration, which could be then used to change the initial conditions of the workflow or even control its execution by plugging or unplugging the desired modules.

Maintainability of the produced code has to be taken into account, it is required to avoid code duplication and preserve full backward compatibility for the modified modules. This is to be done using C++ code generalization techniques and class inheritance. The produced code also has to follow the style conventions of CMSSW, which allows for easier management while presenting a consistent and standardized structure.

2.5. Use scenarios

2.6. Test and validation methods

The presented project follows the CMSSW guidelines as to handling test and validation tasks. The framework provides multiple tools for automatic and manual validation as well as performing code quality and validity checks. The test and validation procedures were divided into two categories: code supervision and results validation.

Code supervision

All code deployed in CMSSW environment is automatically validated against the current set of guidelines which describes code style, structure and logic composition. This step also involves checking the dependency structure. Manual code review comes after positive automatic validation. Code reviewers are assigned depending on the modified or created modules. In this project, the changes usually involved Reco and DQM teams. All assigned groups have to sign the changes in order for them to be merged into the main branch. The goal of this procedure is twofold, it ensures that all of the affected domains within CMSSW are informed about the changes and reduces the possibility of bug introduction. Authors and reviewers are aided by the automatic test infrastructure which helps to discover and diagnose possible compilation and runtime errors.

Another aspect of code validation focuses on execution conditions, namely, resource consumption. Strain on the resources could be split into two categories: offline and online. Offline resource requirement refers to the memory size of the output files, whereas online consumption refers to memory and processing cycle requirements during runtime. The first category - Offline is tested automatically per submitted change. Online resources are only periodically measured and are performed and analyzed per workflow.

Results validation

For the purpose of results validation, CMSSW deploys a separate set of procedures. Automatic validation involves unit and matrix tests. Matrix tests are a unique mechanism within CMSSW, which compares outputs before and after the changes for the selected set of workflows. It allows to check if the expected differences in results are visible in the data and if other workflows are unaffected by the introduced changes. Those comparisons could be done on the raw values or histograms, in which case they are compared by bin.

Sometimes not all submitted changes can be tested automatically, in such cases manual tests are often the last and only option to validate the results. By providing a configuration file which describes the inputs, processing, output, and running conditions, it is possible to build a local environment, acquire the resulting file, and manually investigate its contents.

3. Selected realization aspects

The purpose of this chapter is to provide a detailed description of the most critical software components, including the overall logical structure design, implementation details, and configurations.

3.1. Testbeam data converter

As previously stated, one of the biggest obstacles for this project was obtaining data for validation and test purposes. Diamond detectors [19] were never operated in LHC under SAMPIC readout [17] before and therefore the corresponding results were not available. Fortunately, those detectors were operational during special testbeam data-taking. One has to consider different conditions (both hardware and software related) during those data acquisitions and therefore a converting system had to be developed and put in place before feeding the data into the reconstruction step.

The main objective of this converter was to map the conditions existing in testbeam run data to the Run3 diamond detector setup. This involved constructing *TotemTimingDigi* and *TotemTimingEventInfo* objects used by Reco step by utilising a static mapping with testbeam detector IDs linked to the corresponding IDs in Run3 setup. Testbeam data were acquired using 2 detector planes, while the Run3 configuration foreseen 2 RPs for each side of the experiment, each equipped with 4 planes of detection. Therefore, values for those pairs were calculated manually in such a way that 2 detector planes from testbeam data were duplicated across 4 diamond planes in the currently operating station. This station data were then duplicated once again to simulate the two stations setup planned for the LHC Run3. All of those operations were performed by manipulating the corresponding values in 32 bit ID structures (please refer to figures 3.1 and 3.2).

bits: 31-28 **Interpretation:** Detector ID	27-25 Subdetector ID	24 Arm ID	23-22 Station ID	21-19 RP ID	18-17 Diamond Plane	16-12 Diamond Channel	11-0 Unused

Figure 3.1: Logical structure of 32 bit addressing system

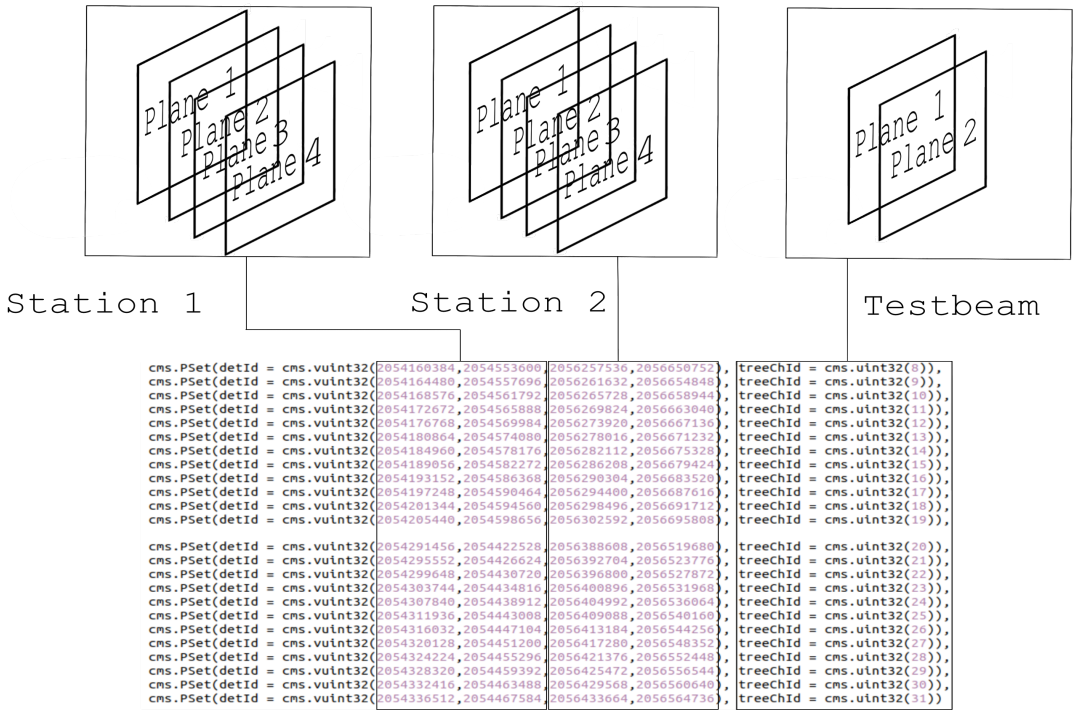


Figure 3.2: Mapping of 2 testbeam channels to 2 diamond stations, 4 planes each

SAMPIC data structure

Already digitized SAMPIC data stored in a *root* file are fed into the conversion algorithm. This multi-layered structure is stored in a columnar dataset called *TTree*, which consists of multiple *branches*, which in turn contain multiple *leafs* representing vectors of data objects.

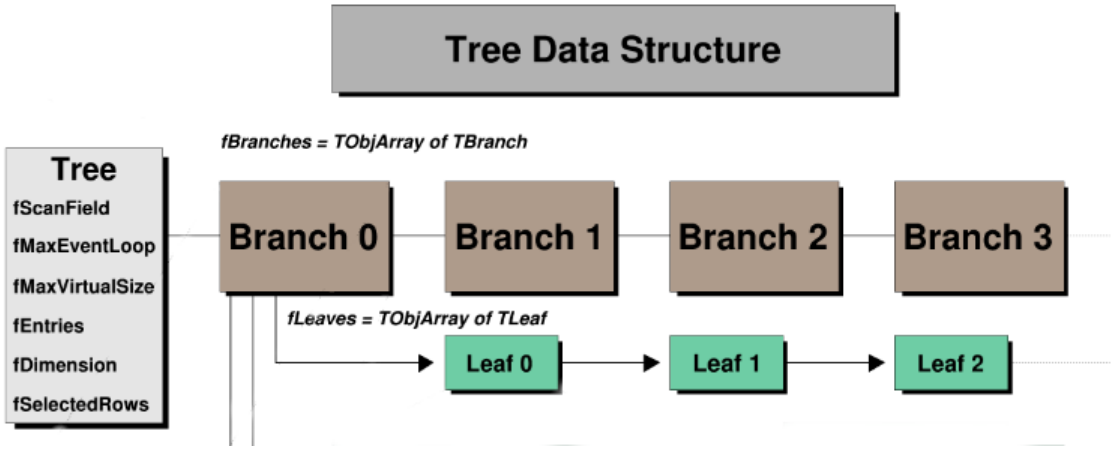


Figure 3.3: TTree structure *https://root.cern.ch/doc/master/classTTree.html*

SAMPIC data is identified by the Event number which corresponds to the entry number in *TTree*. It is important to consider that data fields could be associated with any existing layer and are not restricted to the *leafs*. Each entry contains a number of active channels and the trigger time for the event. Identifiers of active channels, cell buffer info and timestamps are all stored on *channel* level and can therefore be interpreted as integer arrays. Amplitudes are stored on the *sample* level and can be conceptually interpreted as a 2D array of double precision floating point numbers.

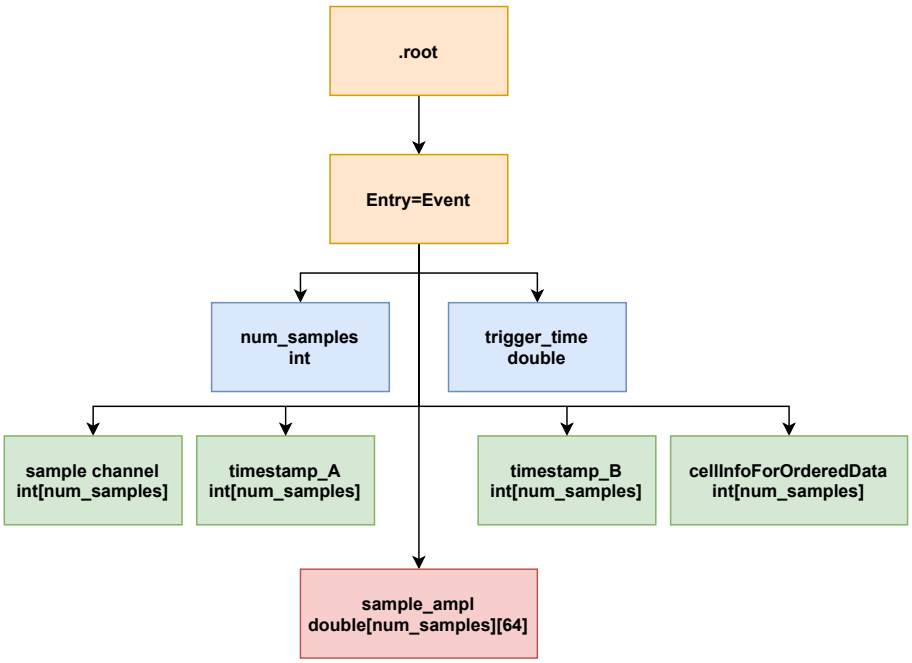


Figure 3.4: SAMPIC digitized data

Conversion algorithm

Producer module in CMSSW is a module that allows to include custom defined C++ objects into the processed event structure. The conversion algorithm could be therefore split into several sections. First one, the configuration layer is responsible for establishing the environment with correct parameters and constant values. The second part, the initialization processes the input parameters, launches the producer, and loads data from *root* files. The last section includes core logic of the data processing and construction of the corresponding objects. Those objects could be then directly attached to the event and processed by later modules. In case of SAMPIC compression, two objects are created - *TotemTimingEventInfo* which contains event level metadata and *TotemTimingDigi* which stores the channel's and sample's data. During the conversion process, several parameters are modified to match the output structure properties. In addition, the input channels are remapped according to the static mapping delivered from the configuration layer (see fig 3.2).

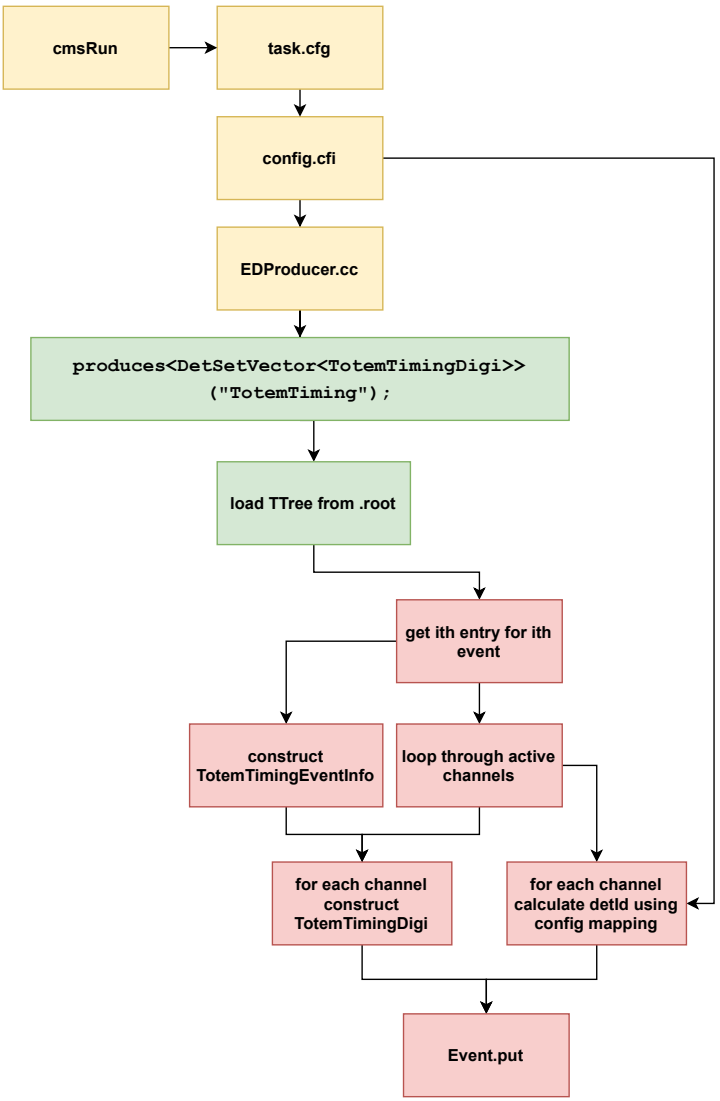


Figure 3.5: Flowchart of SAMPIC conversion algorithm

3.2. Reconstruction

Keeping the differences between Diamond and UFSD [18] detectors in mind, the UFSD Reco modules had to be prepared for receiving and processing both data streams. Changes to the reconstruction stage were applied to several files to ensure compatibility with both UFSD and Diamond workflows. The most important changes to the workflow are mentioned below.

Hits producer

totemTimingRecHitsProducer is the registered module which loads external parameters and controls the main flow of the rechits production. To create the output structures, it invokes the *totemTimingRecHitsProducerAlgorithm*. Modifications to this module were limited to changes within the parameters set.

Hits producer algorithm

totemTimingRecHitsProducerAlgorithm is responsible for creating *recHits* structure, objects that describe position of the particle within a detector. First modification was to generalize the detector identifier, which was done by converting all specialized objects to the base class which represents the geolocation data uniformly across all readouts as a 32 bit integer number (similar to fig 3.1).

Second adjustment was more conceptual in nature and involved inverting the acquired signal to match the one coming from UFSD. Additionally, a new parameter *sampicOffset* was introduced through the top level Python configuration system which allows to manually alter baseline of the amplitude.

All of those changes lie in line with the established guidelines, which require backward support for the legacy UFSD workflow. Because both Diamond and UFSD detectors derive its identifiers from a shared base class, there was no need to further separate the code and the generalized algorithm was fully capable of processing both data streams.

Track fitter

Track fitter *totemTimingLocalTrackFitter* is a module which reconstructs the particle tracks within detector by processing *recHits* data coming out from the hit producer. Changes to the fitter were mostly technical in nature and involved creating a class template and two explicit class instances to account for the differences in the Diamond and UFSD processing. This solution was preferable to duplicating existing files and separation of both modules due to the high degree similarities between the two. It would be tedious to maintain two instances of the same algorithm considering that every future alteration would have to be repeated across different code variations. Using C++ template and explicit template instantiation mechanism, this task was carried out without explicit code duplication. Generic class instances are specialized to UFSD and Diamond workflows at compilation time, producing two separate assembly objects. Those instances are then registered as CMSSW modules with unique names, which could be invoked at the Python configuration level.

Additional change was made to incorporate the second Diamond station into the reconstruction workflow. It proved necessary because up until now all detector identifiers were explicitly listed in the module's source code. It was decided that it would be better to change this mechanism in a way that allows the data to decide which detectors should be considered during the

reconstruction phase, it was possible to achieve this by utilizing the detector metadata stored in *recHits* object of each event.

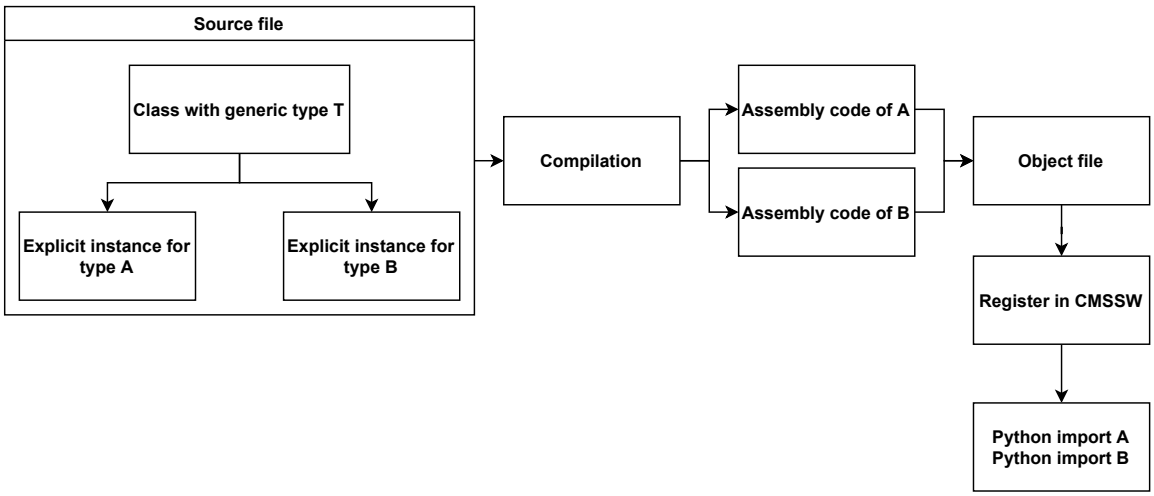


Figure 3.6: Simplified description of template instantiation mechanism in CMSSW

Changes to configuration

Changes to the configuration involved modifying the standard reconstruction sequence to run Diamond SAMPIC workflow which become the default flow for the LHC Run3. UFSD sequence is considered a legacy with its code disabled across standard CMSSW sequences.

3.3. Calibration

There are several types of calibration which should be considered for the Diamond detectors operating under SAMPIC readout:

- ADC calibration - sequence of offsets to the signal amplitude produced outside the main workflow and applied during reconstruction
- Channel alignment - timing offset for each channel produced by the PCL and applied after ADC
- Channel resolution - timing precision produced by an offline workflow and attached to recHits

Channel alignment

Channel alignment is a crucial calibration which synchronizes all data acquisition channels. Without this correction, it would be impossible to reliably reconstruct particle objects as the detection times would prove different from the actual particle arrival.

Channel alignment calibration for Diamond SAMPIC was developed as a part of Prompt Calibration Loop. PCL is an established workflow running in Tier-0 Express step of the online data acquisition system. The main Diamond detector readout (HPTDC) is also run by the PCL sequence, which makes it only natural to include SAMPIC calibration in this mechanism. The channel alignment calibration works in an iterative manner as follows:

1. Load initial calibration data from database (only ADC calibration is provided)

2. Calculate calibration

3. Update existing calibration in database (Prompt)

This is also the exact processing used in the PCL, which processes the automatically generated partial dataset and updates the database record with the calculated results.

Implemented solution consists of two main modules: Worker and Harvester. Worker module is responsible for producing and saving time distribution plots which are then processed by the Harvester. The idea is to have multiple Worker instances working in parallel and producing partial plots which could be then aggregated and loaded into a single Harvester instance.

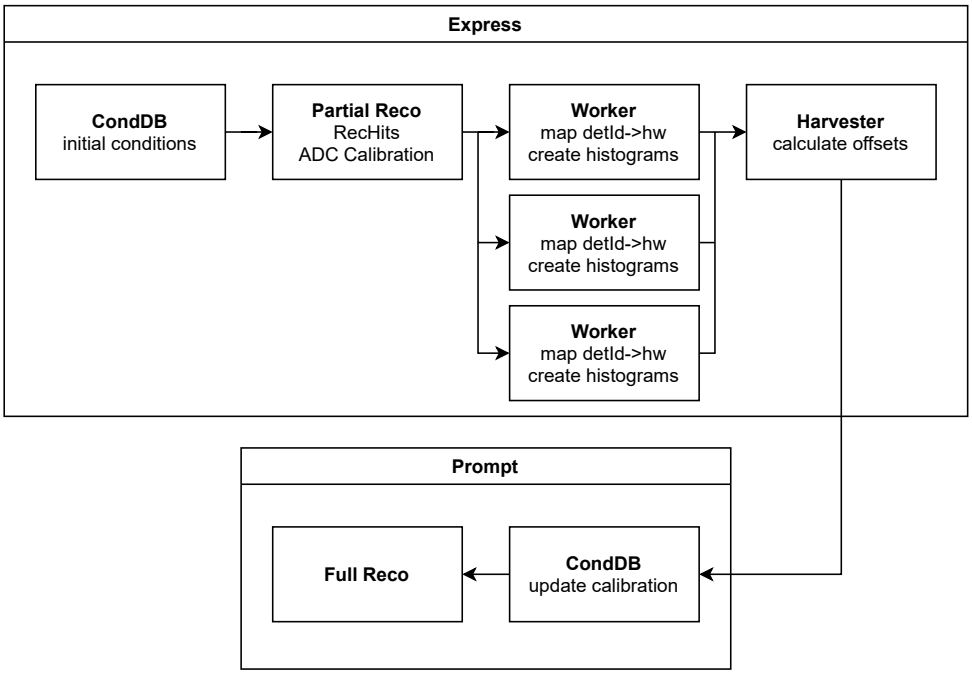


Figure 3.7: Channel time alignment in PCL

Aside from producing time distributions, Worker modules are also constructing a map which links hardware identifiers to detector identifiers. The hardware information is extracted by looping over the Digi data. This mapping could be then decoded with the detector identifier in a Harvester. This allows to harness the power of parallel modules and ensures that all required data records can be accessed in a Harvester in a constant time.

Offsets are calculated in the Harvester module from time distributions by taking the channel's mean and subtracting it from the current calibration. The current calibration can be extracted by requesting *PPSTimingCalibration* from the event structure.

Calibration data format

Output which contains channel offsets can be saved in two formats: Json and sqlite, where Json format is usually preferred for local running and presents human readable data and sqlite output is preferred when running in a production environment with database connection. Logical structure of both formats is identical and organised by the hardware identifiers.

ADC formula represents the calibration with a set of parameters and describes the amplitude offset applied in reconstruction. Data records are organized by a structure of board identifier, SAMPIC identifier, and channel number. Those records contain channel alignment, calibration,

and per channel resolution. Each entry contains 64 cells with n parameters corresponding to the ADC formula. This format allows to load and store bundled calibrations easily and in an efficient way.

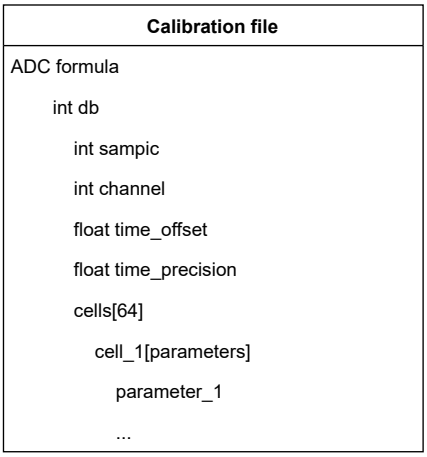


Figure 3.8: Calibration file structure for the Diamond detectors

Database connection

During calibration procedure, algorithm downloads the initial configuration and uploads the one that was updated. This is done by establishing a connection to the CERN's Conditions Database (CondDB). This database stores all conditions in which CMSSW modules are running. Those conditions are often grouped together to form Global Tags, which are whole environments containing parameters, calibrations, and execution settings. Global Tags are split into execution stages of Tier-0 (see fig 2.3).

Channel alignment calibration interacts with *CondDB* at two stages, at the beginning, when it downloads the initial calibration values from the Express tag and at the end when it uploads the results to the Prompt tag. It is important to observe that the calibration doesn't update the same tag but rather utilises two different tags lying at different processing stages (see fig 3.7). The initial calibration is therefore never updated through automatic workflow but rather adjusted manually when necessary. Prompt tag however, is overwritten by the automatic workflow and therefore always contains the latest, most precise calibration which can be later used in a Prompt reconstruction.

3.3.1. Validation output

Aside from delivering channel offsets, Harvester module is also producing channel-by-channel validation plots. Those plots show time distributions with a new calibration already applied, which makes it easy to spot any inconsistencies in the data which may point to possible hardware failure. Experts will rely on such plots to react to the detector issues. In the standard workflow, channel alignment is applied as part of the calibration sequence in reconstruction. By allowing to review the calibration effects before it is applied, one can choose to rerun the calibration procedure or correct the offsets before the invalid data are upload to reconstruction.

Given the running conditions of PCL, it was important to keep the memory and processing power requirements to the minimum and therefore validation histograms could not be filled in a Harvester module as this would require additional memory allocation and refilling of all histograms entries. Taking those limitations into account, another approach was considered. Time

distribution structures are already being allocated and filled by the Worker instances and then merged in a pre-step of Harvester execution. This allows to apply the offsets to already existing distributions without additional memory allocation. Unfortunately, those shifts cannot be applied directly as this would involve some heavy operations, such as moving every histogram bin. Solution to this issue was not to modify the data but rather scale the time axis itself so it corresponds to the calculated shifts. This approach was only possible due to the fact that entries in the validation plots are offset by the same fixed scalar value for all data of one channel within the same run.

Manual calibration module

Additional module was developed to attain the need of manual calibration, which allows to apply an arbitrary calibration payload to the given dataset outside of the reconstruction stage. It was implemented as an Analyzer which outputs channel-by-channel and aggregated time distribution plots. Differently to the PCL Harvester it is performing a direct shift on the data which is a costly operation, but given that the module is running offline, this is not a major issue. On the positive side, such outputs can be then processed by a different analysis.

Existing of such modules proved necessary due to the fact that the PCL can not deliver aggregated data comparisons, which may prove useful when checking for inconsistencies and evaluating calibration quality.

Testing scenario

Testing the channel alignment code proved to be difficult because nor LHC SAMPIC data, nor Global Tags in the database existed at that time. Considering this, a unique procedure was designed and put into place. Usually PCL workflows can be tested automatically but given the circumstances, the module had to be validated using a manual sequence. To address the Global Tag issue, two tags were uploaded into the CondDB, one with the Express destination and one with the Prompt, both containing a precalculated payload which evaluates to zero calibration. Having those test tags in place, the Worker module was fed with simulated data coming from the converter module (see fig 3.9).

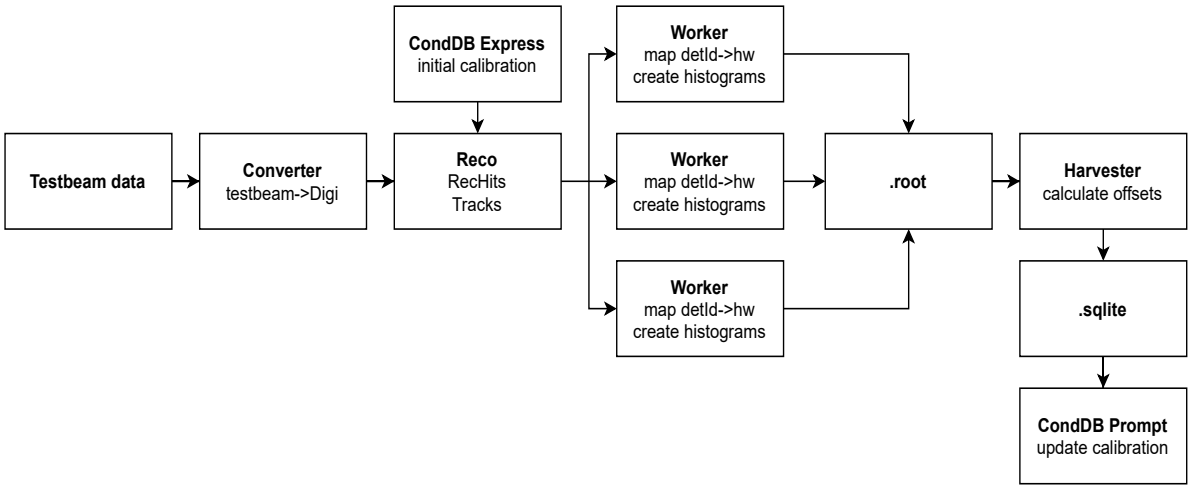


Figure 3.9: Testing sequence of SAMPIC channel alignment

Integration with PCL

Worker and Harvester modules with their corresponding parameter configuration are attached as one of the PCL's workflows. This is done by registering such a workflow in the Python configuration structure. Specified workflows are automatically run during Tier-0 Express stage.

3.4. Data Quality Monitor

Data quality modules provide a fast and reliable mechanism to evaluate various outputs by presenting them with a set of histograms. In terms of SAMPIC readout, two DQM modules were developed and included into the standard workflow.

From the implementation and structural perspective, both components share the same core programming and the differences between the two can be incorporated using flow control mechanisms. Taking this into account, SAMPIC DQM was designed as a single class with a boolean parameter controlling which workflow is currently allowed to run.

DQM in CMSSW

CMSSW provide base class *DQMOneEDAnalyzer* which can be used to extend and build upon. The class itself delivers some basic utility capabilities, such as histogram booking, which allocates memory for the required histograms. More importantly, inheriting from this module guarantees thread safety operation by preventing concurrent runs from occurring while, at the same time, allowing for concurrent lumi sections to be processed.

Dynamic structure generation

SAMPIC DQM was designed to minimize the number of hardcoded and configuration passed values by utilizing meta information coming directly from the event stream. That way, the histograms are allocated and filled only for the detectors that are present in the data, which reduces memory and processing overhead. Additionally, module uses *Geometry* records to extract and apply correct shifts and positions when histograms are being filled with data.

This dynamically generated histogram structure is also reflected in *root* output file. All histograms are hierarchically organized and automatically labeled with detector metadata, such as identifiers, which allows to distinguish and localize different sectors, stations, pots, planes, and channels.

Online DQM

Online DQM module is focusing on delivering hardware-related validation. It's expected to run with minimal memory and processing resources due to the fact it's executed in the Express stage of Tier-0 and as such is responsible for delivering the first feedback after the reconstruction stage.

Online DQM module is processing data and creating the histograms in a hierarchical manner. At each level, a different scope of data is being analyzed which allows to create aggregated plots by utilising already processed, lower level data. In the SAMPIC DQM, we can distinguish the following layers of processing:

- Global - which produces aggregated plots from all detectors
- Sector - which produces aggregated plots from all detectors within designated sector

- Pot - which produces aggregated plots from a single Roman Pot
- Plane - which produces aggregated plots from a single detector plane
- Channel - which produces plots from a single channel

At each level, the corresponding plots are being filled either from the data stream itself or the compatible histogram at the lower layer.

Offline DQM

Offline DQM follows the same pattern designed for the Online DQM with few important changes. Offline module runs in the Prompt stage of Tier-0 and therefore is more focused on the data itself and not on the validation of the hardware operation. Considering this, Offline DQM don't need to provide plane or channel level validation and therefore those plots could be dropped. This means that the module is in reality producing a subset of the histograms produced by Online components. While this is true, one has to remember about the different purposes that the two modules serve. DQM Online processes only a fraction of data, hence it runs in Express, whether Offline DQM processes the entire dataset.

Testing scenario

SAMPIC DQM module was tested in a scenario where all incoming data were simulated using the converter described in 3.1. All resulting histograms were validated against their counterparts coming from HPTDC DQM.

Additionally, due to the heavy strain that histograms put on the resources, the memory consumption was measured. The measurements were taken using *Valgrind Massif* heap profiler tool which is capable of recording memory operations, creating allocation graphs, and identifying memory leaks. A memory snapshot was taken during DQM Online operation (which generates a full set of histograms) showing allocation requirements below 150MB.

Integration with CMS systems

Both Online and Offline parts of the SAMPIC DQM were included in the standard DQM workflow using the top level Python configuration structure. This allows the module to be run in Tier-0 environment. It is also possible to load the resulting plots automatically into the DQM GUI which allows to display GUI layout with the histograms. DQM modules can be also used within CMSSW in offline workflows in remote or even local environments.

4. Work organization

The purpose of this chapter is to describe the project management methods including planning and realisation steps.

4.1. Project aspects

The project was a part of joint collaboration between AGH Institute of Computer Science and the CMS-PPS. As a part of this collaboration, the author was registered as a CERN User and was given access to restricted tools and resources. The author worked both remotely and on site

with the CERN experts to ensure the quality of the product. The project was done within the CMMSW framework and as such it followed the DevOps methodology, including Continuous Integration and Continuous Deployment practices of CMS. Even though this project was developed by a single person, it wouldn't have been possible without the support of others, special thanks are due to:

- Leszek Grzanka - thesis supervisor and one of the people responsible for AGH-CMS collaboration, his technical and administrative supervision allowed for the continuous development of the project both remotely and on CERN site
- Valentina Avati - on site supervisor and the software expert, who has been of invaluable help to understand the requirements of the project, the software structure and work ethics of the CMS-PPS
- Edoardo Bossini - on site supervisor and the hardware expert, responsible for the Diamond detectors and the new readout, his contribution in resolving the software and hardware issues and validating the results was invaluable for this project

4.2. Component design and implementation

Design and creation of each of the software components was divided into the following steps:

1. Understanding the underlying concept and component's role in a system
2. Setting up requirements and guidelines - in particular resources constraints, backward compatibility and production environment
3. Source code analysis - including reviewing existing, similar components and designing overall structure
4. Gathering and preparing necessary data - in particular preparing datasets structure, data cleaning and format parsing
5. Initial design review - covering base structure and proposed software solutions
6. Implementation and validation - including full code implementation alongside with user side configuration and initial results validation
7. Final review and deployment - including code review, unit and integration tests and deployment to CMSSW framework

All of those stages were done in close collaboration with CERN's physicists and software developers, where the most important concepts were often discussed directly in person or using various VOIP communication systems. Taking into account the wide effect range of the project, it was often necessary to communicate and synchronize the work with multiple teams responsible for different parts of CMSSW codebase. Cooperation was also crucial to finding and resolving errors that arose in the validation stage.

Deployment and validation procedures in CMSSW are streamlined using *github* and *jenkins*. Before creating a pull request to the main branch of CMSSW, changes and results were presented to and accepted by a group of experts during a scheduled, recurring meeting. Validation stage consisted of the code review and automatic tests, after which changes were signed and merged with the software's main branch.

The reason for the presented work structure is twofold. Firstly, it lies in accordance with CERN's guidelines and requirements and secondly, it implements a reliable mechanism for spotting and correcting conceptual errors relatively early on. Additional advantage is that while the validation and deployment procedure can take a long time, it doesn't delay the development of other modules, which is often independent and could be done in parallel.

4.3. Project schedule

Work schedule for this project consisted of a general schedule and individual component development schedules.

The general schedule was prepared during the initial phase of the project and consisted of components described in fig 2.3 with approximated delivery dates which were estimated using PPS timeline for LHC-Run3 preparations and the development plan of CMSSW.

Individual component schedules were set as an initial phase of the component design process, which allowed them to be much more flexible and take into account the dynamics of continuous CMSSW development. Those schedules often followed deadlines set by the release dates of a framework's snapshots.

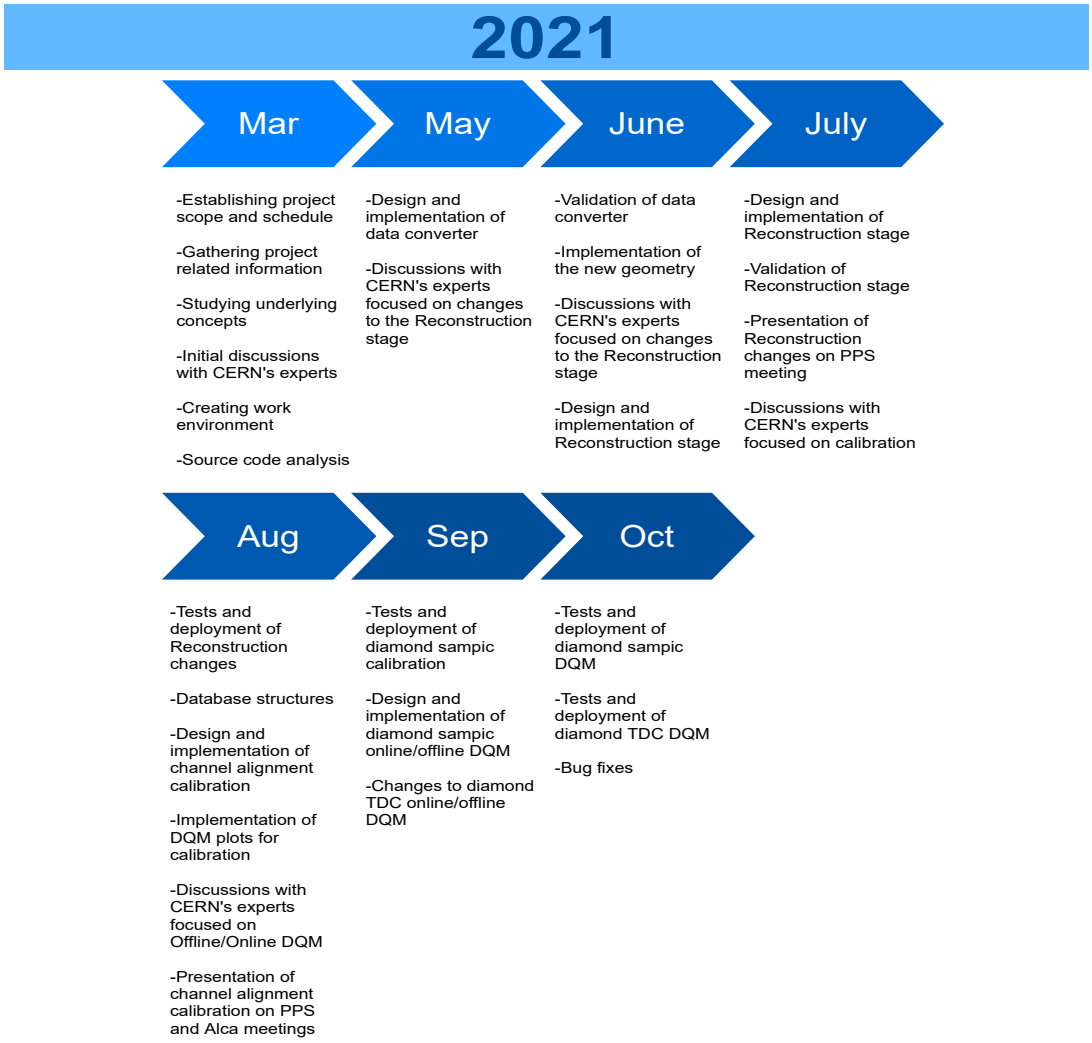


Figure 4.1: General schedule of the project

4.4. Codebase organization

Codebase was organized using *github* and *git* VCS. Repository was created as a fork of CMSSW with each component's code split into a separate branch. To ensure that the project is compatible with the main branch, its contents were periodically rebased to the newest framework's release.

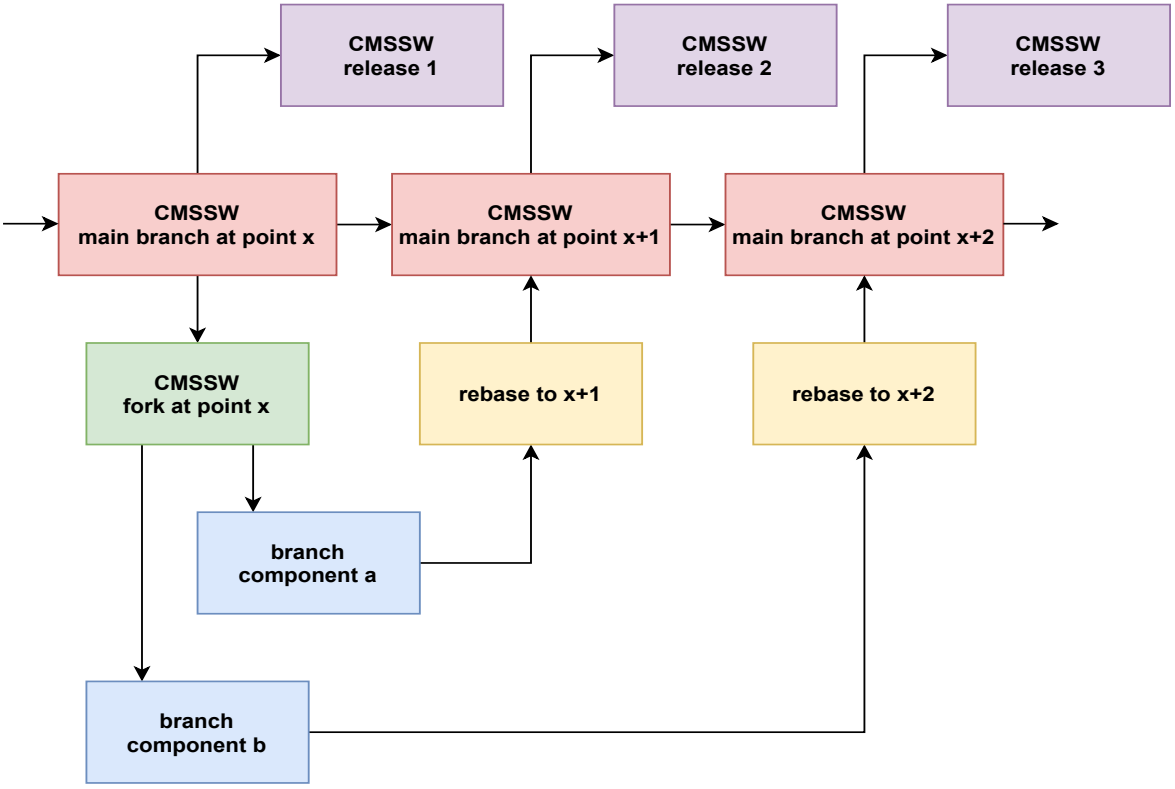


Figure 4.2: Codebase organization in relation to CMSSW

4.5. Future work

Future work was defined at the start of a project as a set of possible features and analysis that, due to the limited time period can not be considered part of a project's scope. Those possible future tasks are as follows:

- Prepare new DQM GUI layouts for diamond SAMPIC data - this feature is in development but is considered low priority and is not a part of the initial project's scope
- Add additional calibration to diamond's reconstruction stage in HPTDC workflow to account for nonlinearities - this component is in development but is not considered a part of a project
- Test software with real raw diamond SAMPIC data - this evaluation is planned to occur after detectors installation at LHC (early 2022)
- Compare HPTDC and SAMPIC readouts - this analysis is planned to take place after all necessary data will be gathered (early 2022)

4.6. Component development history

Date	Task/Event	Additional description
23.02.2021	Technical discussion with CERN experts	Meeting focused on waveform simulation and digitization.
24.02.2021-17.03.2021	Initial concept and data analysis	-
18.03.2021	Technical discussion with CERN experts	Meeting was focused around thesis work schedule and implementation of input data generator module. Especially the issue of different detector geometry (Diamond vs UFSD) was considered.
23.03.2021	Meeting with CERN developers	Presentation of the initial data converter concept to CERN developers. Meeting was focused around the different geometry issues (Diamond vs UFSD) in the data structure. Possible mapping was discussed. The module schema was agreed and approved. Tree operations efficiency was discussed.
27.03.2021	Meeting with the thesis supervisor	Meeting focused on the general scope of the project and technical documentation.
06.04.2021	Discussion with CERN experts	Meeting was focused on the data generation module. Two issues were discussed: empty events in the provided root files and the structure of TotemTimingDigi which was causing issues with amplitudes printout Solutions to both issues were presented. Further work involving modifications in the reconstruction module was discussed.
07.04.2021-20.04.2021	Implementation of data converter Design of reconstruction module	Implementation and validation of the data converter. Reviewing the reconstruction code and analyzing necessary modifications.
21.04.2021	Discussion with CERN developers	Presentation of data converter and discussion related to the tests and deployment.
21.04.2021-17.05.2021	Implementation and validation of reconstruction module	-
10.05.2021	Discussion with CERN developers	Analyzing issues with incorrect amplitudes returned from reconstruction module. Signal was inverted to solve this problem.
18.05.2021	Discussion with CERN developers	Presentation of the reconstruction module and discussion related to the tests and deployment.
19.07.2021	PPS Offline meeting	Presentation of the reconstruction module and obtained results. Module was approved for pull request.
20.07.2021-14.08.2021	Code cleanup and bug fixes	Code was formatted to fulfill CMSSW guidelines.
14.08.2021	Reconstruction module was merged into CMSSW	Pull request was approved by the review team [13].

Table 1: Reconstruction module - development history

Date	Task/Event	Additional description
06.04.2021-10.07.2021	Review the initial concept and analyze existing information	-
11.07.2021	Meeting with CERN experts	Discussion focused on PCL system and its role in diamond detector calibration.
12.07.2021-15.07.2021	Review and modify existing source to take into account second detector station	-
12.07.2021	Meeting with CERN experts	Discussion focused on ADC and channel alignment calibration. Conceptual design for the calibration module was also discussed.
13.07.2021-18.08.2021	Design and implementation of PCL workflow	Module consisted of Worker, Harvester, DQM and top level configuration.
10.08.2021	Meeting with CERN experts	Discussion was focused on addressing the database payload issues. At the time, the conditional database was lacking appropriate records to perform any live tests with remote payloads. The concluded solution was to create additional records for the test purpose.
11.08.2021-26.08.2021	Implementation of database payload transfer. Creation of a new database record. Upload of new initial calibration payload.	-
20.08.2021	PPS General Meeting	Diamond SAMPIC channel alignment workflow was presented at PPS General Meeting. Module was accepted for pull request.
26.08.2021-05.09.2021	Maintain pull request, implement the suggested changes	Few issues were spotted during the review process. Those issues had to be appropriately addressed.
30.08.2021	AlcaDB meeting	PCL workflow and database relation was presented at AlcaDB meeting. Tests and deployment were discussed.
06.09.2021	Diamond SAMPIC channel alignment workflow was merged with CMSSW	Pull request was approved by the review team [5].

Table 2: Calibration module - development history

Date	Task/Event	Additional description
20.04.2021	Meeting with CERN experts	Meeting was focused around the reco module which is the second stage module in the workflow, we have discussed possible issues with calibration config files and talked about using dqm module for output validation.
27.04.2021	Meeting with CERN experts	Discussed issues with Data Quality Monitor and possible modifications to its source code.
27.04.2021-11.05.2021	Adapt diamond's DQM for use in diamond SAMPIC reconstruction workflow	-
11.05.2021	Meeting with CERN experts	Meeting was focused on the results delivered by Data Quality Monitor and reviewing the possibilities of adding additional plots containing proton tracks.
12.05.2021-01.06.2021	Add new geometry to SAMPIC DQM	Access new geometry shifts from the associated event structure. Modify the plots to use geometry data.
01.06.2021	Validate the compatibility with Run3 geometry, include the second station in DQM	Modify the plots by adding a second station, validate the geometry using DQM output.
13.07.2021	Validate changes to geometry	Create validation plots with SAMPIC DQM.
24.08.2021-03.09.2021	Review existing diamond's DQM module	DQM of diamond TDC had to be re-evaluated for Run3.
04.09.2021-08.09.2021	Implementation of diamond SAMPIC DQM	Additional track plots and track correlation plots were implemented on top of a standard TDC plots. Module was validated with a data converter.
09.09.2021-17.09.2021	Modifications to diamond TDC DQM	Plots cleanup, additional track and correlation plots.
18.09.2021-27.09.2021	Online/Offline switching and validation	Plots were split to Online/Offline DQM and validated with data converter (SAMPIC) or real data (TDC).
08.10.2021	DQM-DC meeting	DQM components were presented during DQM-DC meeting and were approved for pull requests.
18.09.2021-15.10.2021	Working with review teams to solve all issues, code cleanup.	Several issues related to the configuration had to be addressed before DQM components were validated.
14.10.2021	diamond SAMPIC DQM was merged into CMSSW	Pull request was approved by the review team [9].
15.10.2021	diamond TDC DQM was merged into CMSSW	Pull request was approved by the review team [8].

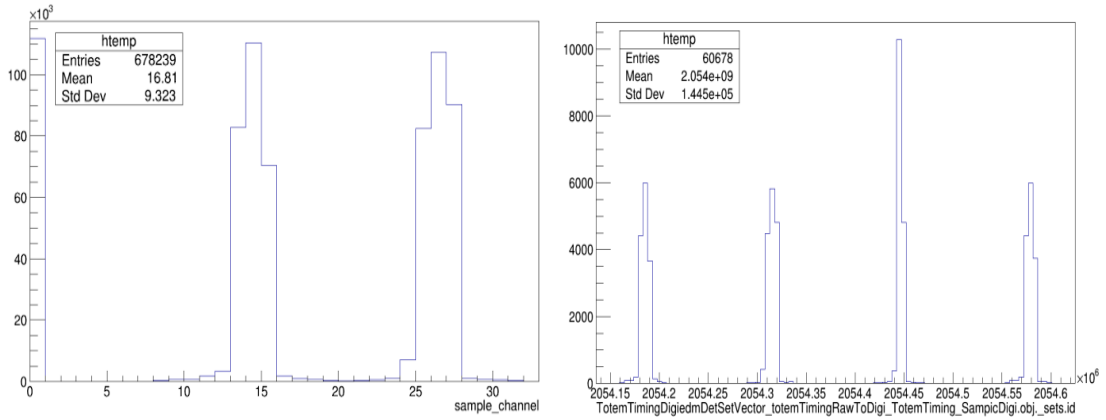
Table 3: DQM module - development history

5. Project results

The purpose of this section is to present the project's results including output validation and performance measurements. Additionally, this chapter aims to describe the existing project's limitations and related possible future improvements.

5.1. Testbeam data converter

Data converter output was tested using modified *TotemTimingDQM* which delivered validation plots for the testbeam digi.

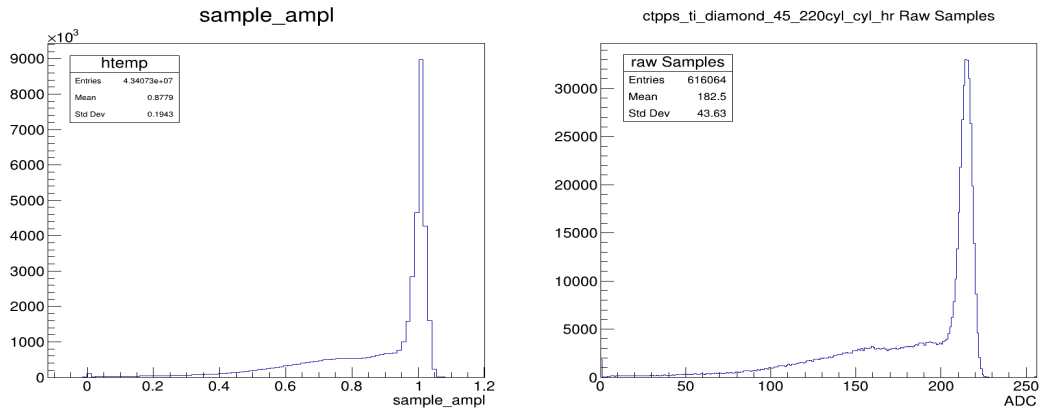


(a) Input entries per channel

(b) Output entries per plane id, filtered partial dataset

Figure 5.1: Testbeam converter validation plots

Plot fig 5.2(a) show active channels in the input data which are then duplicated following schema fig 3.2. Histogram fig 5.2(b) presents the active planes in the output structure. Using those id values, it is easy to cross-validate the outputs against the provided mapping and confirm its correctness.



(a) Input amplitude [V]

(b) Output amplitude [ADC counts]

Figure 5.2: Amplitude plots

Outside of the output logical structure, the results content were also validated by comparing input-output values.

5.2. Reconstruction

Reconstruction was validated using *DiamondSampicDQM* outputs coming from new *DiamondSampicLocalReconstruction* workflow. Schema fig 5.3 describes how the crystals and channels are aligned on the detector board. Figures fig 5.4 and fig 5.5 show representative plots coming from reconstruction. To test the reconstruction process, a module was run with data coming from different crystals. It is important to observe the relation between channel and crystal mapping and the corresponding reconstruction plots. All plots below present either entries in the x[mm] axis of the detector board or entries in the channel pad. It should be noted that the distribution of those entries closely follows the presented channel mapping and crystal position on the board.

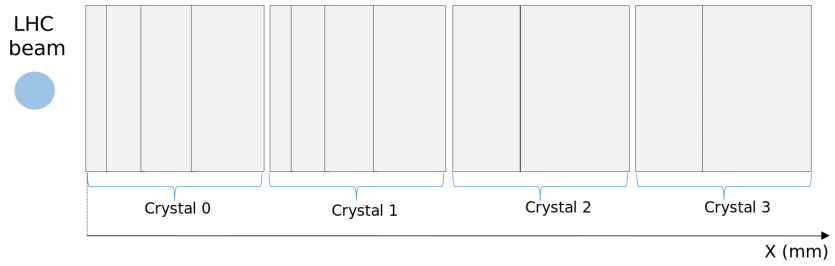
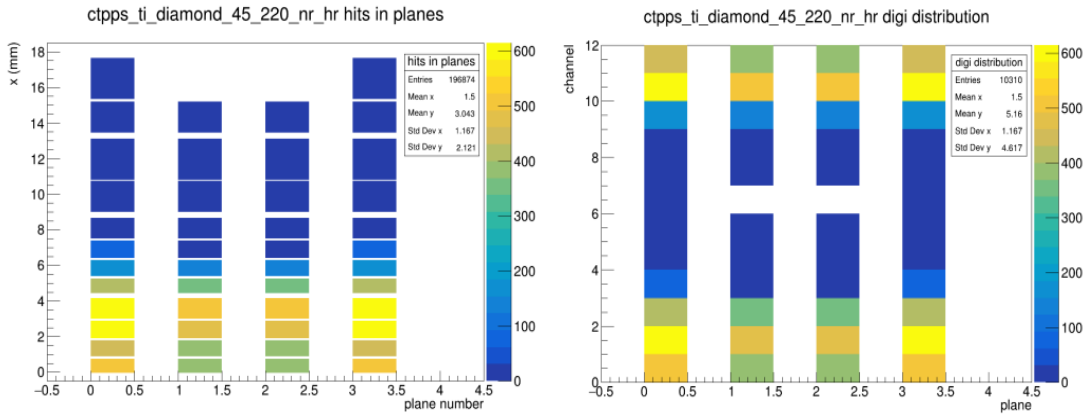
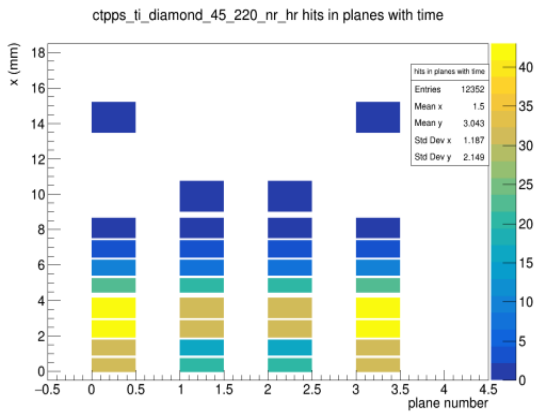


Figure 5.3: Diamond detector channel indexing

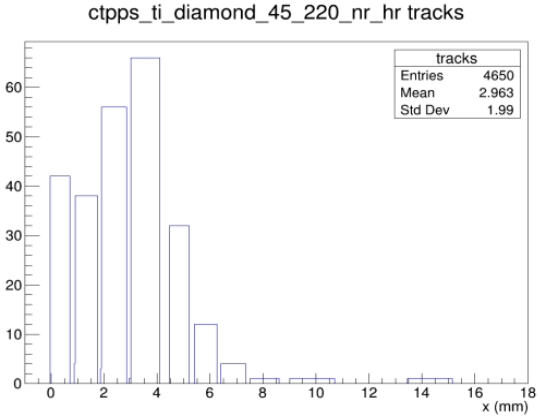


(a) Detector hits in planes

(b) Digi distribution in planes

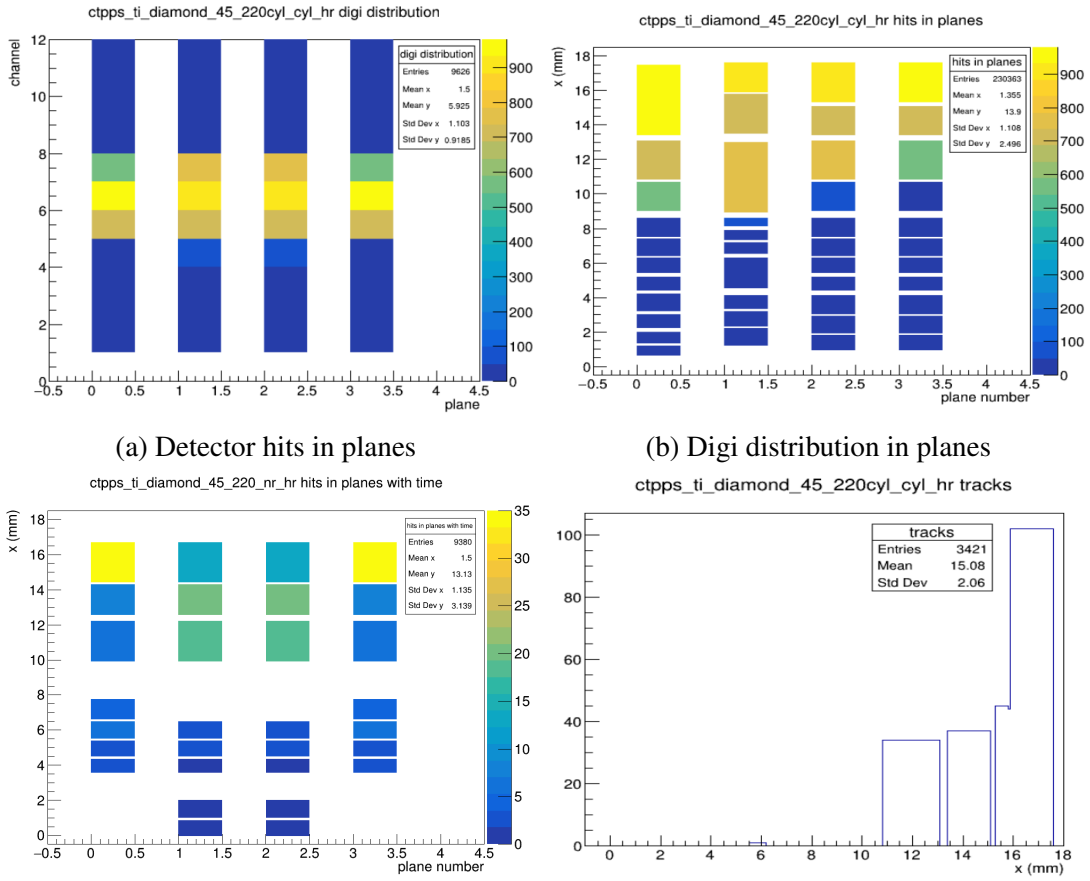


(c) Detector hits with time



(d) Tracks in x axis

Figure 5.4: Reconstruction results for testbeam data, with crystal 0 exposed to the particle beam



(a) Detector hits in planes

(b) Digi distribution in planes

(c) Detector hits with time

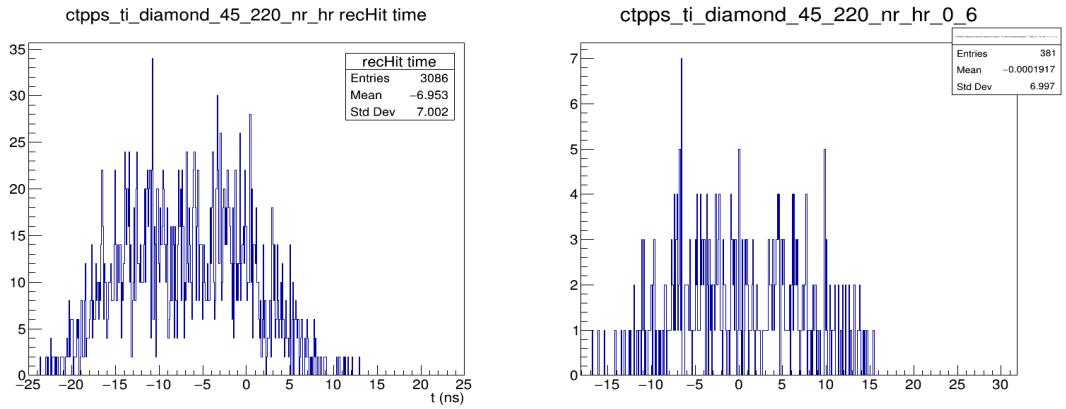
(d) Tracks in x axis

Figure 5.5: Reconstruction results for testbeam data, with crystal 1 exposed to the particle beam

5.3. Calibration

Calibration was validated using the approach described at fig 3.9. Obtained results were compared with the corresponding plots coming from uncalibrated *Reconstruction*. Shift in the timing distribution can be observed at fig 5.6. It is important to mention that PCL workflow does not have capabilities to produce aggregated data histograms like the one coming from DQM (a). Instead, it is utilising axis shift to present per channel histograms (see 3.3.1). Channel alignment PCL of SAMPIC was also tested with *performance suite* to evaluate RSS memory requirements displayed at fig 5.7.

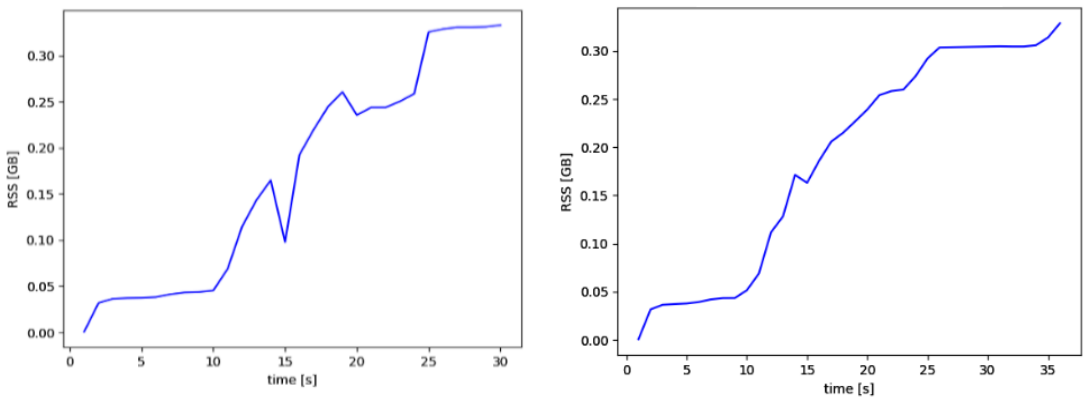
The main limitation of this PCL workflow is the workload done by workers who have to fill in large quantities of histograms. This results in relatively long processing times. Thankfully, those operations are independent and can be computed in parallel. At this moment, there are no further improvements planned for this workflow as both memory and time requirements were met.



(a) Uncalibrated time distribution of channel 6, full dataset

(b) Calibrated time distribution of channel 6, reduced dataset

Figure 5.6: RecHits time distribution



(a) Worker RSS memory

(b) Harvester RSS memory

Figure 5.7: Memory requirements of PCL workflow

5.4. DQM

Data Quality Monitor was validated against the counterpart Diamond HPTDC output. Both modified modules were validated in Online and Offline modes. Given that Offline histograms are a subset of the Online output, all presented representative results fig 5.9 were recorded in Online mode.

The main limitations of the DQM modules are high memory and processing time requirements. All histograms are filled entry by entry and memory consumption strongly depends on the resolution of the plots. On the other hand, given that all DQM outputs are histograms, the memory requirements do not scale with the amount of input data. General PPS DQM memory tests were conducted during the deployment stage using *Massif* and yielded a result of 583.7 Mb of total memory heap consumption with 145.1 Mb directly assigned to SAMPIC DQM.

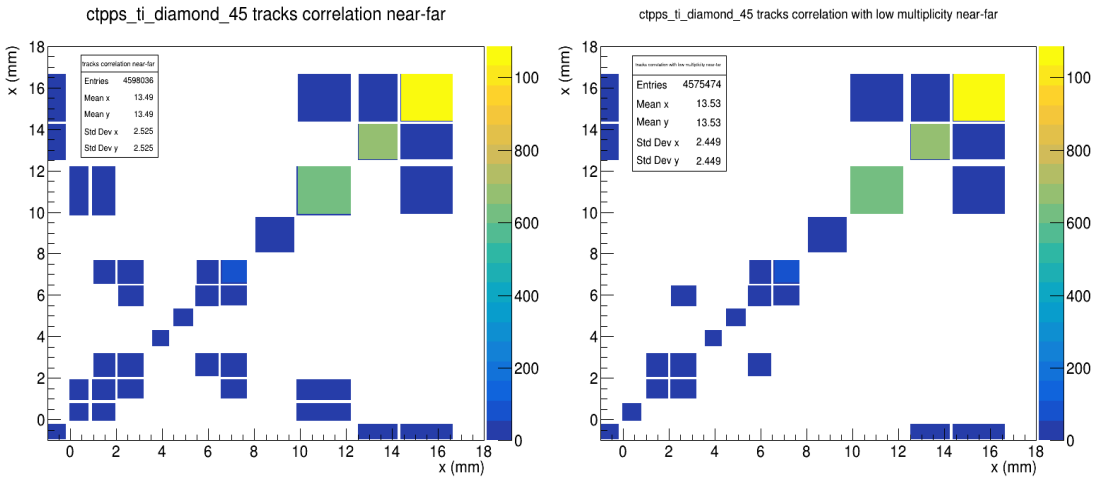
Further future improvements to those modules include new DQM GUI layouts and additional plots concentrating on per lumisection data. Those plots are still investigated as they could pose a problem when running in the distributed parallel environment of CMS Tier-0.

Track correlation

Tracks correlation plots deliver information about the matching between the two detector stations placed on the same side w.r.t. the IP5. The plots are aggregated for the data within each sector. Histograms are filled with all possible combinations of two tracks x distributions. This inevitably resulted in some incorrect bins being filled due to the fact that each event can be associated with many tracks. To address this issue, an additional plot containing only tracks with low multiplicity was created.

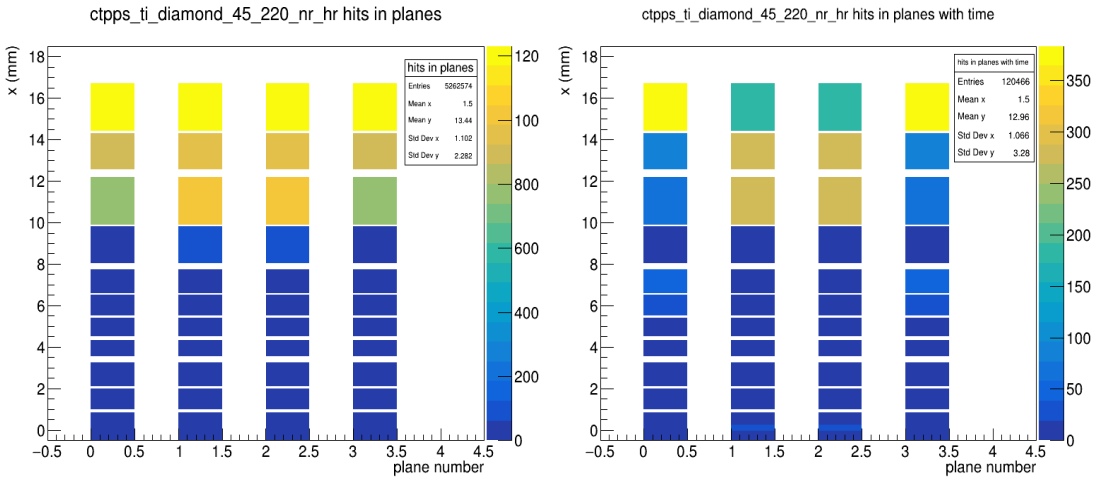
Hits in planes

Hits in planes present detected particle hits in x,y coordinates of the detector plane. The generated heatmap is limited to the detector geometry, which results in a pattern representing the individual pads of the sensor. Each pad is being displayed in real scale. Pads are of different size to optimize the channel average occupancy.



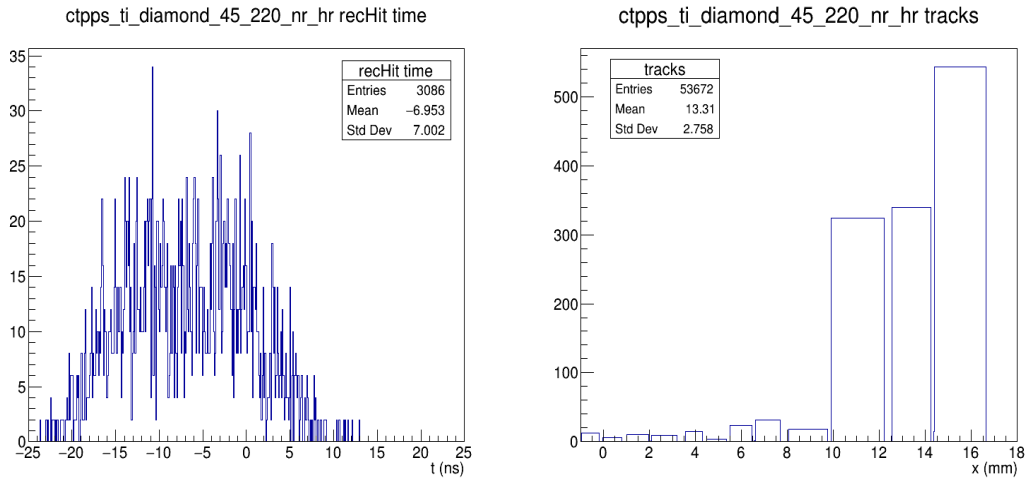
(a) Near-far tracks correlation

(b) Near-far tracks correlation, low multiplicity



(c) Hits in planes

(d) Hits in planes with time



(e) RecHits time

(f) Tracks

Figure 5.9: Examples of Sampic DQM plots produced for 100k events

5.5. Conclusions

The project development started March 2021 and was fully developed and deployed during the following months, with the last component successfully merged with the framework on 15.10.2021. All requirements have been met and the readout system is scheduled for real environment tests in the first quarter of 2022. This will most likely result in further changes to the software source code and its running configuration.

Additional works are progressing for both SAMPIC and HPTDC readouts for Diamond detectors, including new layouts for DQM GUI and HPTDC nonlinearities compensation [12]. Those features will make use of the structures and source code developed during this project.

Personally, I hold the time spent on this project in high regard. It gave me the opportunity to work for one of the most renowned research institutions in the world and improve upon my abilities in software development and project management aspects. This work offered the invaluable experience of working alongside many respected scientists, without whom this project could not be finished. I'd like to offer my special thanks to: Leszek Granka (AGH thesis supervisor), Valentina Avati (CERN supervisor), Edoardo Bossini (CERN supervisor), and all other CMS-PPS collaborators who helped with the development of this project.

References

[1] About CERN. https://home.cern/about. Accessed: 2021-12-19.

[2] About LHC. https://home.cern/science/accelerators/large-hadron-collider. Accessed: 2021-12-19.

[3] ALICE. https://alice.cern/. Accessed: 2021-12-19.

[4] ATLAS. https://atlas.cern/. Accessed: 2021-12-19.

[5] Calibration workflow PR. https://github.com/cms-sw/cmssw/pull/35029. Accessed: 2021-12-21.

[6] CMS. https://cms.cern/. Accessed: 2021-12-19.

[7] CMSSW repository. https://github.com/cms-sw/cmssw. Accessed: 2021-12-20.

[8] Diamond HPTDC DQM PR. https://github.com/cms-sw/cmssw/pull/35454. Accessed: 2021-12-21.

[9] Diamond Sampic DQM PR. https://github.com/cms-sw/cmssw/pull/35445. Accessed: 2021-12-21.

[10] EOS. https://eos-web.web.cern.ch/eos-web/. Accessed: 2021-12-19.

[11] LHCb. http://lhcb.web.cern.ch/. Accessed: 2021-12-19.

[12] Nonlinearities compensation PR. https://github.com/cms-sw/cmssw/pull/36537. Accessed: 2021-12-21.

[13] Reconstruction module PR. https://github.com/cms-sw/cmssw/pull/34759. Accessed: 2021-12-21.

[14] ROOT data analysis framework. https://root.cern/. Accessed: 2021-12-29.

[15] M. Albrow, M. Arneodo, V. Avati, J. Baechler, N. Cartiglia, M. Deile, M. Gallinaro, J. Hollar, M. Lo Vetere, K. Oesterberg, N. Turini, J. Varela, D. Wright, and C. CMS-TOTEM. CMS-TOTEM Precision Proton Spectrometer. *iNSPIRE HEP*, 2014. https://cds.cern.ch/record/175379.

[16] V. M. Azzolini, B. C. van Besien, D. V. U. m. Bugelskis, T. U. Z. m. Hreus, K. F. Maeshima, J. U. O. m. Fernandez Menendez, A. V. U. m. Norkus, J. F. F. Patrick, M. C. Rovere, and M. A. C. Schneider. The Data Quality Monitoring software for the CMS experiment at the LHC: past, present and future. *iNSPIRE HEP*, 2018. https://cds.cern.ch/record/2701776.

[17] E. Bossini. Development of a time of flight diamond detector and readout system for the totem experiment at cern, 2016. https://cds.cern.ch/record/2227688.

[18] E. Bossini. The proton timing system of the totem experiment at lhc. *Proceedings of Science*, 2018. http://cds.cern.ch/record/2710220.

[19] E. Bossini and N. Minafra. Diamond Detectors for Timing Measurements in High Energy Physics. *frontiers in Physics*, 2020. https://www.frontiersin.org/articles/10.3389/fphy.2020.00248/full.

[20] C. D. Jones, Cornell University, Ithaca, M. Paterno, J. Kowalkowski, L. Sexton-Kennedy, W. Tanenbaum. THE NEW CMS EVENT DATA MODEL AND FRAMEWORK. Technical report, CMS Collaboration, 2006. http://cds.cern.ch/record/2743738.

[21] M. Oriunno, M. Deile, K. Eggert, J. Lacroix, S. Mathot, E. Noschis, R. Perret, E. Radermacher, and G. Ruggiero. The roman pot for the lhc. In *Proceedings of EPAC:2006*, 2006. https://accelconf.web.cern.ch/e06/PAPERS/MOPLS013.PDF.

[22] The CMS and Totem Collaborations. Proton reconstruction with the Precision Proton Spectrometer (PPS) in Run 2. Technical report, CMS Collaboration, 2020. http://cds.cern.ch/record/2743738.