UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Master Degree in Physics

Final Dissertation

Upgrade of the DAQ system in the Veto Counter at NA62 and performance evaluation

Thesis supervisor

Prof. Piero Giubilato

Thesis co-supervisor

Dr.ssa Giovanna Lehmann Miotto

Candidate

Dr. Marco Ceoletta

Academic Year 2022/2023

Contents

List of Figures

List of Tables

Abstract

NA62 is a medium-sized particle physics experiment located in the CERN North Area beam facility. Its main objective is to study the ultra rare kaon decay channel PNN ($K^+ \longrightarrow \pi^+\nu\bar{\nu}$). Given the extremely low branching ratio of the signal, muon and photon vetoing is of paramount importance and most of the subdetectors in NA62 are dedicated to this function. In 2020, a new sub-detector named Veto Counter has been installed. This device is designed to reduce the spurious contributions of secondary interactions happening within the beam line.

NA62 currently uses the TEL62 digital board as the backbone of its data acquisition (DAQ) system, which is a highly improved and upgraded version of the TELL1 board originally developed for LHCb. Following the ever growing requirements of the NA62 experiment, the limitations of the TEL62-based readout are showing more and more. These include lack of radiation hardness and readout rate limitations of the aging Time to Digital converter modules used. An upgrade of the readout system using a new custom time to digital converter (TDC) in combination with the modern FELIX card developed for the ATLAS experiment is thus envisioned. The Veto Counter is equipped with a dual TEL62 and FELIX readout and is used as a test bench in the evaluation of the new readout.

In this work the performance of the Veto Counter is studied and a comparison between TEL62 and FELIX readout is performed. Furthermore a control system for the newly developed FELIX TDC is designed and tested on the Veto Counter platform.

A general introduction on the physics studied at NA62 will be given in Chapter 1. In order to fulfill its experimental goals NA62 requires a specific design; in Chapter 2 we will discuss the most relevant design choices regarding trigger and experimental structure. An overview of the main sub-detectors is given in Chapter 3 while in Chapter 4 a description of the current Trigger and Data Acquisition system of NA62 is given.

The FELIX readout upgrade effort will be detailed in Chapter 5, together with a description of the main limitations of the TEL62 readout. This upgrade is expected to involve all the sub-detectors at NA62 currently using a TEL62 system. Software design of the *FELIX control system*, developed in this work is described in depth in Chapter 6. Chapter 7 will give a description of the Veto Counter sub-detector; being a test bench for the new readout technology it features a dual TEL62-FELIX readout. Finally, the performance study of the Veto counter and its readout is presented in Chapter 8. As an aside, in Appendix C some observation on a possible use of the Veto Counter to aid in upstream track extrapolation are presented.

Chapter 1

Physics Introduction

1.1 CKM matrix and Unitary triangles

The Cabibbo-Kobayashi-Maskawa (CKM) matrix is a "unitary"[1] 3×3 matrix describing the mismatch between freely propagating quark states and weakly interacting quark states. The former named *mass eigenstates* $(d\,s\,b)$, the latter being *weak eigenstates* $(d'\,s'\,b')$.

$$\begin{pmatrix} d' \\ s' \\ b' \end{pmatrix} = \mathfrak{M} \begin{pmatrix} d \\ s \\ b \end{pmatrix} = \begin{pmatrix} M_{ud} & M_{us} & M_{ub} \\ M_{cd} & M_{cs} & M_{cb} \\ M_{td} & M_{ts} & M_{tb} \end{pmatrix} \begin{pmatrix} d \\ s \\ b \end{pmatrix} \tag{1.1}$$

$|M_{ij}|^2$ are proportional to quark flavour transitions.

The unitary nature of the CKM matrix forbids *flavour changing neutral currents* (FCNC), this means $\Delta S = 2$ weak transitions are forbidden completely and $\Delta S = 1$ are allowed only with charged currents interactions. In other words it is not possible to observe a FCNC. This holds for all three quarks families and is an extension of the so-called GIM (Glashow-Iliopoulos-Maiani) mechanism for first and second quark families.

Consider a one-loop box diagram that involves W exchanges. Even though Z^0 exchanges prohibits flavour changing neutral current (those are flavour neutral) the process admits FCNC albeit at a very small level on the scale of W mass, the level is proportional to the mass squared difference of quark masses exchanged in the box diagram[2].

A general 3×3 unitary matrix has 9 degrees of freedom; the CKM matrix may be parametrized in various ways and has ultimately 4 degrees of freedom:

KM: Uses Cabibbo's angle θ_1 together with the similarly defined θ_2, θ_3 and a CP violation phase angle δ.

Standard: Based on Euler's angles, it includes as well the CP violation phase angle δ.

Wolfenstein: This parametrization is more complex and has the advantage of being a power expansion, enabling whatever degree of precision is required by appropriate highest order.

In this introduction we will refer to the Wolfenstein parametrization that is ideal to introduce unity triangles. In the Wolfenstein parametrization the CKM matrix is written as function of four independent parameters $\lambda, A, \varrho, \eta$. The expansion is carried out as power of $\lambda = |M_{us}| \sim 0.22$.

By expanding to $\mathcal{O}(\lambda^5)$ with the reparametrization:

$$\varrho' = \varrho\Big(1 - \frac{\lambda^2}{2}\Big), \quad \eta' = \eta\Big(1 - \frac{\lambda^2}{2}\Big). \tag{1.2}$$

[1] With current best estimations of CKM values [14] there is a 3 sigma deviation from unitarity in first row!

[2] in the original GIM the $u - c$ pair, in the generalized GIM also t.

one gets

$$\mathfrak{M} = \begin{pmatrix} 1-\frac{1}{2}\lambda^2-\frac{\lambda^4}{8} & \lambda & A\lambda^3(\varrho'-i\eta') \\ -\lambda+\frac{1}{2}A^2\lambda^5[1-2(\varrho'+i\eta')] & 1-\frac{1}{2}\lambda^2-\frac{\lambda^4}{8}(1+4A^2) & A\lambda^2 \\ A\lambda^3(1-\varrho'-i\eta') & -A\lambda^2+\frac{1}{2}A\lambda^4[1-2(\varrho'+i\eta')] & 1 \end{pmatrix} + \mathcal{O}(\lambda^5) \quad (1.3)$$

This form is what will be used in the following.

As a consequence of CKM unitarity one may write:

$$\sum_k |M_{ik}|^2 = \sum_k |M_{ki}|^2 = 1 \quad (1.4)$$

where i represents a quark family. This is the *weak universality* relation and implies that all the $SU(2)$ pairs are equally coupled to the weak interaction carrier bosons.

Together with (1.4) unitarity naturally poses six more constraints on M_{ij}, that can be written as

$$\sum_k M_{ik}M_{jk}^* = 0 \qquad \forall\, i \neq j. \quad (1.5)$$

Each of these relations describes a so-called *unity triangle*: these triangles have all equal area (its value is proportional to the CP violation phase) and their spatial arrangement depends on quark field phases. Unitary triangles are invariant with phase transformations, in the sense that their area remains unchanged under rotations in quark phase space.

Lets focus on the relation

$$M_{ud}M_{ub}^* + M_{cd}M_{cb}^* + M_{td}M_{tb}^* = 0. \quad (1.6)$$

and construct its triangle in the Wolfenstein parameters plane, shown in Fig. 1.1.

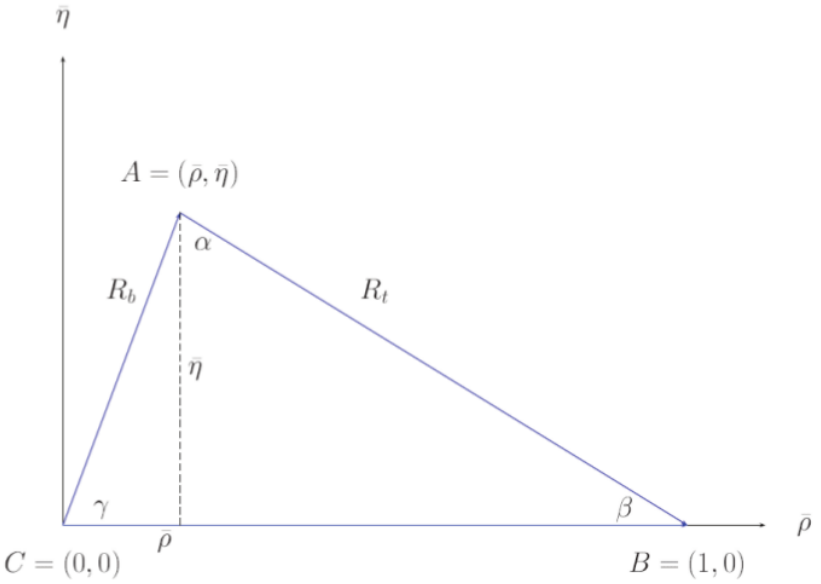

Figure 1.1: The unitarity triangle for (1.6), [1]

In this triangle length R_b, R_t are related to M_{ij} with the following

$$R_b = \frac{|M_{ud}M_{ub}^*|}{|M_{cd}M_{cb}^*|} = \sqrt{\varrho'^2+\eta'^2} = \Big(1-\frac{\lambda^2}{2}\Big)\frac{1}{\lambda}\Big|\frac{M_{ud}}{M_{cb}}\Big| \quad (1.7)$$

$$R_t = \frac{|M_{td}M_{tb}^*|}{|M_{cd}M_{cb}^*|} = \sqrt{(1-\varrho')^2+\eta'^2} = \frac{1}{\lambda}\Big|\frac{M_{td}}{M_{cb}}\Big|. \quad (1.8)$$

From this it is possible to obtain expressions for the associated angles as phases of CKM elements

$$M_{td} = |M_{td}|e^{-i\beta}, \tag{1.9}$$
$$M_{ub} = |M_{ub}|e^{-i\gamma}. \tag{1.10}$$

From (1.7) sides of the unitary triangles are related to CKM elements directly (remember $\lambda = |M_{us}|$). Our knowledge of the CKM elements comes from semi leptonic K, B decays [15] [16]:

$$|M_{us}| = 0.2252 \pm 0.0009, \quad M_{cb} = (40.9 \pm 1.1) \times 10^{-3}, \tag{1.11}$$
$$\frac{|M_{ub}|}{|M_{cb}|} = 0.089 \pm 0.012, \quad M_{ub} = (4.15 \pm 0.49) \times 10^{-3}. \tag{1.12}$$
$$\tag{1.13}$$

and enables direct computation of both Wolfenstain parameter $A = M_{cb}/\lambda^2 = 0.817 \pm 0.015$ that represents triangle area (CP violation phase) as well as $R_b = 0.40 \pm 0.06$.

The determinations of A, R_b obtained from tree decays can be used as universal constants and are always valid within SM. In order to build the unity triangle only one determination between R_t, β, γ is thus needed.

1.2 The PNN decay

In this section a very brief introduction to the rare flavor-changing neutral current decays $K \longrightarrow \pi\nu\bar{\nu}$, commonly named *PNN* decays, is given.

In flavor physics rare decays of the form

$$K^+ \longrightarrow \pi^+ + \nu + \bar{\nu}; \quad K_L \longrightarrow \pi^0 + \nu + \bar{\nu} \tag{1.14}$$

are extremely important processes because they are some of the very few theoretically clean ones. This means that the decay *Branching Ratio* uncertainty is dominated by what comes from the determination of CKM matrix elements; according to [17] PNN branching ratio is predicted to be

$$BR = (7.81 \pm 0.75 \pm \mathit{0.29}) \times 10^{-11} \tag{1.15}$$

while first contribution accounts to CKM uncertainty and *second* contribution to intrinsic SM theoretical uncertainty.

Theoretically clean processes are good probes for new physics beyond the Standard Model; their use is especially advantageous in comparison of direct testing due to their sensitivity [18].

In these decays the underlying mechanism is hard GIM so the process dynamics are short-ranged and the amplitude at short range is governed by only one semileptonic operator whose matrix elements may be completely determined experimentally. This means that it is possible to directly test the SM predictions because studying such decays yields in quantities that are directly comparable to those obtained in SM studies. Work on PNN-style decays may also deepen our knowledge of the CKM matrix. The interest in those decay modes does not stop there: given their cleanliness they are an ideal probe for beyond-SM degrees of freedom and thus new physics.

The hadronic decays presented in (1.14) are essentially a consequence of the following quark process

$$s \longrightarrow d\,\nu\,\bar{\nu} \tag{1.16}$$

and this process, according to the Standard Model is originated by a combination of a Z_0 penguin and a double W exchange.

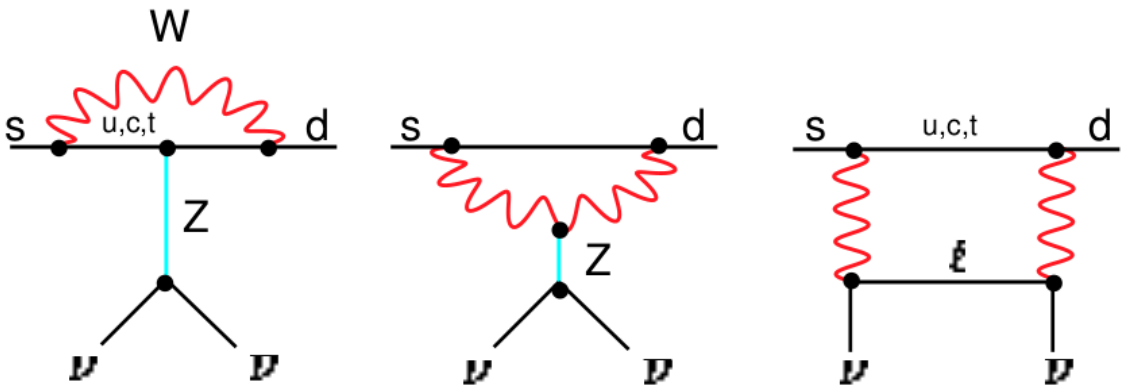

Figure 1.2: Underlying processes for the $s \longrightarrow d\,\nu\,\bar{\nu}$ according to SM [2]

Since those processes are quadratic GIM mechanism the amplitude level for the given "channel" is given by

$$A_i \approx \frac{m_i^2}{m_W^2} M_{is}^* M_{id}, \quad \text{where } i = u, s, t. \tag{1.17}$$

Here the u contribution is negligible, as well as the c contribution for K_L decay. Thus top quark dominates and the process coupling is Fermi-like and of the form

$$\mathcal{H} = \sum_{\ell=e,\mu,\tau} \frac{G_\ell}{\sqrt{2}} (\bar{s}d)_{V-A} (\bar{\nu_\ell}\nu_\ell)_{V-A} \tag{1.18}$$

where G_ℓ is the effective coupling constant.

Given G_ℓ it is possible to relate the Branching Ratios (BR) of the (1.14) to the ones of the well known K_{e3}^+

$$BR(K^+ \longrightarrow \pi^+ + \nu + \bar{\nu}) \propto BR(K^+ \longrightarrow \pi^0 + \nu + e^+)|G_\ell|^2, \tag{1.19}$$

$$BR(K^0 \longrightarrow \pi^0 + \nu + \bar{\nu}) \propto BR(K^+ \longrightarrow \pi^0 + \nu + e^+)|\mathrm{Im}(G_\ell)|^2. \tag{1.20}$$

For further details about these relations one may consult [19].

G_ℓ may be expressed as sum of two contributions, from top and charm lines respectively

$$G_\ell = \frac{\alpha G_F}{2\pi \sin^{\Theta_W}} [M_{ts}^* M_{td} X(m_t^2/M_W^2) + M_{cs}^* M_{cd} X_{NL}^l]. \tag{1.21}$$

In this expression both X coefficients have been computed in [20] [21]. The top contribution of first member has a known expression whose only source of uncertainty is the top quark mass meanwhile the charm contribution has a larger error that needs to be computed averaging on the neutrino families.

Lets consider the CKM matrix in the basic Wolfenstein parametrisation as written in (1.3).

Comparing this with (1.21) one can notice that the charm contribution depends on M_{cd}, M_{cs} and is real; for this reason it will not play a role in K^0 of (1.19), making this process extremely clean.

Moving on, it is possible to directly infer M_{ts}: in the basic Wolfenstein approximation $\mathcal{O}(\lambda^4)$ we have $M_{ts} = -M_{cb}$ and the latter can be determined precisely studying semileptonic decays:

$$|M_{cb}| = (41.5 \pm 0.8) \times 10^{-3}. \tag{1.22}$$

Since M_{cd}, M_{cs} are well known values, and M_{ts} is set by M_{cb} measuring precisely the BR of (1.14) and thus determining G_ℓ allows calculation of M_{td}, according to (1.1). This directly relates to the Wolfenstein parameters ϱ', η' that are the basis of the CKM unitary triangle.

In the unitary triangle the β angle is very well known from the theoretical analysis of CP violation in the $B \longrightarrow \psi K^0$ decay, moreover the length of that (R_t) is determined from lattice QCD studies of $B^0\bar{B}^0$ oscillations.

The dashed line in this triangle represents $|G_\ell|$ of the $K^+ \longrightarrow \pi^+\nu\bar{\nu}$ decay; determining that can be a very good test of the SM [18] because a comparison between results from $B^0\bar{B}^0$ oscillations studies and experimental data may be performed; moreover the determination of $|G_\ell|$ should have lower theoretical uncertainties. We should note that the displacement from the unitary triangle vertex (1,0) is due to the non negligible charm quark contribution cited before.

Lets now focus on the height of the unitary triangle: this is determined by the rate of $K_L \longrightarrow \pi^0\nu\bar{\nu}$ and provides a direct estimation of the η parameter. Measuring that rate would prove an example of further CP violation (the latter being the measurement of ε'/ε in K^0 system, to that NA48 experiment contributed) with low theoretical uncertainty [22].

One should also not forget that having accurate and clean measurements of BR for (1.14) may also lead to direct evidence of new physics, if the measured values are incompatible with SM predictions. An example of this, many more may be found in [18] [23], involves supersymmetry; in that model the contribution of SM particles such as up quarks and W boson are replaced with BSM ones such as charged H boson, charginos and s-tops. This will propose the candidate diagrams in Fig. 1.3 for (1.16).

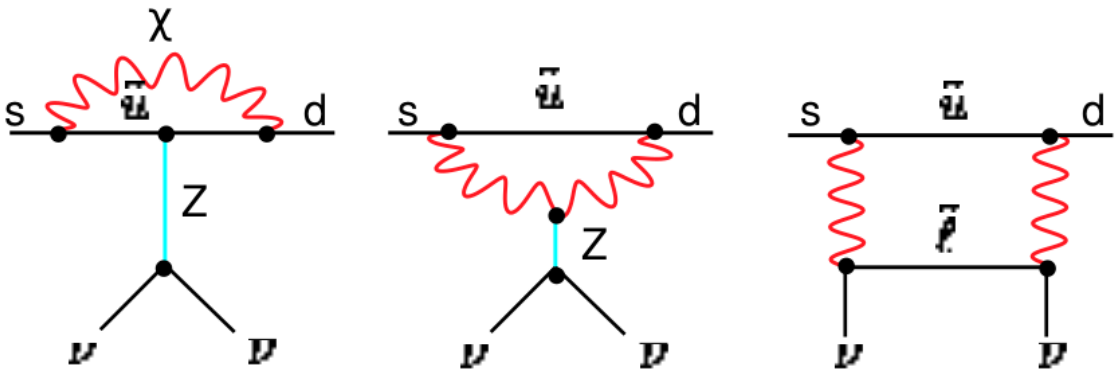

Figure 1.3: Underlying processes for the $s \longrightarrow d\,\nu\,\bar{\nu}$ in the SUSY framework [2]

Chapter 2

The NA62 Experiment at CERN

2.1 Introduction to NA62

NA62 is a medium-sized fixed target experiment located in the North Area at CERN Prévessin site. It is housed in the underground high intensity beam experimental hall (ECN3) previously used for the decommissioned NA48 experiment; of which NA62 reuses a number of sub-detectors and equipment.

Conceptually NA62 is a purpose-driven precision detector designed to measure the rare Kaon decay family

$$K \longrightarrow \pi + \nu + \bar{\nu}. \tag{2.1}$$

Studying these kind of processes can be useful for SM testing and can ultimately lead to the discovery of non-SM signals in Flavour Physics.

The NA62 detector is 270 m long, and most of its length is a dedicated *decay fiducial region*.

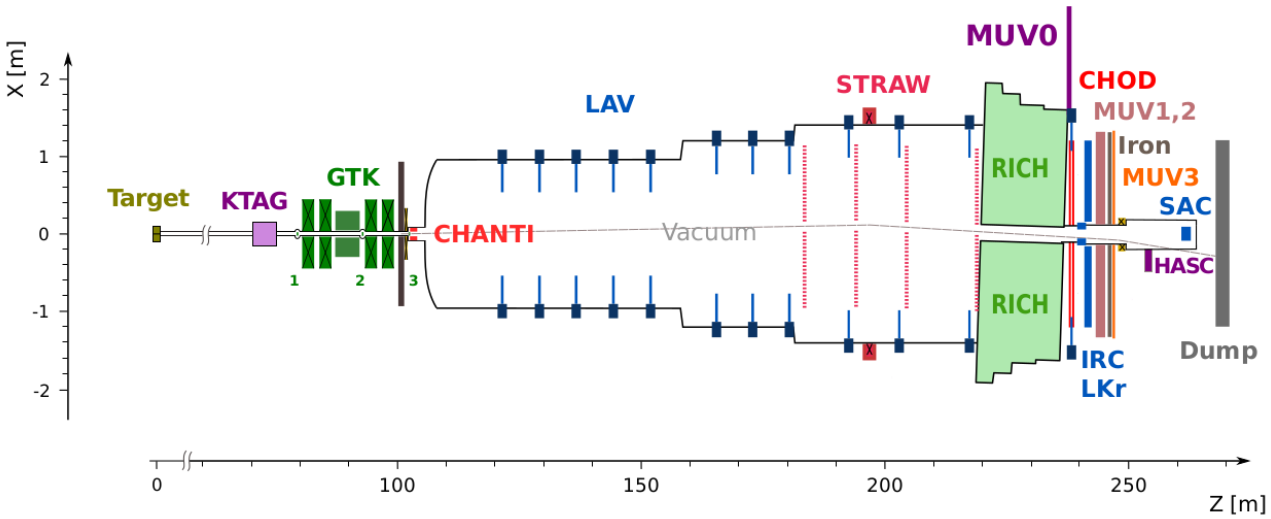

Figure 2.1: KTAG: differential Cherenkov counter; GTK: Si pixel beam tracker; CHANTI: ring stations of scintillator bars; LAV: lead glass ring calorimeters; STRAW: straw magnetic spectrometer; RICH: ring imaging Cherenkov counter; MUV0: off- acceptance plane of scintillator pads; CHOD: planes of scintillator pads and slabs; IRC: inner ring shashlik calorimeter; LKr: electromagnetic calorimeter filled with liquid Krypton; MUV1,2: hadron calorimeter; MUV3: plane of scintillator pads for muon veto; HASC: near beam lead-scintillator calorimeter; SAC: small angle Shashlik calorimeter.

The layout of the experiment along with its main 12 detectors is shown in Figure 2.1 [24]. A detailed explanation of all detectors will be given in Chapter 3.

Incoming K^+ are discriminated from generic charged particles in the KTAG and the GigaTracKer (GTK) provides tracking information for upstream K^+ tracks; note that KTAG and GTK are in the beamline and have to face extremely high hit rates.

Downstream in the fiducial region the decays involving π^0 are vetoed by means of photon Large Angle Vetos (LAVs), while STRAW spectrometer provides tracking of charged particles and RICH is used for π^+ identification and μ vetoing.

Hodoscopes (CHOD and NewCHOD) provide further tracking and timing information as well as energy information. Further downstream, the LKr calorimeter and IRC-SAC small angle calorimeters provide photon veto for intermediate and small angles from the beampipe. MUV1-2-3 are muon veto detectors and are used as redundant muon veto together with RICH.

2.2 Experimental design

The structure and design of NA62 has been optimized based on the ultra rare decay (PNN decay) here shown

$$K^+ \longrightarrow \pi^+ + \nu + \bar{\nu}. \tag{2.2}$$

We will now give an overview of the experiment design rationale.

In order to collect a meaningful sample of PNN events (~ 100) at least 10^{13} K^+ decays need to be collected . This computation is based on an estimated 10% signal acceptance and approximating the PNN branching ratio to 10^{-10} . From this stems the requirement of a background rejection factor[1] of at least 10^{12} [4].

Following these requirements the experiment must possess:

1. Good timing performance, necessary to correctly build the PNN events (K^+, π^+ matching);
2. Rejection of two and three body decays;
3. Hermetic γ, μ vetoing;
4. Precise K^+ tagging to overcome beam purity limitations.

NA62 aims at collecting samples of (2.2): given that neutrinos produced are effectively undetectable a PNN candidate is a set of a kinematic constrained track of a K^+ and a downstream π^+. When we say kinematic constrained we mean that accurate timing as well as spatial and angular information are used to match the two tracks in a meaningful decay; avoiding misassignment of orphan[2] upstream Kaon tracks to beam particles interaction tracks downstream.

PNN decay is ultra rare so a very high background rejection is extremely important: in Table 2.1 we list the most probable K^+ decay channels in comparison with the PNN:

Decay channel	$BR\,[\%]$
$K^+ \longrightarrow \mu^+ \nu$ $[K_{2\mu}]$	63.55
$K^+ \longrightarrow \pi^+ \pi^0$	20.66
$K^+ \longrightarrow 2\pi^+ \pi^-$	5.59
$K^+ \longrightarrow \pi^0 e^+ \nu$	5.07
$K^+ \longrightarrow \pi^0 \mu^+ \nu$	3.353
$K^+ \longrightarrow \pi^+ 2\pi^0$	1.761
$\mathbf{K^+ \longrightarrow \pi^+ \nu \bar{\nu}}$	$\mathbf{\sim 7.8 \times 10^{-9}}$

Table 2.1: Most probable K^+ decay branching ratios compared with PNN branching ratio.

[1] for other Kaon decay modes.

[2] a track whose Kaon decayed in products that were not detected.

Events from channel $K_{2\mu}$ can be rejected by having good particle identification to avoid mislabelling muons as pions; while decays producing π^0 can be discarded with hermetic photon vetoing.

Recall that the most probable decay mode for neutral pions is the channel

$$\pi^0 \longrightarrow 2\gamma; \quad BR \approx 0.99 \tag{2.3}$$

so vetoing events with photon production can reduce the impact of all decay modes including neutral π^0. Nevertheless, γ vetoing is not enough to achieve the desired signal to background ratio and a kinematic selection is necessary.

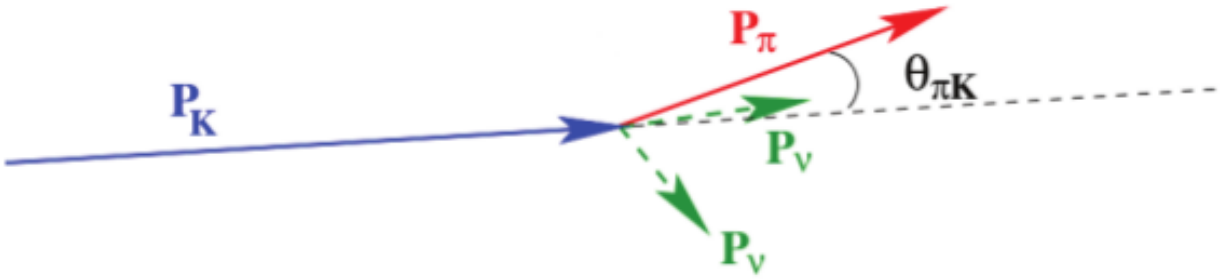

Figure 2.2: Kinematic diagram of the PNN decay. [3]

In the detector a PNN event consists of an upstream K^+ track and a downstream π^+ track: it is possible to define a *missing mass* variable, m_m^2, which represents the mass difference between the theoretical K^+ mass and reconstructed downstream track mass, under the hypothesis it is from a π^+. The missing mass is computed using the Invariant Mass law on the observable quantities of Fig. 2.2:

$$m_m^2 := (P_{K^+} - P_{\text{track}})^2 = m_{K^+}\left(1 - \frac{|P_{\pi^+}|}{|P_{K^+}|}\right) + m_{\pi^+}\left(1 - \frac{|P_{\pi^+}|}{|P_{K^+}|}\right) - |P_{K^+}||P_{\pi^+}|\theta_{\pi K}^2 \tag{2.4}$$

where $\theta_{\pi K}^2$ is the scatter angle between the two particles and P denotes quad-moment. The missing mass may be computed for all decay mode and used as a cutoff for the PNN signal regions.

The possible track four-momentum depends on the scattering angle; constraint plots for an upstream Kaon energy $\mathcal{E}_{K^+} = 75$ GeV are shown in Fig. 2.3, left pane.

The missing mass can thus be used as a kinematic constraint:

- For the $K^+ \longrightarrow \pi^+, \pi^0$ $m_m^2 = m_{\pi_0}^2$ this poses a forbidden value[3] for m_m^2 that divdes the signal region in two.
- For the $K^+ \longrightarrow \mu^+ \nu$ missing mass definition is nonsensical because it assumes the product is a π^+, it will have a distribution spanning even negative values. Can still be used for background rejection.
- For the $K^+ \longrightarrow 2\pi^+ \pi^-$ poses an upper bound in m_m.

[3] ignoring any m_m^2 resolution issues.

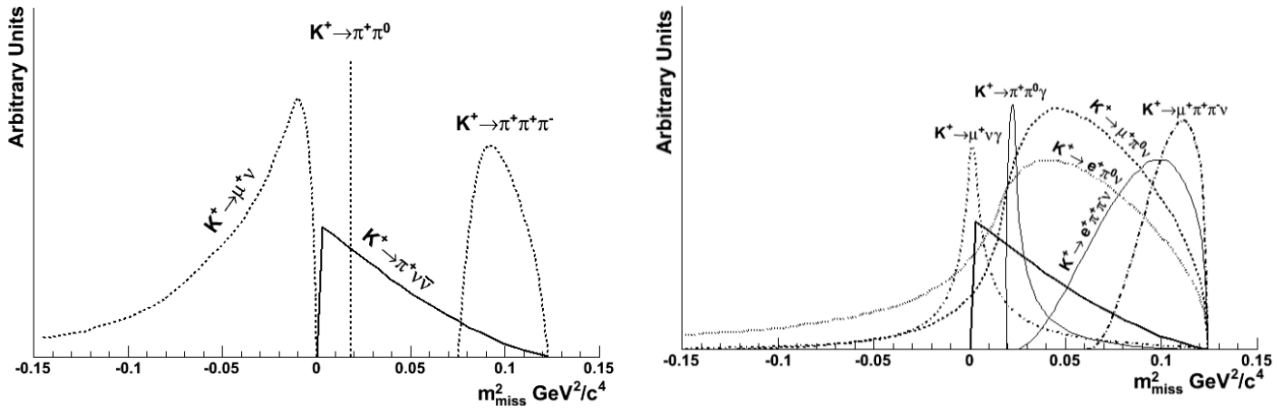

Figure 2.3: missing mass distributions for most relevant kinematic constrained (left) and unconstrained (right) decay modes [1]

From the plot in Fig. 2.3 one may observe that PNN decays are associated with specific ranges of m_m^2, we thus define two signal regions:

$$\text{Region I} \qquad 0 < m_m^2 < m_{\pi^0} - \varepsilon \tag{2.5}$$

$$\text{Region II} \qquad m_{\pi^0} + \varepsilon < m_m^2 < \min(m_m^2(K^+ \longrightarrow 2\pi^+\,\pi^-)) - \varepsilon \tag{2.6}$$

Here ε was introduce to point out that m_m^2 determination has a finite resolution.

The m_m^2 resolution needed by NA62 to perform as requested in terms of signal acceptance and signal to background ratio can be estimated by assuming typical values for veto inefficiencies on neutral π (thus on γ) and μ. This computation yields [2]:

$$\varepsilon \approx 8 \times 10^{-3}\ \mathrm{GeV}^2/c^4. \tag{2.7}$$

and (2.7) poses that K momentum resolution must be equal or better than 0.3%; π momentum resolution equal or better than 1% and angular resolution of the order of $50\,\mu$rad. These values will provide rejection of background tails from the signal regions, while ensuring the requested acceptance. The requirements on resolution described before are the basis of the design of NA62 GigaTracker and STRAW spectrometer, together with their engineering size constraints. Moreover, the high hit rate of GTK poses a stringent request on timing resolution to avoid downstream mismatching.

The GigaTracker is exposed to the whole high intensity beam, of which only a small[4] (below 10%) part is composed by the required K^+; for this reason it has to cope with an extremely high hit rate, of the order of 750 MHz. The GTK provides a 0.2% RMS momentum resolution and 15 μrad angular resolution at the nominal rate.

2.2.1 Choice of beam mean momentum

Previously $P_K = 75\ \mathrm{GeV}/c$ has been used as beam momentum figure; here the rationale of choosing an high energy beam for NA62 will be discussed.

First off, a K^+ beam has been chosen instead of a K^- one due to the higher production rate (R) of positive Kaons starting from 400 GeV SPS primary protons, in fact $R_{K+}/R_{K-} \sim 2.1$. Moreover a K^+ beam is much purer for same required kaon momentum $\left(\frac{R_{K+}/R_{\pi+}}{R_{K-}/R_{\pi-}} \sim 1.2\right)$ while being mostly similar considering all equally charged byproducts.

Lets now finally justify the $P_K = 75\mathrm{GeV}/c$. [25] provides particle production metrics for primary proton interactions on Be targets such as the one used in NA62. According to that study $P_K = 75\ \mathrm{GeV}/c$

[4] See section on beam type choice.

maximises the number of $K^+\,\pi^+$ decays in the fiducial length as well as their K/π ratio and the K^+ decay number over the total hadron flux.

This value also maximises the relative detector system acceptances for the PNN decays:

- Acceptance over protons per second from beam;
- Acceptance over π^+ decays;
- Acceptance over total hadron flux.

Photon vetoing system are more efficient at high energy and having a large beam momentum will ensure mean energy distribution of γ coming from π^0 decay to have a large mean value. Notice that particle selection performed in the RICH requires the particle to be in the range $15\ \mathrm{GeV}/c < P < 35\ \mathrm{GeV}/c$, so products from K^+ decays at $75\ \mathrm{GeV}/c$ must have energy greater than $\sim 40\ \mathrm{GeV}/c$ to be detected properly within the fiducial region boundary.

An high energy beam would pose many problems with the detectors; total hadron flux will increase greatly and the structure of the detectors as well as the whole experiment would need many changes and can even become unfeasible (cost and length of fiducial region limitations).

2.2.2 Hardware vetoing system

As shown in the previous section the majority of background comes from processes that involves γ or μ production (first two most probable processes).

Photon vetoing

Notice that the most probable photon-producing decay $K^+ \longrightarrow \pi^+\,\pi^0$ requires at least π^0 inefficiency 10^{-8} to match required S/B ratio.

Regarding photon vetoing there is an hermetic veto up to $50\ \mathrm{mrad}$, this is achieved with a three level system:

- A Large Angle Veto system (LAV) comprised of 12 stations in the fiducial decay volume, providing veto in $[8.5 - 50\ \mathrm{mrad}]$ range;
- The old NA48 Liquid Kripton (LKr) calorimeter, providing veto in $[1 - 8.5\ \mathrm{mrad}]$ range;
- The Intermediate Ring Calorimeter (IRC) and Small Angle "Shashlik" Calorimeter (SAC), located very close to the beam pipe, vetoing $< 1\ \mathrm{mrad}$.

This system provides $< 10^{-5}$ inefficiency for low angle ($[1-10]\,\mathrm{mrad}$) γ with $\mathcal{E}_\gamma > 10\,\mathrm{GeV}$ and $< 10^{-3}$ inefficiency for $\mathcal{E}_\gamma < 1\ \mathrm{GeV}$.

Muon vetoing

Muon vetoing is also very important and is based on a tandem of two hadronic calorimeters (MUV1 and MUV2) and an iron-shielded fast scintillator wall (MUV3). These units form the Muon Veto System (MUV). Together with the information coming from MUV detectors data from the electronic calorimeter LKr aids in hadronic/electronic shower separation and vetoing. A further vetoing action is provided by the the Ring Imaging Čerenkov counter (RICH).

The inefficiency of the muon vetoing system of NA62 is shown to be around 10^{-5} according to Monte Carlo simulations [4].

Further hardware vetoing

The GTK is a very upstream detector inserted directly in the beam pipe and there is possibility that hadronic interactions in its last station produce unwanted background. For this reason a scintillator detector called CHANTI is installed around the beam pipe and is used as veto.

2.2.3 Particle identification systems

Upstream identification

One of the disadvantages of having an high energy beam serving NA62 is its poor K^+ purity. The $75\ \mathrm{GeV}$ beam used has less than 10% kaon content. All other particles are spurious and their interaction with detectors gas filling can swamp sensitivity. A cut in longitudinal vertex position can effectively solve this. There is the need of blind all but those few Kaons with great timing resolution. This operation is performed by the KTAG; this detector located upstream of the GTK tags kaon traversing time with a $100\ \mathrm{ps}$ resolution.

Having precise time tagging mitigates fiducial gas pressure requirements and enables vetoing candidates coming from downstream interactions of spurious beam particles with e.g. the GTK stations. The CEDAR time tag is the master time information for K^+ used to associate GTK tracks with downstram pion STRAW tracks.

Downstream identification

Identifying downstream particles, together with γ vetoing is the main technique used to limit background processes.

The Ring Imaging ČHerenkov detector (RICH) is used to discriminate between π^+ and other charged particles such as μ, e^+. Experimental testing have shown the RICH to have a $< 1\%$ fake probability in π^+, μ identification; accounting for the largest background decay reduction with an inefficiency of 10^{-2}. Notice that due to its Neon filling the RICH may operate only on $P_{\pi^+} > 13\ \mathrm{GeV}$, being that the Čherenkov treshold of Ne; a π momentum lower cutoff is then applied on events.

Being a gas-filled area, RICH can reduce photon inefficiency due to γ conversions or photo-nuclear interactions happening within its volume. In order to mitigate these effects the downstream hodoscope (CHOD) is used to identify the products of those interactions and veto them.

Background suppression for less frequent decays involving positron products is implemented in conjunction with the LKr calorimeter that, placed downstream, can identify very well $e^\pm$ states that end up after the RICH. The qualities of LKr in positron and electron identification together with its excellent energy resolution were appreciated during NA48 operations.

Another important function of RICH is to provide precise π^+ time tagging for the STRAW spectrometer; in a way similar to KTAG-GTK on upstream K^+. RICH time resolution is around $150\mathrm{ps}$. Significantly better than the STRAW.

RICH can also provide redundant π momentum information that may be used to cross check those from STRAW, albeit with a fourfold lower resolution.

Chapter 3

NA62 Sub-detectors

In this chapter an overview of all the main sub-detectors of NA62 will be given.

3.1 KTAG

The KTAG system is used to tag the incoming beam kaons. This detector is the most upstream operating and is subjected to the full rate beam (~ 750 MHz).

Its main element was based the old West CEDAR of the SPS secondary beam line running on Nitrogen; this is a gas filled differential Čherenkov detector originally used for low momenta beams study [26].

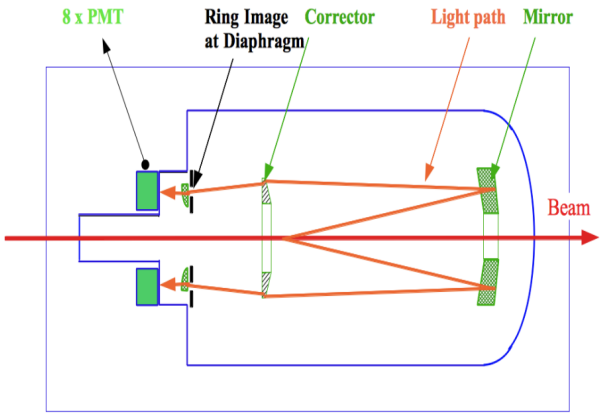

Figure 3.1: Schematic of the CEDAR-W with original 8 PMT readout [4]

Figure 3.2: The CEDAR-W installed on the NA62 beamline with new 64 PMT light collection system

The CEDAR-W consists essentially of a 558 mm diameter, 5 m long vessel filled with gas. At the end of the vessel, the Čerenkov light is reflected by a spherical Mangin mirror onto a ring-shaped diaphragm of 100 mm radius with adjustable aperture width, located at the beginning of the vessel. The pressure of the gas can be adjusted so that only light of the wanted particle type will be transmitted through the diaphragm slit.

In order to be compatible with the high flux conditions of NA62 the original CEDAR had to undergo multiple mechanical and electrical modifications: the original light collection system, based on 8 PMT tubes had to be replaced with a high resolution, fast one based on 8 groups of 8 smaller phototubes each replacing one old PMT. This new system, called collectively *KTAG*, is steerable so to optimize the geometrical configuration and uses custom ASICs for the charge readout. Moreover some structural modifications on CEDAR beam pipe and nose were performed.

The main KTAG system characteristics are reported in Table 3.1, while a diagram of the new light collection system is in Fig 3.3

Parameter	Value
$n-1$	$\sim 142 \times 10^{-6}$
P_{nom}	3.85 bar
θ_K	30.9 mrad
R_K	50 MHz
time ε	< 100 ps
$\Delta\theta/\theta$	4.8×10^{-3}
$\Delta\beta/\beta$	5×10^{-6}
$N_{\gamma\text{Ch}}/N_K$	100
PMT rate	3 MHz

Table 3.1: KTAG with CEDAR-W technical characteristics summary, with R we denote rate and with ε resolution

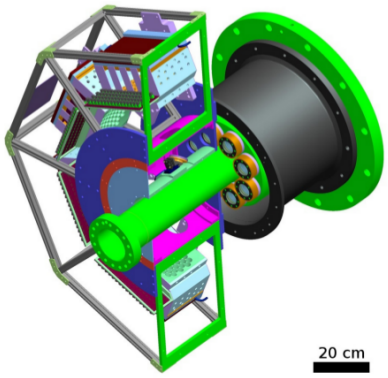

Figure 3.3: New KTAG light collection system with 64 PMTs [4].

It should be noted that CEDAR-W is designed to operate with Nitrogen and its performance is hindered by gas-beam scattering that causes broadening of Čherenkov cones and overlapping of π, K rings; this costing kaon detection efficiency and worsening beam quality [27]. During the whole run time of NA62 the CEDAR was filled with Nitrogen but the need for higher efficiency with increasing experimental sensitivity posed the need for an upgrade[1].

Following that a new CEDAR-H designed to work with Hydrogen while retaining Nitrogen filled CEDAR-W optical quality has been designed and installed in the cavern, keeping the original CEDAR-W as working spare. The 2023 data taking is thus predicted to use the new CEDAR-H exclusively.

3.2 GigaTracker

The GTK is the main Kaon spectrometer of NA62. Its purpose is to track incoming K^+ and thus determine their momenta and direction. According to experimental design the GTK should have a relative K^+ momentum resolution $\leq 0.2\%$, an angular resolution of ~ 16 μrad and a time resolution of less than 150 ps in order to resolve a single track in the high intensity beam. On a single-pixel base the GTK is capable of ~ 200 ps RMS time resolution.

As of 2022 the GTK system is comprised of four hybrid Silicon pixel tracking station named GTK0-3, with GTK0 being a 2021 addition located upstream of GTK1. A group of four dipole magnets placed in the same general area provides beam bending and requires GTK2 to be displaced vertically from the beam direction of ~ 60 mm.

A layout of the original GTK stations is shown in Fig. 3.4.

[1] actually, according to the orignal NA62 design report [4] the CEDAR-W should have been filled with hydrogen, overcoming the optical limitations that the optical elements designed for Nitrogen would pose.

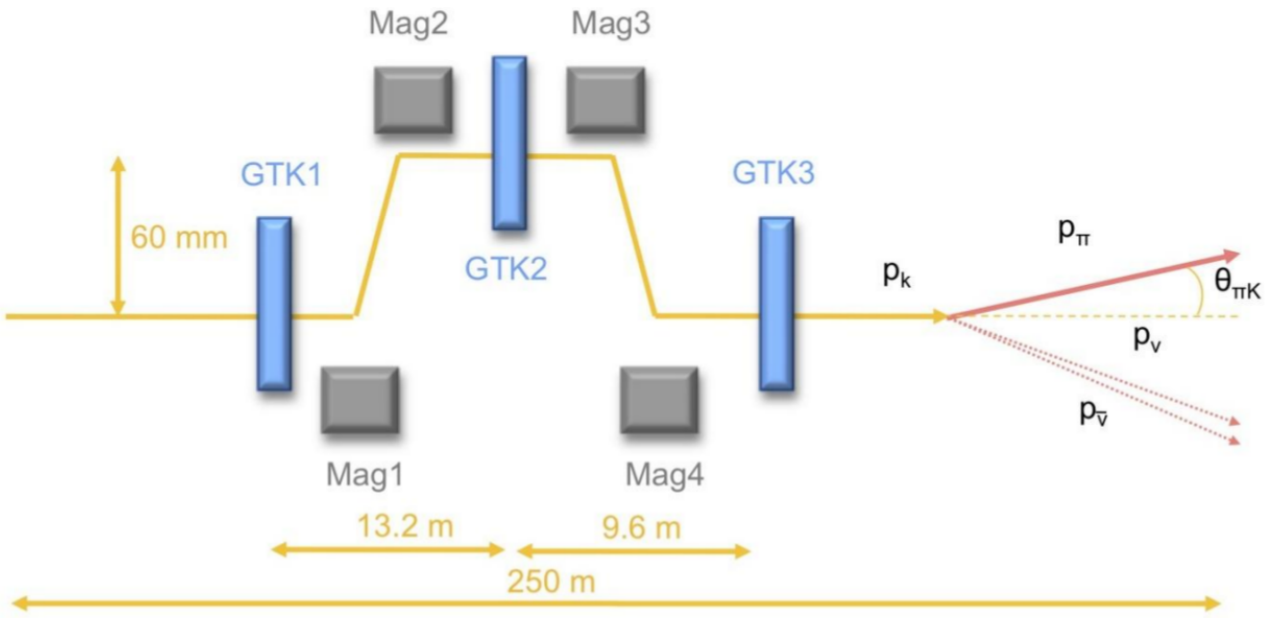

Figure 3.4: Schematic position of GTK1-3 along the beam line; Notice that in the computation of m_m^2 (2.4) P_K comes exclusively from the GTK. [1]

GTK position in the beam line and beam characteristics poses many challenges in detector design: its stations are exposed to a very high and non uniform flux of charged particles with a rate around 1 GHz and with intensity peaks in the central areas of $1.3\,\mathrm{MHz/mm^2}$.

The only detector technology capable of working is such conditions is the Silicon pixel detector: the sensitive element in each station is a $\sim (60 \times 27)$ mm Silicon pixel array immersed in the beam; the array is composed by $(300 \times 300)\ \mu\mathrm{m}$, $200\ \mu\mathrm{m}$ thick pixels arranged in 10 matrices each formed by 1800 units. Detector slab thickness is chosen as a compromise between electron yield per interaction and unwanted secondary scattering. In order to have a successful readout the charge per interaction should be $\approx 15 \times 10^3\, e^-/K^+$.

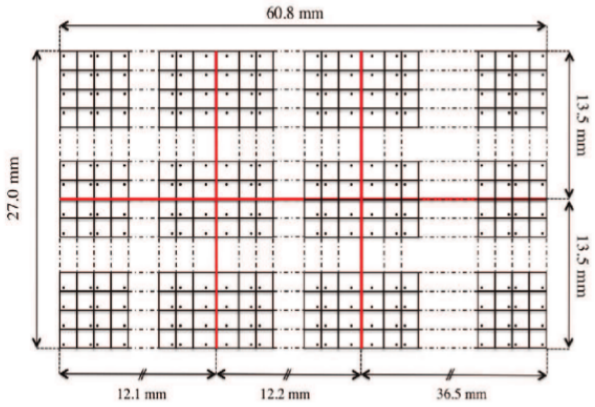

Figure 3.5: Schematic view of a GTK station array, with matrix subdivision an relevant dimensions marked. [4]

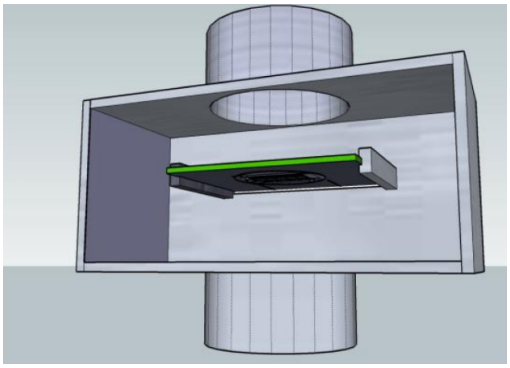

Figure 3.6: Sketch of a GTK station array location in the beam pipe, Silicon array is bonded to the carrier board (green) that provides signal lanes, power and cooling to the module. [4]

3.3 CHANTI

The GigaTracker is located in line with the beam propagation and is inevitably a source of unwanted interactions that leads to fake K decay products in the downstream fiducial region.

While it is possible to use the information of the previous GTK station to achieve some degree of vetoing this is not possible for GTK3. A beam particle interaction in GTK3 may produce, among other charged particles, a π^+ that can reach the STRAW tracker and be misidentified. A sketch of the mechanism is visible in Fig. 3.7.

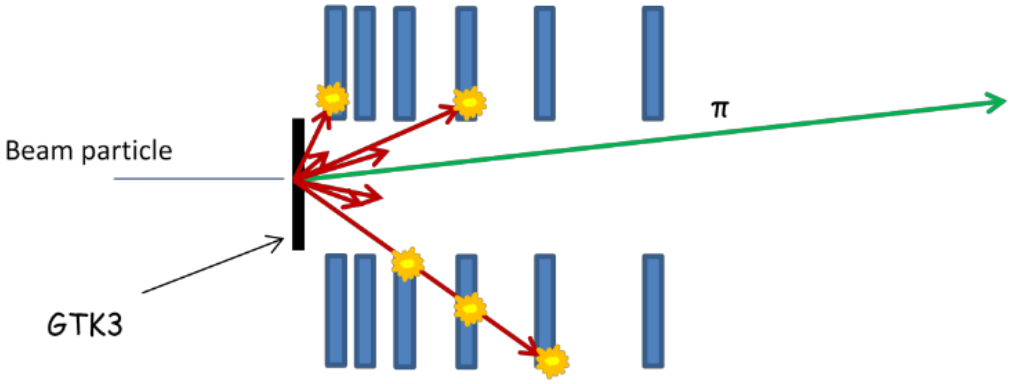

Figure 3.7: Worst-case inelastic interaction of beam particle with GTK3 station, producing fake pion candidate. CHANTI position and interaction points used for vetoing are shown. [4]

This phenomenon has been modeled and simulated and given NA62 design it will provide a fake π candidate in 10^3 interactions. The system then should be able to achieve a target inefficiency of $\sim 10^{-8}$ to counteract. This role is covered by the CHarged ANTI detector system.

The CHANTI is a scintillator detector designed to veto inelastic beam interactions with the GTK stations and the upstream beam collimator; it is also used to reject so called *halo muons* that appear in close proximity to the beam axis. It is built upon 6 rectangular stations; each station is composed by 2 layers of wedge-shaped plastic scintillating blocks arranged in an *horizontal-vertical* configuration where each layer is composed by two half layers of wedge blocks interleaved together. The choice of wedges for the scintillator blocks allows an improved impact position resolution of around 3 mm, better than the one attainable using square blocks by factor ~ 3. An horizontal (vertical) layer is composed by 24 (22) scintillator bars, subdivided into 10-14 (10-12) half layers. The light output of each bar is carried to its dedicated solid state SiPM using Wave Length Shifter (WLS) fibres.

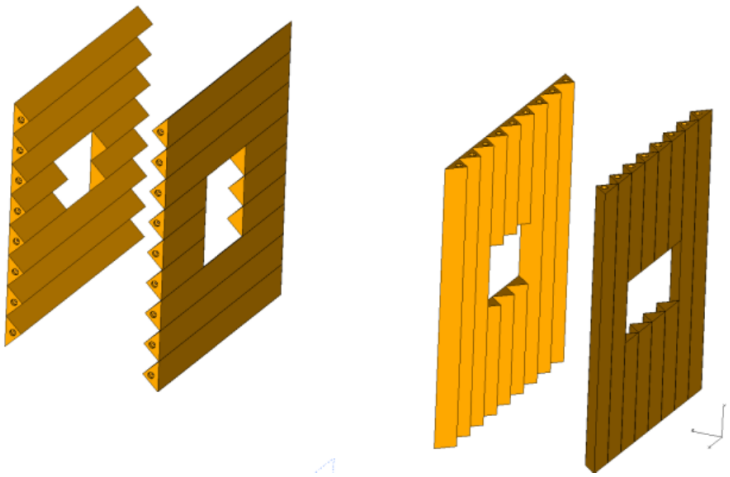

Figure 3.8: Scintillating layers composing one CHANTI stations, both horizontal and vertical orientation layers are visible. [4]

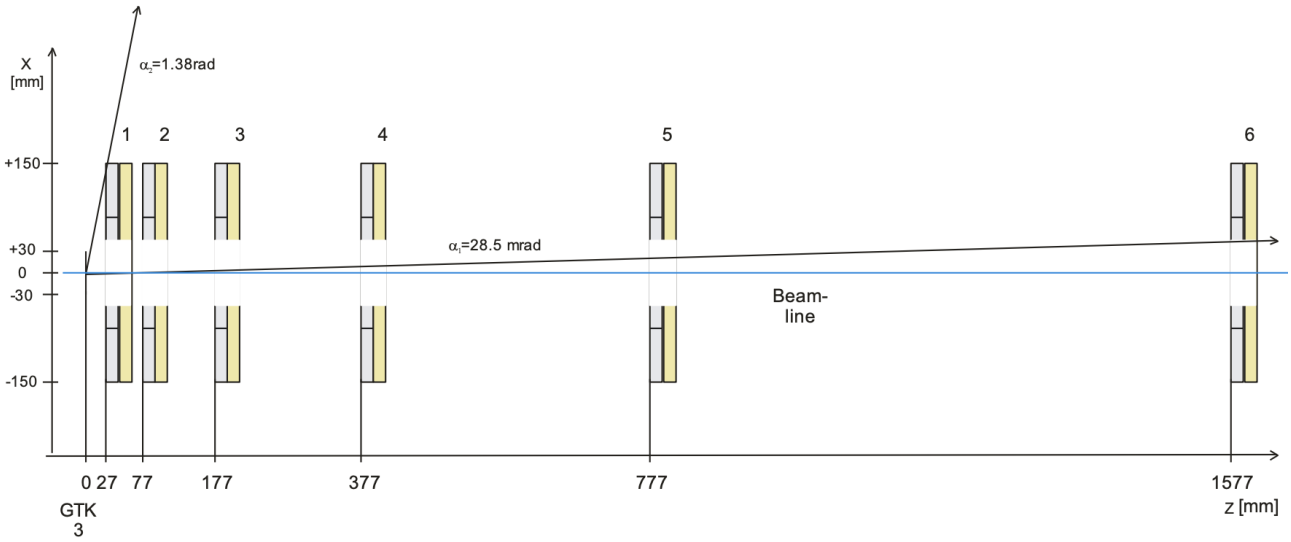

Figure 3.9: CHANTI stations locations along the beam pipe and angular acceptance. [4]

In order to accomodate the beam pipe every station has a (50×90) mm rectangular hole in the middle, the outer side of the layer stack is ~ 300mm. A representation of the stacks is shown in Fig. 3.8 while in Fig. 3.9 we show the station layout along the beam line. This arrangement enables hermetic π vetoing from 30-50 mrad (depending on GTK3 secondary interaction location) up to ~ 1.2 rad. This extremely large angle vetoing complements the other pion veto information coming from LAV, MUV1-2-3 and IRC-SAC that hermetically covers from ≤ 1 mrad up to 50 mrad.

Results from GEANT4 simulations shows the expected CHANTI rate to be ~ 2 MHz, this detector is not used in the hardware triggering but provides a good timing resolution (≤ 2ns) to limit mismatching with downstream pions and thus random vetoing rate; it has also limited tracking capabilities[2] as well as basic deposited charge measurement, that together with position sensitivity are exploited to improve spatial resolution.

The expected vetoing efficiency of the CHANTI is of 95% for total secondary interactions in GTK3, and this efficiency is around 99% if just signal-like events are considered. We remind that a signal-like event for the STRAW is a π^+ downstream trace, so in the case of CHANTI a signal-like event is a secondary emitted pion travelling towards the STRAW whose originating K was completely absorbed in the GTK substrate and the other charged products were intercepted by the CHANTI.

3.4 The Large Angle Veto (LAV)

The LAV system is composed by 12 stations of lead-glass (SF57) scintillator blocks read by PMTs (Hamamatsu R2238) placed between 120m and 240m along the beam line. Following NA62 paradigm of reusing legacy structures and materials the crystals come from the dismantled OPAL electromagnetic calorimeter.

The elementary unit of a LAV station is referred to *azimuth segment* and contains three or four crystals and the relative photomultipliers. A station is built with a number of azimuthal segments forming 4-5 complete rings. Due to the structure of the segments having gaps between the blocks, each ring in the station gets rotated with respect to the previous one in order to provide vetoing hermeticity.

The rings are mounted inside a vacuum vessel, this forms the LAV station and all but the last one are inserted into the fiducial decay tube. Within the crystal blocks there is an opening for an optic fiber connection used for testing and *in situ* calibration.

[2] useful for discriminating halo pions and inelastic interaction products.

Inefficiency of the LAV system has been estimated to be of 10^{-4} after extensive performance testing in Frascati.

A schematic of a LAV station as well as a picture of an azimuth segment is shown in Fig. 3.10 and Fig. 3.11 respectively.

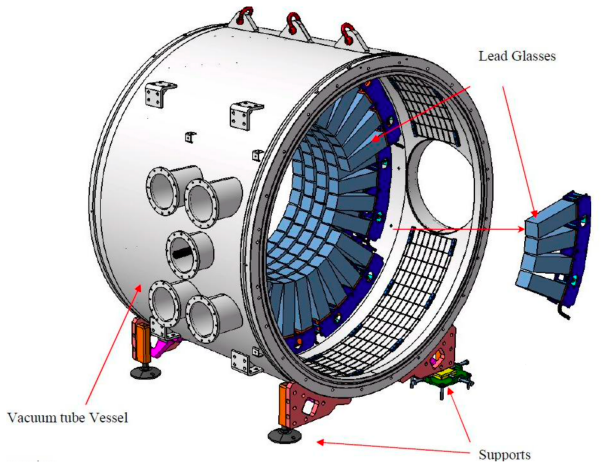

Figure 3.10: LAV station schematic with ring structure clearly visible. [4]

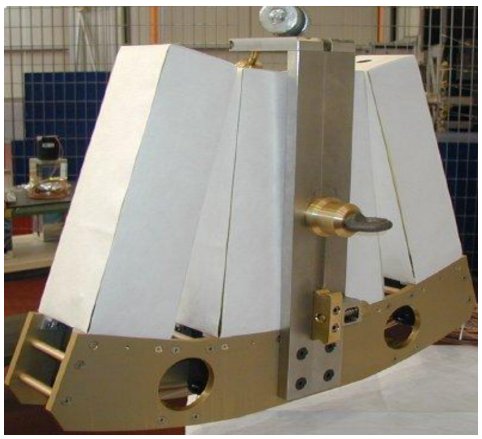

Figure 3.11: Picture of an azimuth module during assembly, wiring and PMT tubes are hidden by the carrier frame. [4]

3.5 STRAW spectrometer

The STRAW is, along with the Giga Tracker, the most important detector of the experiment. This detector must be almost 100% efficient in track reconstruction and may also provide vetoing for multiple charged particles appearing downstream.

STRAW is a spectrometer built with straw tubes; its basic unit is a $36\,\mu\mathrm{m}$ thick, $9.8\,$mm in diameter and 2.1 m long metallized PET straw with Au and Cu cathode coating and a gold plated tungsten anode wire. In order to achieve the required track resolution each station is built by overlaying 4 staggered straw panels, called *views*. The detector is composed by four stations each containing 1792 straw units. Each view is composed of 112 straws arranged in 4 layers. The layout of the views provides an octagonal shaped hole in the middle of the station where the beam may pass without interaction. High rates for the innermost straws are expected anyway, being as high as ~ 500 kHz/straw.

In the middle of the detector a high aperture dipole magnet with a mean field in the center ~ 0.36 T enables momentum measurements for the charged particles; the field is tuned to provide a $256\,\mathrm{MeV/c}$ kick to the incoming beam.

The choice of a straw tracker in vacuum is dictated by the stringent track efficiency and resolution requirements on momentum and position, those being:

$$\frac{\delta P_{\text{track}}}{P_{\text{track}}} \leq 1\%, \quad \delta\theta_{\pi^+K^+} \leq 60\ \mu\text{rad}. \tag{3.1}$$

The tracker elements must also be chosen to minimize unwanted scattering of the beam, especially near the first chamber; all secondary interactions happening at this level may produce unwanted tracks that will worsen background rejection of the experiment.

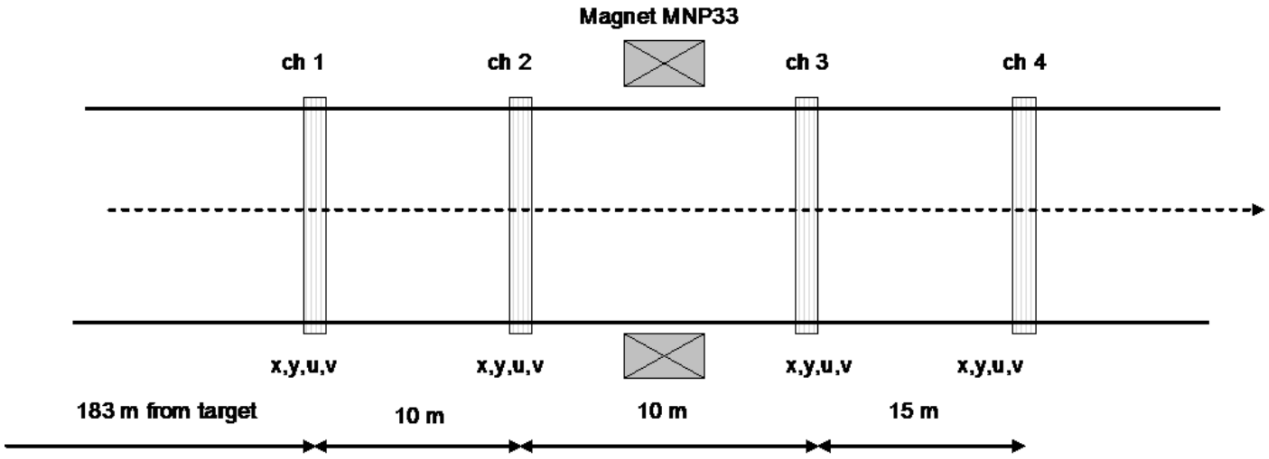

Figure 3.12: The STRAW spectrometer and its location along the beam pipe. [4]

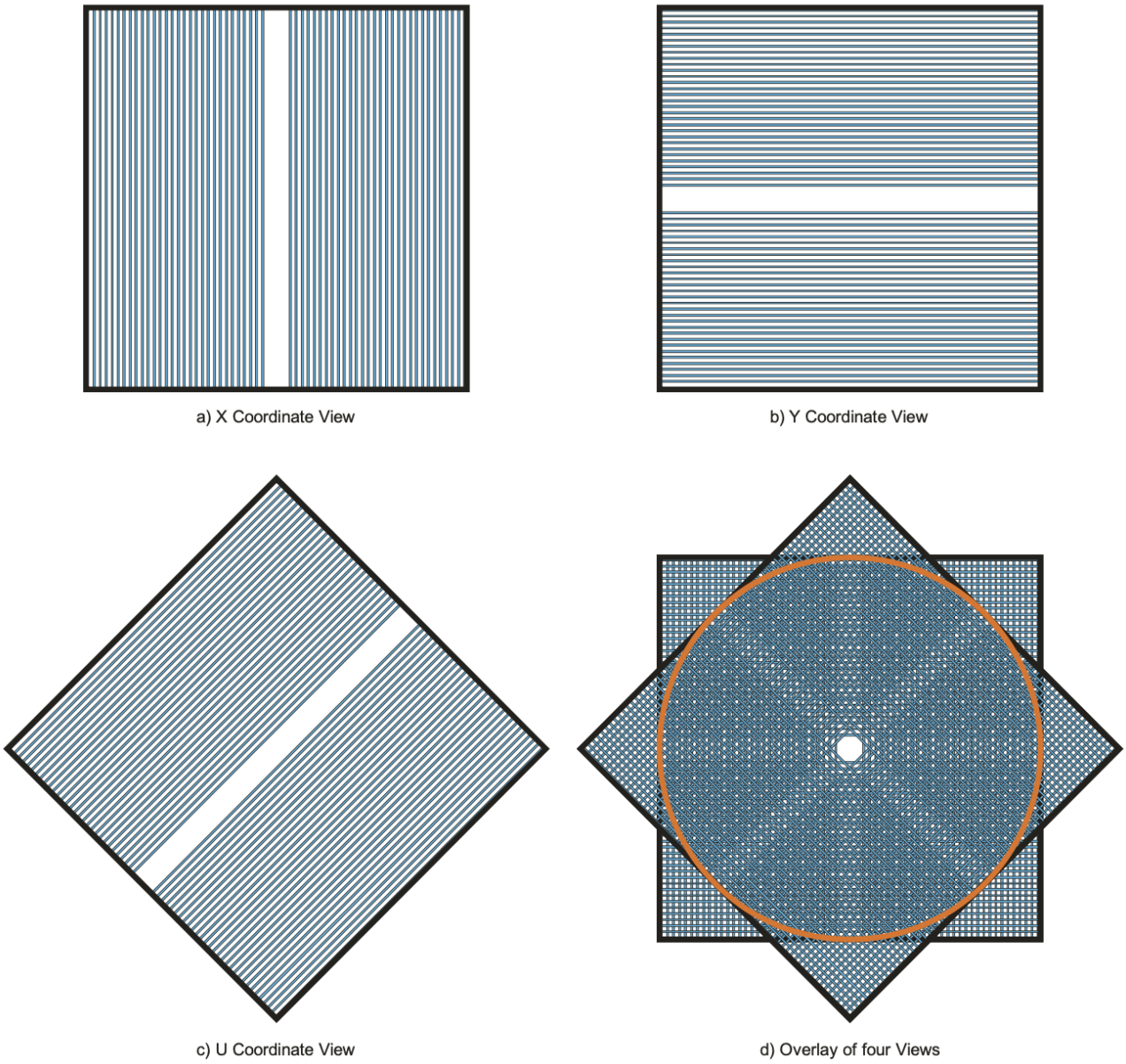

Figure 3.13: Schematic view of the views composing a STRAW station. The stack is built following the *xyuv* scheme, where x, y are vertical and horizontal sections and u, v are rotated by $\pm 45^\circ$ from vertical. [4]

3.6 RICH

The Ring Imaging ČHerenkov counter is located downstream the STRAW spectrometer inside a 17 m long, 3 m wide vessel filled with atmospheric pressure Neon gas acting as Čherenkov medium. An evacuated beam tube of 17 cm in diameter allows the main beam to traverse the detector without substantial interaction.

On the downstream side of the vessel a *mirror mosaic* composed by two mirror groups reflects the Čherenkov light towards the upstream wall, where is detected. This mirror system avoids the beam pipe shadow to interfere with the detector system.

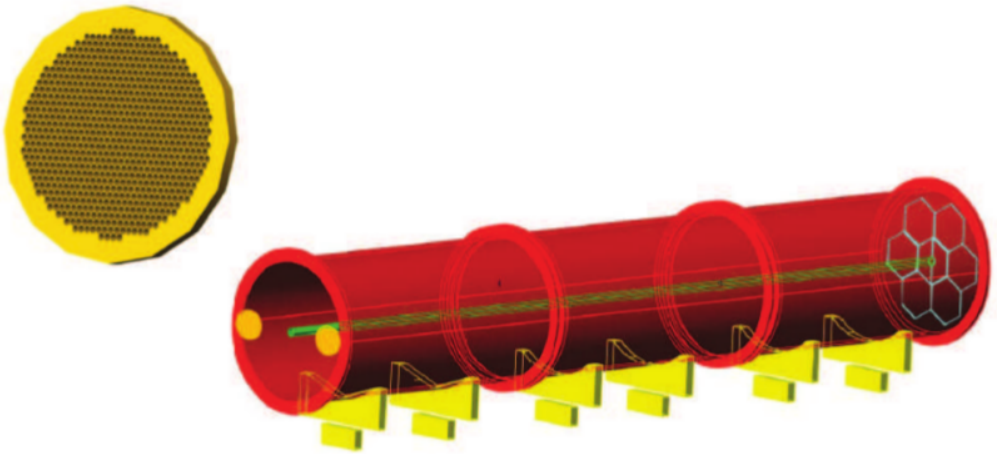

Figure 3.14: Schematic of the RICH vessel and of the detector wall with pixel cutouts. On the downstream wall the mirror mosaic is visible. [4]

Čherenkov light is collected by the detector system located on the upstream wall; the active elements are 1000 Hamamatsu R7400-U03 photomultipliers, having a round photocatode 8 mm in diameter. The detector wall has round pixel 18 mm in diameter, and the smaller PMT tubes are coupled to the wall pixels by means of Winston cone light guides.

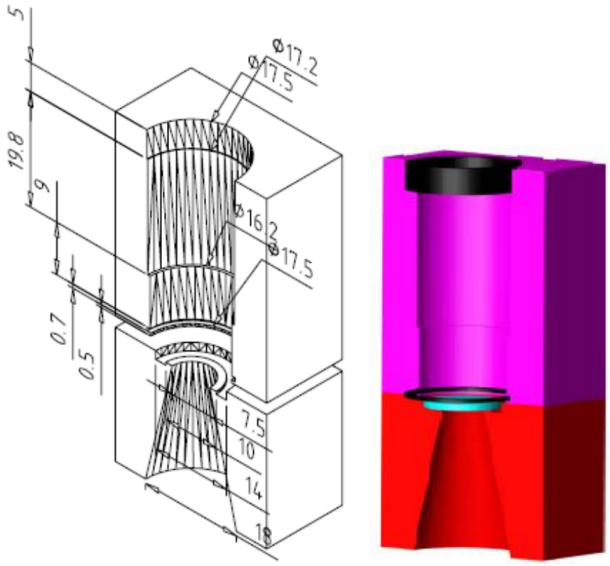

Figure 3.15: Mounting hardware for one RICH photomultiplier tube; showing location and use of the Winston cone. [4]

The RICH system is used mainly for μ background suppression, providing an inefficiency of $\sim 10^{-2}$ for μ^+ between $15\,\mathrm{GeV}$ and $35\,\mathrm{GeV}$ and high resolution downstream crossing time measurements. It also has a role in the low level trigger; the trigger system will be illustrated in 4.2.

3.7 Hodoscopes

NA62 includes two downstream hodoscopes, one (CHOD) inherited from the NA48 experiment and another, more modern, installed to overcome CHOD limitations regarding resolution and hit rate.

3.7.1 CHOD

The NA48 CHArged Hodoscope is mostly used for simple vetoing and triggering purposes at NA62. It is a basic hodoscoped composed of two 64 channel views built with Bicron BC408 scintillator slabs arranged in $x - y$ configuration; slab have a length of 1 m and have a square profile. Each channel is read by an XP2262B PMT tube coupled to the slab by means of a plastic fishtail light guide.

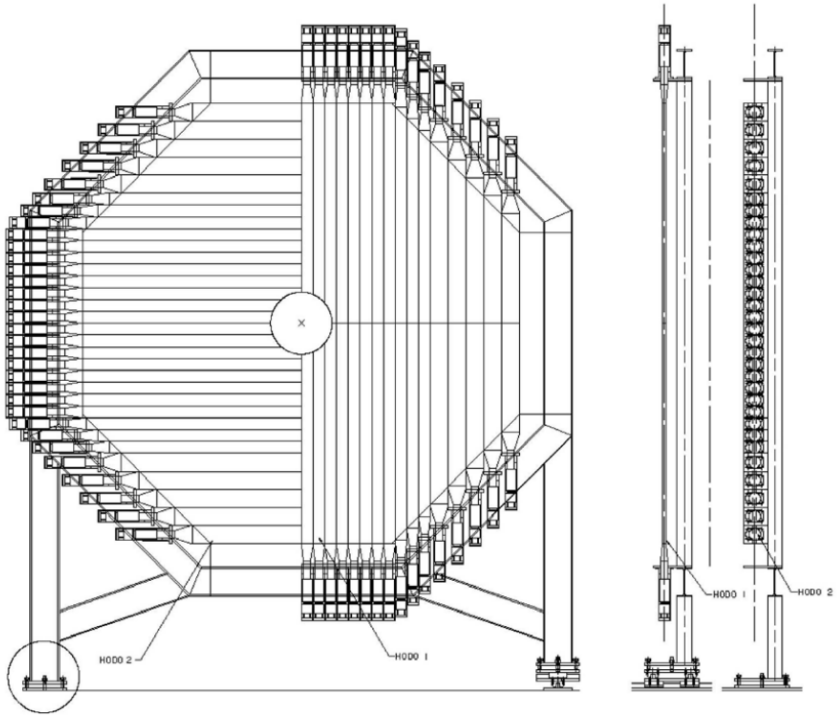

Figure 3.16: Schematic of the NA48 CHOD system. [4]

The detector is located immediately downstream to the RICH and its used to veto secondary interaction events happening around the mirror mosaic; similarly to what is done with CHANTI and GTK3. CHOD is part of the L0 triggering system together with RICH, complementing its information on downstream charged particles.

3.7.2 NewCHOD

NA48 CHOD has high dead time due to light propagation in the slabs and slow phototube readout. A newer, faster design hodoscope has been developed.

NewCHOD uses a tiled design to improve dead time and fast SiPM readout. The detector is arranged in four quadrants each built upon 38 thin (3 cm) tiles of plastic scintillator. Due to the high density of the active elements the readout SiPM arrays, one PM per tile, are coupled to the detectors by means of WaveLength Shifter (WLS) fibres. The SiPM used are MicroFC-30035-SMT; this choice together with tile material enables a maximum rate of 500 kHz. [28] [29]

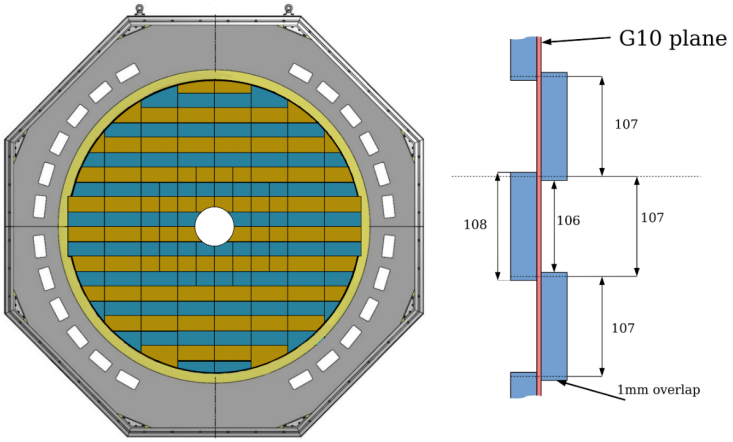

Figure 3.17: Schematic of the NewCHOD system. [4]

3.8 Liquid kripton calorimeter

LKr is another detector reused from NA48. It is a world-unique liquid kripton electromagnetic calorimeter. This type of calorimeter provides very high resolution due to Kr extremely low Moliere radius (6.1 cm) and short radiation length.

The device uses $\sim 10\ \mathrm{m}^2$ of ultra pure liquid kripton as drifting medium. Inside the vessel, around the 16 cm diameter beam pipe traversing it, a tower structure houses the readout electrodes grouped in 13212 cells of $2 \times 2\ \mathrm{cm}^2$ cross section. A cell has one anode and two cathodes in the form of Cu-Be ribbons. The drift distance is 1 cm either side of the anode. In order to improve the response the cells are arranged in a zig-zag geometry along the flight path. Voltage bias and spatial arrangement of the cell elements must be kept very stable to avoid loss in performance, and measures were taken to insure their stability. A neutral hodoscope composed of scintillating fibres with PMT readout [5] is installed inside the vessel in the expected maximum shower development area. It is readout by spare channels of nearby detectors.

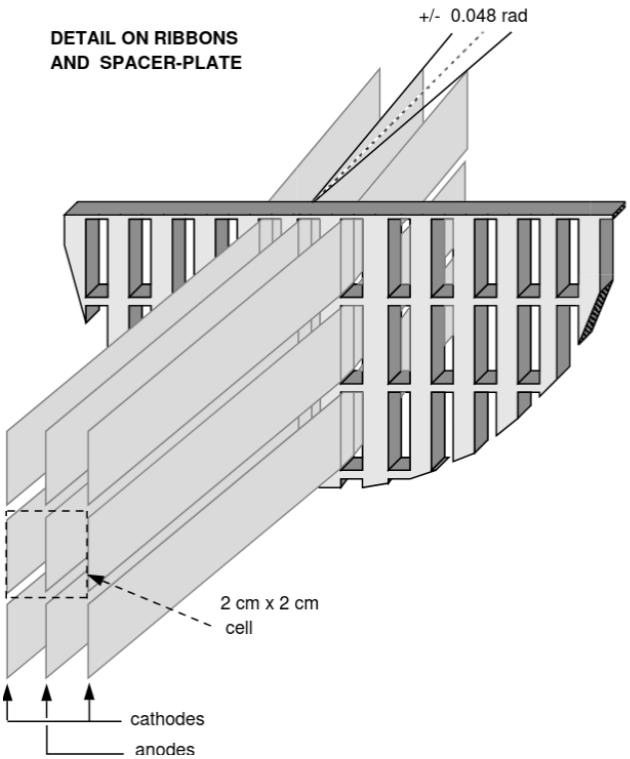

Figure 3.18: Structure of basic cells of the LKr calorimeter. [5]

In NA62 LKr is mostly used as a vetoing system, and the original readout from NA48 is no longer sufficient to keep up with the increased rate[3]. For this reason a new board (Calorimeter REAdout Module or CREAM) has been developed and used. LKr has also a role in NA62 low level trigger, and complete data sending happens only to L1 selected events; the readout system has provision to accomodate this feature.

3.9 Small angle veto (SAV)

Complementing LAV and LKr vetoing a system of two small Shashlyk-style calorimeters is included.

A Shashlyk calorimeter is a special type of electromagnetic calorimeter designed to detect incoming particles by making them shower. It is built by interleaving high atomic weight material such as lead with scintillating slabs. Readout light is brought out by WLS fibres; the characteristic aspect of this topology being that those are routed through holes in the lead plates, allowing a compact construction at the expense of a small detection efficiency loss[4] [30]. This type of calorimeter is ultimately a *single channel* detector; the high rate is not split among PMs and the mutiplicity is needed only to match the total scintillator area and geometry.

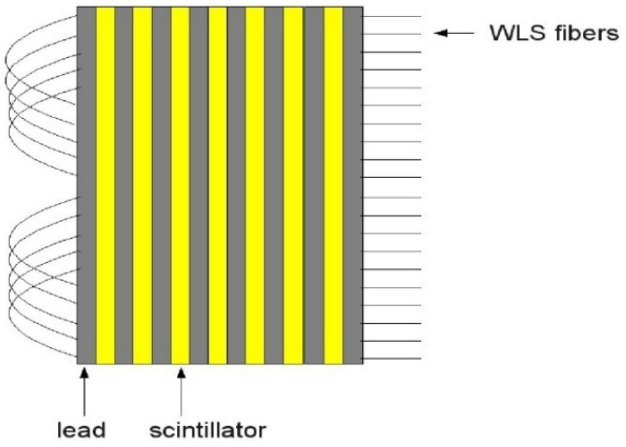

Figure 3.19: Outline of a Shashlyk-style calorimeter, fibers are passed through holes in the absorber plates and routed to the PM system. [4]

These calorimeters need to be located extremely close to the beam pipe in order to catch all decay photons at low angles but far enough to avoid the unmanageable count rate due to the beam halo. High count rates in single channel detectors such as these requires high time resolution to be useful as veto. The count rates are expected to be around hundreds of kHz for SAC, coming mostly from Kaon decays and substantially higher (~ 10 MHz) in the IRC, mainly due to the beam muon halo.

3.9.1 Intermediate Ring Calorimeter (IRC)

This detector is placed downstream of the LKr calorimeter. It is built as two ring sections; the upstream section composed by 25 scintillator-lead layers with 29 cm external diameter and 12 cm internal while the downstream one of 45 layers and 29 cm, 12.2 cm diameters. This difference in diameter forbids particle impinging directly on the detector edges only in the second section. Due to the expected energy distribution of incoming particles an interaction this far downstream won't shower and will then be missed.

In order to further minimize the loss of efficiency and mitigate the one due to the Shaslyk construction the downstream section has a 40 mrad rotated hole matrix with respect to the upstream one; this

[3] NA48 readout maximum rate is approx. 10kHz, expected NA62 one is 1MHz.

[4] That is caused by the loss of absorber from the holes, in NA62 this is not anissue since the detector can be tilted to maximize absorber cross section.

provides higher cross section avoiding particles traversing the detector along the WSL fiber holes. Moreover the outer radius is not concentric with the inner one for both sections, breaking the detector axis rotational symmetry.

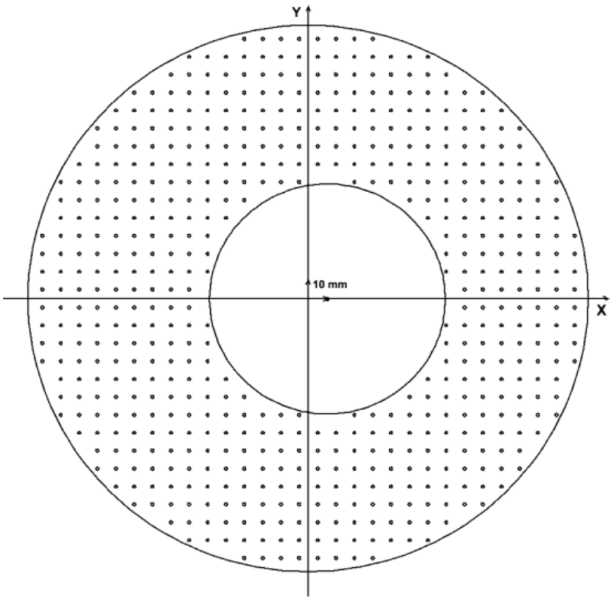

Figure 3.20: Layout of a IRC detector section. [4]

As was said in the introduction Shaslyk calorimeters are single channel, high rate detectors and readout is challenging. If the expected shower size is known it is possible to partition the detector and essentially make it multi channel. In the case of the IRC simulations provide an incoming photon mean energy of $25\ \mathrm{GeV}$, so a segmentation in four quadrants each having its own readout is performed via metallization.

3.9.2 Small Angle Calorimeter (SAC)

SAC is located at the end of the experiment, before the beam dump. It covers the region left by the LKr and the IRC.

It is a block detector comprised of 70 lead plates-scintillator layers, outer dimensions being $(205 \times 205 \times 220)\ \mathrm{mm}$; this equates to $\sim 17X_0$. Absorber plates and scintillator tiles have the same dimensions of $(205 \times 205 \times 1.5)\ \mathrm{mm}$; with absorber plates having 484 through holes drilled with 1.5 mm diameter; the holes are arranged in a 22 $\times$ 22 square grid.

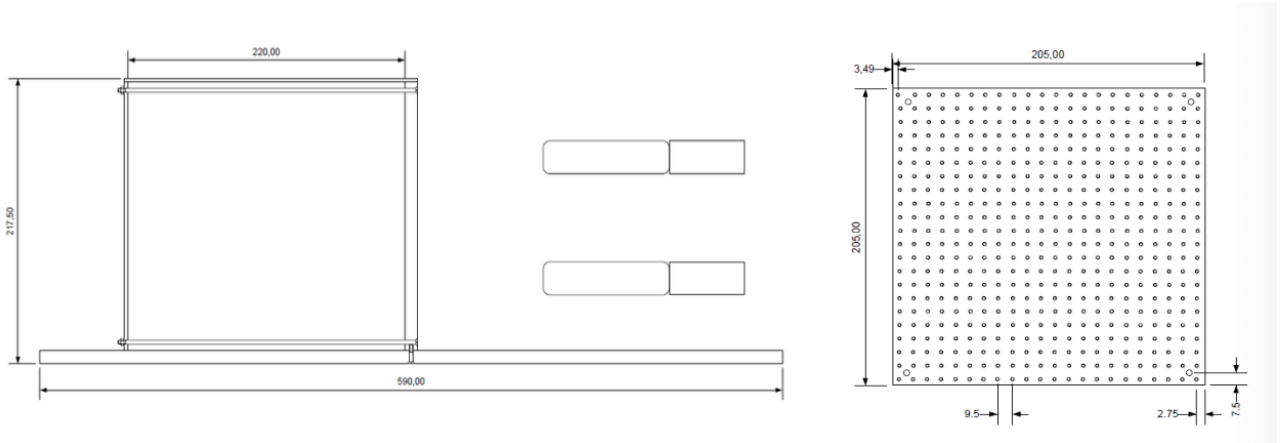

Figure 3.21: Layout of the SAC detector and the absorber plates. [4]

The SAC detector is housed in a round vacuum chamber around 7m long and 1m in diameter. In order to fine tune SAC position the detector is fixed to a moving platform whose position may be regulated externally; to avoid the Shashlyk inefficiency due to reduced cross section the whole detector is slightly tilted with respect to the beam pipe.

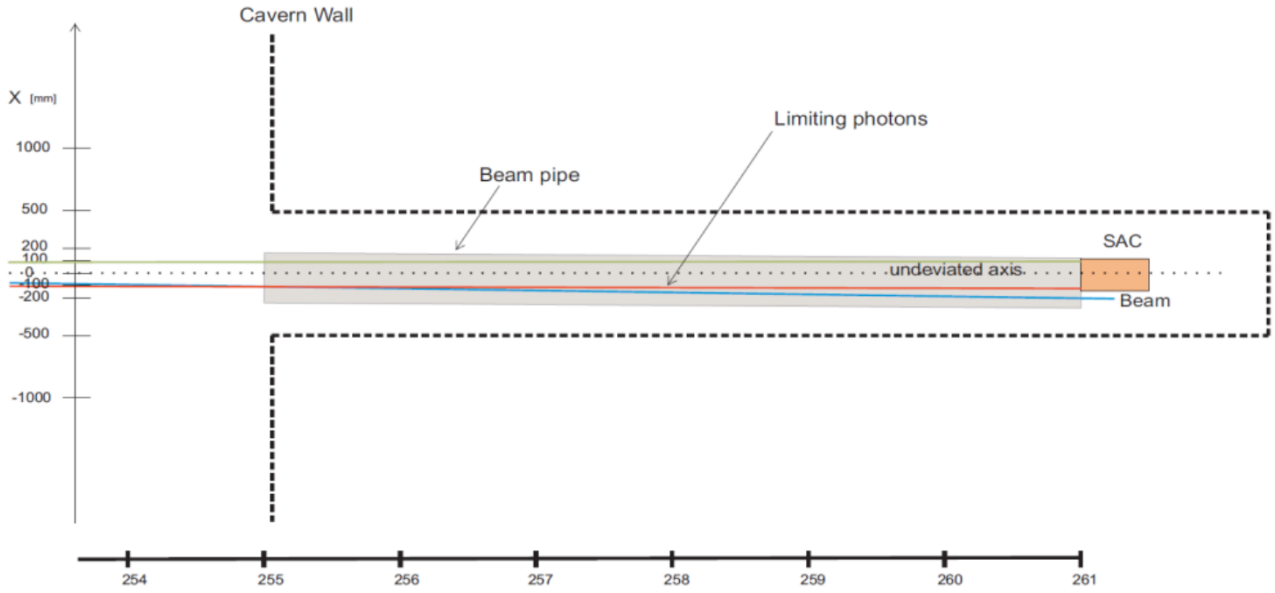

Figure 3.22: Location of the SAC detector in its vacuum vessel. [4]

Chapter 4

TDAQ at NA62

In this chapter an overview of the Trigger and Data Acquisition system (TDAQ) in NA62 will be given. This include a description of the triggering schemes as well as of the network and server infrastructure used for the data taking.

4.1 The SPS machine cycle

Being a fixed target experiment the trigger system of NA62 has to be designed to work synchronized with the SPS machine: the SPS unit of operation is a machine cycle (or *supercycle*). A supercycle has a duration of less than a minute and during it the machine may serve one or more customers (beam lines).

In Fig. 4.1 we show a typical SPS supercycle at the time of writing.

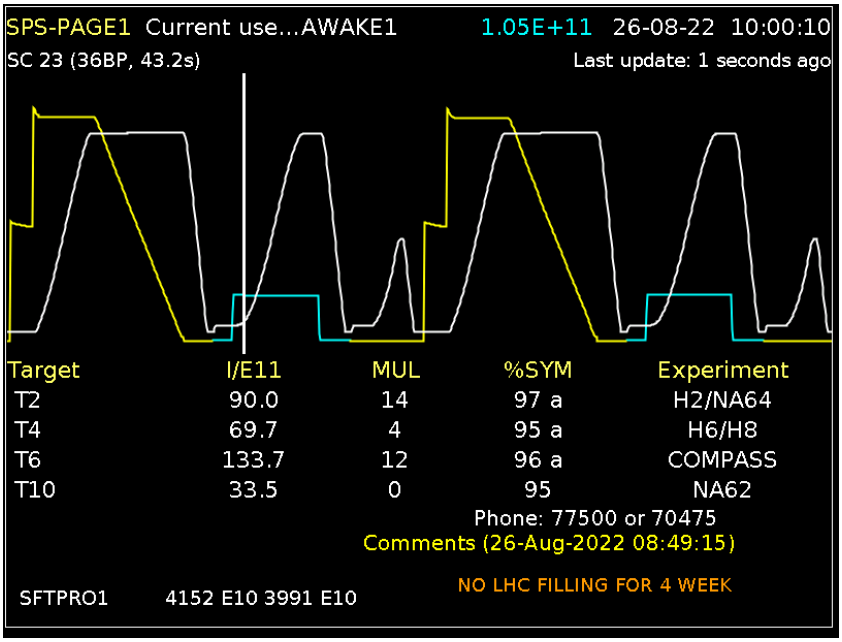

Figure 4.1: An SPS supercycle. the vertical white line shows current time within the cycle. Yellow line is proton current in SPS, blue line low intensity beam current and white line is extraction magnets current. Below the super cycle pictorial representation a list of the north area targets along with real time beam intensity is shown.

The part of supercycle concerning NA62 is the so called *burst*. This roughly corresponds to a slow extraction of the SPS beam towards the fixed targets; the TDAQ system has to be synchronized with it in order to collect events.

Proton current in SPS increases in a two-step fashion, each step corresponding to a PS injection; after current reaches the required level the extraction magnets are ramped up and the beam is slowly extracted when magnets reach nominal current. This time is called the *flat-top* and currently lasts 4.8 s[1]. During flat-top proton current decreases; this is the *spill*.

SPS provides timing signals to the experimental facilities corresponding to various phases of the supercycle, the WWE signals. These are interpreted to provide synchronization to the acquisition systems. In NA62 the most important product of this interpretation are the Start Of Burst (SOB) and End Of Burst (EOB) hardware signals. These happen some time before the actual spill starts and some time after it ended.

4.2 Trigger of NA62

4.2.1 Timing, Trigger and Control network

The backbone of NA62 triggering system is the TTC network. This optical link is composed by two unidirectional channels, one carrying the low level (L0) trigger signal and the other carrying trigger type and SOB-EOB timing information.

The TTC link is based on the ~ 40 MHz *master clock*. This clock provides timing to the whole experiment: all detectors use this information to compute time; they are equipped with a 32 bit master clock counter, providing the *timestamp*. Being 32 bit the timestamp dynamic range is ~ 25 ns and this is used as a rough timing measure. The timestamp is not precise enough to comply with NA62 timing requirements, thus it is complemented by an 8 bit *fine time* word that is counted from a high performance local[2] oscillator locked to the master clock. Fine time resolution is ~ 100 ps. Timestamp counter is started and reset by the SOB and EOB respectively, while fine time is rolled for each timestamp value. Both words are sent back to the acquisition system for data consistency checking and detector time alignment.

4.2.2 Trigger topology

NA62 has 12 sub-detectors; the detector raw data rates (19 GB/s for the GTKs alone) and the number of channels (~ 75000) make a "triggerless[3]" system unfeasible. Moreover, while being more flexible, a triggerless approach is useless to the type of research done at NA62: the vast majority of detector data comes from already well known and highly probable decay channels.

Following these considerations an hybrid three level triggering system is used:

- L0 hardware level trigger implemented on readout cards, rate ~ 1 MHz;
- L1 software trigger, based on subdetector information, rate ~ 100 kHz;
- L2 software trigger, based on partially reconstructed events, rate ~ 10 kHz.

L1 and L2 are collectively called High Level Triggers (HLT) and are implemented in the acquisition farm. The trigger system reduces the data throughput to less than 3GB/s for the whole experiment.

The detectors are divided in L0 detectors and L1 detectors; L0 detectors will send their data to the farm if a L0 trigger signal is issued, while L1 detectors will do the same only if an L1 signal is issued. Format and type of data being sent is dependant on trigger type.

When an SOB signal is issued via the TTC all detectors start filling up their buffers with timestamped data. A selection of detectors provides L0 triggers and this signals the front ends to transfer their data from buffers the acquisition farm. There L1 decision is made and data is ready for L2 evaluation. This is a general outline of the mechanism ignoring many details and exceptions. Further details on the High Level Trigger (HLT) are given in the section 4.5.4.

[1] flat top duration is limited by magnet heating.
[2] unique for each subdetector.
[3] Acquire-all and software-select.

The L0 trigger

The hardware trigger is provided by a selection of detectors and is based on simple constraints such as number of tracks in a tracker or no events in a veto. The main detectors involved in L0 are CHOD, NewCHOD, RICH, MUV3 and LKr. The front end of these systems will provide the *L0 trigger primitives* to the *L0 Trigger Processor* (L0TP). L0 primitives are transmitted to L0TP on a dedicated bus. An L0 trigger primitive is encoded in a 32 bit word that includes timestamp information and an encoded version of the conditions that are satisfied. All primitives are computed asynchronously and their production is severely time bound. Trigger primitives contain important information regarding beam characteristics and detector health: for this reason a *primitive parallel acquisition*, running on a dedicated cluster and providing data to one of the merger nodes downscales and saves them for reference and monitoring purposes.

A positive L0 trigger from L0TP instructs the L0 front end boards to pass their data to the DAQ farm; each detector system will send all data contained in a programmable window around the L0 timestamp, the *trigger window*. This is done to offload precise matching downstream the acquisition system. Data sent after an L0 trigger will be packed in the form a specific data frame; its content will depend on the L0 trigger word issued by the L0TP.

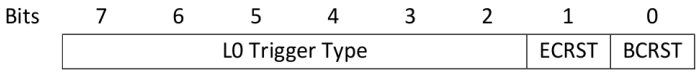

Figure 4.2: Format of an L0 trigger word issued by L0TP to the L0 detectors. The ECRST and BCRST bits of it are used to signal SOB and EOB L0 trigger types. [6]

If no L0 trigger word arrives to the detector in a specified amount of time then the front end boards will flush their buffers and restart acquiring data.

An important difference in L0 trigger behaviour is posed by the calorimeter readout (LKr-MUV1-MUV2). These detectors generates a far too large amount of data to be processed in the L0 time constraint (a L0 primitive algorithm must run in less than $100\,\mu s$). The solution is to have a dedicated L0Calo processor: the calorimeter readout provides skimmed information to it in form of summed signals every 25 ns, these information are small enough to be processed as an L0 primitive by L0Calo. The whole calorimetric data will be transmitted to the farm only after a positive L1 trigger.

The L1 software trigger

While L0 triggers are provided to the front end as clock synchronized signals the L1 trigger is asynchronous. This is due to the less stringent timing constraint on processing.

After a positive hardware trigger the L0 front end boards will send their data to the acquisition farm; there the L1 trigger algorithms are run on the data coming from a selection of detectors, namely the KTAG, LAV and STRAW.

Here we give a brief description of their function:

KTAG: This algorithm aims in rejecting all the L0 events that do not include an upstream Kaon. A K^+ will trigger most of the KTAG sectors in coincidence, thus acceptance criterion is at least five out of eight sectors activated.

LAV: The LAV algorithm suppresses all decays involving π^0 and thus γ; it checks that no more than two hits in ten out of twelve stations are registered.

STRAW: STRAW algorithm poses conditions on the downstream spectrometer tracks; Tracks are reconstructed by means of a Hough transform and some relevant quantities are computed.

- P_z, track 4-momentum along the beam direction, on this a lower limit $3\ \mathrm{GeV/c}$ is immediately applied;

- CDA or Closest Distance of Approach, that is the minimum distance between the tracks and the beam center.
- Z_{v}, the extrapolated position of the downstream track primary vertex.
- $(S_x, S_y), (S'_x, S'_y)$, being the x, y track slopes both before and after the spectrometer magnet; a limit is applied on those values as well, being $|S_x| < 20$ mrad, $|S_y|$ mrad.

When those quantities are computed a selection between *single track events* (STE) and *multi track events* (MTE) is performed based on CDA: multiple tracks all having $CDA < 30$ mm in coincidence are considered part of a MTE. After the tracks are reconstructed further cuts are applied depending on the nature of the run performed: two modes are currently implemented, PNN trigger and Exotics trigger:

- In PNN trigger mode the selection restricts the momentum range to the allowed by the physics ($P_z < 50$ GeV/c) and rejects all MTE or STE that are projected to be out of the fiducial decay region; this imposes a cutoff on both $CDA < 200$ mm and $Z_{\mathrm{v}} < 180$ m
- For exotic searches the constraints are the same as PNN but with a higher z-momentum cutoff of $P_z < 100$ GeV/c and the requirement of a negatively charged end product. This requirement imposes $S_x - S'_x < 0$.

A positive L1 decision will then trigger the L1 front end boards to send their data to the acquisition farm. All the information already computed in the L1 algorithms is saved and packed with the already received data in the so called *L1 packet*.

The L1 system includes an *autopass* feature: a small percentage of L1-rejected events is propagated forward to be used in high level trigger efficiency evaluation.

The L2 software high level trigger

This triggering operation takes place within the acquisition cluster and is based on more complex analysis performed over partially reconstructed events and interdetector correlations. The responsibility of this is entirely in the farm process; every node performs a partial reconstruction. If those conditions are met the event fragments are fit for in depth analysis and are sent to the archive.

A scheme of NA62 trigger flow is shown in Fig. 4.3, while some relevant trigger metrics are reported in Table 4.1

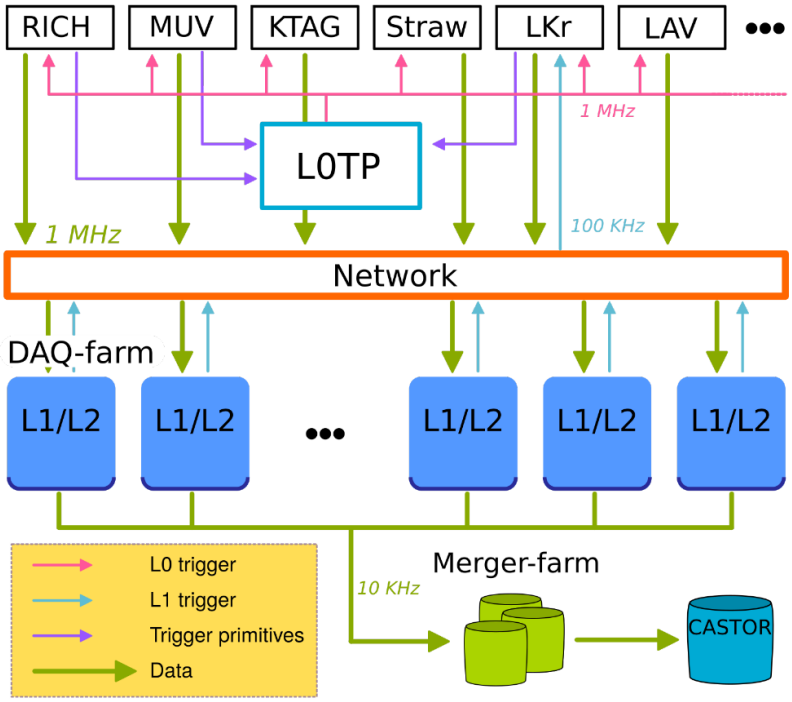

Figure 4.3: Trigger flow scheme of NA62. [7]

Parameter	Value	Description
f(L0) max	1 MHz	Maximum average L0 trigger rate
Δt(L0) min	75 ns	Minimum L0 trigger time separation
T(L0P) max	100 μs	Maximum latency for generation of L0 trigger primitives
T(L0) max	1 ms	Maximum total L0 trigger latency
f(L1) max	100 kHz	Maximum average L1 trigger rate
T(L1) max	1 s	Maximum total latency for L1 trigger
F(L2) max	$\mathcal{O}$(15 kHz)	Maximum average L2 trigger rate
T(L2) max	Spill period	Maximum total latency for L2 trigger

Table 4.1: Main trigger parameters for NA62

4.3 The TEL62-based readout system

The TEL62 [9] is an upgrade of the TELL1 board originally developed by the EPFL (Lausanne) for LHCb [31]; it is intended to acquire digital signals coming from Time To Digital Converters (TDCs) or in general any digital signal encoding information in it rising and falling edge times.

At the time of writing (2022) the TEL62 is the backbone of the readout system of NA62[4]. The TDCs used in this readout are based on the HPTDC ASICs [8]. These ICs support up to 32 TDC channels each. The minimal unit in NA62 is the *HPTDC mezzanine card*, providing 128 channels over 4 ASICs.

4.3.1 The HPTDC ASIC

This custom-designed chip is developed by the CERN microelectronics group and is a *fully programmable*, high performance Time to Digital converter based on the 250 nm CMOS process.

The chip uses a Delay Locked Loop (DLL) as the main TDC element; this DLL is driven via an internal Phase Locked Loop (PLL) that provides a programmable and low jitter clock from the master clock line coming from outside the chip. In NA62 the master clock is the common TTC ~ 40 MHz one; the PLL may be used to increase the internal clock and thus speed up the matching logic and the readout interface, the NA62 TDCs use the PLL in $\times 1$ mode, so only for reducing jitter. Notice that an auxiliary *coarse counter* is used to improve the dynamic range of the timing measurements.

In order to get a high level of integration and thus being able to fit 32 channels into one single IC most of the downstream logic of the TDC is shared among channels and managed by different forms of multiplexing: while each individual channel has a dedicated buffer in the form of 4 hit registers, thus being able of storing up to 4 different time measurements every group of sequential input channels share a common *L1 buffer*. Once a hit is registered on a channel the relevant timing information (DLL state providing the fine time and coarse time counter) is forwarded to the L1 FIFO.

Data in the L1 buffer is then processed by the *trigger system*. This is essentially composed by a *matching function* that selects among the L1 buffer all the events satisfying a given condition that fall in the programmed *trigger window*. Trigger information comes from outside the chip and can be temporarily stored within a small dedicated *trigger FIFO*. Following the trigger decision, data is moved from the 4 L1 buffers to a common, 256 word long, *readout FIFO*. This is then read out by the *readout interface* module that sends the data outside the chip.

A *token system* is natively implemented in the readout interface of the ASIC and provides means to daisy-chain up to 16 chips and make them provide data on a common bus. This is the technique used on the NA62 TDC mezzanine to read out the 4 ASICs.

A schematic of the HPTDC architecture is visible in Figure 4.4.

[4] notable exceptions are the STRAW and the GTK sub-detectors, see Section 4.3.3.

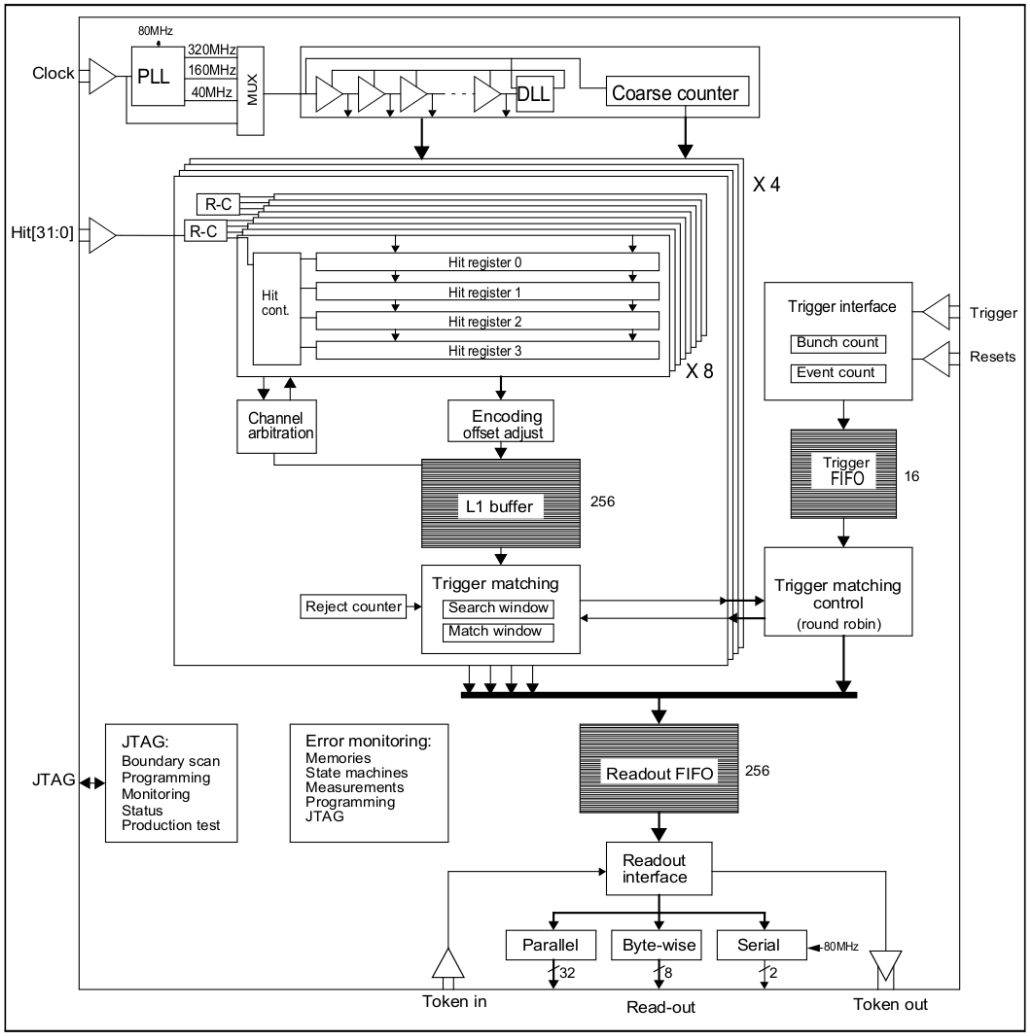

Figure 4.4: Block diagram of the HPTDC structure. Not described previously are the JTAG interface; used for initial setup of the ASIC and testing and the Error monitoring system for redundancy and radiation hardness. [8]

4.3.2 TEL62 readout board

The TEL62 board may install up to 4 mezzanine TDC boards. Using the HPTDC mezzanines employed in NA62 this totals to 512 channels.

Each mezzanine TDC is read out by a pre-processing (PP) FPGA; this device collects data and organizes them in packets referring to 6.4 µs windows. Packets are then written on the PP memory consisting of a commercial 2Gb DDR2 RAM module. The memory is paged and each page contains all the packets related to one time window; stored using a custom packing algorithm implemented in hardware. A block diagram of the TEL62 is shown in Fig. 4.5.

When the trigger line[5] connected to the PP receives a positive trigger all the data contained in a pre-programmed time length[6] around the L0 timestamp is moved from the memory to the Sync Link (SL) FPGA via an internal bus. The SL performs some fast processing on the incoming data and forms the triggered packet, this packet is stored on a fast SDRAM buffer and then extracted and sent downstream via four Gigabit ethernet links. UDP transmission is done on a custom network mezzanine card mounted on the TEL62 base board. Management and control is done via an AUX network link, connected to a microcomputer system (CCPC) that is interfaced to the FPGAs via glue logic.

[5]Can be either L0 or L1 depending on the context.
[6]given as number of 25 ns timestamps.

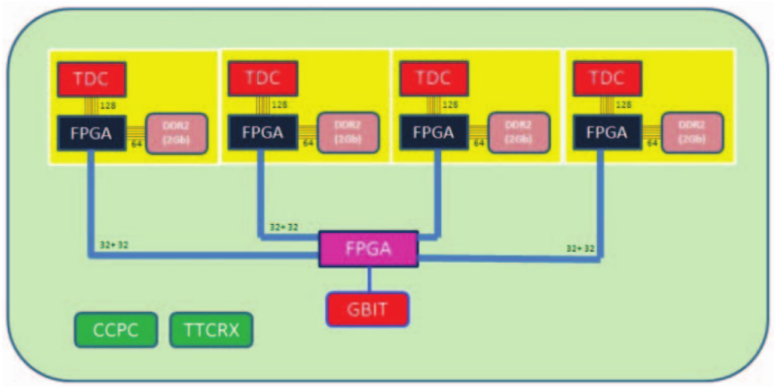

Figure 4.5: Conceptual schematic of the TEL62 board. [9]

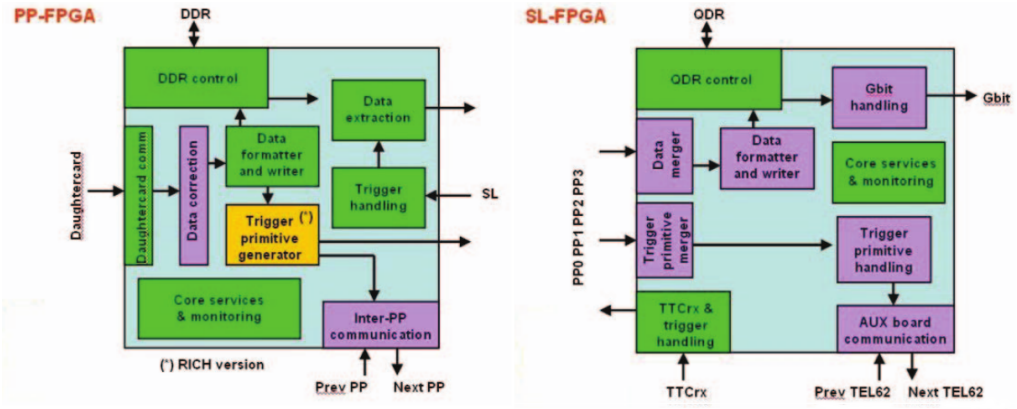

Figure 4.6: Block diagram of the Firmware of PP and SL FPGAs in the TEL62. "Doughtercard" may refer to a TDC board or any other front end digitizing board while "Gbit" refers to data directed to the ethernet output mezzanine. [9]

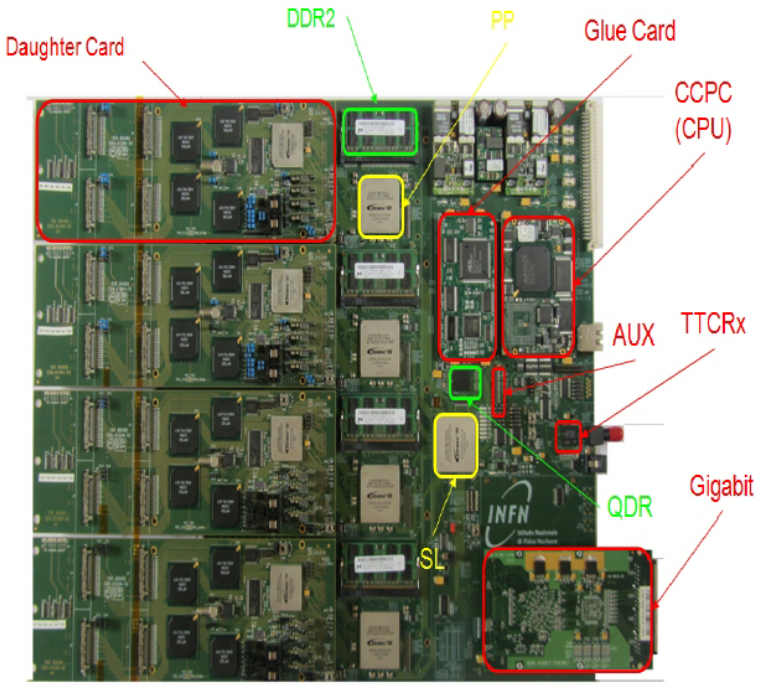

Figure 4.7: TEL62 picture. "Daughter card" may refer to TDC or ADC boards; in this sample 4 NA62 TDCs are installed [9]

4.3.3 Other NA62 readout interfaces

Calorimeter REAdout Module (CREAM)

The CREAM module is used mainly to read data coming from the LKr calorimeter. Number of channels and data output dictated the development of a custom readout board.

A CREAM module complies to the VME standard and is housed in a 1S 6U VME module. It is capable of reading 32 calorimeter channels with 14 bit dynamic range and with at least 10 effective bits. Channels are continuously digitized on board by 40MSa/s ADCs running off the 40 MHz master clock. CREAM features also an independent skimmed data generator for the L0 trigger system; the generator computes the *trigger sums* for the L0-calo processor.

The development of this system has been subcontracted to CAEN [32] by CERN; CAEN is responsible for both hardware design and firmware development.

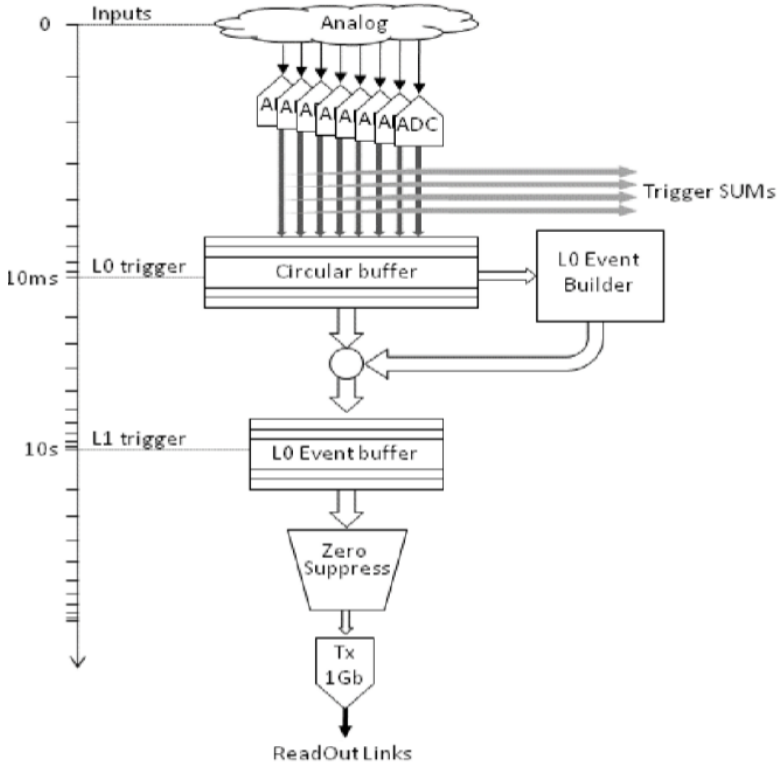

Figure 4.8: Block diagram of one CREAM module. [10]

GTK Readout

The GTK is characterized by extremely high rates. For this reason it uses a fully custom readout system.

An initial stage, based on custom ASICs (TDCPIx [33]) is located directly on the stations, together with a large memory buffer. This first stage receives the L0 trigger signal and upon positive L0 decision transfers all its data to the GTK-RO or GTK-ReadOut. From this stage data may be sent to a dedicated cluster of GTK servers that can pre process the data and forward those to the farm nodes. Both GTK-R0 and GTK cluster are located in the server room, far from the radiation hostile environment of the GTK stations.

STRAW Readout

The STRAW analog readout and time-to-digital conversion is performed by the *Cover* boards, capable of processing a group of 16 straws. STRAW front end is based on the CARIOCA custom amplifier

ASIC [34]. Due to detector requirements the time to digital converters used in the STRAW are FPGA based and implemented using shift registers, this allows for an extremely stable TDC system with very low latency. Downstream of the Cover the digitized data are read by the Straw Readout Boards (SRB); these boards include a memory buffer, L0 trigger signal processing and are directly synchronized with the burst cycle. The whole subdetector is read out by 32 SRB, each view of a tracker station is read by two boards. SRBs also house DCS related monitoring systems.

In order to mend this readout system with the existing experimental setup data is passed from SRBs to TEL62s; these perform event building, act as a bridge to the TTC infrastructure and sends data to the DAQ cluster.

4.4 Data acquisition cluster of NA62

In this section a general description of the NA62 server infrastructure will be given. This includes an overview of the network system as well as a short presentation of the typical management procedure workflow of the machines.

4.4.1 Cluster management

Common configuration tasks and software deployments on the cluster nodes are performed using *Ansible* [35]. Ansible is an agentless provisioning, configuration and deployment suite for Unix and Windows systems.

Configuration steps for nodes; such as package installation, settings and file transfers are defined as *tasks* following Ansible custom descriptive language. A series of tasks defines a *role*, representing a set of steps that are needed for setting up a specific function: an example of role may be "DIM-Interface", containing the installation of the DIM software and the DIM server environment variable addition. Execution of tasks on the nodes, or *hosts* happens through *playbooks*. A playbook is a collection tasks to be performed on hosts, given either as raw tasks or through roles; when the playbook is run an SSH connection to the relevant host(s) is opened and all the required tasks are executed in the order given by the playbook. Hosts may be grouped and groups may be associated to evocative names through the *inventory*, this is a file containing the list of physical addresses of hosts and determining their grouping; when calling a playbook for running an inventory has to be specified so hosts can be resolved.

Most Ansible modules are designed to be *idempotent*; this means that a preliminary check on the status is performed before the actual tasks run: if the state is already the desired one, the task is skipped. This mechanisms assures that a playbook may be run multiple times even when only a subset of tasks are actually required to run again. An idempotent system makes management easier, is less error prone and enables to fast modification deploys. For example: a small change in a configuration file may be deployed on a cluster by running the same playbook used for the initial setup; all unnecessary steps will either be skipped or will not make any harmful change.

4.4.2 Network system

Front end boards from each detector provide data over copper Gigabit Ethernet links. All links employ UDP [36]: this has been chosen over TCP [37] due to its straightforward implementation in FPGAs and lower latency. Their unit of information is called Multi Event Packet (MEP) and contains an aggregation of multiple events, usually 8.

All the links coming from the front-end are connected to a series of *aggregators*. These are network switches that collects all data coming from the front end and routes them to the *core network system*. This is done to reduce the number of network links coming from the experimental hall: aggregators are located directly in the cavern and aggregated data comes to the core network and thus the farm over a smaller number of fast, optical uplinks. This topology has also the advantage of moving all

the sensitive network and acquisition infrastructure from the radioactive cavern to a controlled server room environment, leaving only the aggregator switches and the front end boards exposed.

The core network is composed by four (10-40)GbE routers and one 40GbE router. Each (10-40)GbE router has 48 10Gb ethernet optical links that are connected to the aggregator switches and the DAQ cluster machines. Their 6 40GbE links are connected to one main 40GbE router, forming four redundant trunk links. The use of commercial enterprise network gear for DAQ applications poses some challenges: in their typical use case the traffic is expected to be well balanced on each link and to be roughly symmetric. In NA62 and in DAQ application most of the traffic is unidirectional (from detectors to acquisition) and timed following the SOB-EOB signals. To accomodate this special use case the network infrastructure has to be over designed and custom firmware for the devices as well as specialized network drivers on the servers are needed.

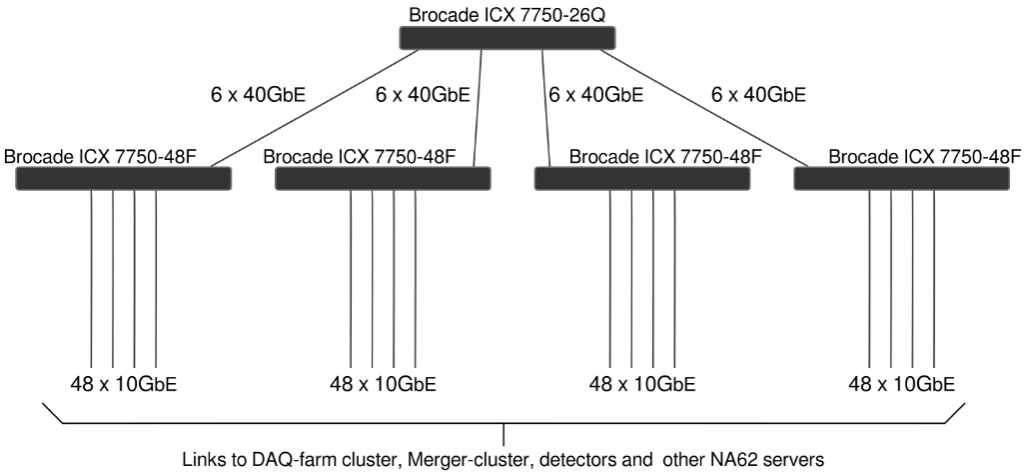

Figure 4.9: Schematic of the Core Network. [7]

Complementing the core network a *DAQ aggregation network* manages the connection from the core network to the various machines in the DAQ cluster. This is based on a (1-10)GbE switch and uses the TCP protocol.

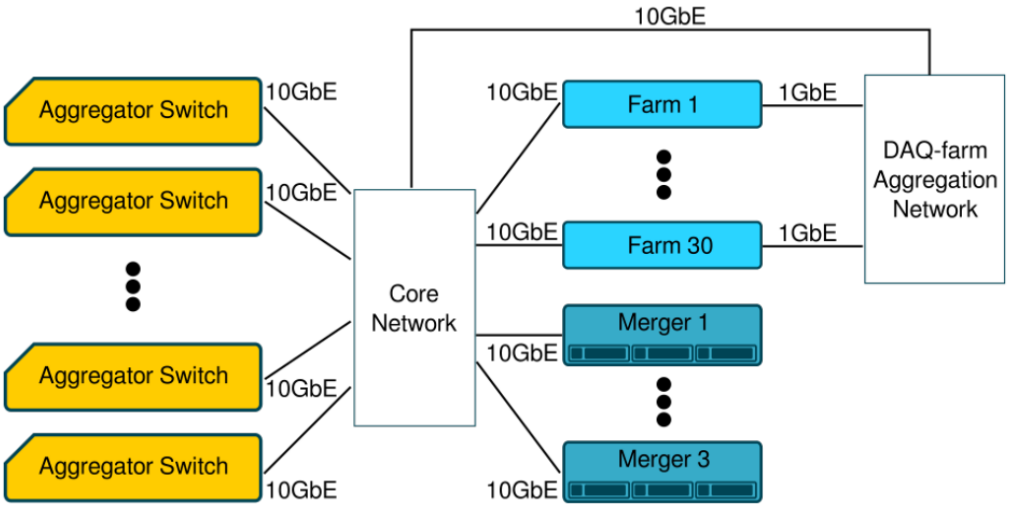

Figure 4.10: Structure of the network system of NA62. Notice the double connection of cluster nodes: directly to the core network and through the DAQ aggregator via slower 1GbE links. Direct core connection is exclusively used for the data coming from the cavern while DAQ aggregator connection is used for all other tasks, such as farm-merger transfer. [7]

4.4.3 Server characteristics

The computing infrastructure of NA62 is housed in a dedicated server room. It includes:

- 6 GTK readout servers; managing the information coming from the Giga Tracker readout system;
- 30 farm nodes (na62farm1-30); these receive the data from the detectors, perform HLT and build serialized events.
  Farm nodes are equipped with dual `Intel Xeon E5-2650 v2` CPU, each containing 8 HT cores and 64GB of DDR3 memory. Network connectivity is provided by a 10GbE NIC with custom drivers managing central network links and a standard driver 1GbE NIC for DAQ interface network links.
- 4 merger nodes (na62merger1-4); used to collect the events from the farm nodes and merge them into raw files, one file per burst;
  Merger machines have different technical specification, Merger4 has been added as part of the 2020 upgrade
  Merger1: dual `Intel Xeon E5645` CPU, 6 HT cores per CPU; 48 GB of RAM.
  Merger2: dual `Intel Xeon E5-2630` CPU, 6 HT cores per CPU; 32 GB of RAM.
  Merger3: dual `Intel Xeon E5-2630` CPU, 6 HT cores per CPU; 32 GB of RAM.
  Merger4: dual `Intel Xeon Silver 4210` CPU, 10 HT cores per CPU; 93 GB of RAM.
  The merger machines provide a first line of storage of the burst files. Merger 4 also hosts the file transfers database. In order to comply with the high throughput situations data storage is handled by RAID arrays with dedicated hardware controllers.
- 4 online monitor nodes (na62monitor0-3); running the online monitor reconstruction software on raw files.
  Each monitor has dual `Intel Xeon E5-2620 v4` CPU, 8 HT cores per CPU; 128GB of RAM.
- 2 development machines to develop, test and deploy the DAQ software (na62farmdev3-4).

4.5 Data acquisition software at NA62

Lets now describe the current DAQ software used within NA62. The software sources are kept in CERN GitLab internal repositories; the most relevant being:

NA62-farm: farm software base repository;
NA62-farm-lib: shared library containing all common software components;
NA62-trigger-algorithms: the high level trigger algorithms used in the farm software;
NA62-farm-merger: merger software base repository.

The software relies on some additional packages and libraries, here we will give an introduction to the most relevant ones.

4.5.1 The DIM interface of NA62

DIM is an inter-process communication library [38], it is used in NA62 as the core for communication between the experiment Run Control (RC) and the DAQ processes.

All software instances for Farm, Mergers and Monitors have a DIM command interface built-in; this listens to commands issued by the RC in the form of messages, for example:

- Start;
- Stop;
- Restart;
- Start Of Burst;
- End Of Burst;

An issue of the Start command will trigger a fork of the DIM interface process and subsequent start of the relevant DAQ software with a set of parameters passed by the run control[7]. A stop signal will trigger software termination and a restart is just a stop and a start in sequence. Other signals may trigger firmware reloads, or rebooting of an hardware node. A very important role is covered by SOB and EOB DIM messages, sent automatically following the SPS cycle WWE messages: The SOB signal is usually used to mark the beginning of the data acquisition time, while EOB to mark the end of it. The specific function triggered by these messages varies among different pieces of software: in the farm is associated with special SOB-EOB event production and is not implemented at all in the merger software (asynchronous running).

Another role of DIM is information exchange: A dedicated machine (DIM server) publishes information that may be read out by one or more subscribing services. The information usually transmitted via DIM can be SOB-EOB timestamps, or hardware monitoring metrics.

4.5.2 The PF_Ring driver

Due to the high throughput the farm nodes need to cope with the 10GbE Network Interface Cards (NICs) to the core network use a special custom driver named `PF_RING` [39]. This improves UDP packet collection efficiency by bypassing the ordinary kernel network stack and scattering the incoming packets on different buffers. A thread pool accesses the buffers and provides the packets to various consumers. The idea is similar to the thread pool system implemented in the merger software, that will be discussed in Section 4.5.5.

`PF_RING` replaces the standard network stack for the NIC; all low level networking operations such as ARP requests and IP defragmentation have to be done at application level, in this case by the farm software. The 10GbE interface is managed by it and is exclusively used for detector data reception from the front end boards.

At the time of this work there is an ongoing effort in replacing the proprietary PF_RING with the open-source Data Plane Development Kit (DPDK) [40].

[7] in case of the farm the set of parameters is named *farm string*.

4.5.3 The ZMQ messaging library

Zero-MQ is a commercial network library [41]. It allows in-process, inter-process and network message exchange either unicast or multicast. The library provides an abstraction layer in the form of *sockets*, representing a communication channel between processes. Sockets come in different types based on the link logic; ZMQ supports standard PUB-SUB connections, PUSH-PULL structures for computing nodes, REQ-REP exchange links, pure bidirectional PAIR links and many others.

Internally a socket may be connected to a specific network port or bound to one, details of these operations depend on the link type. All socket inner operations, such as buffering, scheduling and polling are managed within the library and in principle require no effort in the user code: user I/O is performed by means of sending and receiving functions, in blocking or non-blocking mode.

Information exchange through a socket happens via *ZMQ messages*. Messages are atomic entities of a specified size containing the information to be sent over the link. The library treats all messages equally as blobs of data: there is no type restriction and all the operations on the incoming or outgoing data, such as encoding-decoding, are up to the user to implement based on their application. ZMQ allows also sending multipart messages, this may be useful for large messages. Internal implementation details on message sending formatting or packaging are irrelevant for the user because ZMQ guarantees either complete message delivery or no delivery.

This messaging library is widely used in the NA62 DAQ software: in the following every reference to a network socket refers to a ZMQ socket.

4.5.4 Farm software

The farm software manages the HLT, performs L1-L2 event building and transfers the serialized L2 events to the merger cluster for raw file creation. As of 2022 it also provides a subset of events to the online monitor (OM) machines for fast monitoring, the details of this will be given in Section 4.6. Cluster nodes receives SOB-EOB signals via the DIM interface: after an SOB data may start to be sent by the L0 front end. Following an L0 positive decision each farm node receives $1/30$ of the total MEPs, received MEPs are buffered within the software. Detector data is distributed following a round robin scheme, this results in a transfer rate of $\sim 12 \times 10^4$ MEP/s per machine during burst time.

The core of the farm software is the *event pool*: here event fragments from MEPs are stored, then separated and grouped into L1 events based on timestamp and event number. An L1 event is considered closed once all detectors contributing to it sent their information to the corresponding farm node. If some information is missing from some detectors there is an issue (lost packets) and this needs to be acknowledged and fixed. The same applies for invalid event checksum. Once all L1 events are in the event poll the L1 trigger algorithms may run. If L1 trigger issues a negative decision the corresponding L1 event is scrapped. Meanwhile, in case of positive L1 the farm requests all the missing information; the request is performed by means of Multiple Request Packets (MRPs) sent back to the L1 front end, these are UDP packets containing the requested event IDs. When a given L1 source receives an MRP from a farm node it has to respond with the requested events, this happens via SubDetector Event (SDEs) packets, similar to MEPs but containing only one event per packet due to board firmware limitations. the SDEs will be routed back to the farm node issuing the MRP request. After all L1 sources have sent their data one or more L2 algorithms may run and further thin out the accepted event pool. Once this is done the completed L2 events, each having a size ~ 15 kB gets serialized and sent to the merger cluster via a network socket. Another analogous network socket sends a sub selection of the L2 serialized events to the online monitor machines. Selection is done via trigger flag and L0 trigger word masks. These communication channels are through the farm 1GbE NIC and use the TCP protocol, here the low latency of UDP is not needed.

If the farm software receives an end of burst (EOB) signal through the DIM interface it waits an additional six seconds and then flushes the event poll, ready for a new burst. We recall that the front end boards will send special packets when hit by SOB-EOB signals. The EOB packet for a given sub-

detector contains statistics and metrics collected during the burst and is important for monitoring[8] (for example it contains number of trigger types). These packets get scattered over the whole farm cluster and the corresponding reconstructed events are serialized and sent regardless of their completeness to the sockets once the event poll gets flushed. The merger software needs EOB information and fetching it parsing all 30 EOB events coming from the farm would be expensive and complicated; for this reason each farm node publishes all EOB information it has on DIM, so mergers can subscribe and collect that information easily.

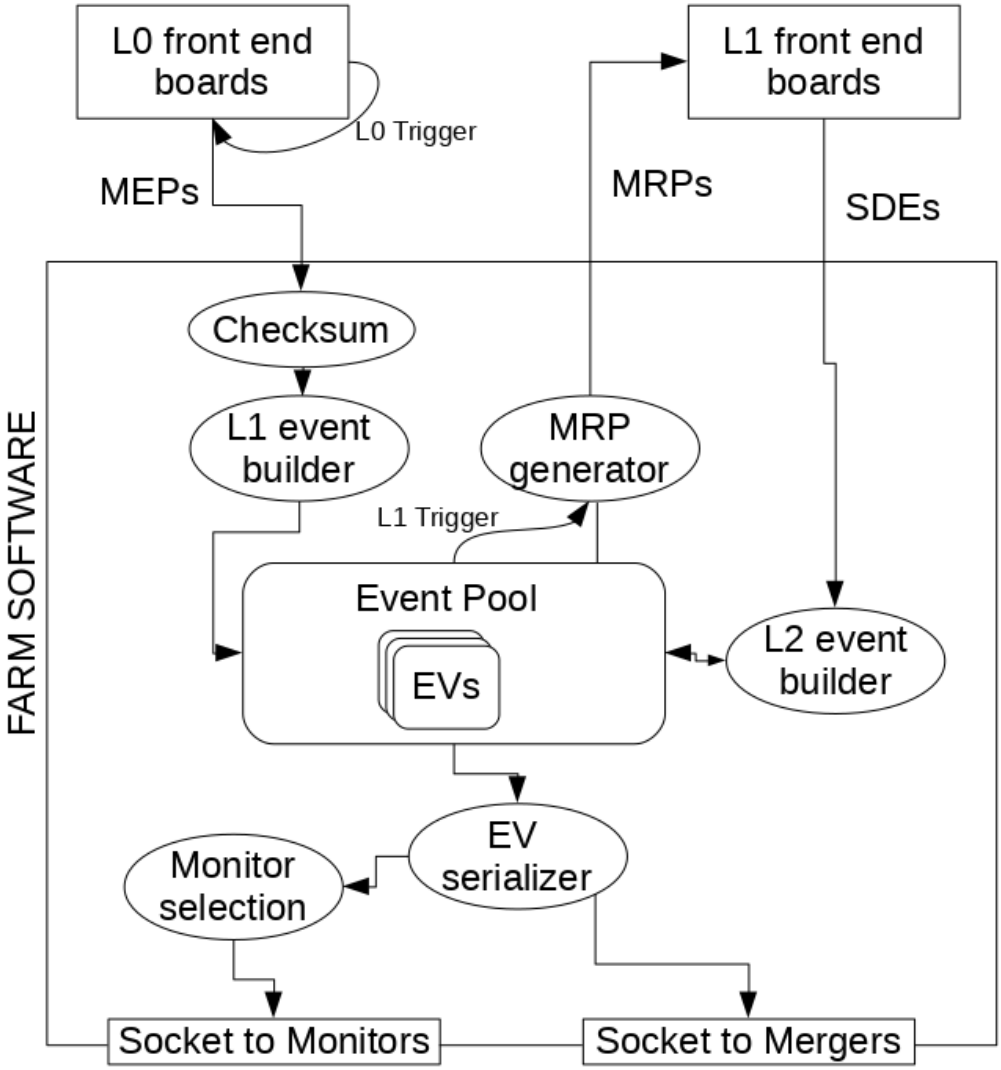

Figure 4.11: Simplified schematic of the farm DAQ software function as of 2022. The network infrastructure carrying the communications is not shown.

4.5.5 Merger software

The merger software is responsible of the creation of raw event files to be stored and reconstructed. A "raw event file" is a collection of events sharing the same burst ID together with detector metrics and end of burst information. Starting from 2022 the merger software may run in DIM synchronous or asynchronous mode, writing time ordered or unsorted raw files. This is part of the SyncroMerger modification that will be introduced shortly.

The core component of the software is the Merger object: this manages the data structure hosting the events. Both data structure type and object behaviour depends on the chosen merger object; choice between asynchronous and synchronous operation is done as a configuration parameter, synchronous

[8] An EOB monitor is used in the control room to check data quality together with the Online Monitor.

merger being a derived type of asynchronous merger. Meanwhile choice between sorted or unsorted file writing is performed at build time choosing the sorted or unsorted asynchronous merger as a superclass for synchronous merger.

Regardless of merger type the ancillary software structure is the same.

Upon startup the main merger software thread instantiates whatever merger type is requested, the writing thread pool and the monitoring services for the Run Control. It then starts an endless loop in which polls the network socket for events coming from the farm. The received events are then enqueued in a `tbb::concurrent_queue` instance within the merger object. A dedicated thread, also within the merger object will deplete the queue and insert the elements in the data structure. This is done to decouple socket event reception to event data structure insertion: if events are being fetched from the data structure to write files it is not possible to insert elements into it; a thread safe event queue can effectively collect incoming data from the socket and pass them to the data structure whenever it is not busy with a file write access operation.

File writing is performed by a configurable size pool of threads; its default number is 4 so the merger software may in principle write four raw files simultaneously. A thread is awaken when events corresponding to a burst start to arrive in the data structure and will fetch the bursts and write them in the raw files. Status of the individual threads in the pool is governed by an `std::condition_variable`. A write timeout check is implemented; should no events arrive for a given burst for a configurable amount of time the thread pool will write the raw file even with missing information.

Besides the input events queue and the merger thread the event data structure topology depends on the merger type:

MergerNoCache — unsorted events

The main structure is an integer-keyed `tbb::concurrent_hash_map` of smart shared pointers to "OnGoingBursts". This type is a container for unordered events in the form of a pointer queue and an EOB packet with the same burst ID. The key of the hash table represents a burst ID, so in principle the structure can accumulate events from multiple bursts at once.

When an event is extracted from the input queue the merger thread checks if the corresponding burst ID is already present in the hash table; if it is present the event will be added to the appropriate OnGoingBurst queue, if it is not a new OnGoingBurst will be created and an awake signal will be sent to the writer thread pool to take care of it[9]. The thread will write the events and the EOB information coming from DIM as soon as they are available, ignoring the overhead produced by the thread safety.

Merger — time ordered events

Instead of an hash table essentially containing unsorted queues for every burst ID received the time ordered merger uses a two dimensional map structure, implemented by an `std::map`. Events in the map are keyed with their burst ID and event number, when elements are inserted into the map by the merger thread the map is sorted by event number. If no events for a given burst ID get fetched from the queue in 10 seconds (a time deliberately chosen to be larger than the spill length) then the corresponding map is considered complete and a writer thread from the pool is awaken and proceeds to write the sorted events to the corresponding burst file, together with EOB information coming from the DIM interface. It should be noticed that `std::map` is not a native thread safe data structure, so in this version thread safety is implemented by means of another map containing pointers to `std::mutex` and keyed by Burst ID: when an event of that burst ID is fetched from the event queue the corresponding mutex gets instantiated.

[9] this will trigger a new thread within the poll to write a new raw file, if no thread are available data will be stored in RAM until a thread is available.

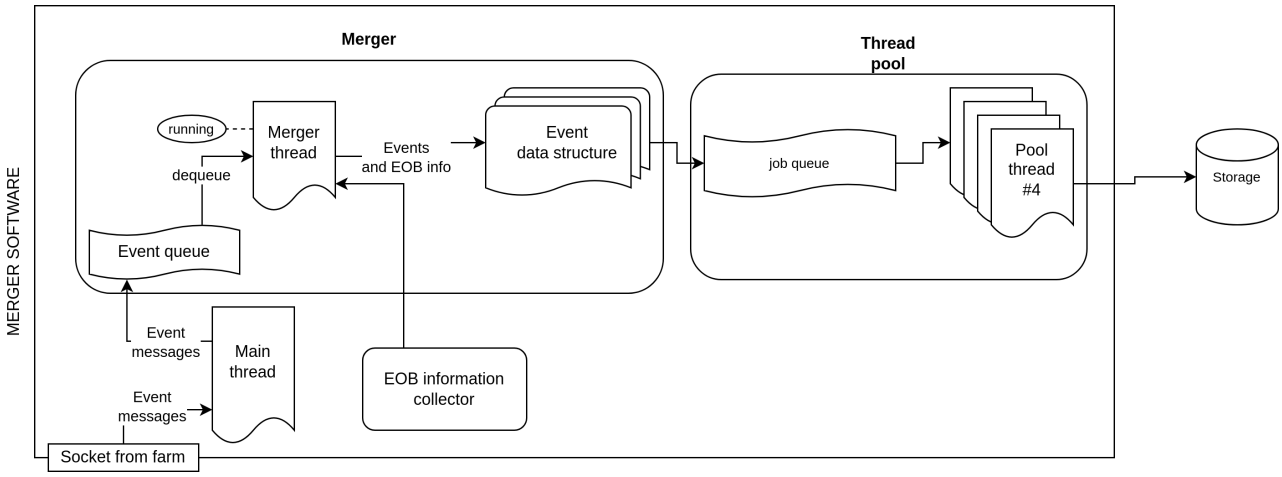

Figure 4.12: Simplified schematic of the Merger DAQ software. Only the general dataflow is shown. Technical details on the event data structure and the various objects are detailed in this section.

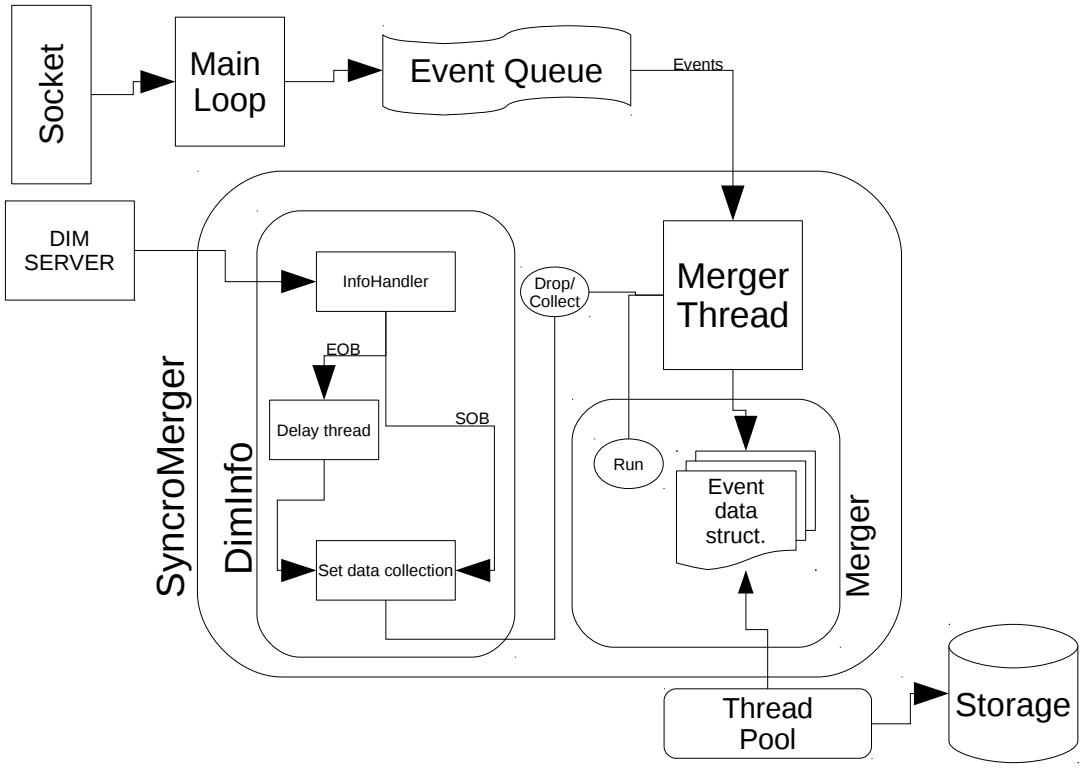

Figure 4.13: Schematic of the SyncroMerger software. Here the delay thread is the function that gets triggered by the EOB from the InfoHandler function. The Drop/Collect flag is part of the SyncroMerger class and is triggered through a SyncroMerger pointer available to the inner class DimInfo.

Synchronizing the merger software to DIM

The merger software was originally designed to run asynchronously from the experiment, especially regarding the spill cycle start and end; the idea being that event data may come from the farm even after the end of burst (in fact the L2 trigger algorithm can run by design in the interspill period) and all priority should be given to event collection efficiency rather than speed.

The SyncroMerger was developed as a derivative of the ordinary, asynchronous merger. This derived class overrides the merger thread function with an implementation that drops or fetches events from the socket queue based on an atomic flag. This flag represents the burst status: it is toggled by the end and start of burst signals issued by the DIM interface. In order to fetch this information the

class includes an instance of the DIM class DimInfo. DimInfo provides a way of subscribing to DIM services and offers several functions that can be overwritten to suit different needs: Syncromerger subscribes to the `BurstTimeStruct`, which contains SOB and EOB timestamps. At every SOB the relative timestamp is updated and the EOB one is set to zero, and updated at EOB. Every time the status of `BurstTimeStruct` is updated a function called `infoHandler()` is triggered and toggles the atomic flag for the merger thread accordingly. This approach has actually one severe limitation: In normal data taking conditions many important events such as the EOB packet may arrive after the EOB signal; a synchronized merger process built the way described before will drop all these events. In the actual implementation it is possible to delay the flag toggle with respect to the EOB by a fixed amount passed in the software configuration string. This *EOB dead time* is implemented by letting `infoHandler` trigger a `Boost::signal()` at EOB; the signal is bound to a function that waits the given EOB dead time and then toggles the flag. Please note that it is not possible to implement this dead time by a sleep statement in the `infoHandler`: that will result in a lock of the DIM listener thread; Since the InfoHandler gets executed at every state change of the burst time structure if its execution takes a long time (in this case due to the sleep) all changes within this time to the burst time structure will be ignored and the calling mechanism of infoHandler will hang.

4.5.6 Monitor software

The online monitor software enables fast data quality checks by the shifter personnel, it runs on the monitor cluster.

A reconstruction software instance provides information on detector health in the form of sub-detector specific plots (Online Monitor). Another monitoring software is the *EOB monitor*; this provides untriggered EOB information on the sub-detectors and may be used to spot issues faster thanks to its lower latency compared to the Online Monitor. L0 trigger primitives may provide information on the detector health, these are collected and stored and a *primitive monitor* software processes them, this software also provides beam quality information. All the plots produced by the monitor software are visible in the control room and shifters can take action if one subdetector shows issues.

Before the current work the online monitor software relied on copied raw files from the mergers; since the OM software is based on the NA62 analysis framework with minimal modification it is designed to work with raw files on disk. Each completed event file was copied to the monitor machines via a service running on the merger cluster nodes. This approach is clearly non ideal for online monitoring purposes: copying a multi-GB file over the network is an expensive operation ($\sim$ 1min) and most of the events contained in it are useless to the online monitor anyway, we remind to Section 4.6 for further discussion.

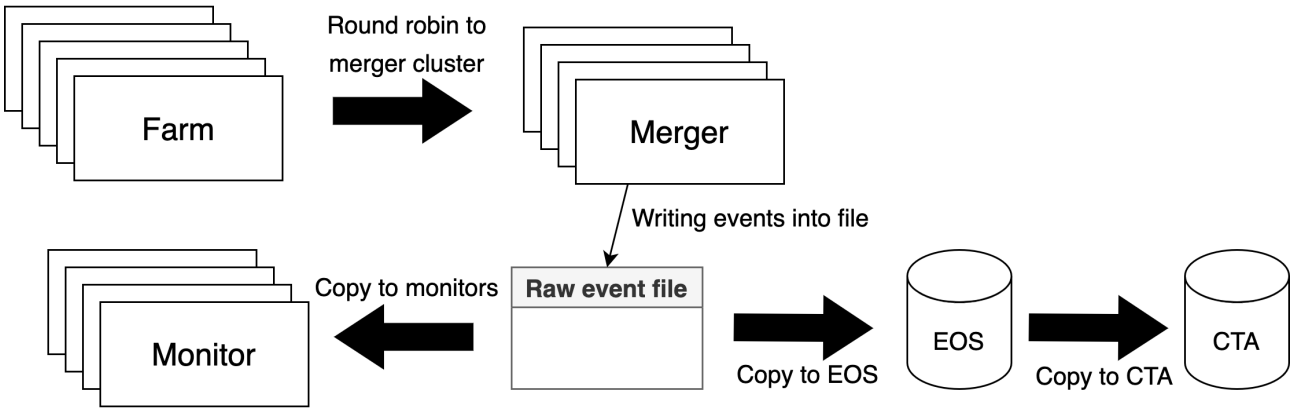

Figure 4.14: Farm data flow before online monitor speedup. Data files are copied from the merger cluster to the online monitor.

4.6 Speeding up the online monitor

In 2022, a proposal has been made to modify the data path for the online monitor in order to reduce latency.

The farm software is already set up to send all events to the merger cluster. With minimal modification it is possible to forward a subset of those events to the monitor cluster: there an instance of the merger software can run and produce the raw files directly on the monitors disks. If all events are sent to the monitors the network load on the farm nodes will double (from 50 Mbps to 100 Mbps), still staying well within the limits of the network interfaces. By applying a selection the load may increase by only 20%, or around 10 Mbps.

The modification has been deployed in summer 2022 together with the new, faster unsorted merger software. Results show an improvement of $\sim 100\%$ on latency.

A further, proposed modification would require an intervention on the online monitor software: instead of reading disk files produced by a local instance of the merger software the monitor software may access the socket directly, collecting and processing the events as soon as they arrive. This would certainly reduce the online monitor latency but its implementation would require significant rewriting of the reconstruction framework code base, an effort deemed to expensive for the time being.

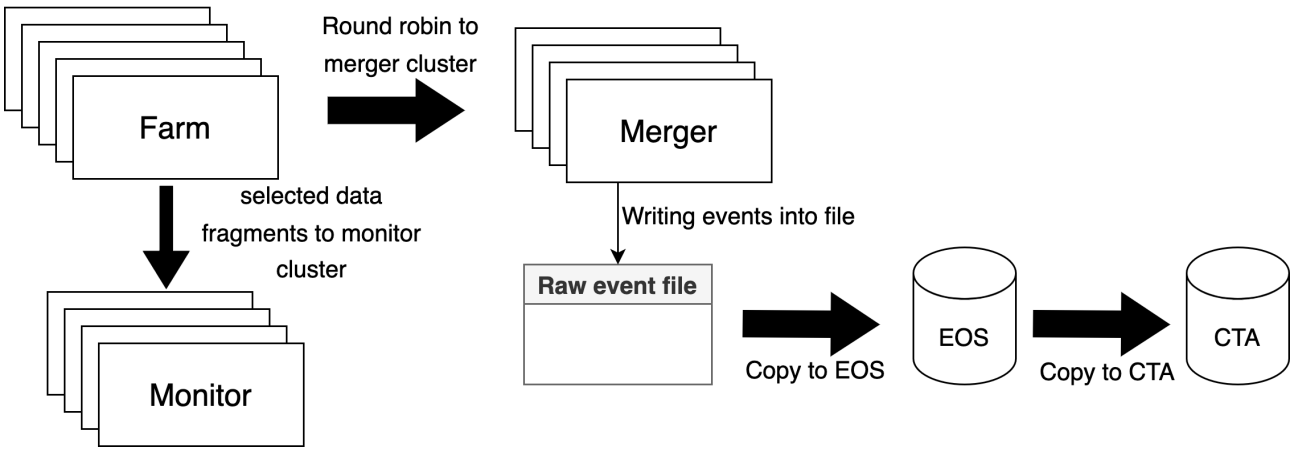

Figure 4.15: Farm data flow after online monitor speedup. The event fragments are directly forwarded to OM cluster, there an event file is created by a merger instance or data are collected directly from the socket.

4.6.1 Farm software modification

Farm software component `Storage Handler` manages the round-robin event sending to the mergers. It includes a sending thread that pushes events to the network socket, this thread consumes the serialized events inserted into a queue by the L2 algorithms.

The original software thread consumed events from the event queue without any further action.

In the new version the event queue is replaced by a queue of `std::pair`[10]; first element of pair being a boolean flag signalling the event is selected for monitor forwarding and second element being the actual serialized event. Other modifications are just doubling of the initialization code for the new network socket instantiation and termination upon software shutdown.

Selection of events is performed in the `SendEvent()` function: here the L0 trigger mask and the trigger flags are extracted using the event-specific function and one settable masks is applied to each string.

[10] logically these are structs containing two arbitrary types.

Bitmasks

A bit mask can be used to verify if a specific bitset satisfies one or more conditions: suppose one wants to filter all 8 bit words having MSB = 1 and LSB = 1, while having bit 2 low. One can show that the previous condition can be written as:

$$(S \ \& \ 0b10000001) \ \& \ (\bar{S} \ \& \ 0b00000100). \tag{4.1}$$

An equivalent form may be written using logical or. The words used in these sort of tests are called bitmasks.

4.6.2 Selection masks

According to Fig. 4.2 the L0 trigger word is composed by a 6 bit L0 type and 2 special bits. The OM software requires several special trigger types, here we list them together with their relative bit word:

- Synchronization (0b100000);
- SOB events (0b100010), EOB events (0b100011);
- Random (0b10110x);
- Calibration (0b11xxxx);
- Monitoring (0b101000).

Notice that the OM monitor does not need *Physics type* triggers, those have type always in the form (0b0xxxxx). Non Physics triggers can be selected using the following mask:

$$TM = 0b0111111 = 0xFC = 252. \tag{4.2}$$

This mask leaves also other non physics triggers, such as Choke on/off, but their number is negligible and thus is not an issue[11].

Regarding the trigger flags we request only the calibrations, these are identified by 0x04 and thus the flag used will coincide.

The final selection condition is then:

$$(\text{L0Word} \ \& \ 0xFC) \ \text{OR} \ (\text{Flags} \ \& \ 0x4). \tag{4.3}$$

[11] Having multiple masks for trigger type would render configuration of the software more difficult for minimal gains.

Chapter 5

NA62 DAQ Upgrade

In this chapter an overview of the DAQ upgrade proposed in April 2020 [42] is given.

The upgrade is intended to replace the aging TEL62 readout board and its associated HPTDC front-end Time to Digital Converter (TDC) with a new TDC module read out by the FELIX card, developed for the ATLAS experiment. This new readout system addresses many lingering issues of the current TEL62 one and will pose the basis for future nominal intensity and beyond nominal intensity runs, as well as setting a new standard for future Kaon generating experiments.

NA62 operated with lower than nominal intensity from the start of data taking in 2016 all the way to 2018; after 2018 the intensity was eventually brought to 100% and while the TEL62 readout managed to cope with these conditions it has no headroom for future improvements and higher than nominal intensity runs. While the TEL62 performance was marginal a severe bottleneck was shown with the L0 trigger system, given that a FELIX readout has the potential of not needing it in the current form the upgrade may prove advantageous.

5.1 TEL62 readout system limitations

The HPTDC [8] mezzanine used in the current TEL62 readout system has very limited on chip buffering; it is completely dependant on the TEL62 card to provide the required trigger signals, to buffer the data and to forward them to the server acquisition farm through network. The design constraints of the HPTDC as well as the limited buffering and processing ability of the TEL62 readout poses a limitation on trigger topology as well as on event rate.

In 4.3.1 we described the general structure of the HPTDC chip: in order to cram together 32 channels per chip and keeping a reasonable silicon footprint it is necessary to share most of the downstream logic among channels. This is achieved through multiplexing the hits in the L1 buffers and multiplexing those buffers into a single output queue. While multiplexing enables high integration it is at the source of many performance limitations.

5.1.1 Channel merging bottleneck

When pulses arrive to a given channel (refer to Figure 4.4) the edge timing is forwarded to the hit buffers. Since there are 4 hit buffers per channel there can be at most 4 hit timestamps in the registers[1]. These hit buffers act as a first level *derandomizer*, enabling the TDC to disentangle up to 4 different edges. The hit registers should be flushed as soon as possible into the common L1 buffer but this operation is bound to the status of the other 7 channels of the group; since the L1 buffer is a shared resource and can serve only one producer at a time. If the L1 buffer is not free to be filled and more hits arrive to the hit registers the oldest ones are dropped, meaning that some edges of the pulses will be lost.

[1] two if the TDC is used in *pairing mode* and considers the raising and falling edge of an event pulse as a single entity.

The L1 buffer priority policy is simply based on channel number: lower index channels within the octet have higher priority compared to higher index channels. Lets consider only one 8-channel group: for now we ask that all channels are subject to the same (sufficiently high) hit rate, the hits are completely uncorrelated and the pulses are coming from a source that has zero dead time. In this configuration a significant hit loss can be observed on high channels. This shows precisely because high channels are served with lower priority by the L1 buffer and are thus bounded to drop events. If the assumption about zero dead time is dropped then the hit loss will decrease: this because a longer dead time enables the hit registers to be flushed before new hits come to the TDC inputs. On the contrary, having correlated hits and large differences in hit rate on different channels will result in an apparent loss of efficiency of the derandomizers for the channels; increasing the loss rate. This is worse if the most active channels are the high index ones[2] within the octet.

HPTDC performance simulations showed that hit rates up to 1 MHz can be accomodated on all channels with low losses even when the correlation between channels is 100% (same signal on all 8 channels). If pairing mode is not used, such as in NA62, then the maximum hit rate under those conditions can be up to 2 MHz per channel. We should notice that this evaluation only considers one active octet of a single chip, so no effects from L1 and readout buffer multiplexing and from trigger latencies. While the readout buffer multiplexing latency is a concern only when more than one octet of channels is used in the TDC chip the trigger latency is always present.

5.1.2 L1 buffer bottleneck

Once the events are passed from the hit registers to the L1 buffer their permanency time is dependant on the trigger matching function latency. Given that the L1 buffer is filled from the hit registers its occupancy is not related to the hit correlation between channels[3] but still it is sensitive to the timing distributions of the hits reaching the channel octets; anyway the most important contribution to this effect is the trigger latency.

The best operation of the TDC is achieved with the L1 buffer occupancy less or equal to 50%. The trigger latency and the detailed features of the trigger matching function (such as the trigger window size to match hits) determine the speed at which the buffers are flushed and the data are forwarded to the common readout buffer.

If the L1 queue is full for whatever reason the oldest events will be dropped and the matched events missing parts will be flagged appropriately.

5.1.3 Trigger FIFO bottleneck

In extreme conditions it is possible that the trigger FIFO, used to store the incoming triggers from the outside world, overflows. These overflows may happen if the trigger rate exceeds a threshold dictated by the speed of the trigger matching function, the size of the trigger matching windows and the raw hit rate per channel. If an overflow happens the trigger logic will take care of generating dummy trigger events to be used in place of the lost triggers, this will ensure synchronism of the system at the expense of a reduced trigger efficiency.

5.1.4 Reaodut FIFO bottleneck

The readout FIFO is the final buffer within the TDC before the outside world (in the case of NA62 this is the TEL62 board). If the readout of the TDC is slower than the rate the queue is filled there will be event loss. As a general rule it is advised to mantain a readout rate that is at least double than the readout queue filling rate to provide a reasonable headroom to readout rate spikes. Notice that in case of a full readout queue the trigger matching[4] will start to drop events until some space in the queue is freed.

[2] thus the lower priority ones.

[3] since that effect has been already taken care by the hit registers.

[4] in the NA62 readout.

5.1.5 TEL62 board and readout design limitations

Given the previously discussed limitations on the TDC hardware and on the corresponding readout board it is clear that the system can exhibit issues when used in high rate conditions, such as in higher than nominal luminosity in NA62 future runs. Moreover, the strong reliance of this readout on hardware triggering (4.2) poses limitations to the experiment future developments: first of all a trigger system like this is not very flexible, meaning that it can perform a selection based on very few and scarcely processed pieces of information. Also, due to the front end hardware limitations in the TEL62 all L0 processing and preliminary buffering must happen very close to the detectors. This results in having a lot of vulnerable devices in the experimental cavern; a very harsh environment for sensitive electronics. TEL62 housed in the cavern are exposed tho high radiation fluxes; damage to the main components is a concern as well as firmware corruption. The issue is mitigated by implementing remote firmware flashing procedures but overall readout efficiency is compromised.

During NA62 operational life the beam intensity is expected to increase up to the nominal; after 2018 the intensity was brought to 100%. While TEL62s and the existing L0 hardware managed to perform acceptably the front end HPTDC could not keep up to the increased data rate and event loss was observed. This observation was the basis of the proposal for a new readout system [42].

5.2 The new readout system

The underlying novel idea is to move as much delicate logic as possible away from the experimental cavern to the server/counting room.

A new radiation hardened TDC module based on FPGAs has been developed and is intended to digitize data directly, without any L0 buffering; from this module all the data will be transferred to dedicated servers via several high speed optical links. Data transfer from the TDCs to the servers is managed by the FELIX card, developed and used in the Atlas experiment.

Having all the data now subjected to L0 triggering on the FELIX servers will enable the implementation of a more efficient and flexible L0 triggering mechanism. Being a software trigger it can operate asynchronously and with variable latency, thanks to the very extended buffering capacity of the servers. This will enable an higher L0 algorithm complexity and consistently higher data rates, possibly letting the DAQ system to run acceptably at higher than nominal intensities. An L0 trigger decision may then be taken asynchronously and with variable latency; commanding an L1 request to be forwarded to the farm via the dedicated FELIX servers. A system like this will be very flexible, at this point it would be possible to even compute the L1 primitives directly on the FELIX servers, those may be sent to the farm that will collect L1 event packets and run L2 algorithms only.

This is a very ambitious prospect and it is intended to be developed during the years. The idea is to start the FELIX upgrade by reading out L1 exclusive sub-detectors in parallel with the TEL62 readout. In this phase the readout can be tested and features may be added. The following activity would be to include also high-rate L0 detectors, in order to build an interface between the FELIX card and the existing hardware trigger processor. After this is done all eligible detector readouts may be replaced.

The first detector to be used for this purpose is the new Veto Counter: its low number of channel and relatively low rate are good for a demo project. With the experience from the Veto Counter the next subdetector to be upgraded would be the CHANTI, with higher rate and channel count. As part of FELIX-L0 integration effort a FELIX readout for the KTAG is also planned, this has significantly high rate (~ 50 MHz) and is is part of L0.

5.2.1 TDC board

The novel TDC is based on the knowledge acquired during the STRAW readout board (SRB) development. This TDC, similarly to the ones used in the SRB is based on *shift registers* and counters; a different approach to the HPTDC chip, widely used in NA62 and based on delay elements (a DLL chain). A shift register topology limits the minimum time bin size to $\sim$ 100 ps but allows to have high bin size precision and linearity; provided that an high quality clock is used[5]. Moreover the performance of such a TDC is independent to temperature fluctuations and power supply voltage, it is stable and does not require periodic calibration. The STRAW spectrometer in fact uses more than 8000 of such TDC without the need of any calibration.

The new TDC is based on a `10CX105YF780` FPGA, this part is from the CYCLONE 10GX family and offers an high degree of radiation hardness thanks to non-existent Single Event Latch-up (SEL), improved Single Event Upset (SEU) and it has active error detection and correction in both configuration and user memory [43].

We point out that in the idea of the FELIX readout system the TDC board is the only critical component that will be located in an harsh radiation environment; and is extremely radiation hardened by design. Each TDC card has its own built in DCS management module; this is based on an ATMega microcontroller and communicates with the DCS system over a dedicated 100Mbps network interface. DCS-style commands such as voltage power cycle may be issued over this link, the preferred communication is using DIM.

For PCB form factor, it was decided to use 6U EURO card, which allows to put 2 TDC FPGAs on single TDC board. Each may be configured with 64 TDC channels with 0.78 ns bin size, 32 channels with 0.39 ns bins or 16 channels with 0.195 ns bins. The input connector depends on TDC resolution and detector.

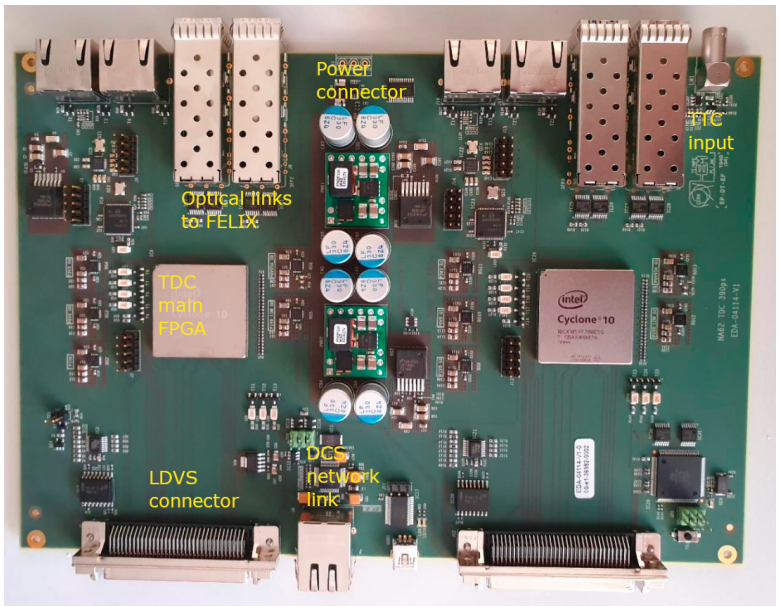

Figure 5.1: Picture of an NA62 TDC module for FELIX readout. The main elements of it are labelled.

5.2.2 The FELIX Card

The FELIX (Front End LInk eXchange) card is a PC card developed by the ATLAS TDAQ collaboration as part of the general readout upgrade of Run 4. It provides an interface between the front end readout and a commodity network, the FELIX readout unit being a standard server machine with one or several FELIX cards installed.

[5] In a shift register TDC those parameters are entirely dependant on the FPGA clock quality.

Incoming data from the front end are routed through bidirectional optical links. The current firmware for the card supports either GBT or FULL transmission from front end to host (FE-H) and GBT only from host to front end (H-FE). FE-H is the main data channel, while H-FE is used for control and timing signals.

The GBT protocol is a custom-designed communication encoding developed at CERN for data transmission in Large Experiments, its maximum theoretical rate is 4.8Gbps. [11] GBT may work in radiation hard mode with full error correction or in *wide* mode, with no error correction; providing measured rates of 3.2Gbps and 4.48Gbps respectively. The logic unit of this protocol is a 120bit *frame*, containing a 4 bit header for frame type and optical receiver locking, 4 bits for DCS and monitoring applications, 80 bit payload and 32 reserved bits for frame error correction or payload based on mode. A physical link configured for GBT protocol can host up to 40 virtual links, *elinks*. If radiation hardness is not needed and data rate requirements exceeds wide mode GBT the FELIX card supports an high bandwidth *FULL mode*; this applies the usual GBT protocol for H-FE link while using a simple protocol with 8b/10b encoding for FE-H. In FULL mode the maximum bandwidth is 9.6Gbps.

Hardware design

FELIX is a standard form factor PCIe x16 Gen.3 card in $(270 \times 110 \times 1.58)$ mm. The board is powered using an 8pin ATX power cable and uses three `LTM4630A` 18A voltage regulators. Currently it hosts 48 bidirectional links; optical TX and RX are managed by 4 receiving MiniPODS and 4 transmitting ones. A MiniPOD is a self contained optical transciever with up to 12 links support and a maximum throughput of 14Gbps. Fibres are bundled and brought from the MiniPODS to MTP connectors on the card bracket.

All MiniPODs are connected on board to the main FPGA, a Kintex Ultrascale `XCKU115FLVF1924-2E`; this part has two PCIe 8x endpoints that are joined by a switch IC `PEX8732` and the resulting 16x bus is routed to the card PCIe edge connector. An additional timing mezzanine card (TMC) is installed to provide the main FPGA an interface with a TTC or equivalent timing system; different mezzanines are designed to support different timing standards. Clock generation is based on the external TTC reference: the main timing clock[6] is passed to an `ADN2814` PLL and recovery chip which provides a 240 MHz derived clock. A jitter cleaner chip (`Si5345` or `LMK03200`) provides the clock directly to the FPGA and the MiniPODs.

FPGA firmware management is done via a microcontroller (MCU): upon bootup the FPGA gets programmed with one of the firmware images stored in an on board 2Gb flash memory; the memory can store up to 4 different images and choice is done with physical board jumpers. It is also possible to access the microcontroller via either SMBus or PCIe to trigger a firmware loading. An additional, direct FPGA JTAG interface is provided for development purposes; if this is used no action will be performed by the MCU and programming is up to the user after bootup. JTAG interface allows also to program the flash memory, as an alternative way to direct PCIe access.

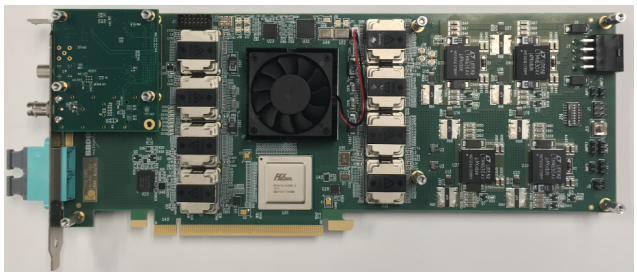

Figure 5.2: Picture of the FELIX card without fibres connected to the MiniPODs, on top left the TTC daughter card is visible. Under the heatsinked main FPGA there is the PCIe bridge chip, right side of the card is mostly power supply. [11]

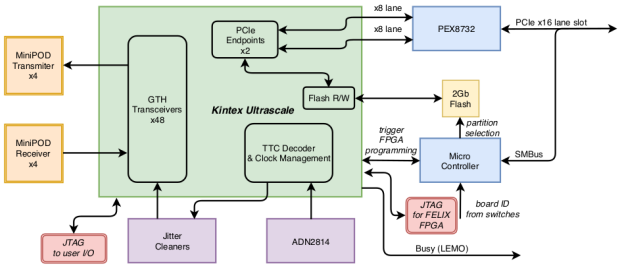

Figure 5.3: Block diagram of the FELIX card. [12]

[6] for TTC standard 40 MHz clock.

Firmware design

The firmware has to interface the custom optical links to an off-the-shelf server architecture, it is composed of various modules:

- **GBT-FPGA:** provides encoding/decoding function for the GBT protocol, including the active error correction and data scrambling. It receives the 120 bit GBT frames and encodes them in packets compatible with the central router
- **Central router:** this is the main interface module of FELIX; it connects to the GBT-FPGA modules and queues up data to be DMAed. Incoming frames in the GBT-FPGA contains data regarding multiple elinks, here payloads for each elink are routed in their own FIFO and then to the PCIe interface. Before inserting packets in the queues it also adds additional information in the form of End of Packet words; router operation is deeply linked with TTC information incoming from the TTC decoder core.
- **Wupper core:** its purpose is to access the intermediate queues and perform the PCIe DMA operations to or from the host. There are 2 Wupper implementations, each controlling 8 PCIe lanes, 64Gbps throughput. The Wupper uses the default Xilinx PCIe core to perform operations on the server bus.
- **TTC decoder core:** this module manages the TTC information distribution; serial TTC data are sampled and govern a Finite State Machine (FSM) which then controls and forwards the status of the card.

Due to physical constraints of the FPGA used in the design the DMA relies on two Wupper instances, each controlling 8 lanes of the 16x bus, maximum total theoretical transfer rate is 128Gbps, as per PCIe Gen.3 specifications.

Data are internally multiplexed on an elink basis before being forwarded on the PCIe bus for DMA. Each elink has a 1K x 16b FIFO as a buffer and the transfer manager inside the Wupper manages these FIFOs by enqueuing data in form of *blocks*; FIFO selection is done via multiplexers. The blocks are then pushed to memory via PCIe while the software may perform reassembly.

The path (H-FE) is similar but reversed.

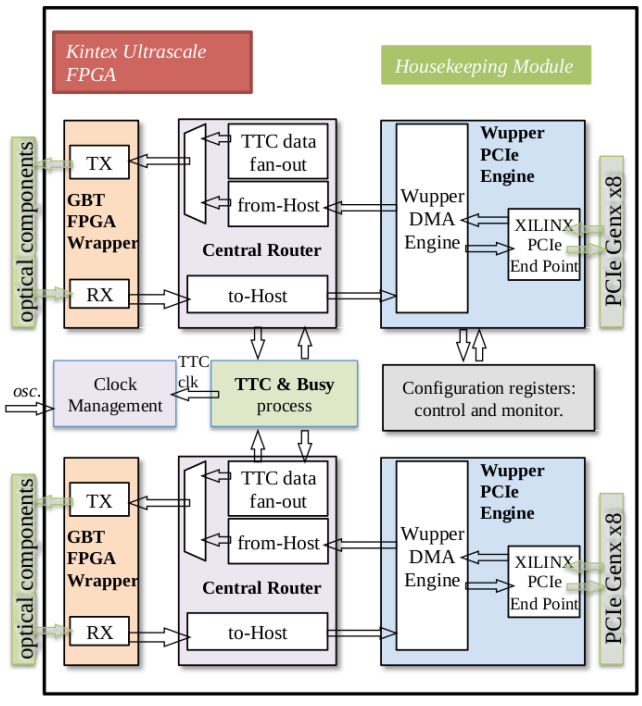

Figure 5.4: Block diagram of the FELIX firmware, notice that the Wupper engine employs further configuration and management logic to perform DMAs to the memory: the configuration registers determine memory paging. [11]

NA62 modifications for the FELIX firmware

The FELIX firmware has been slightly modified in order to be compatible with NA62:

- The TTC decoder core was designed to operate with ATLAS-type signals. This "LHC-centric" operation was not desirable, instead interpretation of the SOB-EOB signals is needed.
- FELIX ATLAS firmware does not support control elink readback for the NA62 TDC control channel.

Adapting the TTC core required a deep modification; the native *bunch reset counter* and *event counter reset* signals from atlas are reinterpreted as SOB-EOB.

The control path modification is simpler: when the front-end TDC recieves a read command via the FlxUpload library it replies with a shortchunk containing the register being read and its content. In order to implement a functioning readback in software this shortchunk has to be routed to a dedicated FE-H elink where it can be then intercepted by the main wrapper software. An incoming readback packet from the TDC is encoded with the same 8b/10b format of the data packet, but with an extra bit flag in the packet header. The GBT decoder core for every elink has been modified to route all readback packets to the next elink number. For example, GBT decoder for elink N will forward all readback packets to elink $N + 1$, while all the others will get forwarded to elink N.

Chapter 6

FELIX Control System

6.1 Introduction

In NA62 there is need of having a dedicated channel to send commands and configuration values from the Run Control to the front end electronics. In a FELIX system this accounts to get the relevant commands to the desired FELIX server and to forward those through the FELIX card to the TDC boards. Essentially TDC commands are performed through register writes on the onboard memory. Readback of the TDC registers is also important; it can be useful to check whether or not a write operation has been successful as well as reading TDC information (such as Board ID and similar) from the Run Control. The control system has been developed from scratch as part of this work.

In order to implement this features a modification of the FELIX firmware is needed, detailed in Section 5.2.2.

6.2 Requirements

An effective front-end control system for NA62 FELIX readout sub-detectors should:

R1: Be compatible with whatever front-end device gets connected to the FELIX card.
R2: Work reliably regardless of the SPS supercycle instant the command is issued.
R3: Provide feedback on its operations.
R4: Be seamlessly integrated in NA62 existing architecture (Run Control) for both issuing commands and feedback messages.

6.3 Design ideas

The control system may be composed by a number of *Control server* modules running within the main NA62 FELIX Wrapper on the FELIX servers and by a *Control client* running on a different machine, most likely a Run control one. Commands may be sent from the client machine to the FELIX server over a standard network link.

Given the requirement of feedback (**R3**) every operation that involves a read should provide a read-back of the relevant register(s) as a response. This may be done either in the client side by having a reading command implemented and issuing that after a write or directly on the server side. In this case a specific flag in the client-server format may trigger the readback sending.

In order to fulfill **R4** the Client application should be based on a *Control Client library* that can be easily integrated in the Run Control software; the application will then be used for standalone or testing operations while all regular control operations will be scheduled and managed through the Run Control. The requirement **R1** may be guaranteed by the control client library design; it should be possible to send arbitrary payloads to arbitrary registers and read any register; specific meaning of

the operation will be front end dependant. The library shall provide built in payloads and registers for use with a FELIX TDC front end as internal constants.

A command sent to the front end has to be issued as soon as possible; since the TDC accepts such commands only when *out of burst* the control system should listen for DIM start and end of burst signal and schedule the operation accordingly in order to comply with **R2**. Given that the FELIX server has direct access to DIM information in the form of `Boost` [44] signals in the FELIX Wrapper the scheduling may be implemented on the server side.

6.4 The fupload tool

The FELIX code base provides many different low level tools, mostly for configuration and management [12]. An interesting candidate to be the starting point of the development of a control system is *fupload*. It was the basis for the creation of the control message DMA system.

This piece of software enables data transfer from the FELIX server to the front end across any GBT elink (in H-FE direction). It can either work with user-made files defining its input or with predefined built-in test patterns. Many upload options such as elink width, DMA timing and upload speed can be also set upon software invocation.

Fupload supports the following input files:

- ASCII files containing binary data in plaintext;
- Binary files containing binary data;
- Binary files containing raw bitstreams to transmit without any further operation.

Internally the software is based on a low level library named `FlxUpload`, this is what manages the DMA transfers of the messages from the host server memory to FELIX and then to the front end device. `FlxUpload` has different functions to *prepare* the incoming data (from file) to be uploaded, depending on their format.

An `FlxUpload` instance is created within the fupload main process, this is used to call the required functions to prepare the data and perform the uploading.

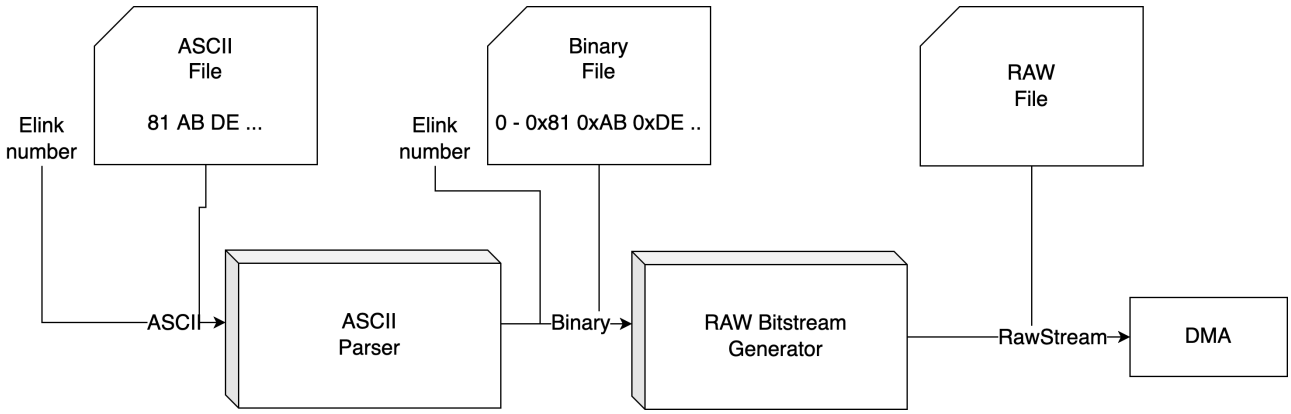

Figure 6.1: Schematic of the data preparation operation performed by FlxUpload on the fupload input information. The elink number for the uploading may be provided as an input parameter (ASCII and binary files) or directly in the RAW stream; in the former cases elink information will be encoded in the generated bitstream by the stream generator.

6.4.1 TDC payload format

An operation on the TDC can be a register read or write; that is described by a bit sequence in the fupload input file. We remind that fupload is *front end agnostic*, this means that it attempts transfer any data on any elink; what is described here is the specific data format for the FELIX TDC. The sequence contains (in order):

- One bit flag indicating operation type (0b0 = read, 0b1 = write)
- 63 bits representing the register.
- 32 bits of payload only if it is a write operation.

A single file may contain one or more operations: in ASCII mode each operation should be terminated with a *newline* while in binary mode an appropriate termination sequence has to be used.

Here we show some example of operations together with a description:

```
Read reg. 0x7123456                 --- 71 23 45 67
Write 0xDEADBEEF on reg. 0x7123456 --- F1 23 45 67 DE AD BE EF

Read reg. 0x10                      --- 00 00 00 10
Write 0xDEADBEEF on reg. 0x10       --- 80 00 00 10 DE AD BE EF
```

Notice how the bit sequence is split into bytes and written in hexadecimal format.

Read/Write flag bit and register most significant byte are merged:

```
Write reg. 0x71 ... --- 0b1111 0b0001 ... = 0xF1 ...
Read  reg. 0x71 ... --- 0b0111 0b0001 ... = 0x71 ...
```

Internally these operations are converted into a *raw bitstream* to be DMAed. The raw bitstream for one operation is always composed by four 64 bit long *blocks*. The two most significant blocks are always zero, while the two least significant contain the payload (if it is a read operation the payload is zeros), the register and the R/W flag bit. Blocks are logically arranged in bytes similarly to what happened in the operation file. In the raw file endianness of the words is reversed.

After the register data a 32 bit *trailer* contains information about bitstream length and elink to upload to.

6.4.2 Bitstream trailer computation

The trailer is in the form $xy\ \ zh$; byte y represents the word length of the bitstream: all read operations have no payload so $y =$ 0x4 while all write operations have $y =$ 0x8.

Bytes x, z, h encode the elink: elink number E is passed to fupload as an argument; it is a 12 bit hexadecimal number. Trailer bytes are computed with:

$$x = (E \,\&\, \text{0xF}) * 2; \tag{6.1}$$

$$h = (E \,\&\, \text{0xF0}) * 2 >> 4 + \text{ofw}; \tag{6.2}$$

$$z = (E \,\&\, \text{0xF00}) * 2 >> 8; \tag{6.3}$$

$$\text{where ofw} = x\%\text{0x10}. \tag{6.4}$$

The symbol $\%$ denotes integer division.

Here we show the content of the two LSB bitstream blocks for the operations discussed before; elink number chosen is 0x000:

```
Write 0xDEADBEEF on reg. 0x71234567 --- 00 00 00 00 00 00 ef be ad de 67 45 23 f1 08 00
Read reg. 0x71234567                --- 00 00 00 00 00 00 00 00 00 00 67 45 23 71 04 00
Read reg. 0x10                      --- 00 00 00 00 00 00 00 00 00 00 10 00 00 00 04 00
Write 0xDEADBEEF on reg. 0x10       --- 00 00 00 00 00 00 ef be ad de 10 00 00 80 08 00
```

6.5 Software architecture

The control system is composed by a control client library used by the control application (the Run Control) and a number of control servers instantiated within the FELIX main wrapper. Lets give further description of those components.

6.5.1 The Control Client library

This library is the API to be used by the Run Control or any application wishing to exploit the FELIX control channel. A control library should be instantiated on an elink basis; elink number and network port to the FELIX server are defined upon object creation. Ports are chosen using a dedicated *port computation rule*, based on elink; the host application has only to provide the required elink number and the hostname.

The client library allows the creation of read/write *control messages* via dedicated functions; differentiation between operation types is implemented using function name overloading. Message creation functions provide an appropriate raw bitstream for the requested operation; this is stored within the client instance in a dedicated memory area and then forwarded to the network socket within the client. The control message storage area can hold at most one bitstream at the time; since that message is immediately sent, all the buffering is performed by the socket FIFO. Given the asynchronous nature of the DMA upload on the server side the client library constantly listens to the socket for incoming readback messages using a dedicated nonblocking receiving thread. Whenever a readback message arrives it is pushed in a queue; applications using the control library may pop incoming readbacks and use them however is needed. All responsibility for queue management is thus upon client applications.

The library supports in principle message creation with any payload and register address[1]; to ease operation with NA62 TDCs a set of common registers (Tab. B.1 in Appendix B) together with common payloads are provided as library constants and can be accessed by the host application.

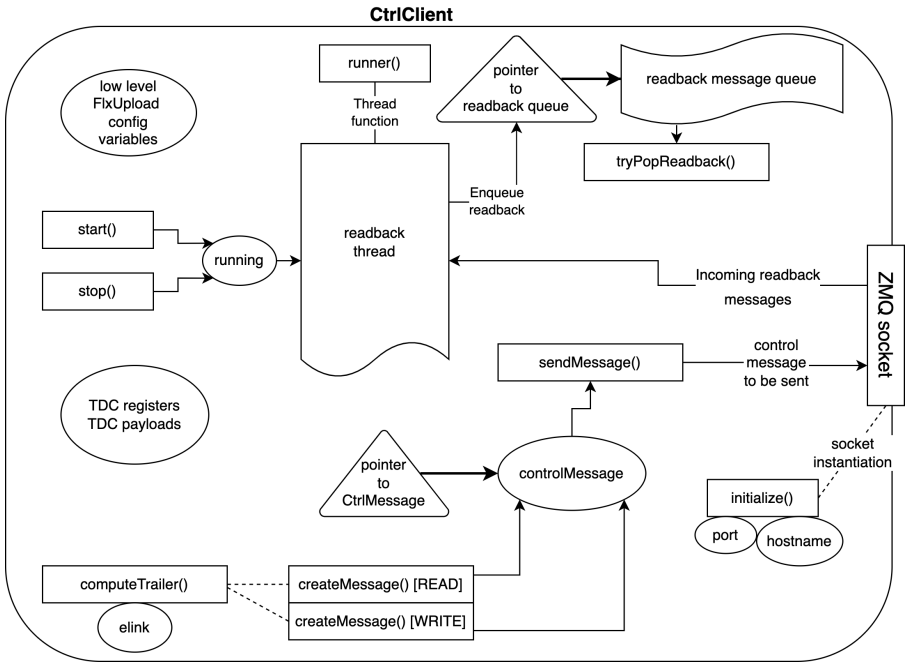

Figure 6.2: Schematic of Control client library.

[1] within the legal bounds.

6.5.2 FlxUpload extension (FlxCtrl)

The management library provided within the FELIX code base supports data preparation only from files; fupload being designed to either work with an operation file or with its built in data generation algorithms. The functions provided in FlxUpload perform all parsing operations and write the bitstream in a buffer within the FlxUpload instance; that buffer then gets flushed and its content uploaded to the front end.

A subclass of FlxUpload named FlxCtrl has been developed. FlxCtrl has public access to all variables of FlxUpload and includes a function to directly fill the buffer with the provided hexadecimal bitstream. The idea being to offload the actual bitstream computation to the client side and using the FlxCtrl instance only to perform the upload. In the original fupload software the FlxUpload instance was constructed and destroyed respectively short before and short after the upload operation had been performed. The upload requires physical access to the card thus once an FlxUpload is instantiated on a given elink of a given halfcard it will lock the whole halfcard until that instance is deleted. This means that if another FlxUpload is created to write on another elink within the same halfcard that operation will fail.

While the former is not an issue for fupload[2] it is an issue for the control system: in principle there can be an arbitrary number of upload operations to be performed on an arbitrary number of elinks spread on multiple half cards; each of those elinks will be managed by one instance of the control server. The chosen solution is to design FlxCtrl as a shared resource between control servers:

FlxCtrl is designed as variation of the well-known *singleton* pattern [45]. FlxCtrl provides a function that returns a structure containing a pointer to an FlxCtrl instance, to a mutex and to some other relevant metrics; arguments passed to that function specifies the halfcard the instance is expected to serve. All the instance management is handled within the FlxCtrl class:

- If the requested halfcard has already one existing instance[3] then the function returns a pointer to that instance.
- If the halfcard is not served by any instance, the corresponding instance is created and the corresponding pointer is returned.

The mutex is used by the control server to lock the FlxCtrl for itself whenever it needs to perform an upload.

Internally, this pattern is implemented using a static variable that keeps track of the number of FlxCtrl instances that get created; all the pointers to those, to the corresponding mutex and some useful variables are stored in a static data structure that gets filled/accessed by the "get instance" function. The constructors are made inaccessible from outside the FlxCtrl class to ensure the intended usage. Upon destruction, memory used by the FlxCtrl instances is freed automatically.

A schematic of the structure of FlxCtrl is visible in Figure 6.3.

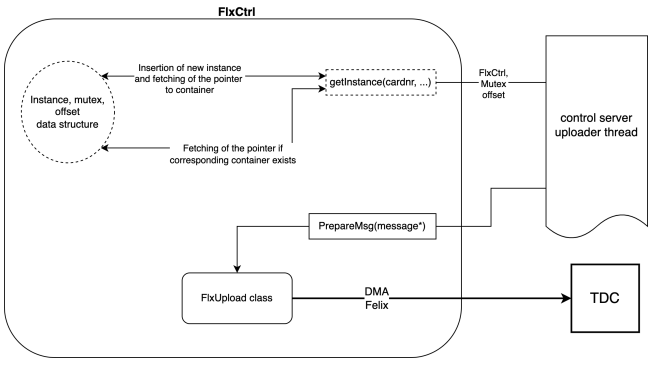

Figure 6.3: Simplified schematic of the FlxCtrl library structure

[2] it is intended to be run every time an operation has to be performed on an elink. Only one instance at a time.
[3] because the get instance function has been already called by another control server whitin creation.

6.5.3 Control Server library

The *control server library* is the receiving end of control messages; designed to interact with the FELIX card through FlxCtrl.

An instance of CtrlServer serves one elink and provides a control messages uploading function. This function is implemented using two dedicated producer-consumer threads: the *receiving thread* listens for control messages without blocking and upon reception enqueues them in a dedicated FIFO. Meanwhile, the uploader thread checks if the queue contains messages and, if messages are present and the detector is *out of burst*, it performs an upload to the TDC through the infrastructure provided by the *FlxCtrl* library, see 6.5.2 for details. Since the server library is designed to work within the main wrapper, the burst state information used within the uploader thread is provided by the wrapper EOB handler signals, to which the control server instance is connected to. A schematic of a control server is shown in Figure 6.4.

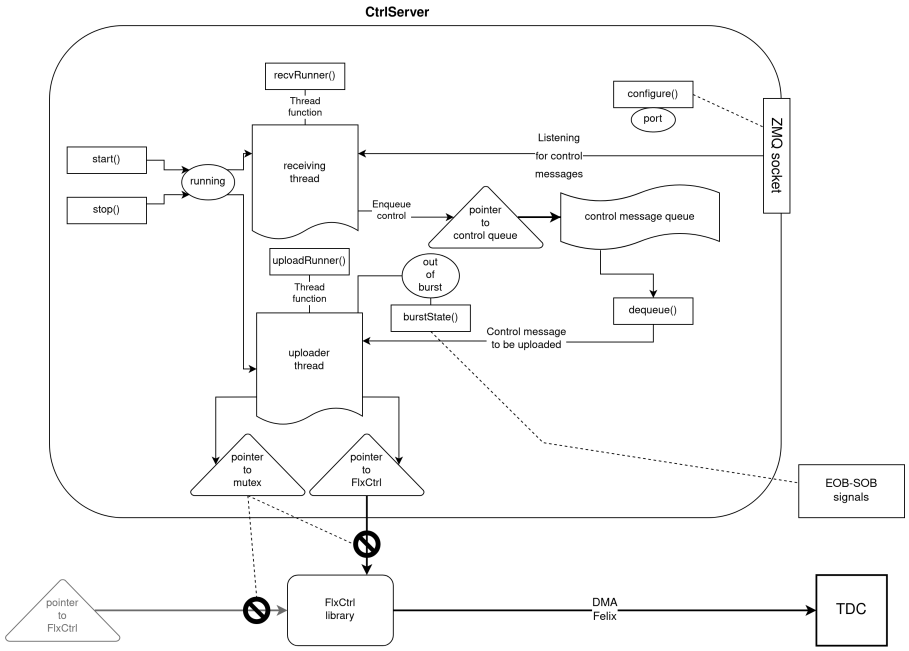

Figure 6.4: Schematic of the Control server library.

6.5.4 Message format

The control system uses two message types; readback and control. Both are implemented as C++ *packed structures*. The control message is a container for the raw bitstream generated within the control client. Board number and DMA index are sent together with the two non-zero bitstream blocks. The readback message contains the register being read and its content; it is sent by the control server after a read operation and it is essentially a reformatting of the information coming in the control shortchunk received by the FELIX card.

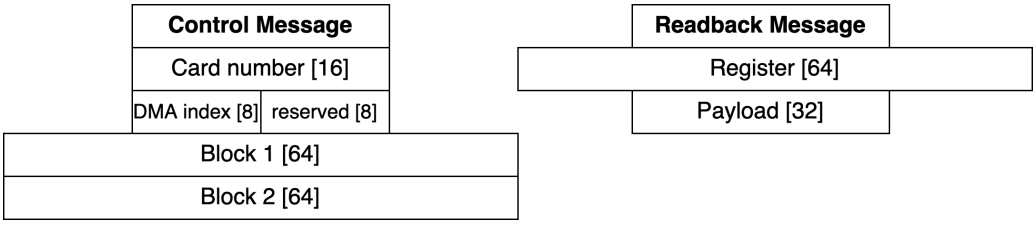

Figure 6.5: Data structures for the control system.

6.6 Integration in the NA62 FELIX wrapper

Let us now discuss how the client and server are included in the existing NA62 infrastructure.

The wrapper takes care of instantiating and managing the control servers. It receives a list of control elinks for both half cards in its configuration file. These two lists are used to initialize the control parsers and create the necessary instances of the control servers for each elink. Internally, the control servers create the necessary FlxCtrl instances. Control server mapping to elinks is provided by a two-dimensional `std::map` of control server pointers, keyed with elink number and half card. When the control elinks are configured, the appropriate map pointer is associated to a control server, SOB and EOB signals are connected to the onBurst flag and the parser gets a custom shortchunk parsing function for the control elinks. This parsing function is issued every time an incoming shortchunk on the control elink arrives from the TTC. The function has access to the appropriate control server instance through a pointer, and fills the internal readback structure of the server and sends the message to the client.

This provides the bridge between the existing data readback system in the FELIX wrapper and the control servers.

Chapter 7

The Veto Counter detector

7.1 Introduction

In NA62 background rejection of common K^+ decays is extremely important. Such decays are vetoed using the techniques discussed in Chapter 3 and the entire NA62 detector is built for this purpose. An important source of background comes from charged π candidates produced in beam interaction with the GTK stations. As was detailed in 3.3 the worst situation happens when the π^+ enters the fiducial decay region and travels all the way to the STRAW spectrometer, all other secondary generated particles are absorbed in the upstream collimator and thus not detected by the vetoing system and there is one undecayed K^+ in coincidence.

During Run1 (2016-2018) this source of background was extremely problematic for the data taking [46]. Several solutions were tested at that time; both hardware such as the addition of a *rainbow collimator*, named TCX and the installation of copper plugs in the GTK magnet holes to absorb the stray pions as well as new software triggering schemes. All these solutions were not able to fix the issue due to either reducing the detector acceptance or being not effective enough.

The Veto Counter is a dedicated subdetector designed and installed to specifically detect the secondary generated particles that will otherwise just be absorbed in the collimator.

7.2 Detector structure

The Veto Counter is composed of three identical scintillator stations and a photon conversion lead plate. Station 1 and 2 are located before the TCX collimator while Station 3 is immediately downstream of it; in order to accomodate them the beam pipe in the area has been replaced with a reduced diameter one. This allows the active elements of the detector to be as close as possible to the beam.

A station is composed by multiple active *tiles*, 40 $\times$ 120 mm or 20 $\times$ 120 mm in size; the smaller tiles are used the closest to the beam pipe, where the activity is the highest. A tile is characterized by its *Tile ID* and *Station ID*. The Tile ID represents the position of the tile with respect to the detector y-axis: tile 0 is the topmost one while tile 10 is the bottom one. This pattern is the same in all three stations. Each tile is glued to two opposing fishtail light guides attached to photomultipliers. Conventionally the right side (facing downstream) is called *Saleve* side while the left side is called *Jura* side[1]. One tile then corresponds to two channels in the data acquisition system. A sketch of the structure is shown in Figure 7.1.

Mechanical support of the tile assemblies is provided by the PMT collars; the station frame allows easy mounting of those parts.

[1] from the names of mountain ranges in the local area.

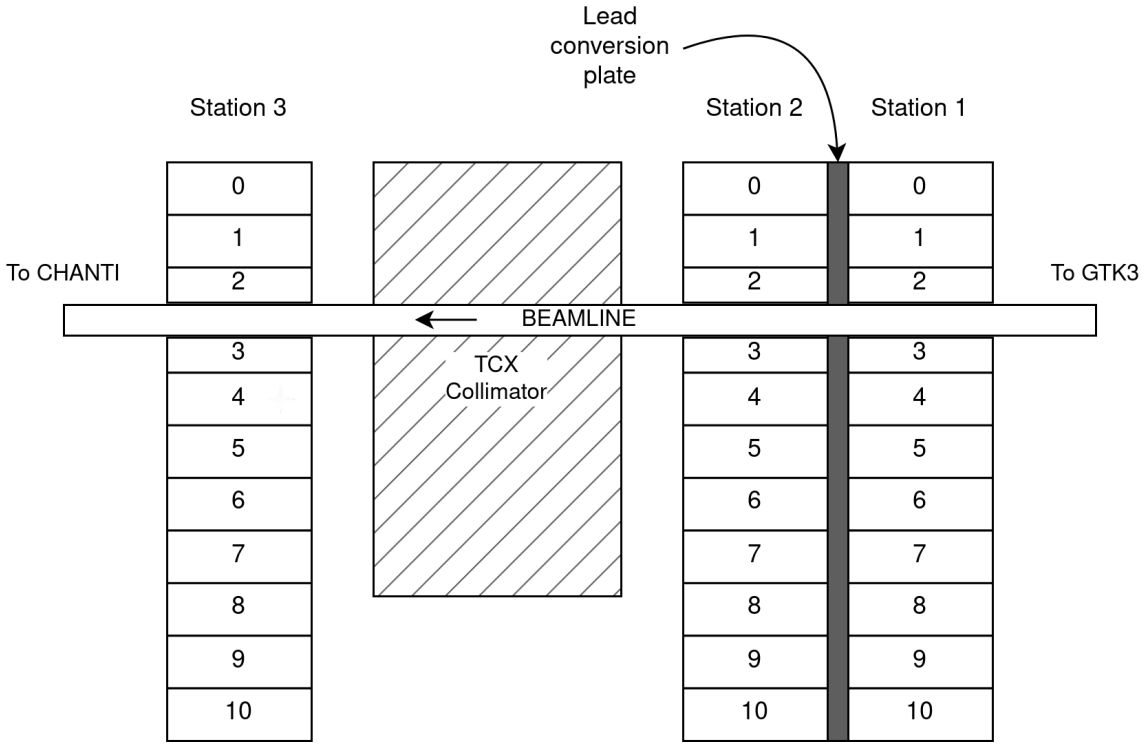

Figure 7.1: Sketch of the Veto Counter structure. Numbers within tiles are Tile IDs. Notice that Tile 2-3 are of the smaller type: due to the high activity near the beam pipe smaller tiles are necessary to avoid excessive rate in the DAQ.

7.2.1 Station 1 and 2

These two stations are contained in the same assembly, together with the lead plate. The system is composed by two boxes named *top* and *bottom*; this is referring to their location with respect to the beam pipe. The top box is 126 mm tall while the bottom box is 328 mm tall; width of the boxes is the same at 670 mm. Both boxes have an M16 eyelet either side to attach them together in the assembly. All parts are machined aluminum and include a *mounting frame* to attach the scintillator assemblies; integrity is improved with the addition of two 20 mm pillars. Matching covers, 49 mm in depth allow complete closure of the boxes; outer covers have removable lids (30 mm depth) to access the cabling assemblies of the PMTs.

Inside the boxes the scintillator assemblies are installed perpendicular to the beam line; with the lead plate sandwiched between the tiles. A rectangular hole 90 mm wide allows the passage of the beam pipe; there is no contact between the boxes and the beam pipe. The boxes will also house the PMT socket and polarization board. The whole assembly has a size[2] of $(486 \times 750 \times 198)$ mm and weighs 26 kg.

The γ conversion plate is composed by a sandwich of five, 5 mm thick lead plates riveted together. The smaller, top one, includes 4 *mounting ears* one either side; the bottom plate, due to larger size, has 10. These mounting ears are screwed to *plate support brackets* attached to the box pillars. Plates weigh 4 kg and 12 kg respectively.

7.2.2 Station 3

The layout of Station 3 is similar to the one of the Station 1 and 2 but there is no lead plate and only one layer of scintillator modules. The backing plate (facing downstream) is a flat piece of aluminum 27 mm thick. Weight of this station assembly is 9.7 kg, top and bottom boxes accounting for 3 and 6.7 kg respectively.

[2] (height $\times$ width $\times$ depth)

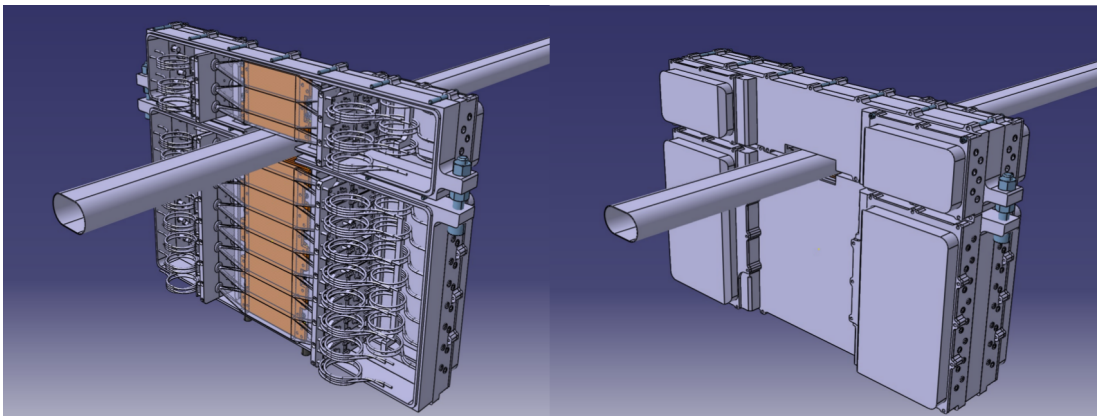

Figure 7.2: CAD drawing of the Station 1 and 2 assembly, left pane shows internals while right pane the closed and assembled module. [13]

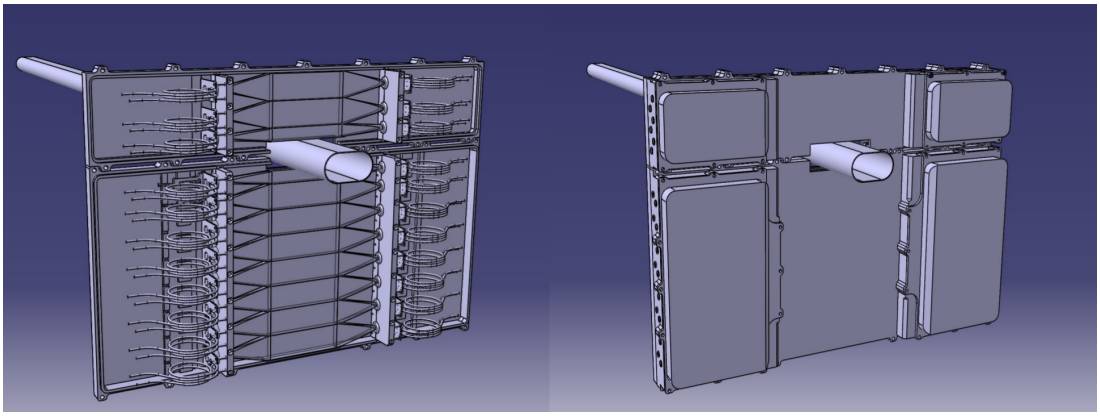

Figure 7.3: CAD drawing of the Station 3 assembly, left pane shows internals while right pane the closed and assembled module. [13]

7.3 Subdetector supports

The Veto Counter stations needs to be attached to either side of the rainbow collimator assembly. To enable adjustment the mounting system has to provide freedom of movement along x, y, z axis, given the beam line hole clearance constraints.

Given the weight of Station 1 and 2 assembly it is supported from top and bottom while Station 3 only from the top. Support brackets are attached to the TCX collimator assembly and are of two types for top and bottom support respectively. Given the increased load on the TCX assembly due to the weight of the stations an additional support frame is secured to it to provide further longitudinal rigidity.

7.3.1 Top support

TCX assembly has already a set of mounting holes on the top, both downstream and upstream side. This is where the top support is secured for both station assemblies.

The top support bracket is composed of a plate attached to the TCX mounting holes that houses a larger cylindrical piece that attaches to the station assembly; the cylindrical shape allows for 3D freedom in the form of rotation and sliding motion.

7.3.2 Bottom support

Support at the bottom is provided by the cross beam of the TCX collimator cage: an U shaped bracket is clamped to the beam with screws, this also provide a certain degree of adjustment in z, x directions.

Another two plates form a T support that is fastened to the U bracket and on which the Veto Counter station can stand. The T assembly may slide on the U bracket and get fastened in any position thanks to oval screw cutouts; this provides y axis movement; similar slots along x provide further x travel.

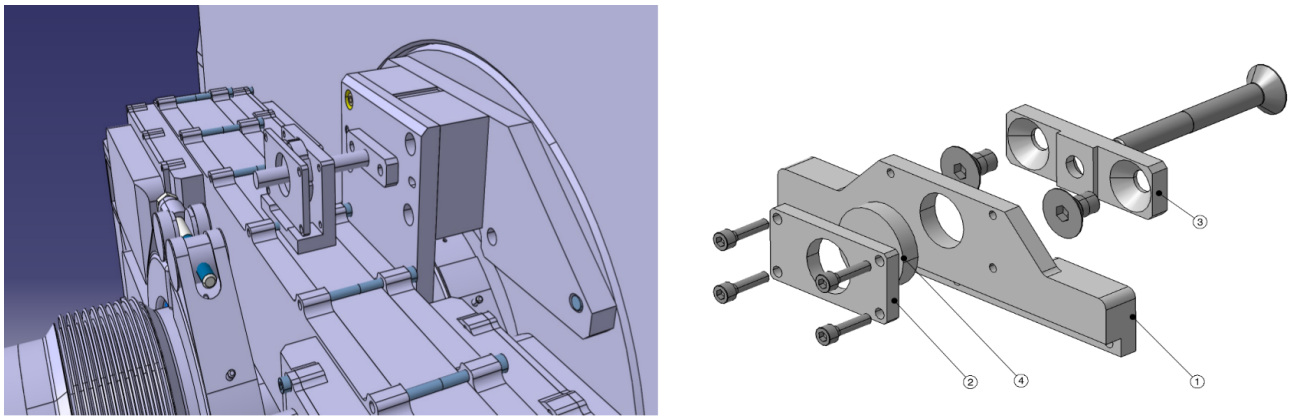

Figure 7.4: Assembly of the top support to TCX (left pane). CAD drawing of the top support (right pane): (1) pivot bracket attached to the Veto Counter assembly, (2) pivot cylinder support and lock, (3) bracket to TCX mounting holes, (4) cylindrical pivot. [13]

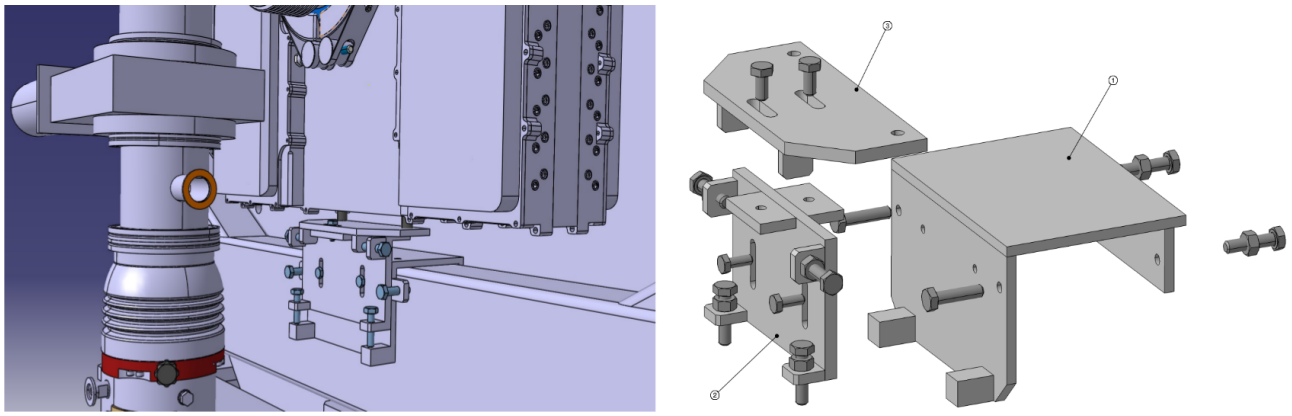

Figure 7.5: Assembly of the bottom support to TCX (left pane). CAD drawing of the bottom support (right pane): (1) U bracket sitting on the cross beam, (2) vertical plate with y movement, (3) top plate attached to bottom Veto Counter box; (2) and (3) form the T bracket. [13]

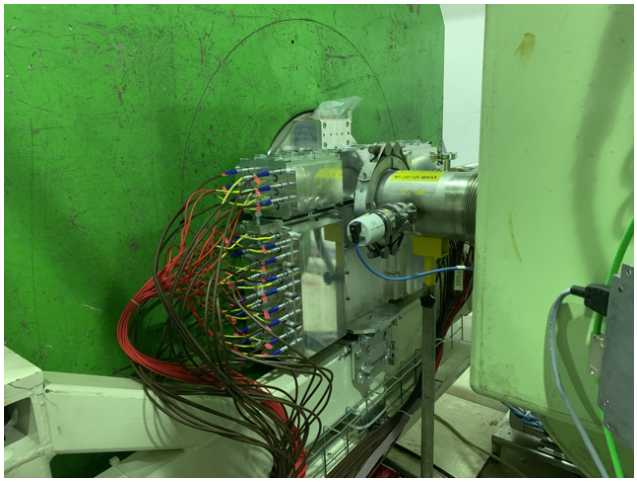

Figure 7.6: Picture of Veto Counter station 1 and 2 installed in the designed location.

Figure 7.7: Picture of Veto Counter station 3 installed in the designed location.

7.4 Front end

The Veto Counter PMTs are R9880U[3] and the tiles are made of Bicron BC408 plastic scintillator medium. Given the fast nature of the signals produced by the scintillators and the photomultiplier an high speed front end is required. The front end boards are custom-designed active Constant Fraction Discriminator modules (CFDs) with differential input and reading one PMT tube each. While CFD support fully differential input signals they are used in single mode in the Veto Counter.

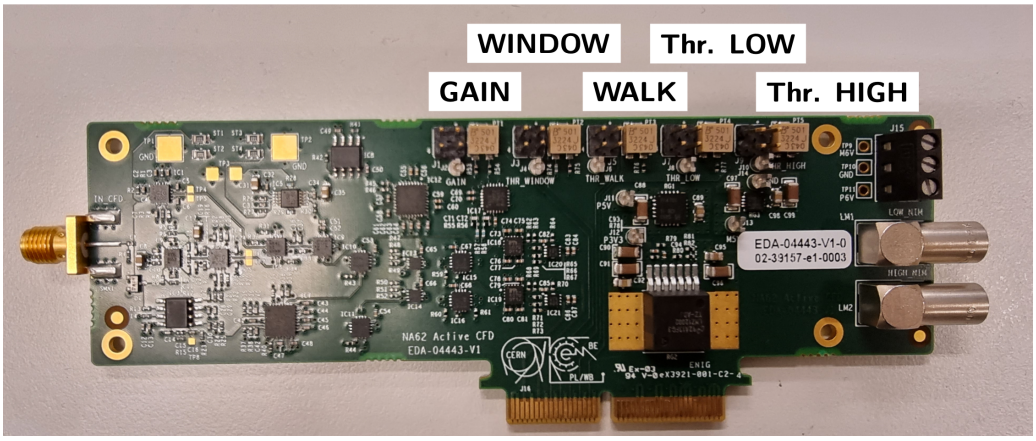

Figure 7.8: The CFD module used in the Veto Counter; setting of relevant parameters such as tresholds may be done via external control through DAC or manually with trimmers. The golden SMA connector on the left is the input port, while the right LEMO connectors are NIM-compatible outputs for low and high threshold. Card edge connector is to be plugged in the motherboard and provides LVDS outputs and power to the CFD. For testing purposes or if only one CFD is used it can be powered independently of the motherboard using the screw terminals above the LEMO outputs.

The design of this CFD is slightly novel: instead of directly feeding the comparator with signal and inverted-delayed signal those are summed and limited before being compared. This avoids the limitation in common mode voltage typical of high performance comparators and allows for extremely high dynamic range (up to 1:100) and high linearity. To perform this an analog front end to the comparator is required, a schematic of which is available in Figure 7.9.

Firstly the input signal passes through an optional[4] input balun to be symmetrized and to filter out common mode noise. Following that a low noise amplifier (`ADA4938`) provides a $\times 3$ amplification and its output signal is feed to a delay line and to an intermediate amplifier while inverted amplified signal is feed forwarded directly to the summing chip. The summing chain is composed by two `THS3491`, one performing the aforementioned amplification of the delayed signal and the other performing signal summation. Finally the output signal passes to a diode voltage clamp and a slew rate limiter based on an `LMH6559` then goes to the digital comparator.

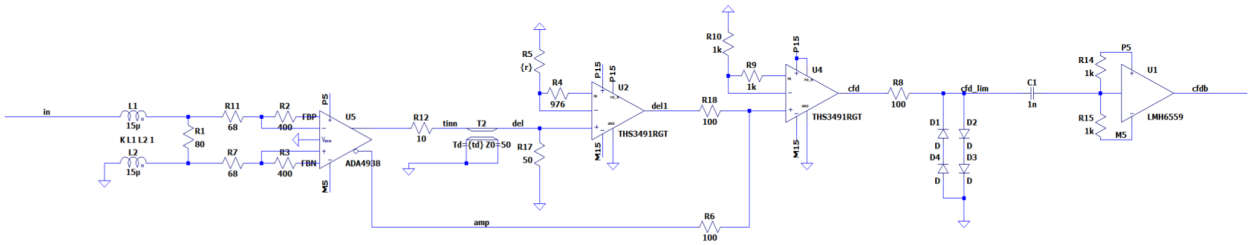

Figure 7.9: Simplified schematic of the Analog CFD circuit. Initial differential amplifier has gain 3 and provides both normal and inverted outputs; the delay line for normal signal is $\sim$ 1 ns and is implemented using an *in-PCB microstrip trace*; amplifier for delayed signal provides gain of around 5.

[3] spare units from the KTAG subdetector.
[4] used only with differential signals.

After this analog treatment the signal will be forwarded to a dual threshold comparator. All relevant parameters of the circuit are voltage controlled and may be set with trimmers or from DAC based on jumper configuration. DAC configurations and other CFD parameters may be set via the motherboard's ethernet interface.

The Veto Counter requires multiple CFD units, those are housed on a dedicated motherboard that provides power and LVDS interfacing to readout system. Moreover, an ethernet link on the motherboard can be used for CFDs control and setup purposes. The motherboard is designed to accomodate 16 TDC, thus a 32 channel TDC needs 2 motherboards; these are in the same 6U Eurorack format as the FELIX TDC.

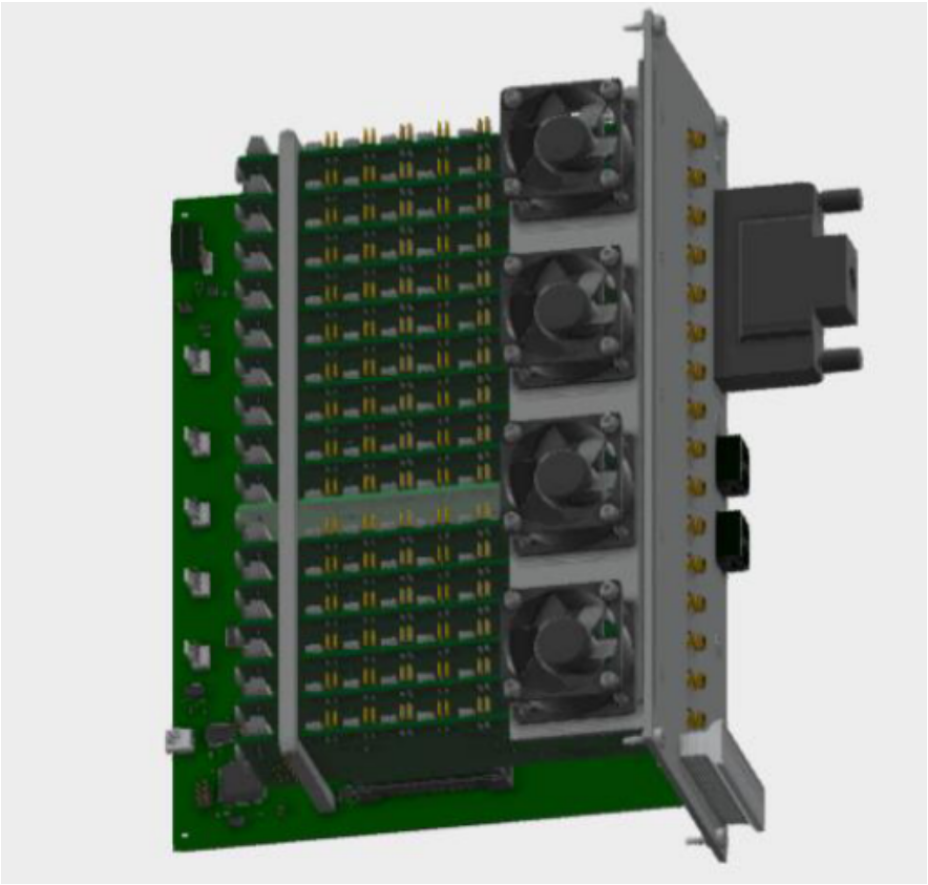

Figure 7.10: CAD drawing of the CFDs and motherboard assembly, LVDS connectors are on the front panel together with SMA input connectors.

7.5 CFD operation

The PMT pulses reach the CFD via the SMA input connector. All pulses that do not cross the *CFD threshold* (set to -30 mV) are discarded, while pulses above the CFD threshold go through the CFD and to the comparator circuit. The output of the comparator will depend on the amplitude of the CFD signal with respect to the *low* and *high* threshold:

- If the CFD signal is not crossing either threshold both high and low output will provide a 34 ns long pulse.
- If only low threshold is crossed then the low output will be a pulse whose length is the *time over threshold* (ToT) of the signal, the high output would be a 34 ns long pulse.
- If the signal crosses both thresholds then the respective output pulses would have ToT length for the respective threshold.

The operation is shown in Figure 7.11. Both low and high output is provided in Pseudo-NIM format on the LEMO connectors or through the card edge as LVDS signals.

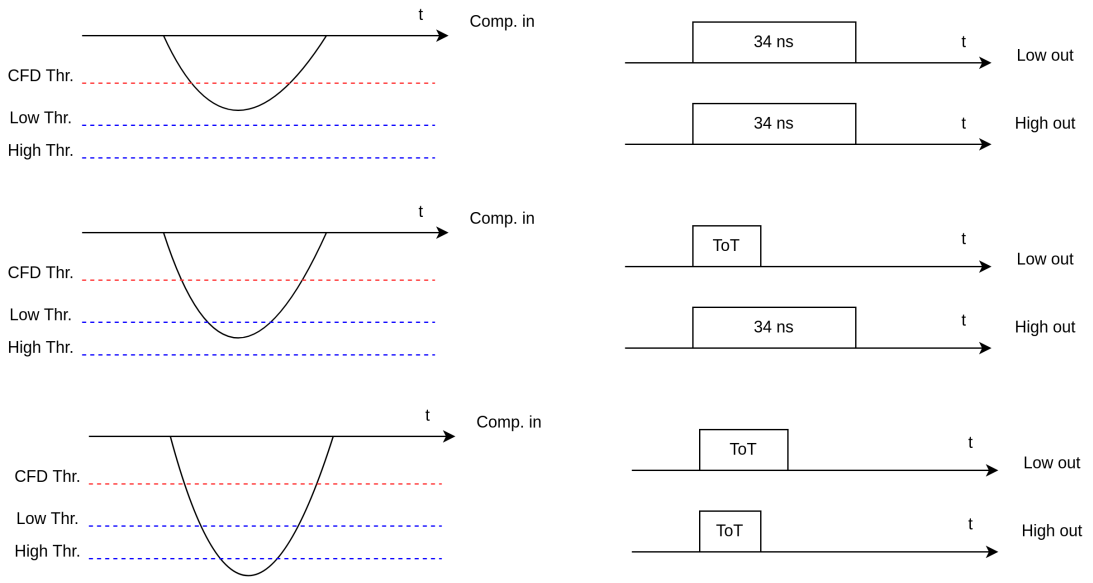

Figure 7.11: output shape of the CFD module for different amplitude of PMT pulses.

7.6 Readout system

The Veto Counter has a dual readout: one CFD module can provide the output comparator signals both in *pseudo-NIM* format through the LEMO connectors or in LVDS format via the motherboard. The LVDS signals from the motherboard are directly connected to the FELIX TDCs; being 22 channels per station and 66 channels total 4 CFD motherboards are used totalling 64 channels with two left unconnected and not being read[5] (Tile 10 of Station 3). Each phototube is connected to its own CFD module via a 7 m long coaxial cable with SMA connectors, see Figure 7.10. At the motherboard the output signals for 16 CFDs are aggregated on a 32-pin LVDS connector and each motherboard is connected to 1/2 of a FELIX TDC card, see Figure 5.1. TDCs are connected with custom FELIX optical links to FELIX card, located in its server in the farm room. The FELIX server has dual Intel Xeon Gold 6346 processors, 256 GB of RAM and and a dual x710 NIC for network communication. Being this an L1 exclusive detector data sending to the server follows the usual MRP-SDE packet scheme described in Chapter 4.

Legacy TEL62 readout is based on the pseudo-NIM outputs from the CFDs: the *low threshold* NIM signal is taken out from each CFD through a LEMO coaxial cable to dedicated NIM to LVDS converters; these provide an LVDS level signals that are again aggregated and brought to the spare channels of the ANTI0 TEL62 readout. The whole Veto Counter can be read by half a TEL62 (two HPTDC).

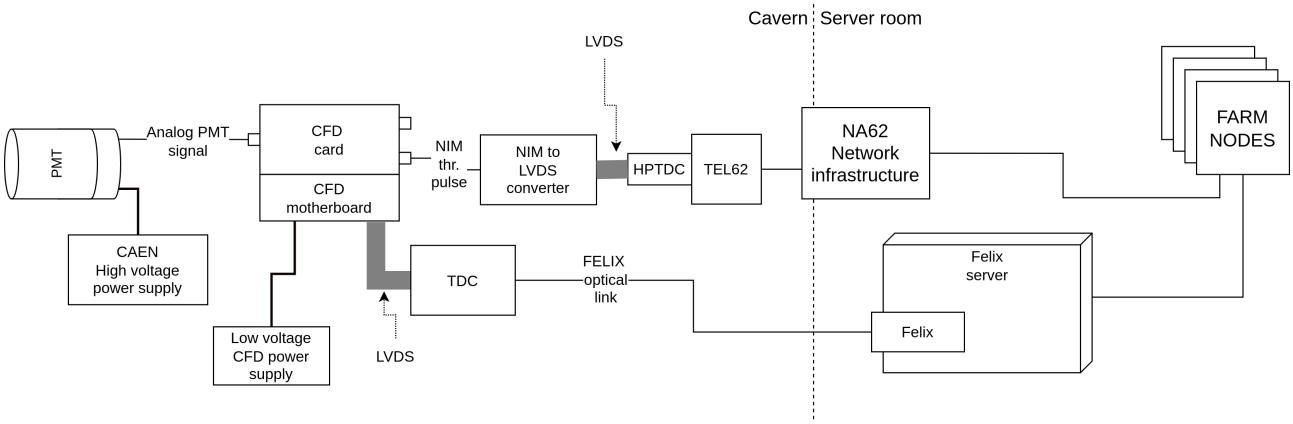

Figure 7.12: Block diagram of the dual readout system of the Veto Counter.

[5]The unread tile is very low activity and not needed.

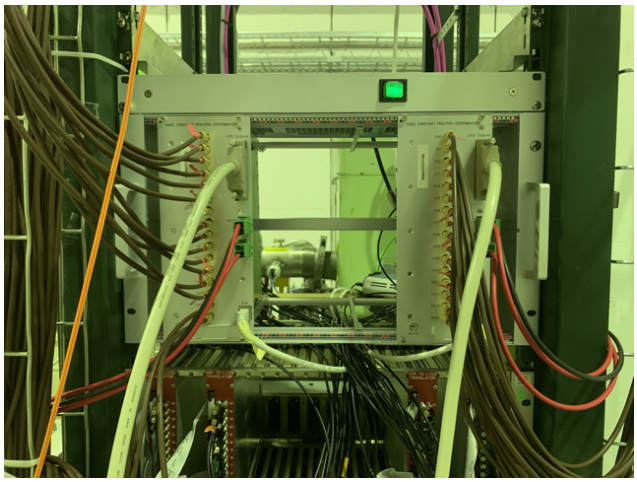

Figure 7.13: The CFD motherboards installed in the rack /1

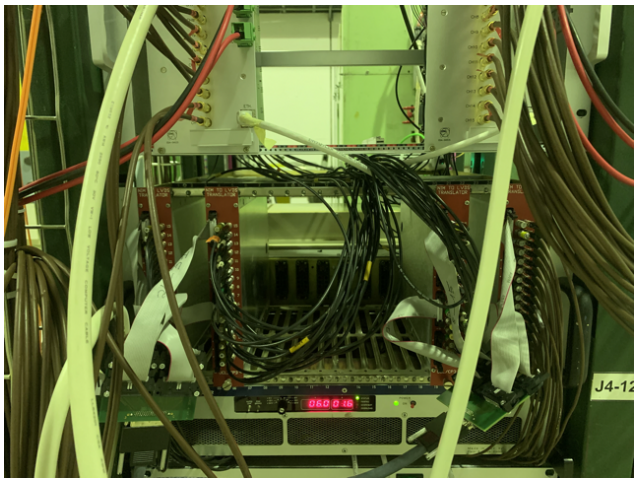

Figure 7.15: NIM to LVDS converters used in the TEL62 readout. Flat cables are the LVDS outputs going to the TEL62 TDCs and black cables are input NIM signals.

Figure 7.14: The CFD motherboards installed in the rack /2

Figure 7.16: The TDC modules for FELIX readout

Figure 7.17: General view of the Veto Counter station in the experimental hall. The rainbow module in the middle is the TCX collimator, stations are located at either side of it while the racks contain all the required power supplies and readout modules.

Chapter 8

Veto Counter performance studies

In this chapter we will describe the procedure designed to evaluate the performance of the Veto Counter. Since the subdetector is read out by two independent systems, one of which under development the goal of this work is to verify the performance of the FELIX readout against the proven TEL62 one. When talking about performance we refer to the quality of the reconstructed data obtained from the new readout system in comparison with the ones from the legacy system.

In order to quantify this metric the study is divided in two phases: at first a direct comparison between the raw data will be performed; these are the digitized hits for the readout system and are usually an intermediate step in the event reconstruction process performed by the NA62 Framework [47]. In principle the two readouts should register a comparable number of hits, being connected to the same front end. This analysis will reveal any low level issues in the system such as dead time between samples and buffer latencies.

The second phase of the study deals with the reconstructed events for each readout. The raw hits are processed by the reconstruction algorithms and the product of this operation are hit candidates with physical significance. Since during data analysis one is interested in working with those it is important to verify that the candidates makes sense and are compatible between the readouts. A good metric for observing this is the *efficiency*. Two different efficiency algorithms are developed and discussed and their results compared between the two readouts. Ideally one should expect a similar efficiency for those systems, the legacy TEL62 one taken as a reference. We should point out that subtle differences in the raw hit performance may have little or no impact to the overall detector efficiency computed with the designed algorithms. Those differences thus may not dramatically matter from an ordinary data analysis point of view[1].

While capable, the Veto Counter is not designed to provide positional information on the incoming particle hits. A simple estimation of the flight time for a photon emitted within the tile may be used to perform a positional calibration; the position information provided will be compared with the one provided by the extrapolated track in order to emphasize any systematic effect that may arise in the extrapolation procedure.

[1] for example, a missing hit on one of the two PMTs can show up in the hit analysis, but the corresponding loose candidate may be considered regardless in the efficiency evaluation.

8.1 Detector efficiency

The *efficiency* of a subdetector is defined as the ratio between the number of detected events and the total number of events that occurred in it. For example the intrinsic efficiency of a scintillation crystal would be the ratio between the photons emitted by the medium over the number of interactions that happened in it. Similarly, considering a whole subdetector, the efficiency for a given process will be defined considering the number of events expected in the detector over the registered event number. This metric will take into account all sources of inefficiency within the subdetector; an estimation of the total efficiency may be done evaluating each part separately and multiplying the results together.

The definition given before is purposefully vague. With "detected events" one may consider for example all the events of a given type that occur in the detector in a given time span; while with "total events" one may consider the expected number of interactions as from a Monte Carlo simulation of the system or events registered from a similar detector that is expected to be sensitive to the same phenomena. Similarly one may consider the already reconstructed candidates as detected events, and use this as a starting point of an efficiency algorithm.

8.2 Digitized data study

We start our analysis by observing the digitized data coming from the readout. These are obtained from the raw data with minimal processing. They are generated by performing the reconstruction of the raw data using the NA62 framework and stopping it before the candidates are built.

Given the high computational resources needed to perform the reconstruction as well as the size of the resulting data files this study is feasible only on a very small subset of data. In particular this work was based on Burst 634 of Run 12565. This specific burst has been taken roughly at the end of the 2022 data taking and thus is representative of the final subdetector status by the end of the data taking period. Together with 634-12565 two more bursts (51-12252 and 302-12277) have been analyzed. While the previous was a nominal intensity burst; these other two have been acquired with a very low intensity beam, $\sim 25\%$ and $\sim 10\%$ of nominal intensity respectively. The results of this study is presented in 8.2.6.

8.2.1 Digitized data format

Before describing the study it is important to point out how the digitized data are formatted: each *trigger event* is associated to zero or more *hits*; these represent a reading by one of the PMTs. A hit is associated to a *channel*, that maps uniquely to one phototube. As detailed in Chapter 7 there are two PMTs reading each tile. It is common to name one side *Jura* and the other *Saleve*[2].

A channel ID is a 3 digit number xyz where $x = 1, 2, 3$ represents the station and $yz = 00, 02, \ldots, 10$ represents the tile. All channels from Saleve side have an offset 50 on the tile number; so for example:

$$\text{Channel 252} \longrightarrow \text{Station 2, Tile 2, Saleve PMT,}$$
$$\text{Channel 307} \longrightarrow \text{Station 3, Tile 7, Jura PMT.}$$

In the following we will refer to channels as well as to stations and tiles; in the latter case the data coming from Jura and Saleve sides are merged.

The FELIX TDC also reads the high threshold of the CFD and maps it to channels $1xyz$; as an example Channel 1256 would be the high threshold for Station 2, Tile 6, Saleve PMT. Since these information are not provided to the TEL62 readout they are ignored in this analysis.

A hit is registered when at least one *signal edge* of the CFD comparator output is caught within the trigger readout window, details are available in Chapter 7. The detection time of each edge is available within the digitized data, from this information one may infer the hit time with respect to the trigger time.

[2] these are the names of the mountain ranges at each side of the experimental area.

8.2.2 Total number of hits comparison

The first observation is on the detected number of hits for each readout: each trigger event is associated to a number of hits. In Fig. 8.1 we thus show the histogram of the hits for both readouts.

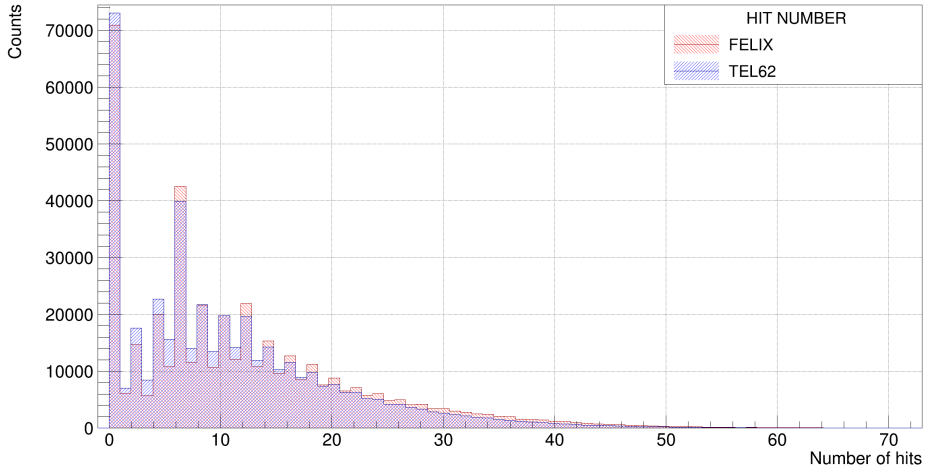

Figure 8.1: Total number of hits per trigger event for the FELIX and TEL62 readout.

We notice that on average the FELIX readout detects a larger number of hits with respect to the TEL62; this is particularly evident in the range $[20; 50]$. It is reasonable to assume that part of the events that got a lower number of hits in the TEL62 readout are cause of the excess in the low count area (below 10 registered hits). The case $N_{\text{hits}} = 6$ is relevant and may correspond to an extremely common condition[3]. The high rate of this event may overwhelm the TEL62 readout due to its already discussed limitations. We will expand this matter further by examining other metrics.

Continuing on this, it is also interesting to compute the difference in number of hits between readouts per trigger event, this is shown in Figure 8.2.

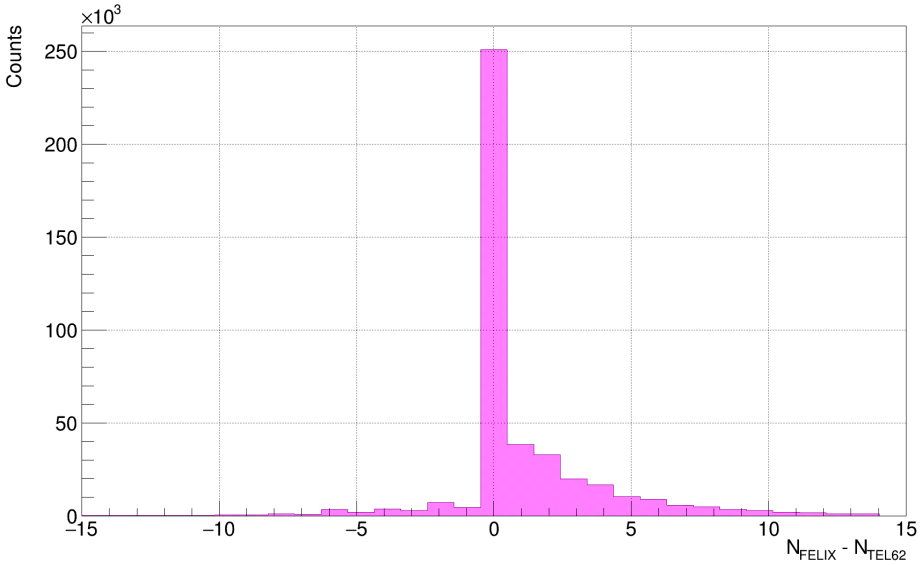

Figure 8.2: Difference in number of hits between the FELIX and TEL62 readout. each entry is the net difference between the two readout number of hits

[3] we would like to remind that the tiles closest to the beampipe have an extremely high activity and maybe it is common to have an event that causes 6 hits.

Again, here we notice more clearly that on average the FELIX readout captures more hits per event comparing to the TEL62. The *skewness* of this distribution is:

$$\mu_3 = 0.939 \pm 0.004; \tag{8.1}$$

8.2.3 Detected edges comparison

In the previous section we considered the total number of hits; that is we discarded all information on the individual hit such as channel or timing. Lets now focus on the timing proprieties of the hits; in particular the *detected edges*.

The trigger window is 200 ns long and goes from -75 ns to +125 ns with respect to the trigger time. An output pulse from the CFD features a leading and a trailing edge; see Figure 7.11. Any edge at the TDC input and within the trigger window is considered a hit on the given channel. If both edges are captured in the window this is a *complete hit*, meanwhile if only one edge is present it is a *partial hit*, either leading or trailing edge only; we should point out that having partial hits in the readout is not automatically wrong: if a given hit happens very early (very late) with respect to the L0 trigger then only the trailing (leading) edge may be captured. In principle those incomplete hits should not be relevant because the trigger window is specifically tweaked to get the relevant hits for the physics that is being studied. The *raw hit time* is the time of the edge for incomplete hits and the mid-pulse time for complete hits. Regardless, observing the event distribution within the trigger window can be useful to benchmark the readout.

A sketch of the different types of event conditions is visible in Figure 8.3.

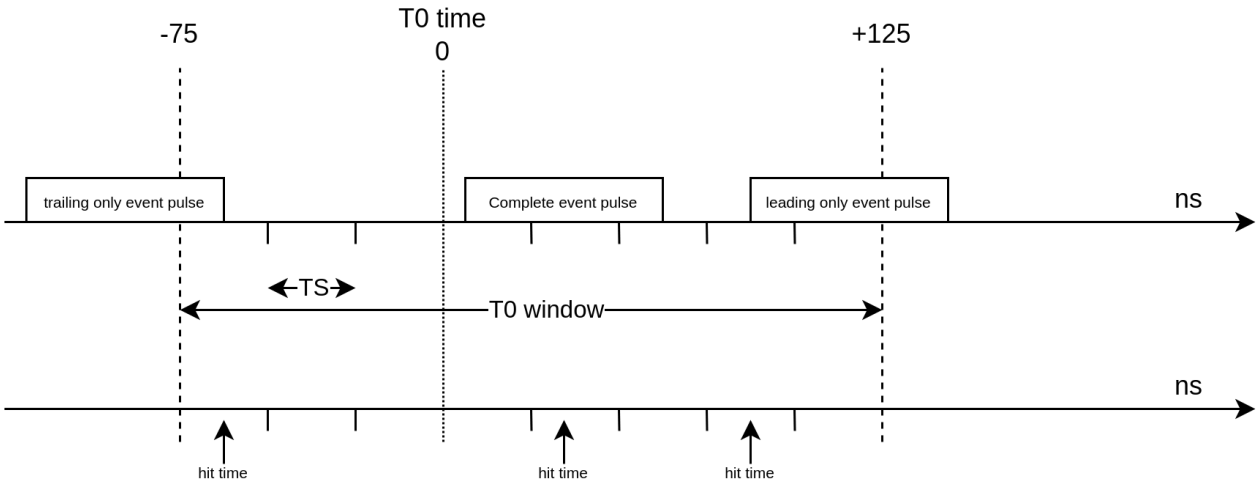

Figure 8.3: Various pulses coming from the CFD comparator associated with different types of events: both edge event (complete) and partial events in with leading or trailing edge only. the time slot (TS) of 25 ns is the coarse time block of the trigger window.

In Figure 8.4 we may observe the number of complete hits and their time within the trigger window.

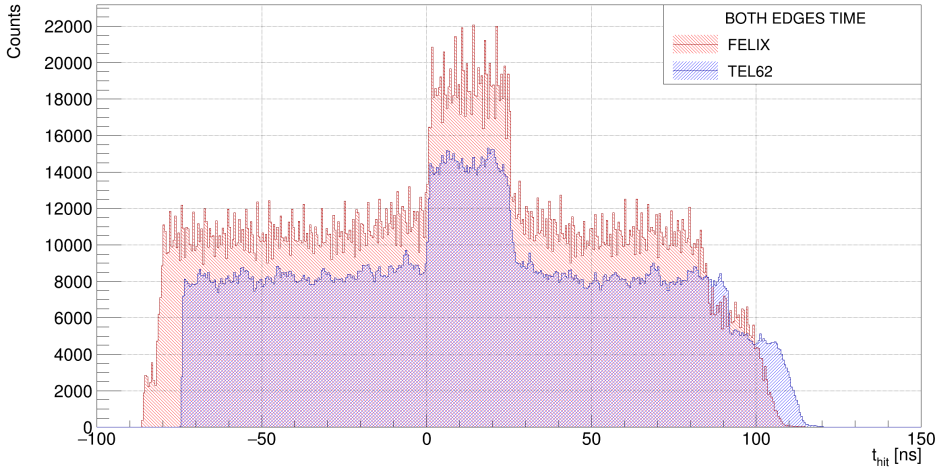

Figure 8.4: Complete hit (both edges) time per readout system.

As said before we can notice that there are significantly more hits in the time slot following the trigger time. This corresponds to what was said before concerning the trigger window tweaking. Moreover we can notice that the number of complete events is significantly higher for the FELIX readout throughout the entire trigger window. The misalignment of the trigger window edges is normal given that these are two different readout systems.

Lets expand this further by checking the incomplete events:

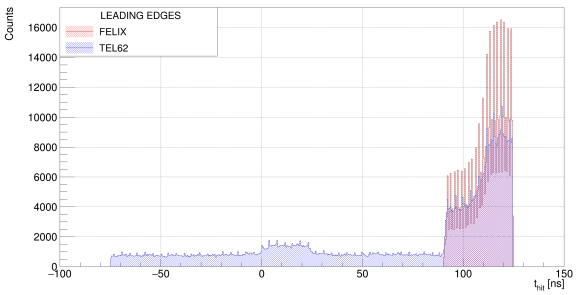

Figure 8.5: Partial hit (leading edge only) time per readout system.

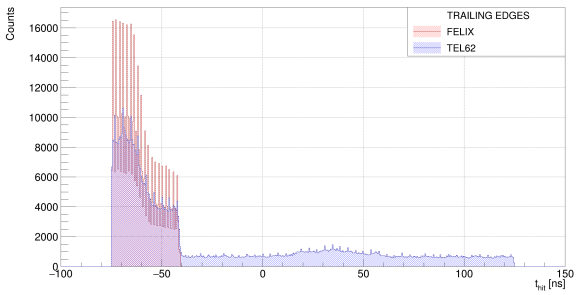

Figure 8.6: Partial hit (trailing edge only) time per readout system.

Notice how the FELIX readout has incomplete hits only appearing close to the boundary of the window, while the TEL62 readout shows a significant number of incomplete hits throughout the whole trigger window with a slight bump coinciding with the first timeslot after the trigger time. Notice also that there seems to be a bias towards some time values in the FELIX readout, this can be due to a firmware limitation. As was seen before the first timeslot is the one that has more events happening and thus maximum strain on the readout. While the peak of partial event near the boundaries is expected (the other edge would fall out of the trigger window regardless) and irrelevant for the physics (those hits are very far from the trigger) the constant background of partial events throughout the window shows that the TEL62 is always losing edges and identifies a fraction of the events with both edges within the trigger window as incomplete.

To conclude this observation in Figure 8.7 the ratio between the number of hits for each event type (leading, trailing and both) is shown.

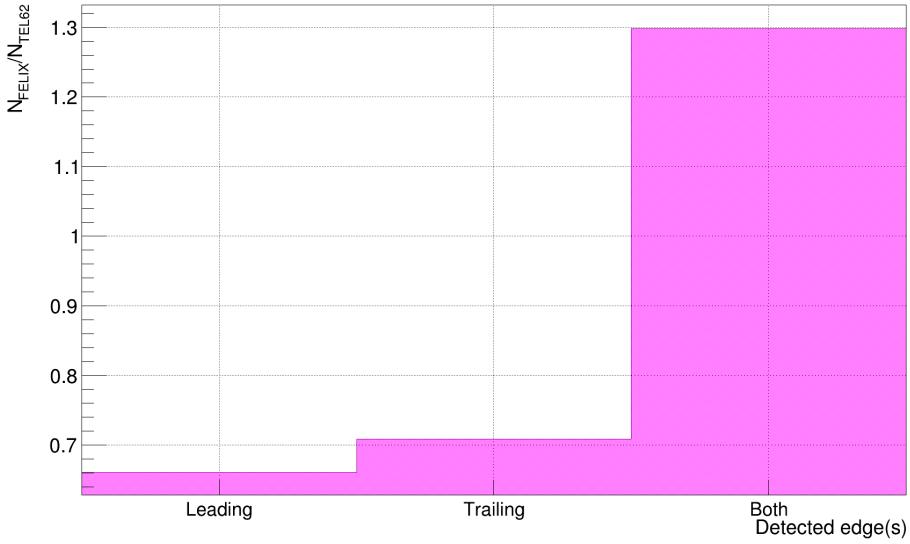

Figure 8.7: Ratio of hit numbers between the FELIX and the TEL62 readout grouped by event type; we observe that "Both" events (complete) are higher in number for the FELIX readout.

Again, the complete events are substantially more for the FELIX readout, while the incomplete are less: on those the difference is mostly due to lost edges in the TEL62 readout, as seen in Fig 8.5 and 8.6

8.2.4 Number of hits comparison

In Section 8.2.2 we compared the total hit number per event without any regard for *where* the hit is coming from. Now we will continue the reasoning followed there by including this information. Lets start with studying the number of hits as function of channel number; the dependency on channel number should be related to the different physics conditions among the Veto Counter stations.

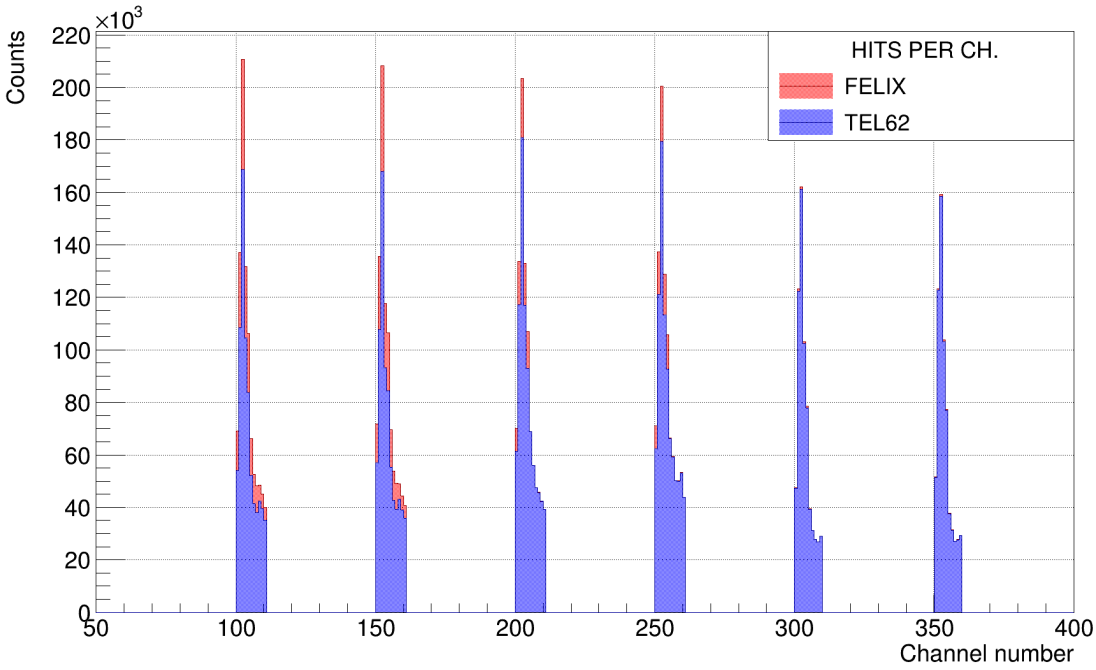

Figure 8.8: Number of hits per channel for both readouts.

From Figure 8.8 we can immediately observe how the activity in the channels corresponding to Tiles 2-3 is the highest; these are the small tiles closest to the beam pipe. In general the tiles corresponding to low Tile ID are the most active. Moreover all the channels corresponding to Station 3 show significant less rate compared with station 1 and 2. This is also expected due to Station 3 downstream of the TCX collimator and thus in a less active area; the hit number difference between readouts is far less evident in Station 3 as well. This is further evidence that the FELIX readout is better than the TEL62 in coping with high rate conditions: on Station 3, where rate is low this limitation is not visible, but gets more and more relevant on Station 2 and Station 1.

In order to better understand what is happening to the TEL62 readout at high rates lets observe the hit ratio as a function of the channel ID: while this representation essentially contains the same information shown in Figure 8.8 here the difference between the various stations is more easily visible.

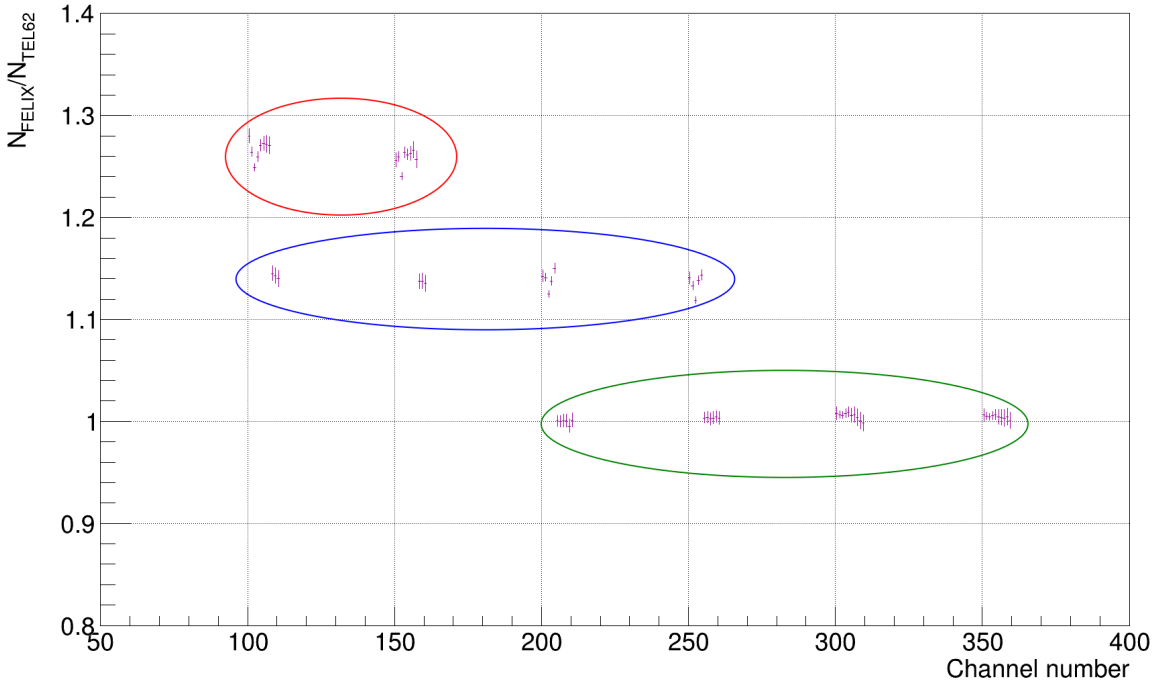

Figure 8.9: Ratio of hit number per channel number.

We notice how the highest number of lost hits happens for Tiles 0-7 in Station 1 with approximately 25% loss (red circle). A lower loss, in the order of 15% happens in Tiles 8-10 in Station 1 and 0-4 in Station 2 (blue circle). For the reminder of Station 2 and all Station 3 tiles the difference is negligible (green circle).

The origin of this effect is to be found in the TEL62 HPTDC operation: this device has a rate limit for every subsequent 8 channels as well as an intrinsic rate limit per HPTDC board [8]. The PMTs are connected to the HPTDCs channels following their natural order: the first 16 HPTDC channels are used to read out tiles 0-7 of Station 1, the subsequent 16 for Tiles 8-10 in Station 1 and 0-4 in Station 2 and so on. It is well possible that the TEL62 readout may hit the 8 channel rate limit on the first 16 channels and, to a lesser extent, on the second subsequent 16 channels.

The FELIX readout uses a completely different, higher performance TDC that is well below its rate limit. So this effect is not present.

For completeness, here we show the analogous of Figure 8.10 but as function of Tile ID and Station ID. It is very easy to spot the different groups of tiles, we also provide the exact value of the ratio for all tiles.

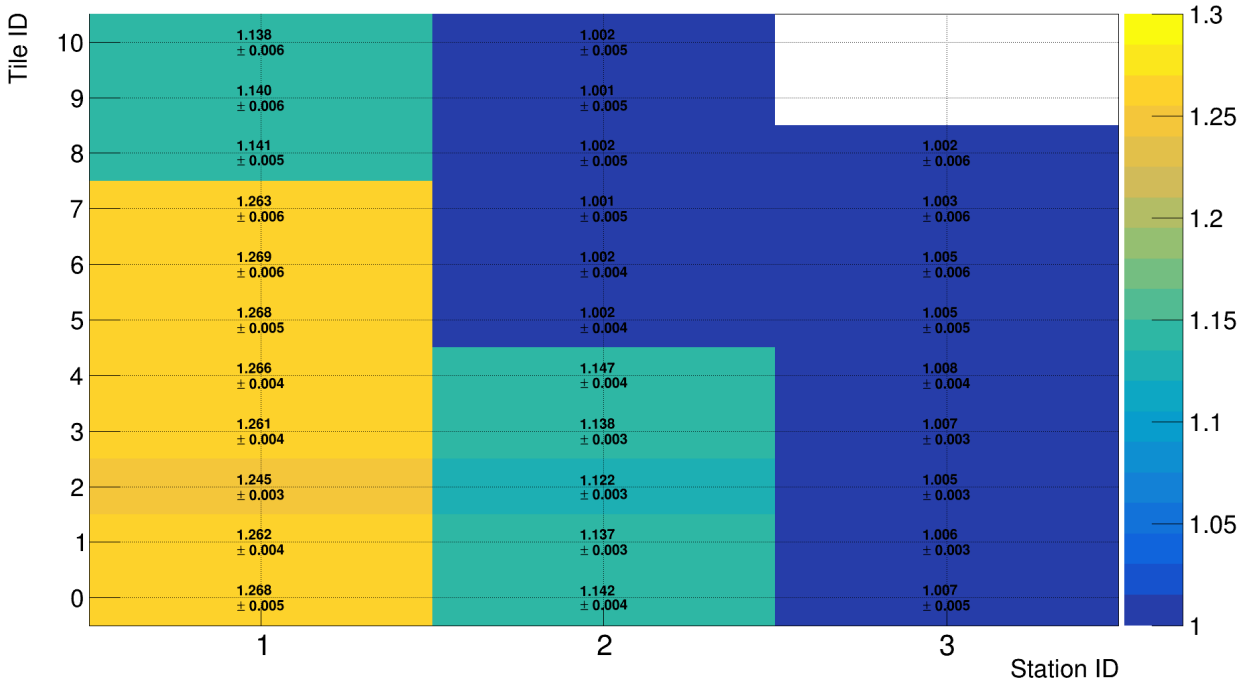

Figure 8.10: Ratio of hit numbers per Tile ID and Station ID.

8.2.5 Hit matching

Up until now we always considered the hits from the two readouts completely separately. While before we compared only the number of hits here we will perform a very simple hit matching to directly compare the performance of the two detectors.

Hit matching algorithm

Given a specific trigger event there will be a variable number of hits from the TEL62 readout and the FELIX readout. Each of those hits will be associated to a timestamp with respect to the trigger time as well as a channel ID.

Our selection finds for every FELIX hit the first TEL62 hit with matching channel and a time separation of no more than 1 ns. Since the FELIX readout proved to be more resilient to high rate conditions it makes logical sense that there will be many unmatched hits for the high rate channels. First of all lets consider the ratio between the matched hits and the total FELIX hits as a function of the channel. This is shown in Figure 8.11.

As was predicted in the previous statement many channels exhibit a dramatic number of mismatched FELIX hits, reaching peaks of $\sim 30\%$. The dependency between mismatched hits and tile activity is well visible; with the biggest effect showing on the low ID tiles of Station 1, particularly on the Jura side.

We should also notice that the number of matched hits is never exactly the same of the FELIX hits; not even in the very low activity channels. This happens because even though the FELIX readout is higher performance than the TEL62 in high hit conditions neither of those readouts has unitary hit detection efficiency, so there can be a small number of hits undetected by either readout for every trigger event. The same hit match ratio is shown on a Tile ID and station ID basis in Figure 8.12.

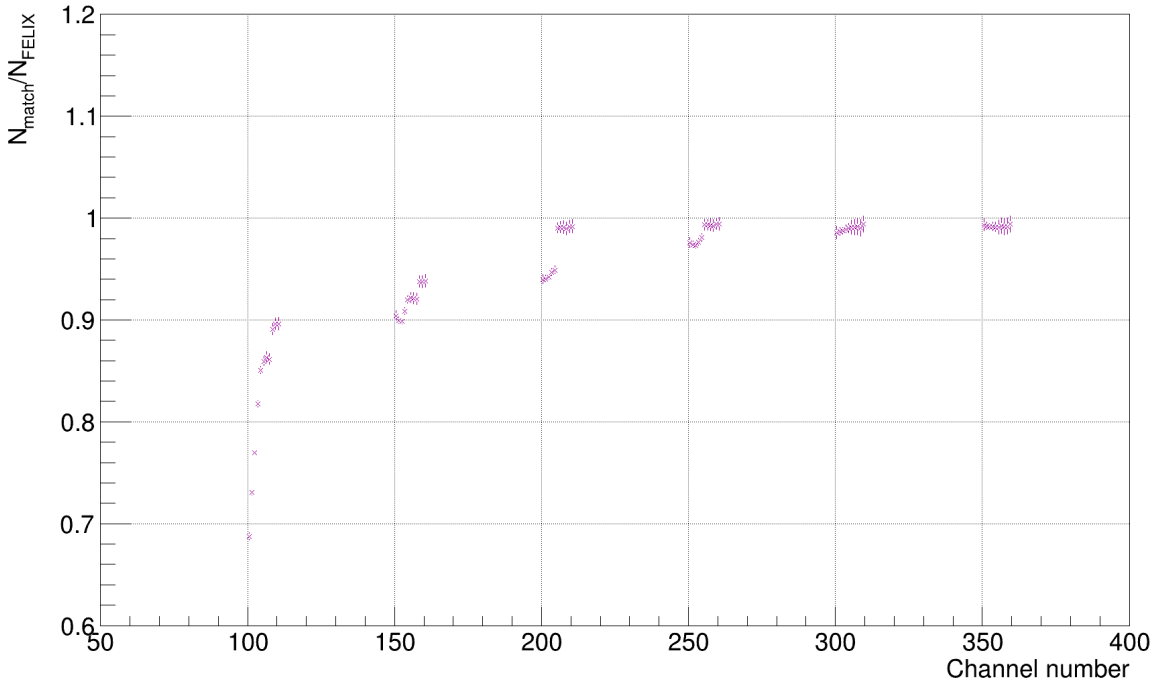

Figure 8.11: Ratio of matched FELIX–TEL62 hits and FELIX hits as function of channel number.

Figure 8.12: Ratio of matched FELIX–TEL62 hits and FELIX hits as function of Tile ID and Station ID.

8.2.6 Low intensity data

Lets now perform the same analysis described before on data taken at very low beam intensity.

At nominal intensity a dramatic difference in total number of hits between the two readouts was observed: this loss of hits from the TEL62 readout was very evident on the most active channels (Station 1, low tile ID). We expect that at lower intensity this loss should be much lower. In Figure 8.13, 8.14 we show the equivalent of Figure 8.2 for the two low intensity bursts studied.

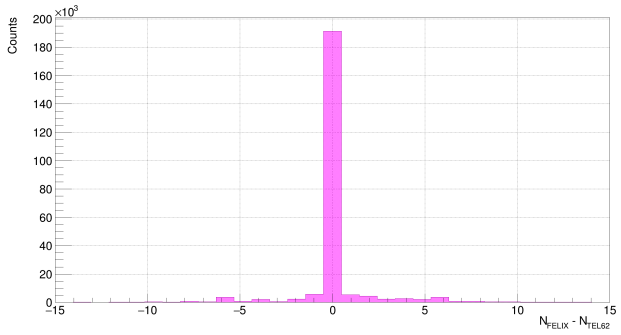

Figure 8.13: Difference in total number of hits between FELIX and TEL62 readout at $\sim 10\%$ intensity.

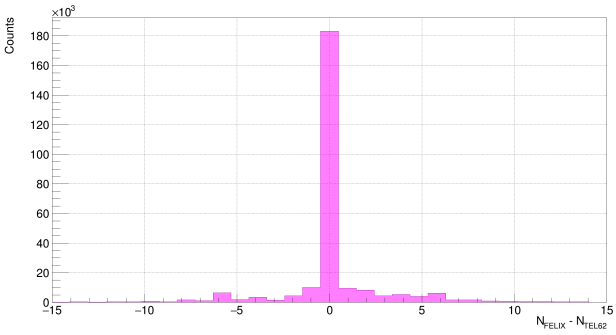

Figure 8.14: Difference in total number of hits between FELIX and TEL62 readout at $\sim 25\%$ intensity.

While Figure 8.13-8.14 appear similar, the data taken at $\sim 25\%$ intensity shows a higher hit loss for the TEL62 readout (right tail of the distribution is larger).

The total number of hits as function of channel does not show the same profile that was seen in Figure 8.8: at low intensity the hit loss is more uniform among channels, the issues observed previously regarding HPTDC channel octets saturating are no longer present and the source of the hit loss has to be searched elsewhere.

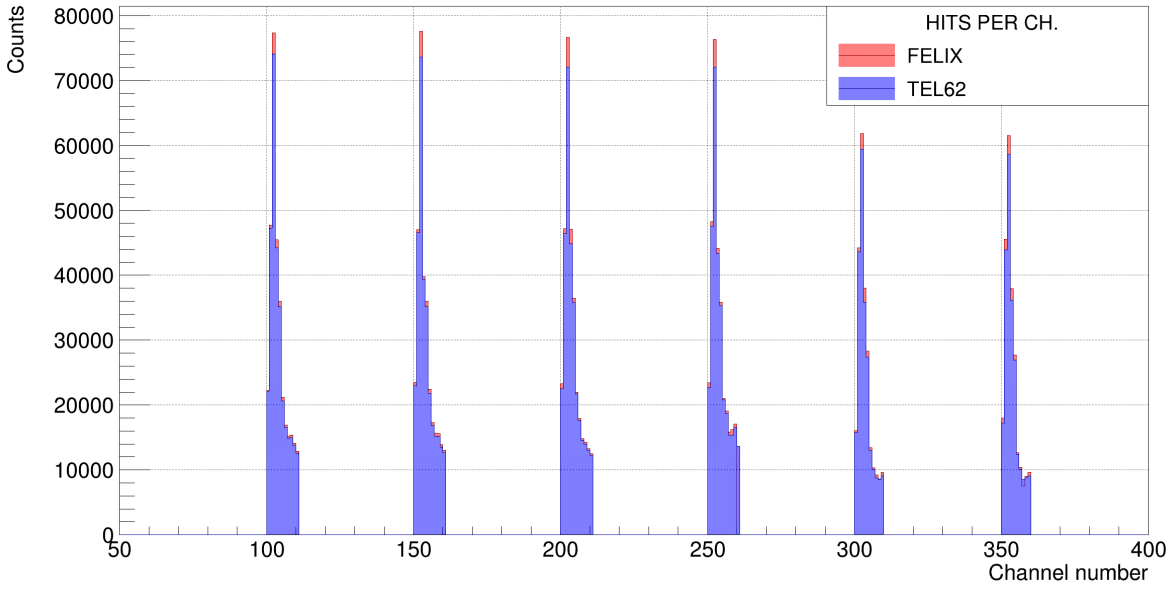

Figure 8.15: Total number of hits per channel at $\sim 25\%$ intensity, notice channel 260 is missing in the FELIX readout due to a fault during the run.

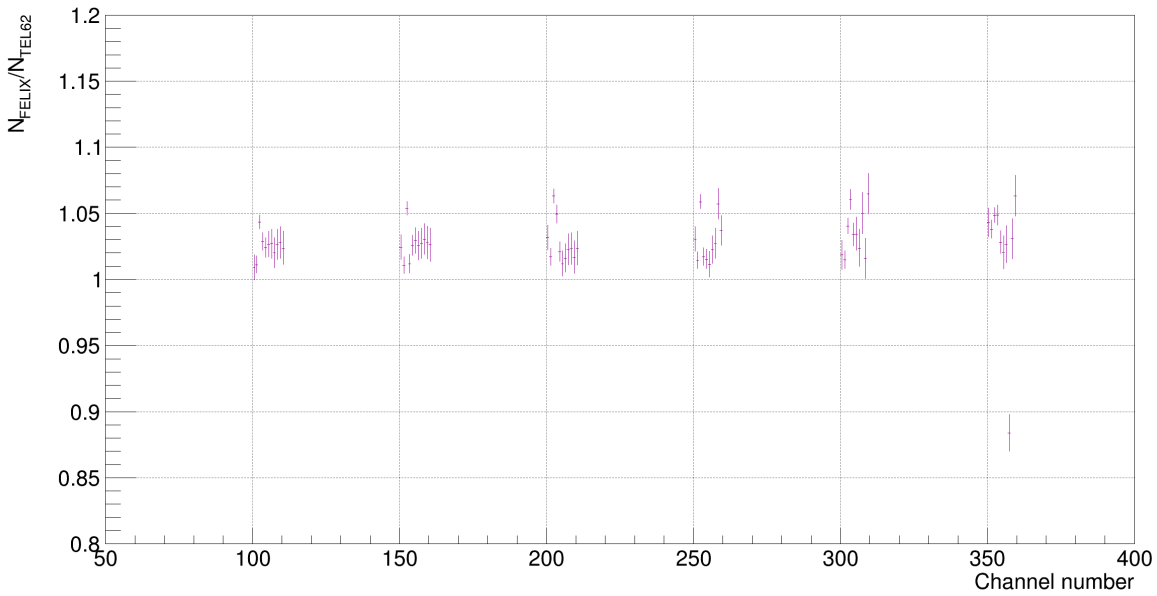

Figure 8.16: Hit ratio between readouts per channel at $\sim 25\%$ intensity. A small hit loss of the TEL62 readout is widespread. Notice the issue at channel 357 and missing channel 260.

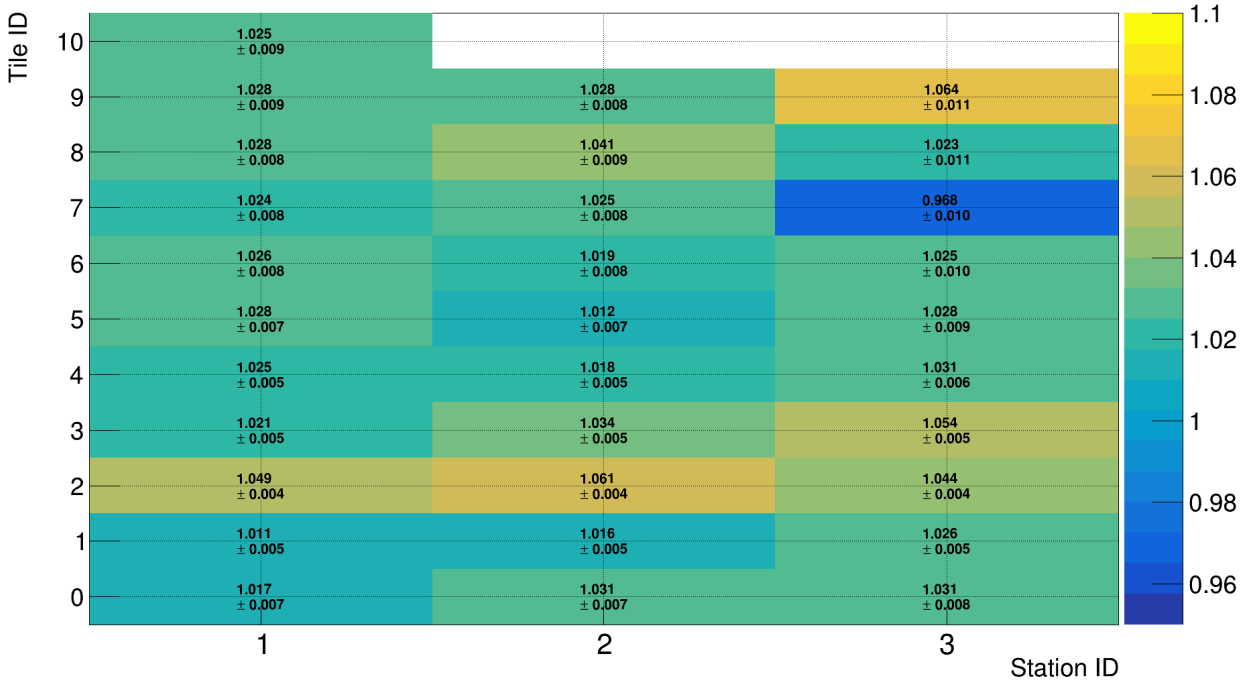

Figure 8.17: Hit ratio between readouts per tile ID and station ID at $\sim 25\%$ intensity. Notice that tile 7 of Station 3 shows a small FELIX readout loss caused by low hit efficiency of the corresponding Saleve PMT (channel 357). Tile 10 of Station 2 is not present due to a hardware fault of the FELIX readout on one of its channels (260).

Regarding detected edges the constant TEL62 hit loss throughout the whole trigger window is no longer present; meanwhile there is a small hit loss at specific leading edge times (border of each 25 ns timeslot) for the FELIX. Partial trailing only events are instead registered throughout the whole trigger window. The low intensity data were taken earlier in the run with a very early TDC firmware version. Most likely these issue is due to a firmware bug fixed in the later revisions.

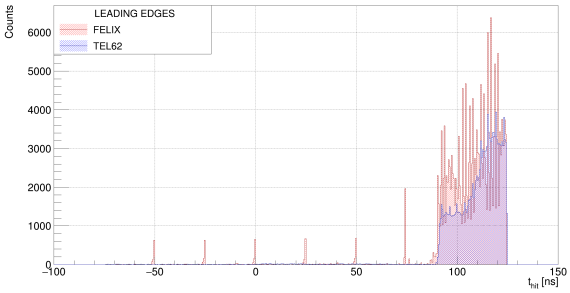

Figure 8.18: Leading edge only events at $\sim 25\%$ intensity.

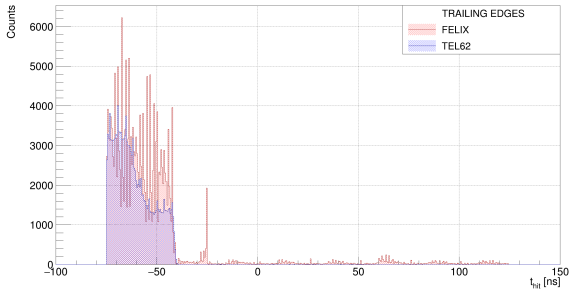

Figure 8.19: Trailing edge only events at$\sim 25\%$ intensity.

Hit matching between the two readouts shows the same high number of mismatched hits at low channels observed at nominal intensity; apart from this peculiarity the hit mismatch between the two readouts is $\sim 8\%$.

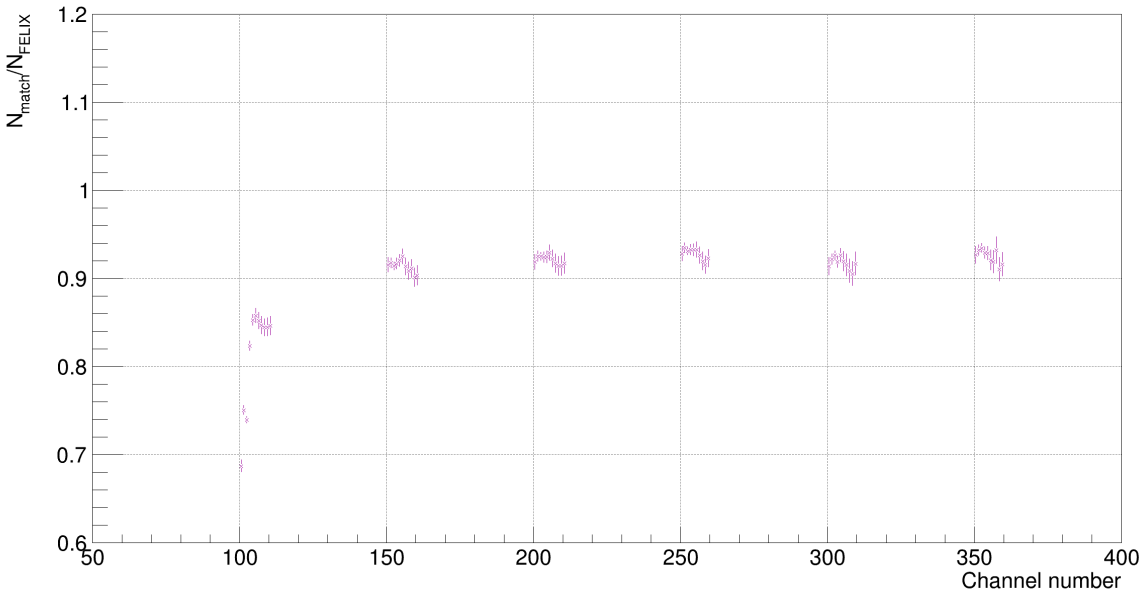

Figure 8.20: Ratio of matched hits and FELIX hits at$\sim 25\%$ intensity.

Again, this is due to the widespread lower hit detection efficiency of the TEL62 readout; while not as dramatic as at nominal intensity it is still present.

8.3 Efficiency study

8.3.1 Introduction

The main operating mode of NA62 is with Kaon beams; in line with the experimental search. One is interested in evaluating the Veto Counter efficiency in localizing and vetoing events that involve a π candidate produced by beam-material secondary interactions (Section 3.3). The algorithms should then run on Kaon runs data and the computed efficiency can be used to estimate the vetoing performance.

Another mode of operation of NA62 is the so-called *muon mode*. In this mode the muon scraper magnets upstream in the experiment are turned off and thus the beam is extremely muon rich. This mode is usually used to test sub-detectors and for tuning the DAQ system. Data collected during a muon run may be used while developing the algorithms for debugging purposes: Veto Counter exhibits an higher hit rate on muon runs, thus testing the selection and the algorithms is significantly faster. A good efficiency algorithm should be able to run on both types of data without modification.

The efficiency is an important metric for detector performance and a comparison between the two readouts can be used to see if the FELIX readout is able to match the legacy readout in a real world application. Given the results obtained in 8.2 a very good result is expected, given that the FELIX readout showed to surpass the TEL62 one consistently, especially in high rate conditions.

Event selection for efficiency

Given the intended use of the subdetector the efficiency is computed using reconstructed data with PNN trigger filtering. The constraints on the events to be used are the following:

- *Only one* downstream π track in the STRAW:
  - That triggered at least 4 straw tubes (good candidate);
  - That has a good reconstruction quality ($\chi^2 < 5$);
  - With a momentum < 40 MeV.
  - In the acceptance range of MUV3 and NewCHOD;
  - Has to be associated in time and space with MUV3: difference between track time and MUV3 time below 10 ns;
  - Either be tightly associated in time and space with CHANTI or ANTI0: difference between MUV3 time and CHANTI (ANTI0) time below 2 ns;

Notice that if a track is associated with a MUV3 candidate its time is assumed to be the MUV3 one. This is due to the superior timing resolution of MUV3 with respect to STRAW. Specifically, for each single track event the algorithm locates the closest in time MUV3 candidate spatially associated to it, and assumes the time of that candidate to be the algorithm reference time (i.e. the *track time*). If there is no associated MUV3 candidate it means the track has no precise enough timing information available and the whole event is skipped. All the subsequent time association checks are done using this MUV3 time and thus are strict. Later in the efficiency computation every time we refer to *track reference time* we are referring to MUV3 time.

The "Halo" Efficiency algorithm

The Veto Counter is located[4] right at the beginning of the decay fiducial region. Given that this subdetector has to veto spurious pions coming from secondary interactions upstream of it one may consider the STRAW data to compute the number of expected events. STRAW is a spectrometer that provides, at reconstruction level, downstream particle trajectories from the track fitting. By knowing the downstream π track it is possible to extrapolate the position it intercepted every VC station[5]. This *expected hit* is then checked against the reconstructed VC candidates in each station and if a candidate satisfies

[4] after GTK station 3 and before CHANTI.

[5] if it intercepted it at all.

a set of timing and positional constraints the hit is considered to have been detected. Events that pass the selection are categorized based on which tile of the VC got hit, this provides the expected hits per tile and is simply done by extrapolating the track position at each VC station plane and checking which tile got traversed. A 5 mm *safety margin* is added on the beamline hole and all around the perimeter of each station.

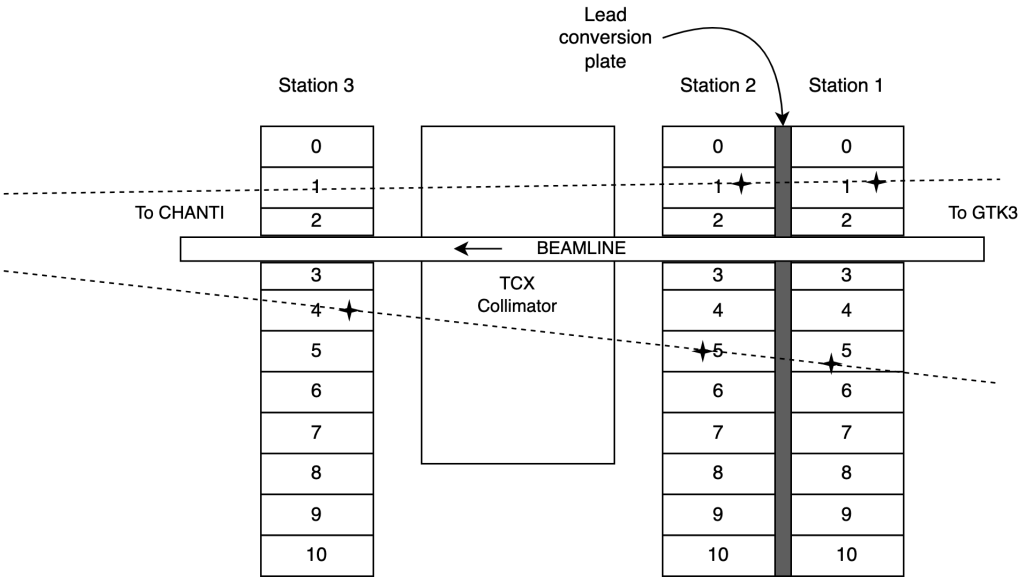

Figure 8.21: Structure of the Veto Counter subdetector together with some extrapolated downstream tracks. Track on the top got registered in Tile 1 of station 2 and station 3, producing a candidate. Track on the bottom gets registered in all stations producing a candidate for all of them.

Every trigger event may provide zero or more *Veto Counter reconstructed candidates*, these are reconstructed hits that happened within that trigger window. These candidates are categorized based on the station they come from. After this the registered hits computation may start for the given single track event.

For any given station each candidate hit tile is compared with the tile supposedly hit by the track: if the tile the track hit is the same of the candidate or one of its neighbours[6] and the time difference between the track and the candidate is sufficiently low (< 2 ns) then that station is considered to be efficient for that track event.

The "self halo" Efficiency algorithm

The Halo Efficiency has some limitations: first of all it relies on extrapolated downstream tracks and this can be unreliable. The VC is located around the beam line and there are significant material obstacles between the fiducial decay region and the subdetector stations: some tracks get extrapolated outside the beam line and while still in VC acceptance the interaction with atmospheric pressure air can scatter the particles and swamp the predicted track position. This is especially true for station 1 and 2 that lie upstream of the TCX collimator and even more for station 1 that has the additional shielding of the photon conversion plate; see Figure 8.21. Moreover the extreme length of the upstream extrapolation can amplify any angular error from the STRAW, predicting an impact position very far from the actual cross position; this is especially relevant on high ID tiles that are far from the beam pipe center. We should also point out that the trajectories are extrapolated through the spectrometer magnet. Lorentz bending of the tracks is considered and corrected but the magnetic field is assumed uniform in the magnet area, a model of the field profile was not available.

The self efficiency algorithm does not use any extrapolated track information; all the data comes within the Veto Counter. Selection of relevant events is performed the same way as the normal halo

[6] TileID ± 1.

efficiency but after that no extrapolated track data is used: if station 3, *reference station* has exactly one tight candidate then we expect the other two stations to register an hit in the same tile or in neighboring ones. The Tile ID of the station 3 candidate is not used, only the timing information. The computation is done assigning to one station the role of *probe station* while the other is the *tested station*. After the efficiency of the tested station is computed the roles are swapped. If the probe station has exactly one tight candidate that is in time with the reference station (< 1 ns) a matching candidate in the tested station is expected: all the candidates in the tested station are checked to see if any matches the hit tile or its neighbours. If a hit is found then the tile is considered efficient.

This algorithms provides efficiency metrics on tiles only, notice also that it is not applicable to station 3 being that used for reference.

8.3.2 Dataset choice

The efficiency has been calculated on a subset of bursts from run 12437; the efficiency algorithms need to run on reconstructed data. Thus, data availability depends on the status of the official reconstruction and event filtering. Since the Veto Counter FELIX readout is still in development and always subject to modification, changes to the system may happen at any time during the run and may not be promptly propagated to all the machinery involved in the official NA62 data reconstruction. Run 12437 was chosen as it was one of the most recent fully reconstructed runs (at the time of writing) for which the official PNN filtered data shows no major issues on the FELIX readout. Performing an unofficial reconstruction for another run would be time prohibitive.

8.3.3 Event selection, timing of tracks

The following plots are intended to show some overall characteristics of the track events and underline their timing characteristics with respect to the track reference time (MUV3 time).

In Figure 8.22 we can observe the time difference between the loose STRAW track time and the associate MUV3 candidate time.

Notice how the time difference distribution has a double peak; the main peak is associated with the MUV3 associated candidates that are very close in time with the track within the same time slot. The second, lower peak corresponds to the MUV3 associated candidates detected in the following trigger timeslot (25 ns apart). The mean value of time offset for the first peak is ~ 400 ps, in line with the detector timing performance. We remind that the STRAW to MUV3 association is performed on a spatial basis only; an histogram of the MUV3 candidate to extrapolated track distance is shown in Figure 8.23.

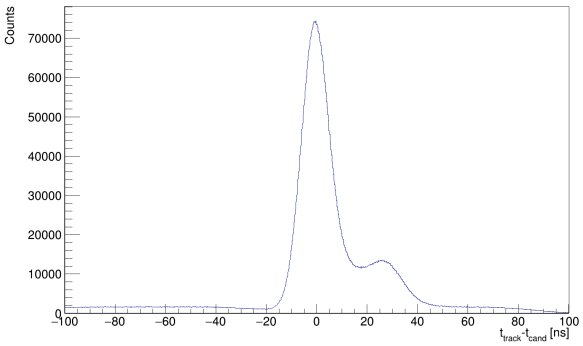

Figure 8.22: Time difference between single tracks and MUV3 associated candidate time

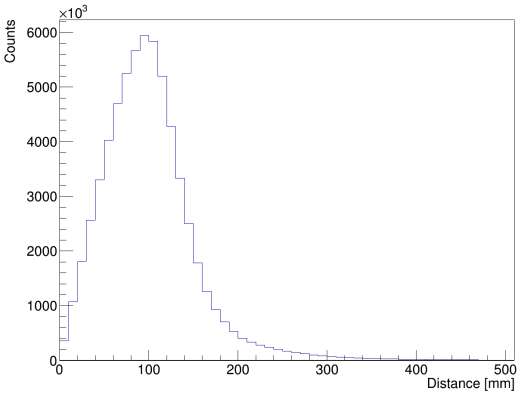

Figure 8.23: distance distribution between single tracks and MUV3 candidates.

Following the MUV3 association the STRAW time is no longer used and the ANTI0 and CHANTI

time associations are performed.

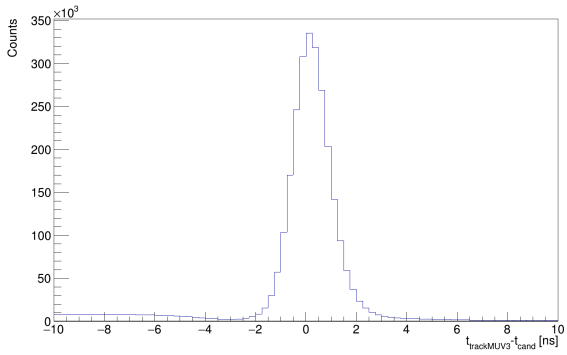

Figure 8.24: Time difference between Track time (MUV3) and associated CHANTI candidate time.

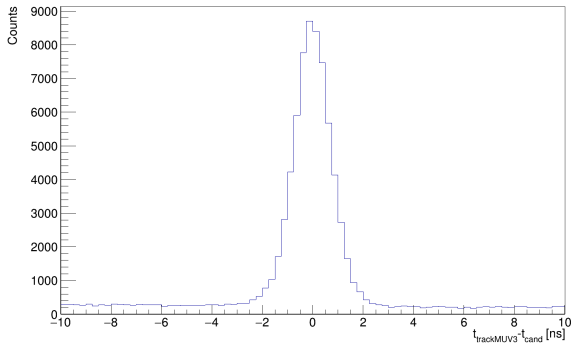

Figure 8.25: Time difference between Track time (MUV3) and associated ANTI0 candidate time.

8.3.4 Reconstruction candidates

Following the event selection it is possible to evaluate the number of Veto Counter candidates for every station. This is interesting because it is the closest metric to the raw hit number studied previously.

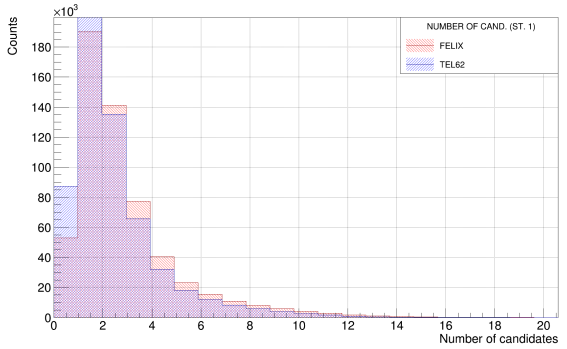

Figure 8.26: Number of candidates for station 1.

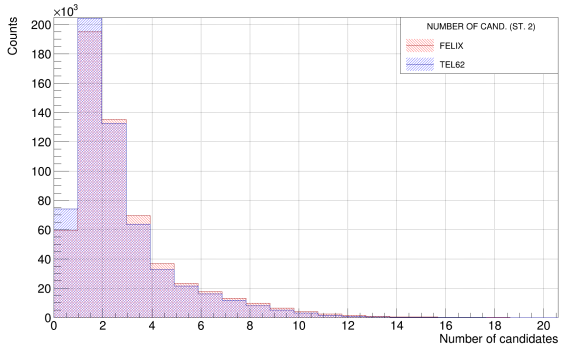

Figure 8.28: Number of candidates for station 2.

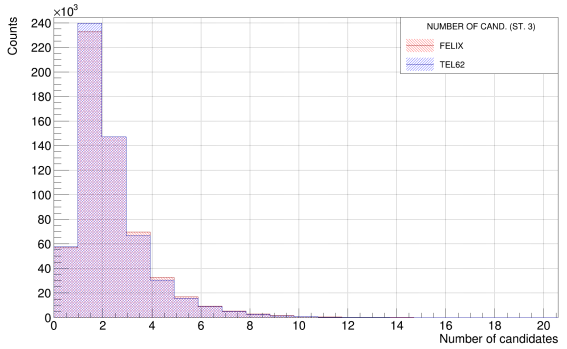

Figure 8.27: Number of candidates for station 3.

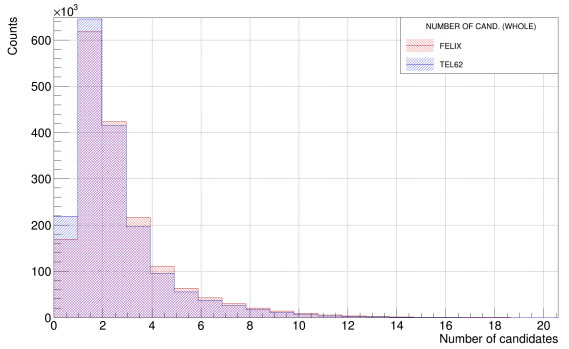

Figure 8.29: Number of candidates for the whole VC.

The difference is great in Station 1 and 2 where we notice an excess of selected events that yielded zero candidates for the TEL62 readout. In station 3 the number of zero candidates events is very similar among the readouts, but a similar situation arises between the events with one candidate and more than one candidate, suggesting the missing candidates that gets digitized by the FELIX readout were coming from missed hits in the TEL62, probably to the intrinsic HPTDC dead time, evident in high event pile-up situations.

8.3.5 The Halo efficiency algorithm

In order to evaluate the size of the dataset needed to estimate a meaningful figure for efficiency the algorithm implementation provides some *summary metrics* on a burst-by-burst basis. These summary metrics are the mean values of expected and detected events for every VC station; from those it is possible to compute the mean instantaneous efficiency of the station and of the whole subdetector.

We start our study with the TEL62 readout. In Figures 8.30-8.32-8.31-8.33 one may observe the trace plot of the instantaneous efficiency.

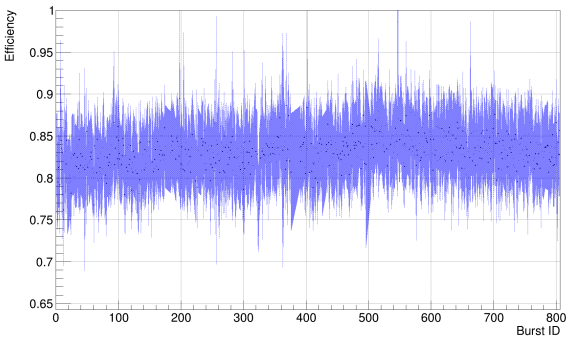

Figure 8.30: Instant efficiency of Station 1 as function of Burst ID. (TEL62 readout)

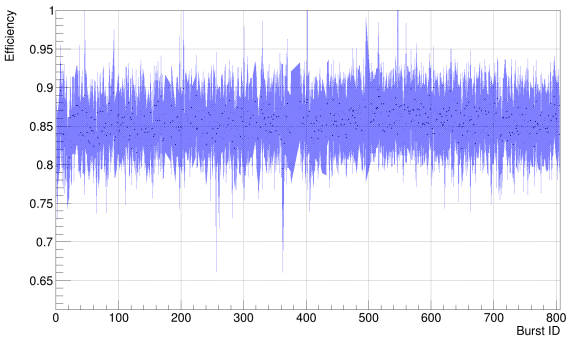

Figure 8.32: Instant efficiency of Station 2 as function of Burst ID. (TEL62 readout)

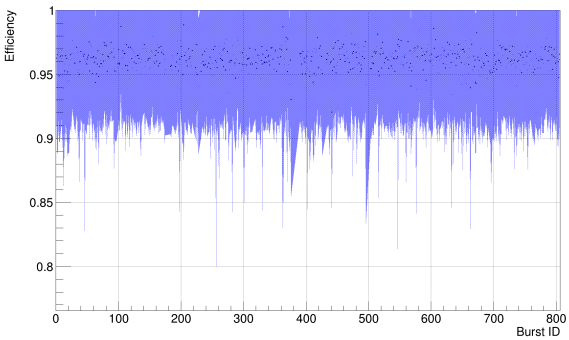

Figure 8.31: Instant efficiency of Station 3 as function of Burst ID. (TEL62 readout)

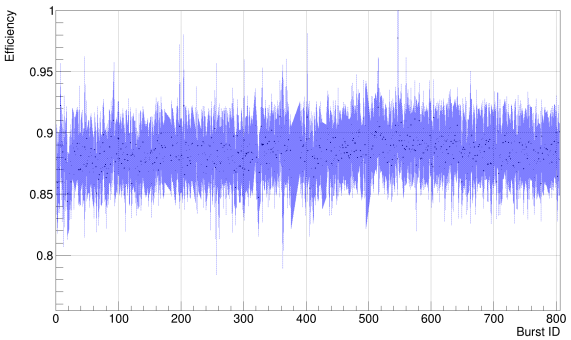

Figure 8.33: Instant efficiency of the whole subdetector as function of Burst ID. (TEL62 readout)

From the previous plots we notice that the mean value of the efficiency in station 1 and 2 is significantly lower than the one of station 3. This is compatible with what was observed in 8.2: station 1-2 are characterized by higher event rate and the TEL readout has been proven to perform poorly in these conditions. Meanwhile the raw hit performance of both readouts is almost identical in the lower rate station 3.

In Figures 8.34-8.36-8.35-8.37 we show the same metrics obtained with the FELIX readout.

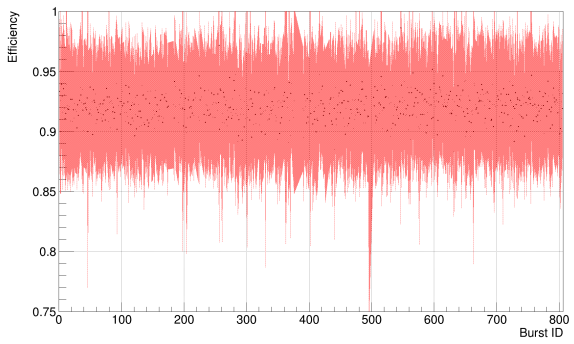

Figure 8.34: Instant efficiency of Station 1 as function of Burst ID. (FELIX readout)

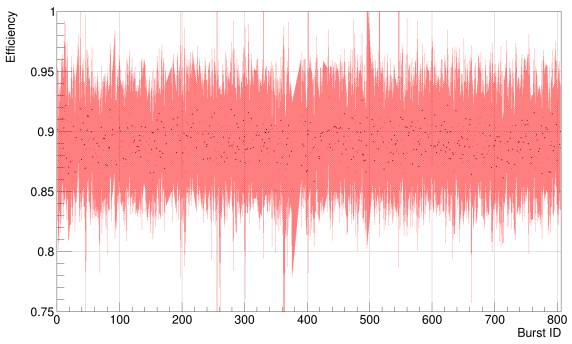

Figure 8.36: Instant efficiency of Station 2 as function of Burst ID. (FELIX readout)

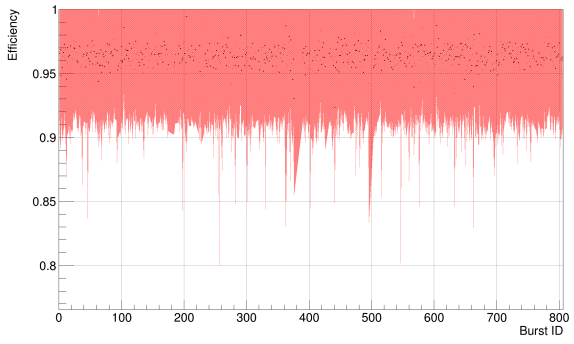

Figure 8.35: Instant efficiency of Station 3 as function of Burst ID. (FELIX readout)

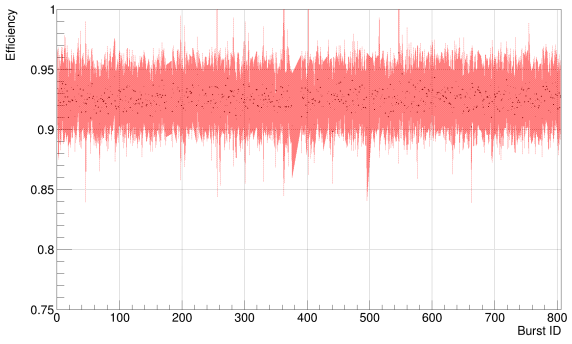

Figure 8.37: Instant efficiency of the whole subdetector as function of Burst ID. (FELIX readout)

While the station 3 efficiency (Figure 8.35) appears very similar to Figure 8.31 the difference between the two readouts is much more visible for station 1-2; The FELIX readout provides a higher overall efficiency in those cases.

In order to better evaluate these quantities we compute the mean and standard deviation of the instantaneous efficiency. To do so the efficiency samples were interpolated with a normal distribution[7].

Station	TEL62	FELIX	% loss
1	0.831 ± 0.019	0.920 ± 0.011	9 ± 2
2	0.855 ± 0.017	0.892 ± 0.014	4 ± 2
3	0.962 ± 0.008	0.964 ± 0.007	0 ± 1
All	0.885 ± 0.011	0.926 ± 0.007	4 ± 1

Table 8.1: Mean Halo efficiency for the Veto Counter and each station

As discussed before the efficiency for station 3 is the same within uncertainty for both readouts, while there is a loss up to $\sim 10\%$ in station 1 (the highest rate one). We also notice that the variance of the efficiency is slightly higher on the TEL62 readout. This point will be discussed further.

[7]We should notice that this approach assumes that the instantaneous efficiency has no dependency with the Burst ID. This hypotesis is, in principle, incorrect. More details on this will be given in Section 8.3.7.

8.3.6 Cumulative efficiency and dataset size

From the instantaneous efficiency plots the integrated efficiency was computed. For the $i-$th burst it is the ratio of observed and expected events over all previous bursts. This metric was used to select the number of bursts to be analyzed: a run is nominally made up of 1500 bursts and analyzing all of them is not necessary to have a good efficiency estimation.

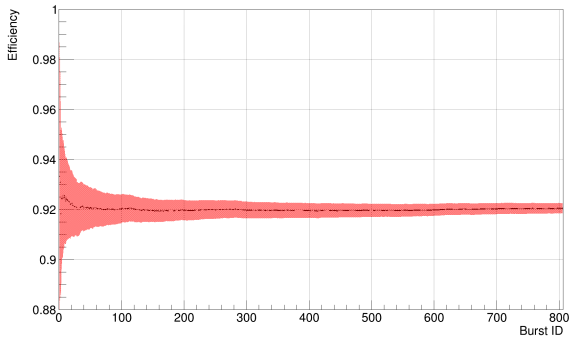

Figure 8.38: Integrated efficiency of Station 1 as function of Burst ID. (FELIX readout)

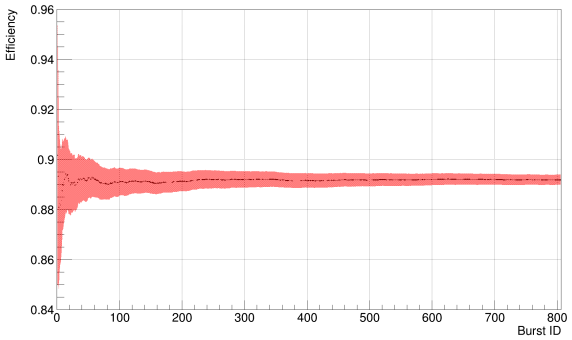

Figure 8.40: Integrated efficiency of Station 2 as function of Burst ID. (FELIX readout)

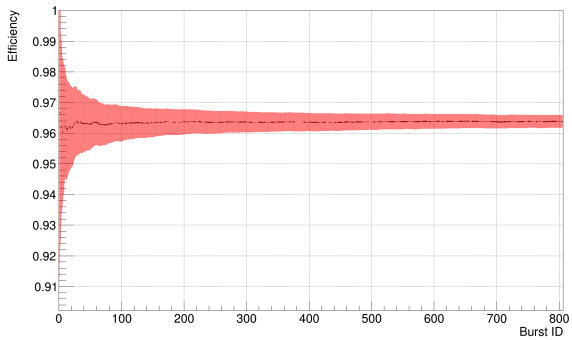

Figure 8.39: Integrated efficiency of Station 3 as function of Burst ID. (FELIX readout)

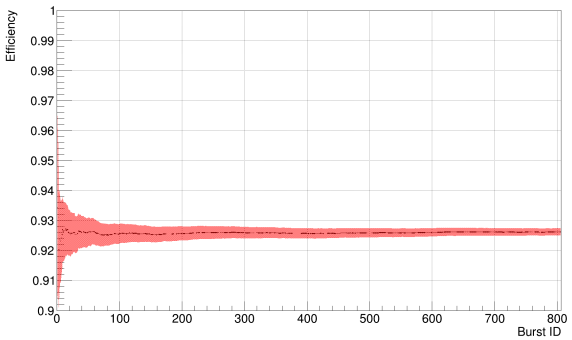

Figure 8.41: Integrated efficiency of the whole subdetector as function of Burst ID. (FELIX readout)

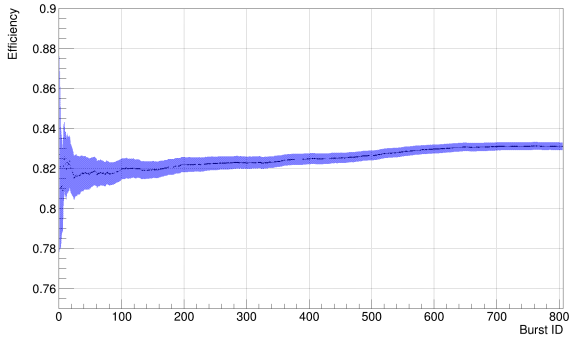

Figure 8.42: Integrated efficiency of Station 1 as function of Burst ID. (TEL62 readout)

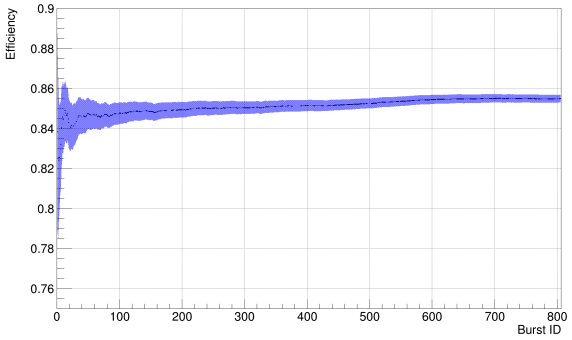

Figure 8.43: Integrated efficiency of Station 2 as function of Burst ID. (TEL62 readout)

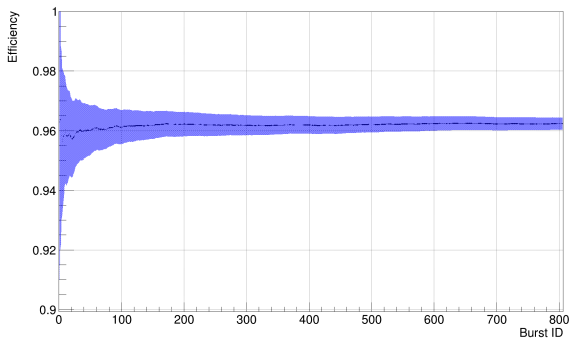

Figure 8.44: Integrated efficiency of Station 3 as function of Burst ID. (TEL62 readout)

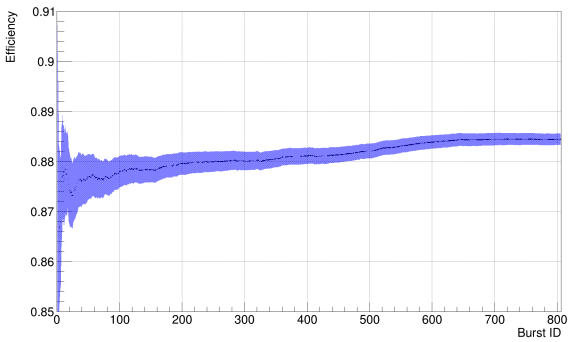

Figure 8.45: Integrated efficiency of the whole subdetector as function of Burst ID. (TEL62 readout)

These plots shows an interesting behaviour for the TEL62 readout on stations 1-2, this may be index of an ignored dependency of efficiency with Burst ID (Section 8.3.7). On Station 3 the trend is stable and very similar for both readouts. From these plots it is also clear that the number of bursts selected for the analysis (~ 800) is sufficient to get a good estimation of the efficiency.

8.3.7 Effect of beam intensity on efficiency

The evaluation of the efficiency in Table 8.1 is based on the assumption that the instantaneous efficiency follows a normal distribution. In other words, the efficiency has no dependency with the burst it is computed on; all bursts are the same.

While this assumption is useful to calculate the mean value shown before it is not strictly true: every burst has in principle different proprieties, whose distributions are unknown and complicated; moreover the readout and acquisition system is not perfect and can contribute to event loss. The most obvious parameter to consider is the beam intensity: this fluctuates during the data taking depending on the SPS machine operation. An higher intensity burst will produce a higher rate in the detector and thus the final efficiency of it may be compromised.

NA62 is equipped with the *argonion scaler*. This small, single channel scintillator is located in the beam dump and provides an unbiased measure[8] of the experimental beam intensity.

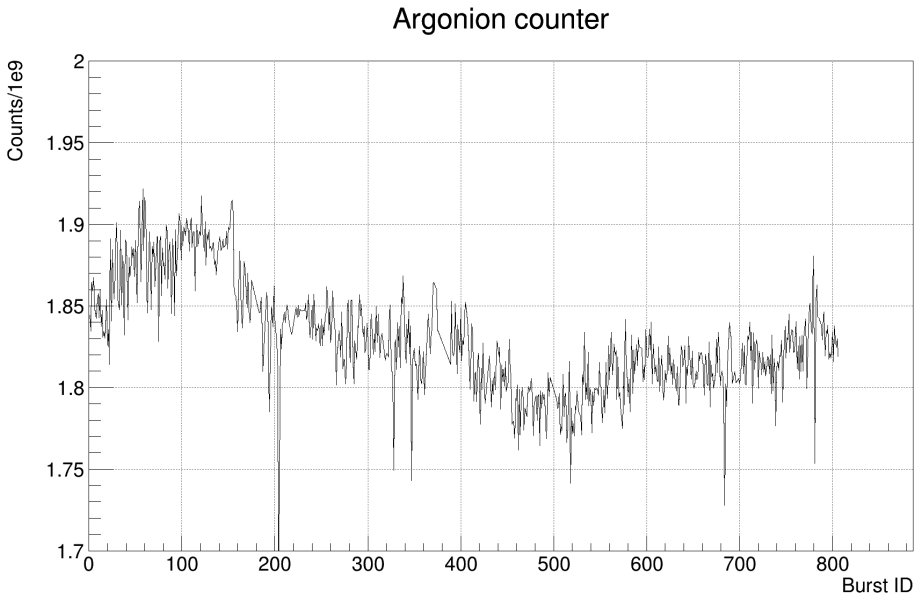

Figure 8.46: The Argonion counter rate as a function of the Burst ID for the analyzed subset of Run 12437

[8]It is not processed within the reconstruction and analysis framework.

During the analyzed run the beam intensity shows a degradation from Burst 100 to 500, followed by an increase afterwards. Since we already noticed that the TEL62 readout is extremely sensitive to increase in data rate lets evaluate if these intensity fluctuations pose an impact on the efficiency. Lets consider station 1 on both readouts. In order to emphasize any dependency effect we perform a running average on the instantaneous intensity with $N = 25$. This acts as a low pass filter which cuts the high frequency fluctuations and makes slower trends easier to observe.

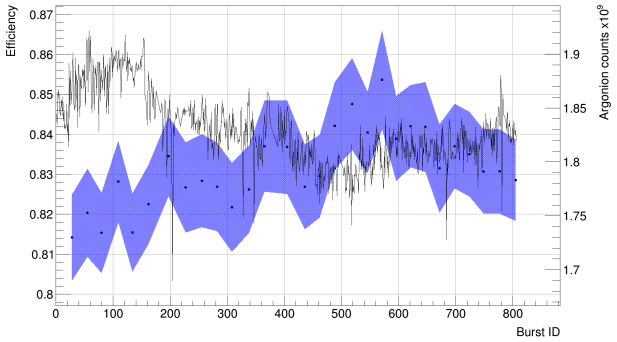

Figure 8.47: Rolling average efficiency of Station 1 as function of Burst ID (TEL62) superimposed with the Argonion counter data.

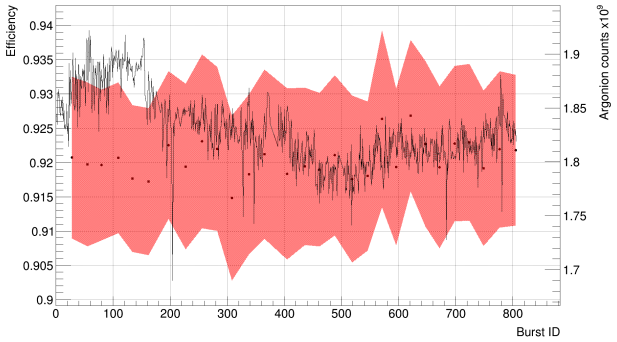

Figure 8.48: Instant efficiency of Station 1 as function of Burst ID (FELIX) superimposed with the Argonion counter data.

While the FELIX readout efficiency shows no appreciable variation with the beam intensity the TEL62 one performs the worst when the beam intensity is the maximum. Notice that the small time offset between the Argonion counter and the averaged efficiency is a feature of the FIR filter that is the rolling average.

We show the same plot for Station 3. Notice how the trend is extremely similar among the two readouts.

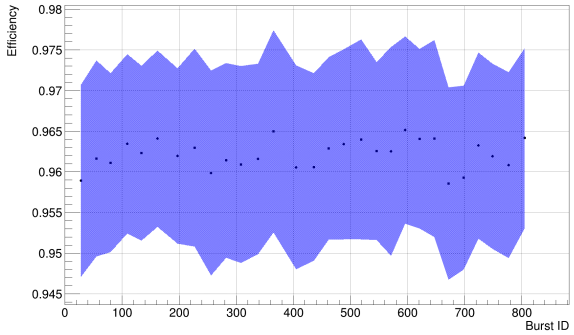

Figure 8.49: Rolling average efficiency of Station 3 as function of Burst ID (TEL62) superimposed with the Argonion counter data.

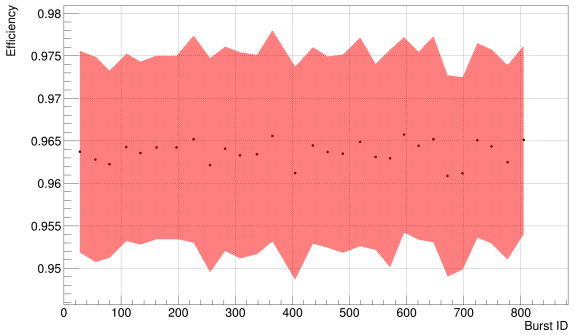

Figure 8.50: Instant efficiency of Station 3 as function of Burst ID (FELIX) superimposed with the Argonion counter data.

Lets now observe the efficiency results in more detail. Since the Halo efficiency algorithm is based on the extrapolated particle tracks it is possible to construct a map of all the expected tracks across the various detector planes. In Figure 8.51 we show the cross section of all extrapolated tracks at station 1.

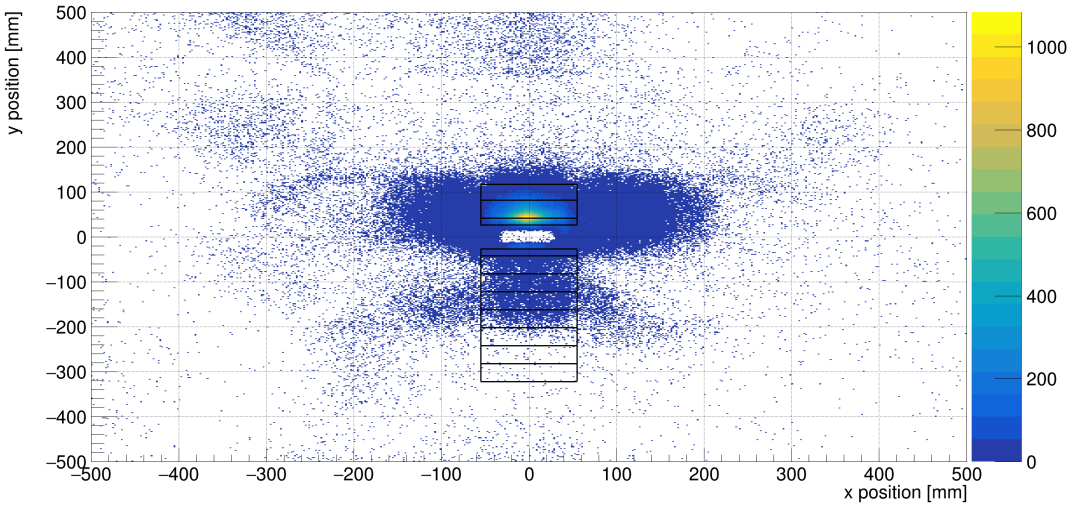

Figure 8.51: Extrapolation of all the downstream single track events from STRAW at the Veto Counter station 1 plane. The border of the various tiles is superimposed (topmost one is Tile 0).

Obviously only a fraction the tracks that traverse the tiles get detected and associated to a suitable Veto Counter candidate. As an example we show in Figures 8.52-8.53 the efficiency of station 1 on a track basis for both readouts.

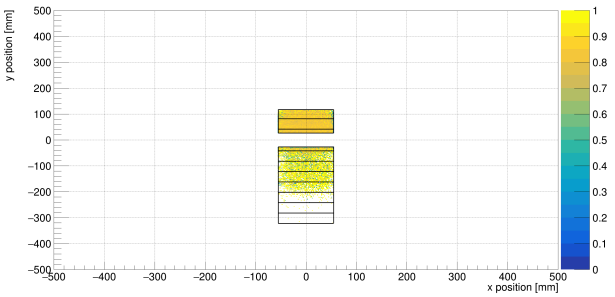

Figure 8.52: Efficiency for extrapolated tracks at the station 1 plane. (TEL62)

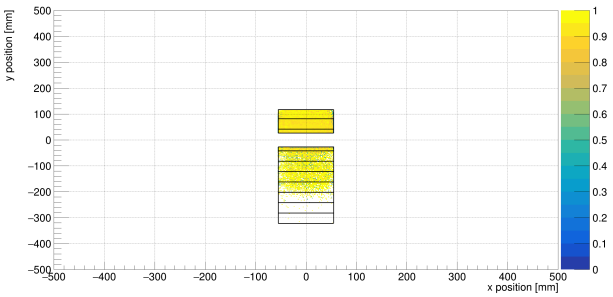

Figure 8.53: Efficiency for extrapolated tracks at the station 1 plane. (FELIX)

Notice how the high number tiles have very low statistics. This is expected and related to both the experimental geometry as well as the dataset choice: the tiles further away from the beamline would have less activity due to the fact that they are far away from the interaction point of the relevant events (product of secondary interactions between the beam and the last GTK station). Moreover their angular position and the beamline structure means that the particles traversing them would have to travel a significant amount of material and air before reaching the STRAW; since the algorithm only performs a naive linear extrapolation from outside the STRAW to the Veto Counter plane it disregards all the scattering effects. There is also the possibility that the particle, even producing a candidate in the Veto Counter will never reach the spectrometer or reach it through a trajectory that is way different than the suggested extrapolated one. This is especially true for station 2 that has the TCX collimator directly downstream of it and even more for station 1 where on top of the TCX there is also station 2 and the lead conversion plate. Tile 9-10 are especially problematic given their line of sight to the STRAW is directly obstructed by some mechanical support structures of the beamline.

Another reason why there are few high angle single track events[9] in the analyzed data is due to them being PNN filtered. The filtering operation applies directly some cuts on the events and the STRAW tracks with such kinematics are already partly cut at the origin and do not show up.

[9] and thus, few extrapolated tracks in high order tiles.

Following a similar format used in the raw hit analysis we show the efficiency for every tile in Figures 8.55-8.54. In these figures we omitted tiles 9-10 for the aforementioned reason, the apparent loss of efficiency is already visible at tile 8. Notice that those tiles were anyway considered in the mean efficiency calculation of Table 8.1.

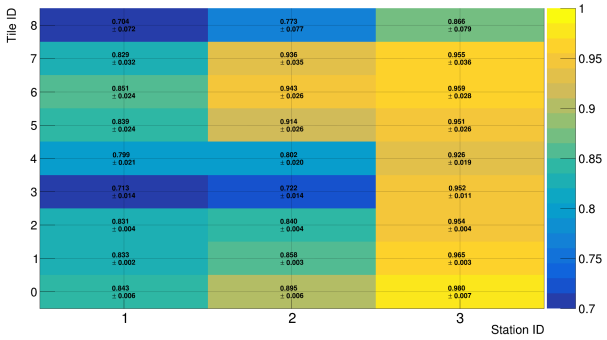

Figure 8.54: Halo efficiency for the TEL62 readout.

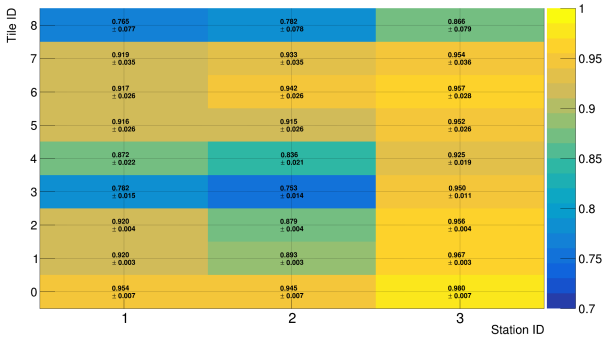

Figure 8.55: Halo efficiency for the FELIX readout.

Lets focus on station 3; in this case the efficiency is very similar between the two readouts; in station 1 and 2 the difference is larger, very visible in station 1. Again, this is due to the event rate across the stations. Tile 3-4 lower apparent efficiency may be due to an issue with the front end since it shows up on both readouts.

It is also interesting to evaluate the ratio between the two readout efficiencies. Here the performance difference is more visible since it ignores any source of inefficiency that is common among the readouts (front end, scintillator...).

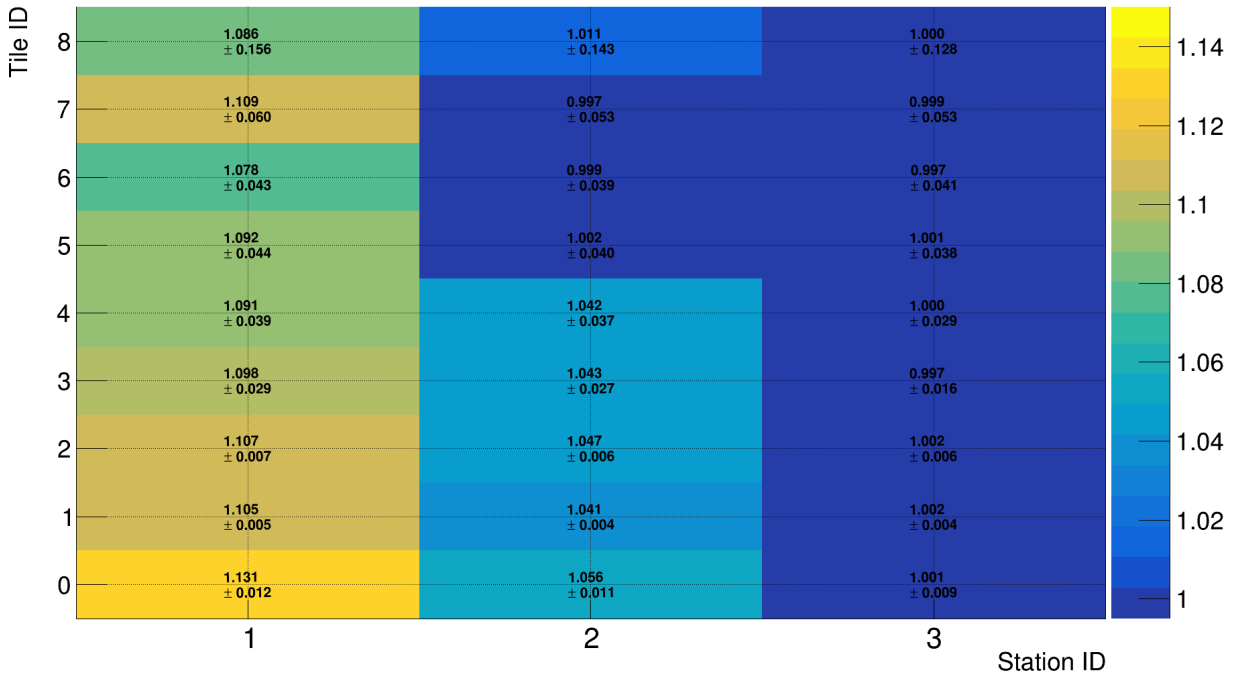

Figure 8.56: Ratio of Halo efficiencies between the FELIX and the TEL62 readout.

Notice how the high number tiles exhibit very low statistics. A weighted average of the efficiency ratios has thus been computed.

Station	Ratio FELIX/TEL62
1	1.108 ± 0.004
2	1.043 ± 0.003
3	1.002 ± 0.003

Table 8.2: Mean value of the halo efficiency ratio between FELIX and TEL62.

8.3.8 Self efficiency algorithm

The Halo efficiency algorithm is not ideal to evaluate station 1 and 2, so in those cases the self efficiency algorithms is preferred. Notice also that the misestimation of the efficiency for high numbered tiles it is not an issue in this case; since the positional information does not come from an extrapolated track but directly from Veto Counter station 3.

In Figures 8.57-8.58-8.59-8.60 we show the same plots of Figures 8.55-8.54, for the self efficiency. Since this algorithm operates with non tight candidates it is possible to evaluate the efficiency of the Jura and Saleve side readout separately.

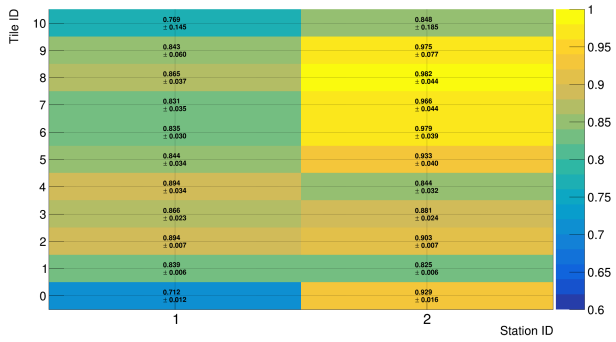

Figure 8.57: Self efficiency of TEL62 readout on Jura side.

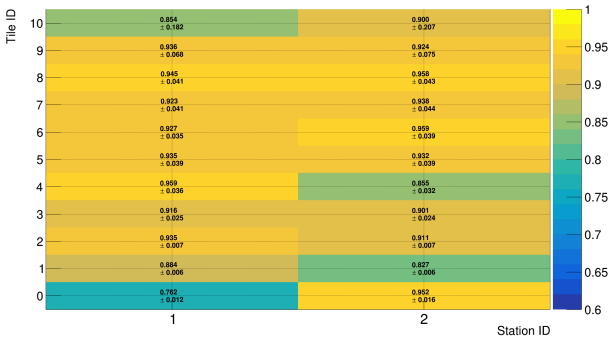

Figure 8.58: Self efficiency of FELIX readout on Jura side.

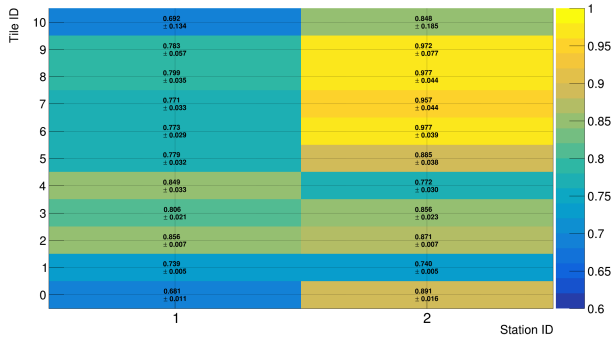

Figure 8.59: Self efficiency of TEL62 readout on Saleve side.

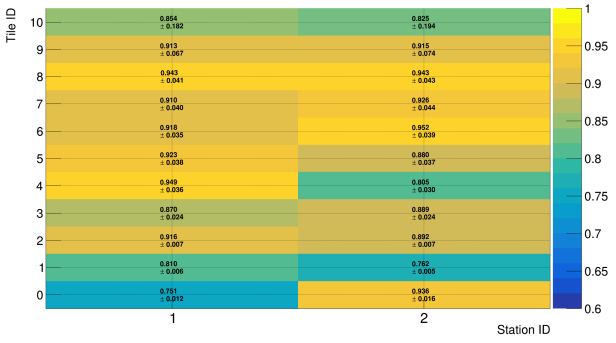

Figure 8.60: Self efficiency of FELIX readout on Saleve side.

We point out that, given the different selection and logic the results of the two algorithms cannot be directly compared. In the self efficiency algorithm there is the requirement of exactly one *tight* candidate in station 3, and the probe station candidate needs to be in extremely tight time coincidence with the reference station. Another aspect to consider when observing the self efficiency result is the lower value for the extreme tiles (0 and 10), especially in station 1; this may be due to a limitation in the algorithm itself: since it relies on station 3 as a reference it is possible that events that were indeed registered in station 1-2 for those tiles are not even cosidered, due to the tracks passing just outside the upper or lower boundary of station 3.

Lets now compute the mean self efficiency for both sides of the subdetector for each readout system.

Station	TEL62		FELIX	
	Jura	Saleve	Jura	Saleve
1	0.885 ± 0.004	0.770 ± 0.004	0.885 ± 0.004	0.848 ± 0.004
2	0.868 ± 0.004	0.798 ± 0.004	0.873 ± 0.004	0.820 ± 0.004

Table 8.3: Mean self efficiency of Station 1 and 2 Jura and Saleve side for both readouts.

Conclusions

In this work the performance of the new digital readout system at NA62 has been benchmarked and compared to the legacy TEL62-based one. The study underlined the superior characteristic of the novel system together with the limitations of the existing one. Everything has been evaluated on the Veto Counter sub-detector developing platform, which proved to be a reliable and useful workbench to test and develop the new hardware and software. The tests showed a better raw hit detection performance of the FELIX readout in all examined data taking conditions; in particular at nominal and beyond beam intensity: the TEL62 system exhibited up to 25% raw hit loss at nominal intensity. The detector efficiency for reconstructed data also shows a marginal improvement ($\sim 10\%$ at best) in high rate conditions, compatible with the findings for the raw hit performance. The possibility of getting positional information from the Veto Counter has been tested with positive results (See Appendix C). A proper calibration may be needed to use this metric officially.

Future developments and further testing may be performed on other sub-detectors in NA62 such as the KTAG and the CHANTI, where new challenges and issues can be addressed and solved; for example the integration with the existing L0 trigger system and higher rate testing. In the foreseeable future the whole L0 infrastructure may be scrapped in favor of a new software-only triggering scheme, for which the FELIX readout is perfectly suited.

The improved performance of the FELIX readout will set a new standard for new runs of NA62 and other future K-beam experiments; becoming a robust and highly dependable solution for modern triggering infrastructures and high rate experimental conditions. Moreover, the absence of obsolete and hard to source components and the extensive use of commodity computing hardware will significantly lower the developing effort and the cost for upcoming experiments.

Part of this research involved the development, integration and deployment of a novel control system for the FELIX custom TDC card. This piece of software has been thoroughly tested and performed as requested in all conditions. As of today (2023) it is integrated in the NA62 software infrastructure and is currently used within the Experiment Run control. Moreover, an upgrade of the online monitoring dataflow has been proposed, developed and deployed as part of this work. This last activity enabled a faster (up to 100%) raw data transfer to the online monitor cluster and greatly improved the promptness of the NA62 monitoring systems.

Appendix A

NA62 event format and storage system

A.1 NA62 serialized event format

The L2 events sent out by the farm cluster aggregates all the information concerning a timestamp for a given burst ID. In this section we will provide a detailed description of the data format, this is important for defining the selection rule for the online monitor (4.6.2).

An event can be seen as a self contained and independent data unit in the context of a run. The structure of an event is given in Fig. A.1 [6].

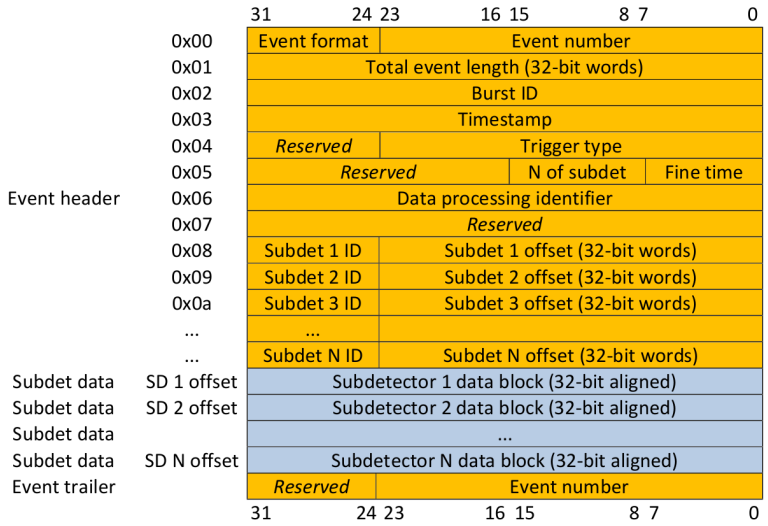

Figure A.1: Serialized event data structure. [6]

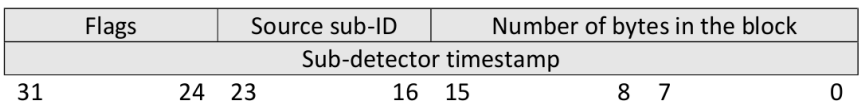

Figure A.2: Subdetector event header. [6]

Lets break down its main fields:

Event format: field used to identify format revision, fixed at `0x62`.
Event number: unique number associated to the event in the burst.
Event length: total number of words composing the event, including the header.
Burst ID: unique burst ID of the event.
Timestamp, fine time: timing information for the event.

Trigger type: a 24 bit word containing L0, L1, L2 trigger words concatenated. Structure of L0 word was shown in Fig. 4.2, L1, L2 are constructed similarly in the farm node.
Number of sub-detectors: number of subdetector data blocks in the event; the format supports 0-255.
Data processing identifier: number indicating level of processing applied to the event; zero represents an event of raw detector readings, higher means some sort of processing has been applied in event building.
Subdetector X ID: numeric ID of subdetector contributing to the event.
Subdetector X offset: Byte offset from event heading bit `0x00` to the given subdetector X data block.

The Subdetector Data Blocks (SDBs) contains the subdetector data and their content and length depends on their source. Every SDB has an uniformed *event header* that is added by the DAQ farm software: it contains size information and subdetector timestamp as provided by L0TP, the latter is used as a checksum test.

A.2 Data storage system at NA62

Lets briefly describe how data gets moved from the merger cluster to the permanent tape storage in the Cern Tape Archive (CTA). The storage system keeps tracks of the data files and initiates all the transfers. Files are first moved from the merger cluster to the Cern EOS where they stay for around 7 days; during this time preliminary reconstruction and analysis may be performed. From EOS files are then moved to CTA for permanent storage.

The main components of the data storage system are:

- A book-keeping system to keep track of files transfer status;
- A gridftp daemon service [48];
- The FTS proxy virtual machine [49];
- The Data Transfer Orchestrator (DTO) software.

The book-keeping is done using a MariaDB [50] database located on Merger 4; there information on file count and transfer state are kept and regularly update. A backup database synched to the main one is kept on Merger 2 and may take over should the master database have issues.

Transfer operations are carried out by the *globus-gridftp-service*, running on the mergers. This service continuously gets information from the FTS proxy and copies files over the network to EOS. Database update and information flow to FTS, as well as submission scheduling are managed by the DTO. This software runs on all mergers and manages the transfers: when a new file is created on one merger DTO submits an entry in the database for a pending transfer and instructs FTS to initiate a transfer; this triggers the gridftp service on the relative merger machine and the file is copied on EOS. Successful transfer will be checked by the DTO that will update the database accordingly; if the transfer fails it is retried three times before giving up. All successfully transferred raw files are deleted from the merger cluster by DTO after 24 hours.

Should all file transfers to EOS stop the merger cluster has sufficient storage for at least 3-4 days of data taking, if the fault is not repaired in this time DTO will start to delete the oldest files on the cluster to acquire new ones.

Appendix B

The FELIX software in detail

B.1 ATLAS FELIX software overview

The ATLAS software code base for the FELIX card is composed by a PCIe driver, a memory allocation driver, a basic library `FlxCard` that is used as an API between the driver and all high level applications, the main FELIX readout application `FELIXcore` and a set of applications for configuration and control `ftools`.

Due to the strict performance requirements of the front end the PCIe driver is extremely minimal and only provides virtual mapping to the FELIX card registers in user space. This topology allows to run the entire low level code in user mode; avoiding expensive context switches and easing debugging and development. Incoming data from the front end are stored in the server memory using RDMA; the main memory structure being a circular contiguous memory buffer, in the the order of $\sim$ 8GB per half card. Allocation of this memory buffer is performed by the memory allocation driver, named *cmem_rcc*.

Both `FELIXcore` and `ftools` rely on a set of low level libraries to perform their operations the most important being `FlxCard`. The design of FELIX core will not be discussed since it is not suitable for NA62 applications.

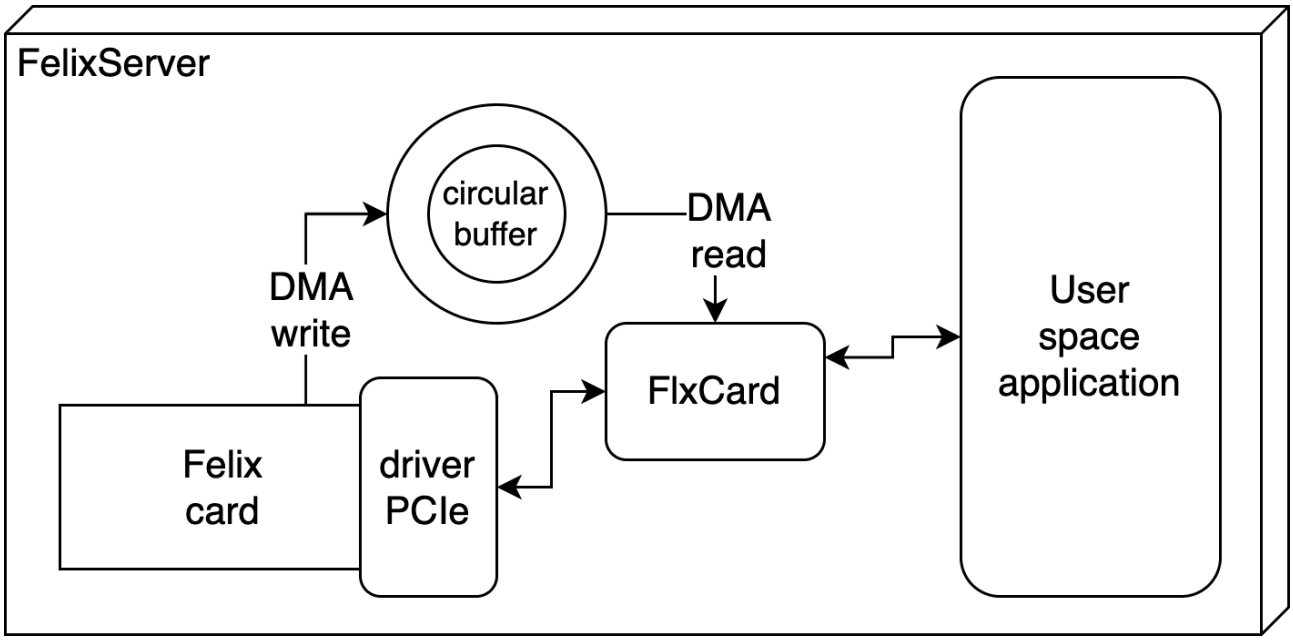

Figure B.1: Basic schematic of the software topology of the FELIX server.

B.1.1 FELIX data format

Data comes to a given elink in the form of arbitrary sized *chunks*. These chunks are decoded and subdivided into *blocks*; a block being 1kB in length and including a 4B header, a trailer and part of the chunk data (*sub-chunk*). If the chunk is larger than a block it is splitted in multiple parts, each having its own trailer and header. Header contains Elink-ID, sequence number and a *start of packet word* while trailer contains information data size and type.

This is the same format that will be used to write the data in the circular buffer. The FPGA firmware includes a *write* pointer to iterate on the DMA buffer, while the FELIX library includes a *read* pointer.

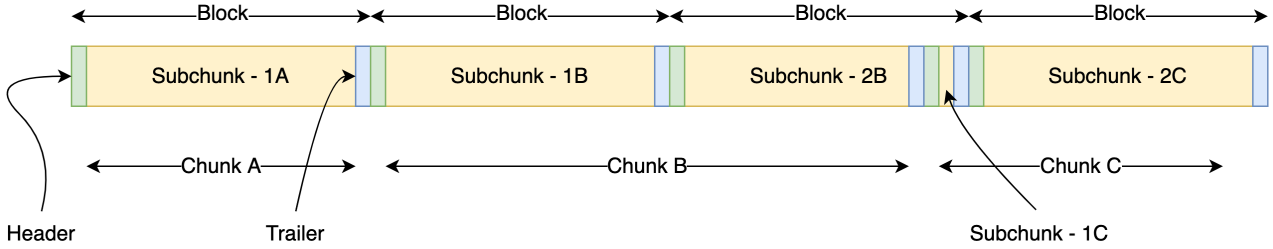

Figure B.2: Example of chunk splitting performed in the firmware; chunk A fits in one block while chunk B,C require multiple blocks.

B.2 FELIX TDC register mapping and format

The TDC control operations performed by the Felix Control system (Chapter 6) are performed via atomic writes in FPGA memory locations. Read or write operations are carried out on a *register* basis; each register represents a 32 bit wide area of the memory and is 63 bit long. A list of the most relevant registers is given in Table B.1.

Register	Meaning	Register	Meaning
0x0	Testing register	0x100	FW version (R)
0x10	CSR, Control	0x110	TDC ID (R)
0x20	Channel mask	0x120	Status (R)
0x30	T0 offset group 0	0x130	Read FIFO
0x40	T0 offset group 1	0x140	Write FIFO
0x50	T0 offset group 2	0x150	Empty flag test FIFO
0x60	T0 offset group 3	0x1000 - 0x103F	I^2C link 0
0x70	T0 offset group 4	0x2000 - 0x203F	I^2C link 1
0x80	T0 offset group 5	0x3000 - 0x30FF	Flash CSR
0x90	T0 offset group 6	0x4000000 - 0x7FFFFFF	Flash memory
0xA0	T0 offset group 7	—	—

Table B.1: Register mapping for the NA62 TDC board; all register addresses are written omitting all leading zeros.

The *testing register* is a free-use register used for testing the control system[1]; *control* register is used to set up various TDC parameters, each having some specific bits in it. Channel state (active or inactive) may be toggled by setting the corresponding bit (0 to 31) in the *channel mask*. *T0 offset* registers may be used to set the timing offset for T0 triggers. All the higher registers are either read only registers for metadata such as FW version and TDC unique ID or pointers to internal TDC data structures such as the R/W queues or the firmware flash. The latter is fully writable through the FELIX card enabling remote TDC firmware flashing from the FELIX server.

[1] not readback by the FPGA.

This register map is memorized in the control client, and used to convert the high level functions (such as "disable channel X, set T0 offset Y") to bitstreams ready to be DMAd.

B.3 Adaptation of FELIX software for NA62

In this section we will describe in detail the NA62 Felix software environment.

The `FELIXcore` library provides a basic API for ultimately managing the physical FELIX card. It is the ideal starting point for building a FELIX wrapper application to be used in the NA62 experiment DAQ.

The name "Wrapper" given to the software emphasizes the fact this application will encapsulate the basic FlxCard I/O interface and provide higher level functionality adapted to the experiment needs. One wrapper instance can control a whole FELIX card, splitting that in two logical "half-cards". Theoretically it is possible to run multiple instances of the wrapper to control multiple, independent physical cards installed on the same server[2] provided that there is no resource clashing.

An NA62 FELIX server will include the basic PCIe driver *as is*, an application for managing the RDMA memory structure (interfacing with the cmem_rcc driver) and the wrapper software. This small application will allocate a number of buffers determined by its parameters that will be accessible in user space via the OS file system.

High level wrapper structure is achieved with multiple modules and auxiliary objects, that we will introduce before discussing the main software. The E-link model, Card wrapper, Block parser and Block router are general pieces of software for managing the FELIX card and have been developed as part of the readout effort for the DUNE experiment. The trigger dispatcher, event builder and the EOB handler are NA62 specific code.

B.3.1 E-links

Elinks are the virtual link units for the FELIX card. Each half card will have a number of active elinks dedicated to different functions (TTC, Data to host,...). While data incoming (FE-H) will use the FULL mode, control messages and TTC will be forwarded in standard GBT mode.

E-link Concept

A class named `ElinkConcept` contains function templates for high level functions that an elink may implement, such as `start, stop, initialize, configure` as well as a collection of type variables that represent significant quantities for the elink. As the name suggests this is a logical *concept* for an elink, without any implementation: it is a *template* for an elink.

A very important object for an elink is its *parser*, For now we just point out that a parser object is part of the elink concept.

E-link Model

The realization of the concept happens in the `ElinkModel` class. Here the high level functions are fully implemented. The elink model defines also a statistics engine, to be used for monitoring purposes when representing data to host elinks. The statistic engine, similarly to the parser, is provided a function to compute the statistic metrics that will be run as a thread. Statistics involve number of processed blocks, error counters and timing information.

Moreover the elink parser, whose existence is imposed by the elink concept, gets a processing thread to run on the incoming data. This is realized by means of a function that will be run in a dedicated *process* thread. In Fig. B.3 we show a complete view of the elink hierarchy.

[2] this can for example serve different subdeterctors.

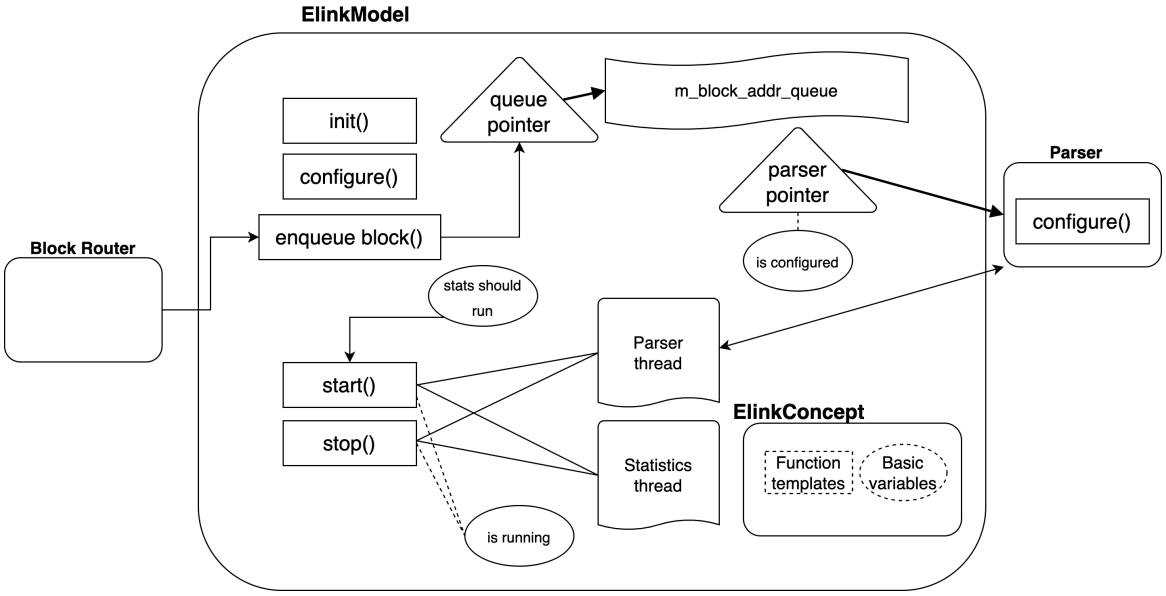

Figure B.3: The elink abstraction hierarchy. Notice the interaction with the associated block router and elink parser.

B.3.2 The BlockRouter

This is a structure which provides access to the elinks for a given half card. A block Router instance contains two `std::map` types. One, named `elinks` associates elinks (represented by their number) to their `ElinkModel`. Technically it is an integer–keyed map of `std::unique_pointer`. The other map is called `elink_block_counter` and relates each elink (again, its number) to a size type; this is the length of the block queues: as discussed in the Section B.3.1 an elink (model) contains a queue of block addresses.

Internally the Block Router uses a lambda function `count_block_address` which accepts a block address as argument and after some sanity on the associated elink, done via the routing maps, enqueues in the block queue. This is done via a specific function defined in `ElinkModel`.

B.3.3 The Card Wrapper

The card wrapper is a wrapper for a half card and it uses the lower level *FlxCard API* directly.

Similarly to what is done for elinks a `CardWrapper` is a subclass of a more abstract `Wrapper` class that acts as a template[3].

Wrapper

Wrapper is the template superclass CardWrapper inherits from. Exactly as elink concept it contains templates for high level type functions and some defining variables for a general FELIX half card.

CardWrapper

This is the Wrapper subclass. It contains more constant values specific for the FELIX card and its intended use together with further function definitions and implementations regarding DMA transfers. Among the objects and variables defined in the CardWrapper there is a pointer to the low level object `FlxCard`, this will be instantiated when the `init()` function is called for the wrapper. It is defined in the subclass because ideally one would want to keep the wrapper concept independent from the specific low level API used in the implementation.

[3]Elink Concept and Elink Model.

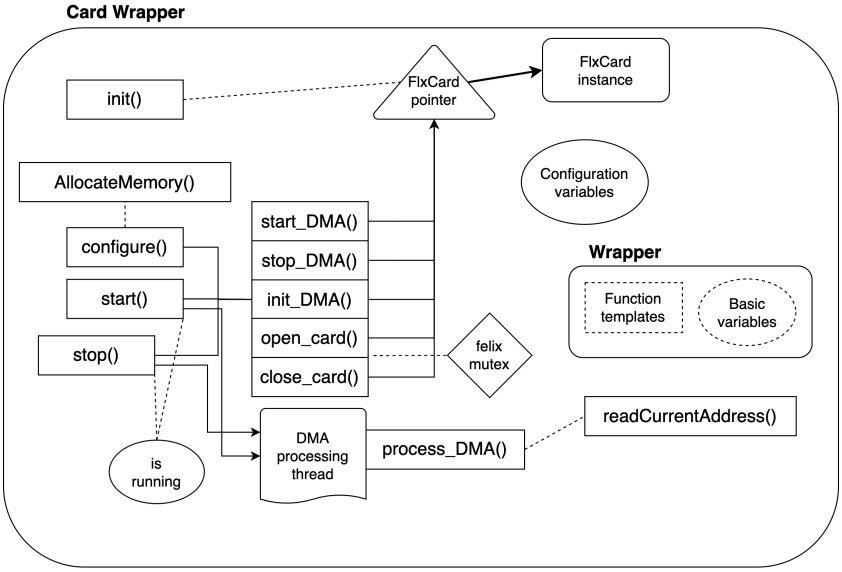

Figure B.4: The card wrapper abstraction hierarchy.

B.3.4 Block Parser

A block parser is an entity that processes blocks coming to and from an e-link. The concept of block parser comes from the low level FELIX code base, since the actual block definitions are determined by the FELIX firmware implementation. The content and type of processing to apply to the blocks depends on the subdetector served by the readout card. For this reason the FELIX block parser provides a way to set it up using a template class providing the implementation.

In NA62 elinks on the card may be used for control (H-FE), to carry TTC information or for data transfers (FE-H). It is clear that data parsing operations will differ between these types. The implementation provided lets the user write their own parsing functions, giving only their signatures. Actual function implementations may be provided as named *lambda functions* in the main wrapper code and bound to the specific wrapper instance for each elink.

B.3.5 The EOB handler

Being a fixed target experiment the software at NA62 needs to have burst status information (namely EOB and SOB signals) available in order to synchronize itself. These information are provided via the DIM interface and a dedicated EOB handler manages to collect the event and provides `boost::signal` entities for the main software to bound to.

Operation of this component is similar to the one of the SyncroMerger described in Section 4.5.5: The EOB handler is a subclass of the DimInfo class; EOB handler subscribes to EOB information services and overrides the `InfoHandler` function with a custom one that forwards the signals and sets some internal flags and metrics such as timestamps. A delayed EOB signal is also provided, this is used for the parser cleanup between bursts.

B.3.6 Trigger Dispatcher

As said in 5.2 a FELIX based system may replace both L0 and L1 trigger sub-detectors. If an L0 detector front end is replaced the FELIX server still needs to provide compatible L0-style trigger primitives to the L0TP; meanwhile if an L1 detector is replaced the server has to listen for L1 trigger decision in the form of MRPs and forward its data to the farm.

These functions are performed in the software by the Trigger Dispatcher. Dispatcher is a superclass for L0dispatcher and L1Dispatcher; it provides a basic template for a Dispatcher concept in the form of template functions and a trigger list data structure. The L0 dispatcher is very simple and only

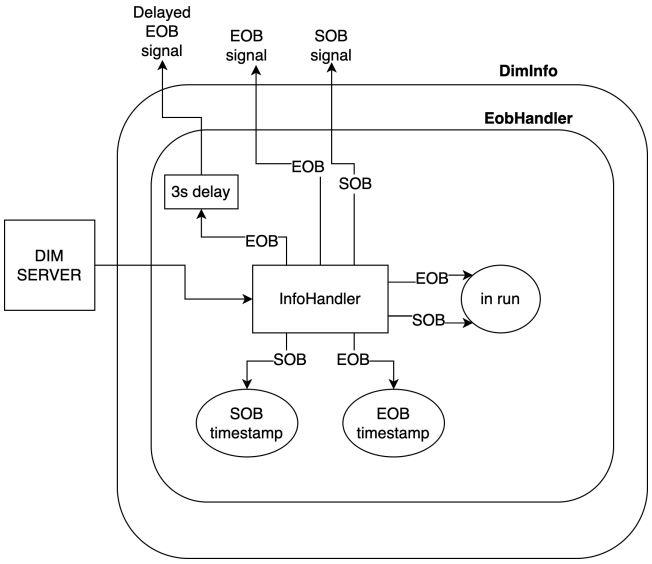

Figure B.5: The EOB handler module.

implements a `sink()` function to write in the trigger list if a positive L0 decision is issued from the L0TP. Meanwhile, an instance of L1 dispatcher has to fetch the incoming MRPs and send the appropriate SDEs to the acquisition farm, interacting with the trigger list structure. These operations are performed by a dedicated thread within the L1 dispatcher and thread state is controlled by an atomic boolean flag.

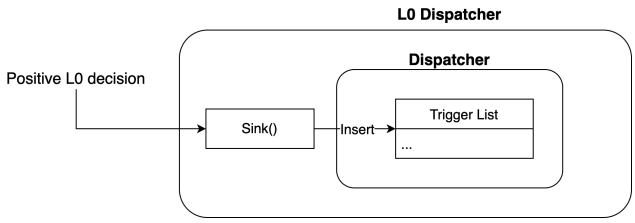

Figure B.6: Schematic of the L0 trigger dispatcher, notice that only the used features of the Dispatcher superclass are shown.

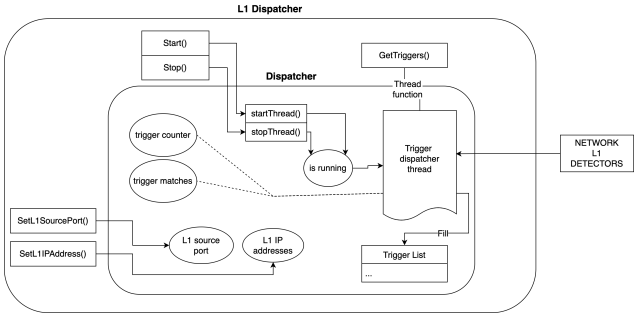

Figure B.7: Schematic of the L1 trigger dispatcher, notice how the trigger list table structure is the same as the L0 dispatcher but instead of directly filling it based on L0 decision there is a whole thread that fills it based on the incoming MRPs.

B.3.7 The Event Builder

The event builder is the software responsible of assembling the subdetector data fragments in actual events, following the event data format specifications for NA62. Essentially an event builder has a dedicated event building thread, a data structure to store the incoming fragments from the front end and a trigger list to store trigger information to pack with events.

Following the same idea of the dispatcher a basic Event Builder template is used as superclass for the specific implementations for L0 and L1 event types. The template contains most of the functionality of the block, including pointers to the relevant data structures for triggers and event fragment storage. L0 and L1 builders differ in the event building thread operation, that is implemented differently in each of them through the `build()` function.

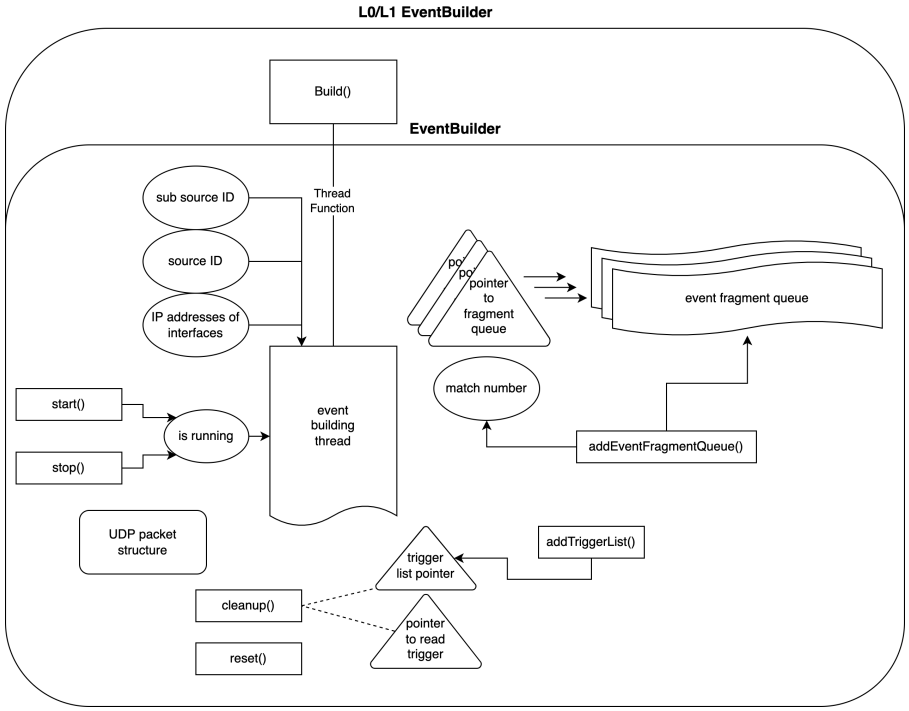

Figure B.8: Schematic of the event builder module.

B.3.8 Main Wrapper

Now that all relevant concepts have been introduced we shall outline the operation of the main wrapper code.

The wrapper software relies on the necessary memory buffers for the elinks to be preallocated by the auxiliary `cmem_rcc` driver and be available within the file system. After verifying that it is the case the main code may execute:

1. An instance of EOB handler is created, now software has access to EOB and Delayed EOB signals.
2. Options from configuration files are read, these include number of half cards to be used, whether the subdetector is L0 or L1 and the elink list for each function (TTC, Data).
3. Based on operating mode inferred from options *trigger dispatcher* and *event builder* are instantiated.
4. EOB and Delayed EOB signals are connected respectively to the reset and cleanup functions of the event builder.
5. Trigger list is created and passed to both trigger dispatcher and event builder as a pointer.
6. Map structures for card wrappers and block routers are created. These maps link each half card to its newly instantiated block router and each elink to a block router structure. Here elinks are represented with an unique integer index.
7. Low level, card specific configuration parameters are read from a *json* string.
8. Using the card wrapper map the requested half card(s) are initialized with their configuration options.
9. Elinks are configured.
10. Event builder, card wrappers, trigger matching threads, and elink models are initialized and started.

The option strings passed in the configuration files containing elink numbers are stored in *elink maps*; these key the half card number to the corresponding TTC or data elinks. One half card may have one or zero TTC elinks at most, while data elinks have no limitations. Instances of elink models are created for each elink to be initialized; then the corresponding parser gets initialized with the relevant options and the corresponding parsing functions are passed to it.

Appendix C

Veto Counter candidate hit position

As was discussed in Section 8.3.5 the Halo efficieny algorithm relies heavily on the upstream extrapolation of the STRAW tracks; a technique proved to be of limited applicability and prone to errors, especially for large beam pipe angles. In this section an attempt in evaluating the quality of such extrapolation is performed.

The metric used to compute the hit position is the *fly time* t_f: once a particle ionizes the scintillator medium of a given tile it produces a light emission that may be registered by one or both photomultipliers. The timestamp of this pulse is encoded in the reconstructed hit candidate and is related to the *x-position* along the tile at which the particle struck. If a signal is generated in both PMTs (tight candidate) the difference between the two hit times is directly proportional to the hit *x-position*.

The time difference between the photomultiplier hit and the light emission is composed by several contributions:

$$t_f = t_{\text{scint}} + t_{\text{lg}} + t_{\text{PMT}} + t_{\text{TDAQ}}. \tag{C.1}$$

where $t_{\text{scint}}, t_{\text{lg}}, t_{\text{PMT}}, t_{\text{TDAQ}}$ represents the time of propagation of the signal along the scintillator medium, the lightguide, the phototube and the frontend electronics and TDAQ system. Since a direct evaluation or simulation of $t_{\text{lg}}, t_{\text{PMT}}, t_{\text{TDAQ}}$ is not possible within this work and would introduce several sources of uncertainty we will restrict ourselves to *tight candidates only*. By doing so and considering the time difference between Jura and Saleve any contribution from anything but the scintillator gets cancelled out since the subdetector is symmetric.

C.1 A model for light propagation in scintillating slabs

Lets consider a bidimensional slab of scintillator of length L and refractive index n; if interactions producing scintillator light happen at one extrema of the slab it can be demonstrated that the maximum and minimum propagation time are provided by : [51]

$$t_{\text{min}} = \frac{Ln}{c}, \quad t_{\text{max}} = t_{\text{min}} \frac{1}{\cos\theta_m}. \tag{C.2}$$

where θ_m is the maximum permitted emission angle with respect to the slab axis. The probability density function of the fly time for an emission at $t_0 = 0$ and thus the total number of emitted photons is given by:

$$\frac{dN}{dt} = -2\pi \frac{Ln}{ct^2} \rightsquigarrow N = \int_{t_{\text{min}}}^{t_{\text{max}}} \frac{dN}{t} d\tau = 2\pi(\cos\theta_m - 1). \tag{C.3}$$

We should notice that in this model the propagation time depends on the x coordinate only.

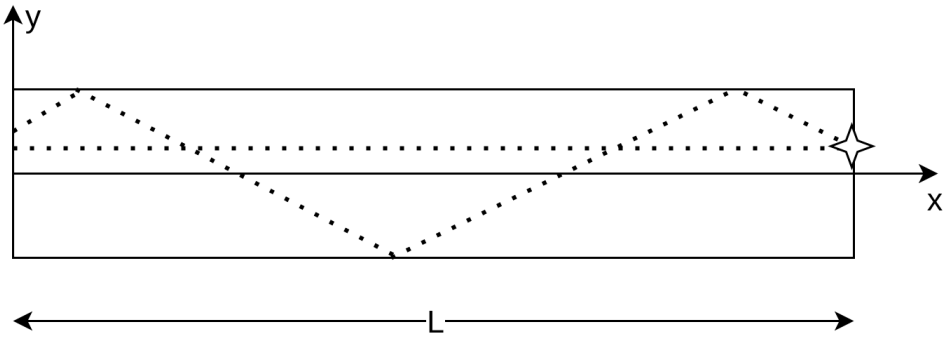

Figure C.1: Minimum and maximum time trajectories in a very thin scintillator slab.

The scintillator material used in the Veto Counter, Bicron B-408 has $n = 1.58$ [52]. Since the tiles are immersed in air it is fair to consider $\theta_m = \frac{\pi}{2} - \theta_c$ with θ_c the critical angle for an air-BC-408 interface.

$$\theta_c = \sin^{-1}\left(\frac{n_2}{n_1}\right) = \sin^{-1}\left(\frac{1}{1.58}\right) \approx 0.685 \quad (\sim 39.3°). \tag{C.4}$$

This choice is justified by the fact that for angles of incidence greater than θ_c the reflectance coefficients approaches zero and thus almost all the incident light is transmitted outside of the scintillator where it is absorbed. In these cases the light intensity reaching the PMT would be extremely low and most likely the produced signal will fall below the CFD threshold[1]. In this very simple model the scintillator medium is assumed perfectly transparent; in reality this is not the case and any attenuation from the material will further hinder propagation for already low intensity reflected light.

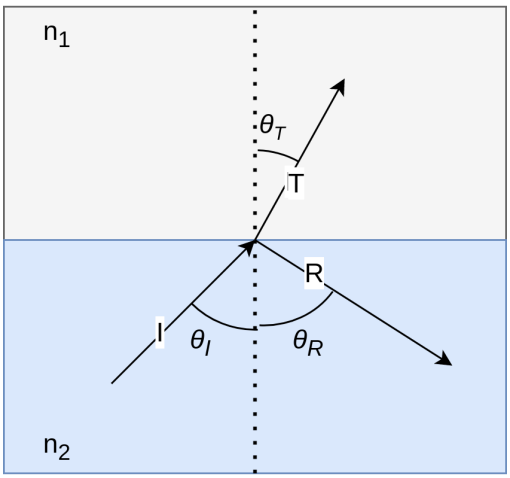

Figure C.2: Light propagation at an interface between two materials with n_1, n_2. All relevant angles are shown.

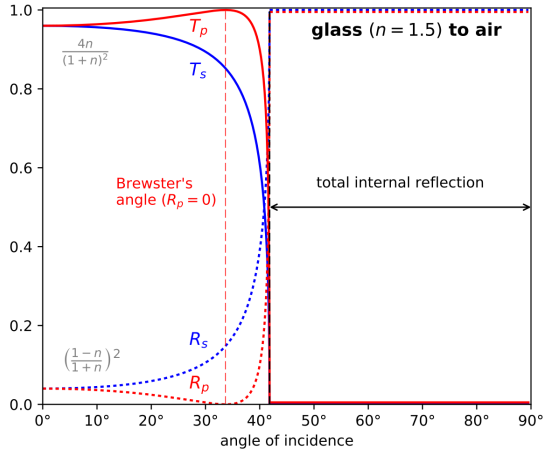

Figure C.3: Fresnel power coefficients of transmittance T and reflectance R for a glass to air interface. Notice how for incidence angles lower than the critical one reflectance is negligible. These coefficients are computed for s and p wave coordinate system; any beam may be written as a combination of s and p polarized waves.

By considering a Veto Counter tile $L = 120$ mm the propagation times are estimated to be:

$$t_{\text{min}} \approx 6.32 \times 10^{-10}\text{s}, \quad t_{\text{max}} \approx 9.99 \times 10^{-10}\text{ s}. \tag{C.5}$$

[1] If the event is sufficiently close to one side the relative PMT may detect the hit, but the further one will not, this will yield to a loose candidate that is not considered in this study.

C.2 Candidate hit time distribution

In this study we consider the same dataset used for the efficiency computation. For all the single track events which satisfies the cuts outlined in 8.3.1 only the tight candidates in time with the tracks $\Delta t < 2$ ns are considered. An example of time distribution is shown in Figure C.4.

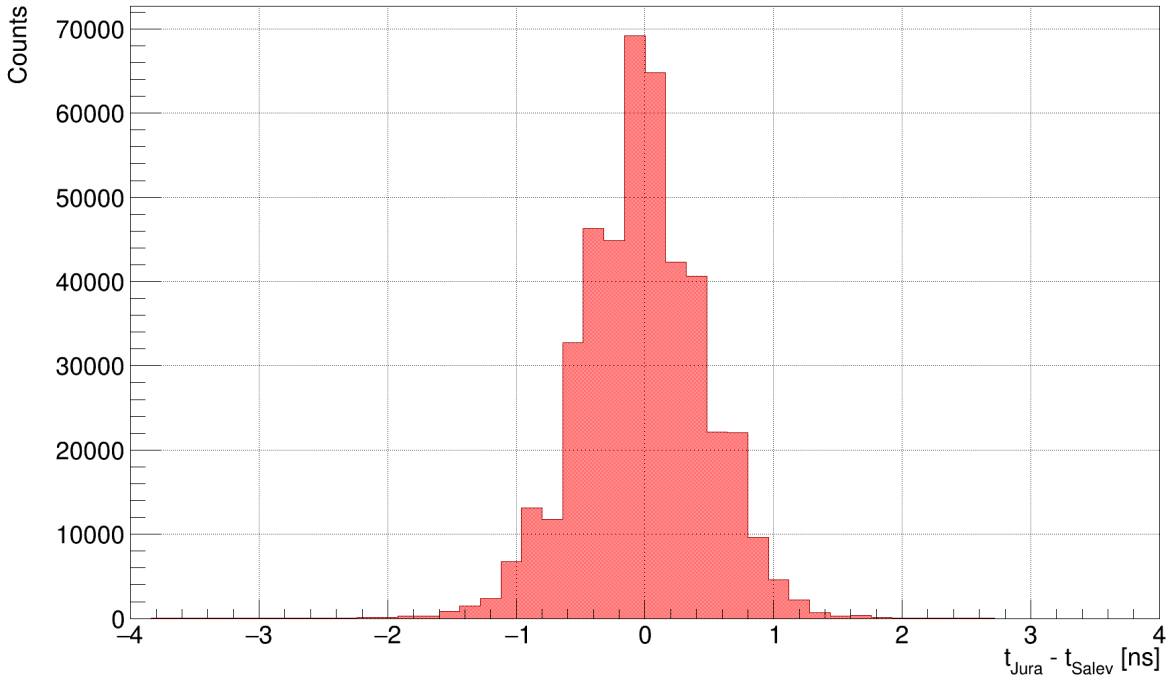

Figure C.4: Time difference distribution for station 3 between Jura and Saleve signals. FELIX readout.

Notice how the mean time difference is centered around zero and how the time values are compatible with the expected theoretical computation of (C.5).

In order to get an estimation of the strike position lets call μ the *calibration factor* between time and position, that is:

$$\mu(\bar{t}) = \frac{x_{\max}}{\bar{t}}, \quad x = \Delta t \mu. \tag{C.6}$$

$\bar{t}$ being the assumed mean travel time across the tile x direction. Lets call R the ratio between the number of candidates whose computed position lies within the detector and the total number of selected candidates. R can be computed as a function of $\bar{t}$; being the number of candidates within the detector dependant by it. Considering a reasonable interval such as $\bar{t} = [0.5; 1.5]$ and selecting candidates from station 3 only[2] we get the following:

[2]This choice is done because station 3 is the one for which the quality of the extrapolated position estimation is the highest. This will be important in the proceeding.

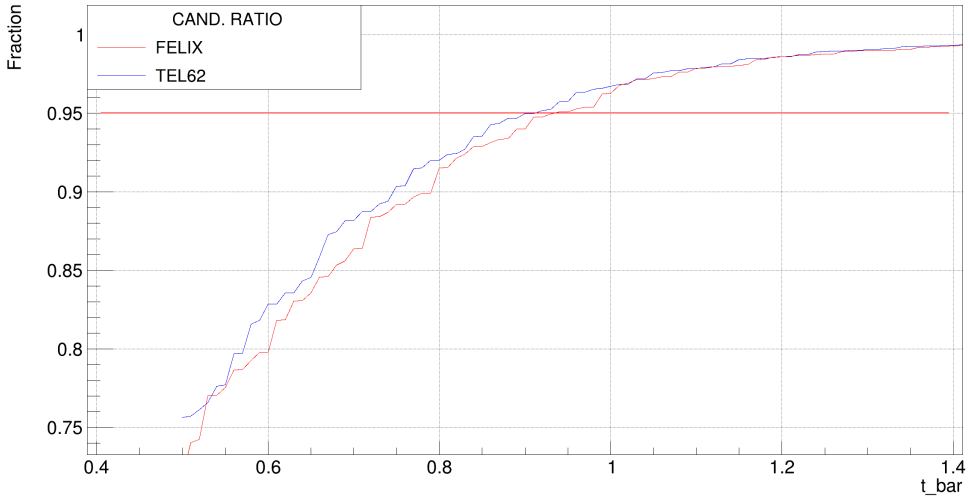

Figure C.5: Ratio between number of candidates within the detector x acceptance and the total number of selected candidates, as a function of $\bar{t}$. Emphasized $R = 0.95$

lets thus estimate $\bar{t}$ as the time value for which 95% of the candidate positions are computed to be within the detector acceptance. That yields:

$$\langle \bar{t} \rangle \approx 0.92 \times 10^{-10}\,\mathrm{s} \tag{C.7}$$

C.2.1 Computed candidate position

With this estimation for the mean time and thus μ lets perform the time calibration and compare the obtained values with the extrapolated ones.

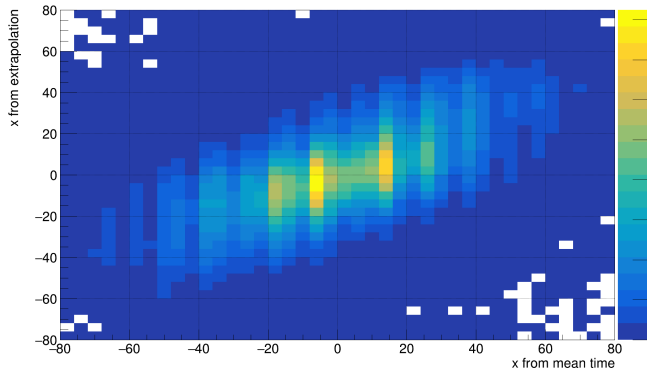

Figure C.6: Correlation plot of the computed and extrapolated position for station 3 candidates. TEL62 readout

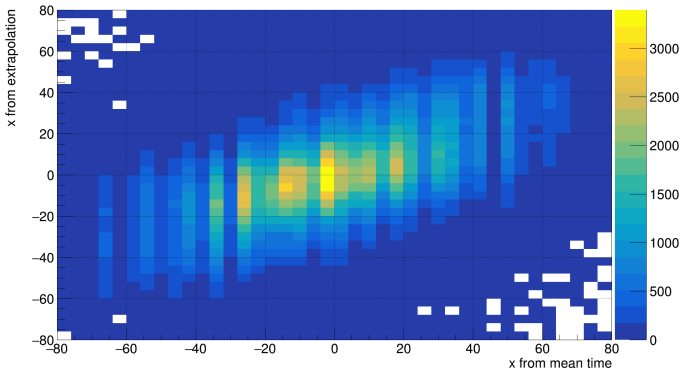

Figure C.7: Correlation plot of the computed and extrapolated position for station 3 candidates. FELIX readout

In Figures C.6-C.7 we can see the computed position from time against the extrapolated position. This comparison was performed for both readouts and the time calibration factor has been computed using the (C.7) estimation. The correlation between the two estimation is clearly visible.

Lets further expand this computation; as was discussed previously the quality of the extrapolation is dependant on the tile ID: high index tiles being the ones with the highest chance of missestimation. While Figures C.6-C.7 include all the tiles from station 3 now we consider one tile at a time and we compute the correlation factor between the two estimations; the chosen mean time is always $\langle \bar{t} \rangle$.

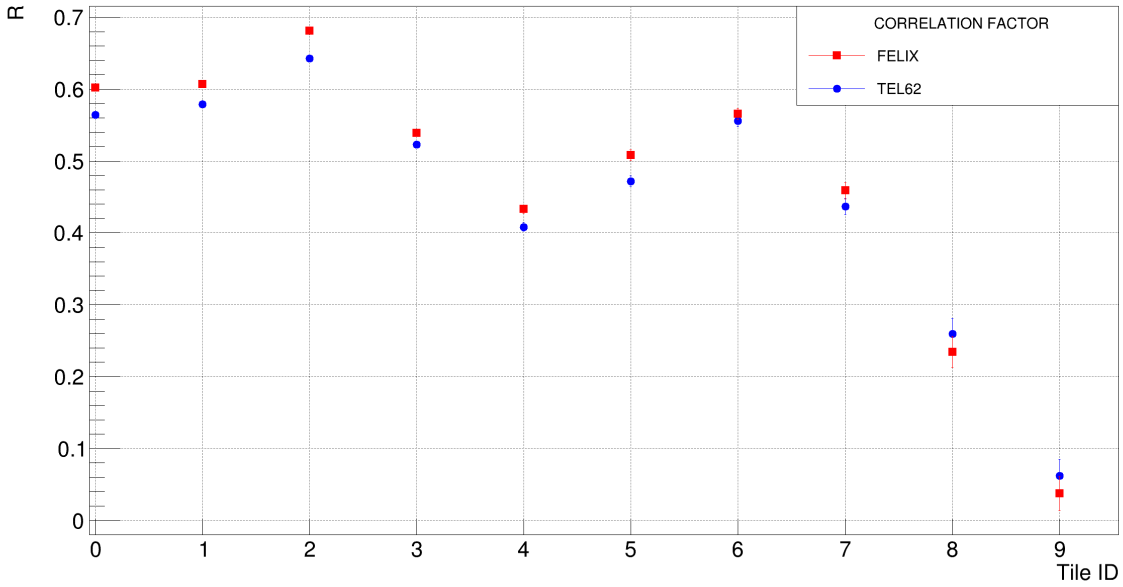

Figure C.8: Correlation coefficient between extrapolated and computed position as function of Tile ID; only station 3 is considered. Both readouts.

It is clear that the readout system in this case plays no difference. The slight variation between TEL62 and FELIX may be due to the lower raw hit efficiency of the former observed in station 3 and does not appear to depend on the Tile ID. The correlation factor appears to be the highest for low index tiles and decreases with Tile ID[3], tile 10 is excluded from this study due to no statistics in the analyzed dataset.

This results confirms what was already suspected regarding the extrapolation technique, the dropoff of r at high Tile ID means precisely that the extrapolation loses validity. This statement is justified by the intrinsically better approach of the time of flight estimation technique: even though the scaling factor may be not well estimated a correlation between the two datasets should still be observed; the extrapolation technique loses accuracy the further away from the beam pipe, while there is no reason to suspect something similar happens for the time of flight estimation since it is based on a physical phenomenon within the tiles. Notice also that the correlation has a double maximum structure; before the dropoff at tile 6. Again this can be justified by the fact that extrapolation is best for the tiles closest to the beampipe; asymmetry comes from the external obstacles and the geometry of the Veto Counter.

[3]please remember that the statistics is also lower for high order tiles.

C.3 Estimation deviations

Lets characterize the candidates for which the two estimations yield different results. In Figure C.9-C.10 we plot the extrapolated position of all the candidates whose computed position falls outside the tile boundary. The distribution appears to have a double peak structure losely centered around zero. This is expected given the nature of the scaling law used to compute the position. Due to the complex interaction of different factors (the extrapolation error as function of the position...) it is not possible to justify this shape further.

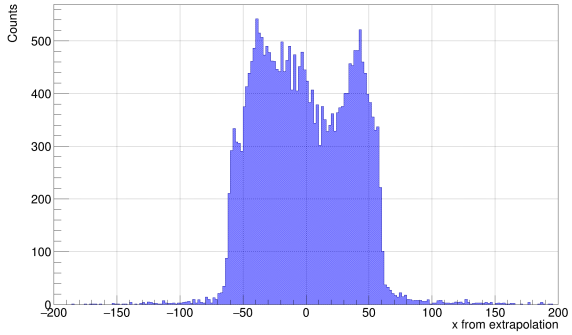

Figure C.9: Extrapolated position of candidates whose computed position lies outside the tile. TEL62 readout

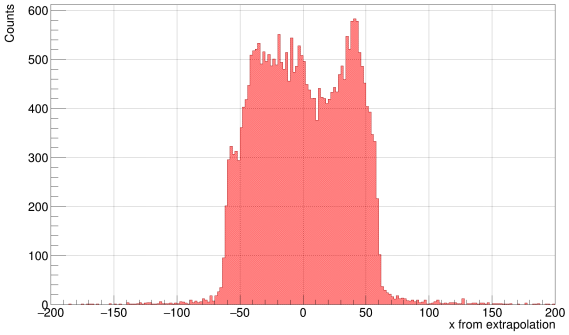

Figure C.10: Extrapolated position of candidates whose computed position lies outside the tile. FELIX readout

In Figure C.11-C.12 we can also observe the distribution of residuals between the two estimations. We then compare it to a normal distribution by means of a *Q-Q plot*.

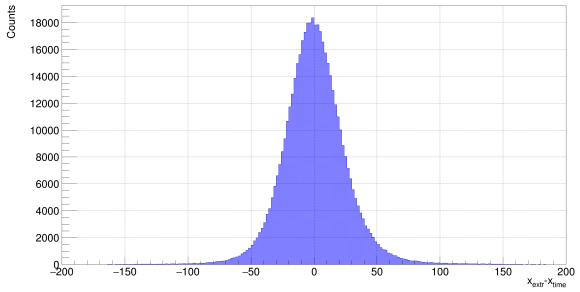

Figure C.11: Residual between extrapolated and computed positions. TEL62 readout

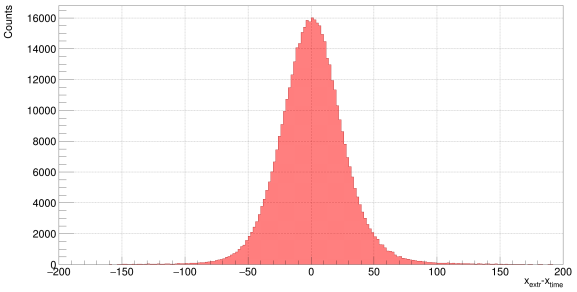

Figure C.12: Residual between extrapolated and computed positions. FELIX readout

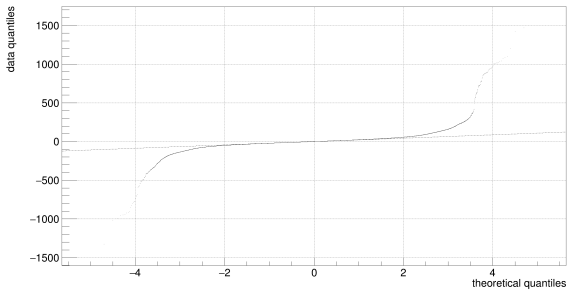

Figure C.13: Q-Q plot for the residuals. TEL62 readout

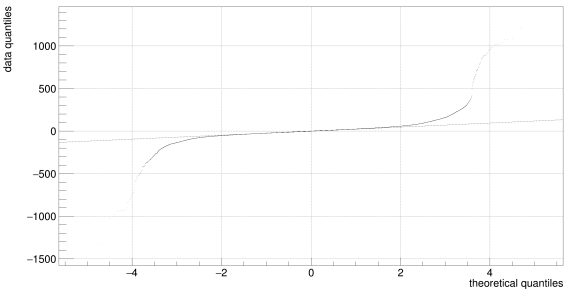

Figure C.14: Q-Q plot for the residuals. FELIX readout

While the distribution appears normal from the Q-Q plot it is clear that it features *heavy tails*; for some candidates the extrapolated position is dramatically different than the computed position.

Appendix D

Considerations on the Merger RAID

As was discussed in Chapter 4 the merger machines are used to merge the event fragments coming from the farm HLT into raw burst files. These files are written on disk by the merger software and then scheduled to be moved to EOS; the typical I/O load on the disks will then be mainly composed by large, periodic writes and somewhat continuous reads performed by the EOS transfer daemon. Raw data file size is generally constant and of the order of GBs.

In the 2022 run an issue was observed with one of the merger machines: merger 4 showed constantly high *I/O utilisation* and often fell behind the other machines in the cluster, meaning that its file write speed could not keep up with the incoming data. This bottleneck caused the file writing to lag with respect to the other machines and the increasing back-pressure caused the software to use up all the available RAM to cache the still unwritten data. Once the memory got all used up the machine would start to drop events; an extremely undesired situation. Moreover the increased load on the remaining nodes would trigger a chain reaction with the lower RAM merger 2 failing shortly after.

During the data taking this problem has been addressed by simply monitoring the RAM usage on merger 4, excluding it from the DAQ as soon as it crossed the arbitrary threshold of 75%, waiting for the files backlog to be written and then restarting the merger software.

After the end of the run an investigation took place and the results of it are discussed in the following.

D.1 RAID technology introduction

RAID is an acronym that stands for *Redundant Array of Independant Disks* [53]. It is a technique where multiple hard drives are used concurrently to store data; by scattering them on multiple units. Using such approach has various advantages, especially regarding data integrity and performance: many RAID implementations feature parity error correction and/or data duplication; this ensures that in the event of a disk failure the data loss can be mitigated and missing data may be reconstructed from the information in the other drives. Moreover, having multiple disks can in some RAID topologies increase bandwidth and throughput, since the R/W operation is essentially performed in parallel.

A RAID may be achieved in hardware or software: a software RAID is managed and created at the operating system level; the disks are independent and handled by software such as `mdadm` [54] that performs all the RAID specific operations. This kind of RAID is very flexible and inexpensive, but may offer lower performance than other options in some circumstances. RAID can also be done using dedicated hardware. This is usually in the form of a PCI-E card that connects to the hard drives via SATA or SAS and handles all the relevant operations on board. Usually these cards have a dedicated BIOS for configuration and the *virtual RAID drive* is seen from the operating system as a standard hard drive. Hardware RAID is OS-agnostic and all the RAID operations are off-loaded from the CPUs to the onboard ASICs. This may be a performance advantage in some cases. Many inexpensive hardware controllers lacks any on board cache and thus their ability to effectively queue operations on the disks

is severely compromised: when data reaches the controller it is scheduled to the disks. While the hard disks have a cache on board it is usually small (∼MB) and unoptimised for the typical RAID operations. The disk cache is usually disabled by the controller itself, having an active cache can also be problematic in case of power loss: due to the nature of the RAID writes if an outage occurs during an operation it may result in a corrupted array because the usual single hard drive protective features are not capable of managing RAID. This is the reason why high end controllers usually have a large onboard cache and a battery backup: individual hard drive caches are disabled and the common RAID cache is used. In the event of power loss the cache will retain the data and write will be resumed at reboot.

In software RAID the caches are usually left active and all operations are done at OS or kernel level. At the time of writing current CPUs are extremely fast in parity computations and in some cases the performance differential between a well optimised software RAID and a cacheless hardware RAID can be negligible. In some cases the hardware RAID may fare worse than a cacheless hardware solution [55].

D.2 The RAID 5 flavour

As said previously a RAID array may be configured in various ways depending on the prioritized feature (throughput, data integrity, fraction of the available storage space usable…); In the following we will give an overview of RAID5; the configuration used in the merger machines, for other flavours one may refer to [56].

RAID5 is a *striping* RAID with *distributed parity*. This means that data blocks are scattered among all the disks in the array; for every block written a special parity block, containing checksum information, is written. These additional data may be used for error correction and data recovery should one disk fail. RAID 5 is a good compromise between throughput and data integrity: the striping insures data writing is performed in parallel among the disks, while the presence of the parity information provides redundancy. In RAID 5 reads can be very fast due to the concurrent access, but writes may slowed down by the parity computations; the parity is calculated *strip-wise*.

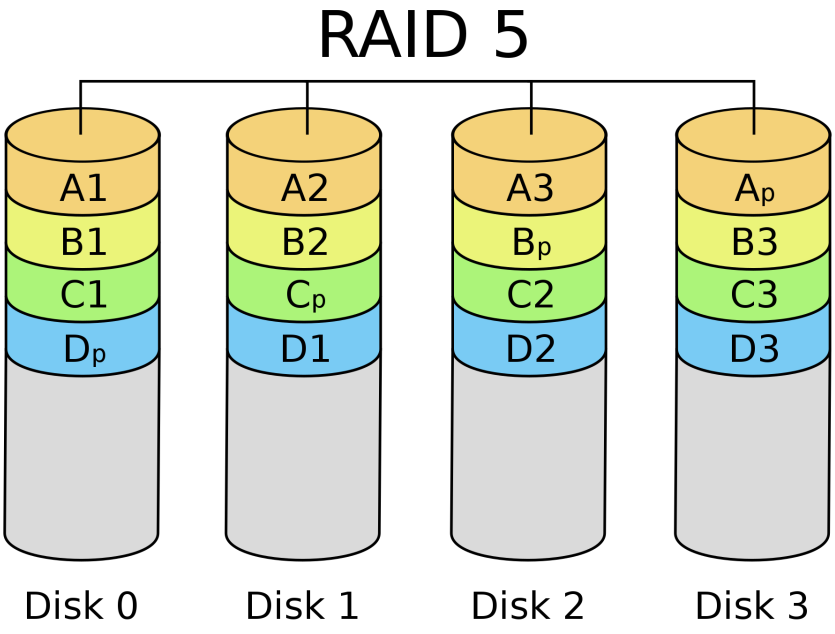

Figure D.1: Schematic of a typical RAID 5 data distribution among 4 disks, the blocks denoted with *p* are the parity blocks for the corresponding *stripe*. Notice how they are scattered among all disks.

D.3 Merger 4 issues

Merger 4 uses an `AVAGO MegaRaid 9440-8i` hardware controller configured for hardware RAID 5. This controller is cache-less and operates by default with disk caches disabled.

During the data taking an extremely high I/O utilisation[1] was observed, approaching and keeping to 100%. This means that, on the point of view of the kernel, the disk is almost never idle and I/O request are piling up. This problem led to the effects described previously. Since merger 4 is a relatively new machine (2020) running with a dated OS and kernel (CentOS 7, CC7) at first this issue was thought to be at software and firmware level. An *identical machine* (Merger 5) was set up and used to test various solutions, all yielded no improvement:

- The kernel was upgraded to the latest officially supported by CC7 (`3.10.0-1160.76.1.el7`);
- The motherboard BIOS was upgraded to the latest version;
- The RAID controller firmware was upgraded to the latest version;
- The entire machine was migrated to CentOS stream 8;
- The entire machine was migrated to Alma Linux 9;
- A dedicated write buffer for file writes was added within the merger software, on top of any kernel write scheduling.

These tests were carried out in parallel with the standard data taking, with merger 5 added temporarily to the TDAQ system and in the Run control.

Following the end of the run more in depth testing could be performed. At that stage testing was done with dummy data. At first merger 5 was configured in a pure software RAID and this significantly increased the performance of the system: I/O utilisation was lower and no major lagging was observed with test data. Confident with this information the same operation was performed on merger 4, together with BIOS and firmware upgrades; results observed on merger 4 were subjectively similar with merger 5. In order to quantify these improvements a testing procedure was designed and performed on the merger clusters, it is discussed in the following sections.

D.4 Benchmarking the merger RAIDs

Given the type of operations the arary is subjected to it was decided to perform a benchmark focusing on *data writes*. The benchmarking software used is `FIO` [57]. FIO is a very versatile benchmark utility that can reproduce almost any disk workload, being fully configurable.

The testing consisted of *sequential writes* of different *blocksizes* varying from 4 kB to 10 MB; the maximum I/O depth[2] was set to 32 and all the writes were performed in *direct I/O* mode to avoid any kernel interference in the testing. Since the merger software can in principle write up to 4 files concurrently the tests were run with a variable number of worker threads, 1 to 64. Testfiles to be written are 10 GB in size.

In the following only a small subsets of relevant results are shown.

D.4.1 Baseline: merger 1 and 2

The first testing done did not involve directly merger 4-5. In order to get a baseline from machines that never exhibited any issues regarding writing speed merger 2 and 1 were tested. Merger 1 is the oldest merger in NA62 with an `ADAPTEC AAC-RAID` card, while merger 2 is newer and uses an `MegaRAID SAS 2208` card.

In Figure D.2-D.3 we can observe the throughput for a blocksize of 256 kB.

[1] that is, the ratio of idle time vs busy time of the drive.
[2] maximum number of pending I/O requests before disk acknowledgment

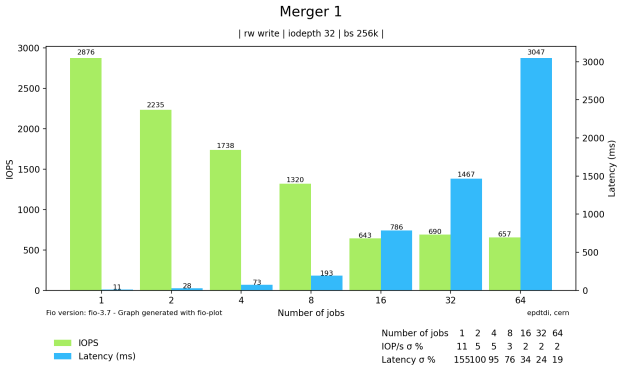

Figure D.2: Merger 1 IOPS and latency for different thread count.

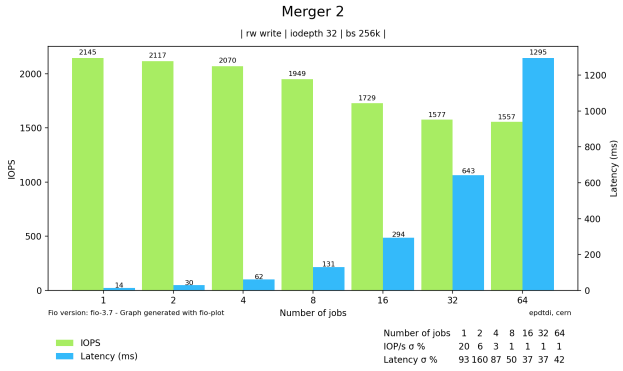

Figure D.3: Merger 2 IOPS and latency for different thread count.

The main difference between these two machines is that the new card seems to fare better than the `ADAPTEC AAC-RAID` when it comes to handling parallel writes, with significantly more consistent throughput. The `ADAPTEC AAC-RAID` only seems to hold a margin in single threaded mode (result is consistent among block sizes and only one size is shown here).

D.4.2 Effect of disk cache on merger 5 HW RAID

The RAID controller from merger 4 and 5 is identical and cache-less. For this reason one of the test performed was to check how much enabling the disks caches would improve the performance. Since the servers are protected from power outages the added risk of data corruption is negligible.

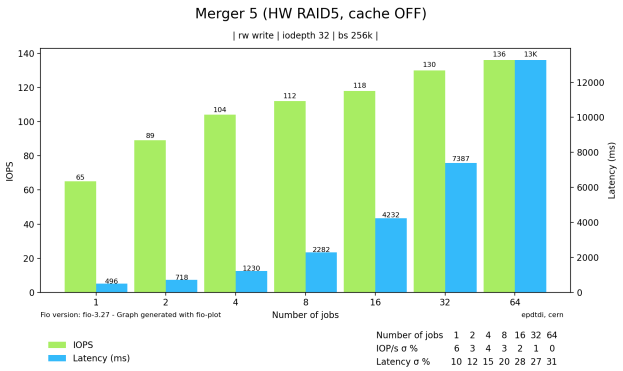

Figure D.4: Merger 5 IOPS and latency with disk cache disabled.

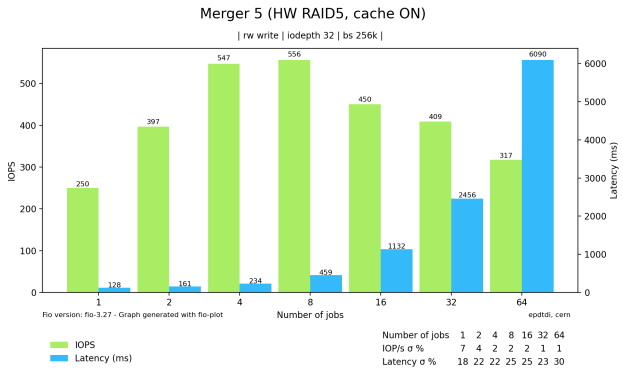

Figure D.5: Merger 5 IOPS and latency with disk cache enabled.

It is clear that, at least on the test carried out, performance of the array with cache turned on is significantly higher, in both latency and throughput. Specifically, considering smaller block sizes, where the effect of the on disk cache is more relevant the difference in throughput approaches one order of magnitude; especially for low number of concurrent writes. At higher block sizes the difference is somewhat lower, but still significant (around three times the throughput).

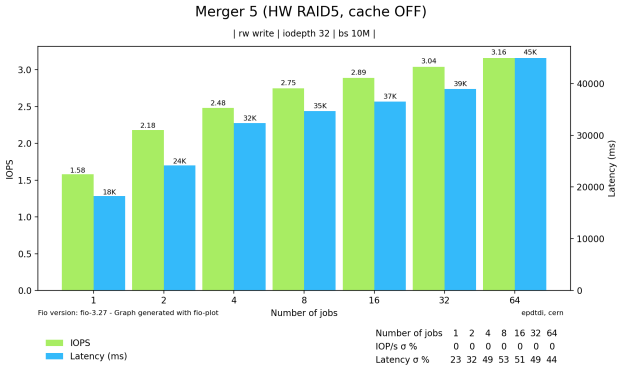

Figure D.6: Merger 5 IOPS and latency with disk cache disabled, large blocksize.

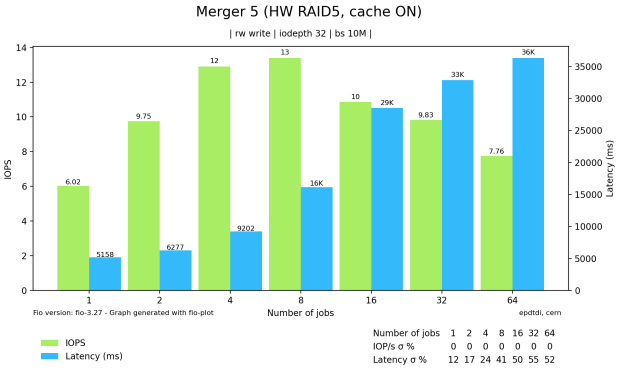

Figure D.7: Merger 5 IOPS and latency with disk cache enabled, large blocksize.

D.4.3 CPU C-states and performance

After realizing that turning on the cache significantly improves the performance of the hardware RAID further investigation was carried out: In some instances the CPU power saving features can compromise disk performance. For this reason all energy saving features were turned off on merger 5 and the CPU C-states [58] were forbidden up to C2.

These operations did not improve the RAID performance in any appreciable way, as we can observe in Figure D.8.

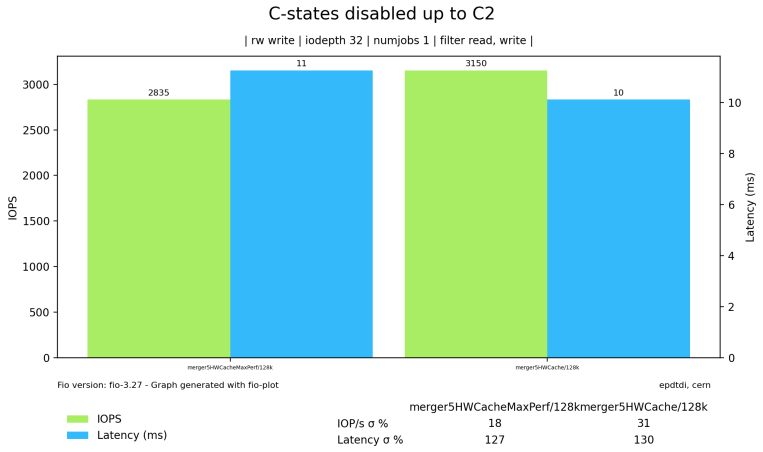

Figure D.8: Difference in performance with C-states enabled or disabled, HW RAID with disk cache, 128 kB block size and 1 thread. Left with C states disabled, right enabled.

D.5 A software RAID solution

Given the extremely poor performance of the hardware RAID on merger 4-5[3] a pure *software RAID* has been tested: the RAID controller board on merger 4 was set up as a *pass-through* (Just a Bunch of Disks, JBOD) and an appropriate software RAID array was build using `mdadm`; the performance of this system has been compared with the hardware solution on merger 5.

While the best hardware RAID configuration (disk caches enabled and maximum CPU performance) is significantly faster for block sizes under 128 kB the software RAID outperforms it for any larger blocksize. This may be due to the hardware RAID controller lack of cache: Small block sizes can

[3] even with disk cache on the comparison with merger 1-2 shows more than one order of magnitude of difference in throughput.

be written promptly by the hardware controller while the software RAID suffers from OS-level optimizations. For higher block sizes the situation swaps, the software RAID performs significantly better, most likely for better scheduling.

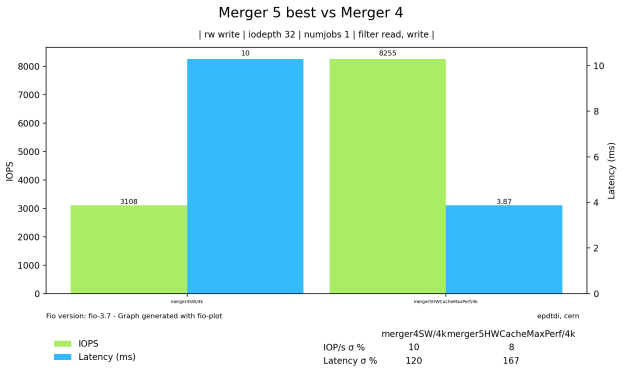

Figure D.9: Comparison between best HW RAID (right) implementation and software RAID (left), 4 kB block size.

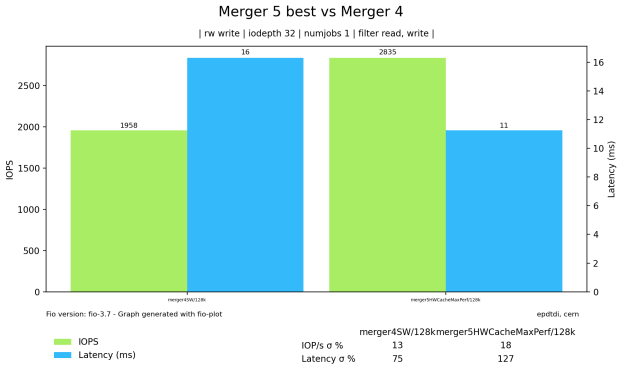

Figure D.10: Comparison between best HW RAID (right) implementation and software RAID (left), 128 kB block size.

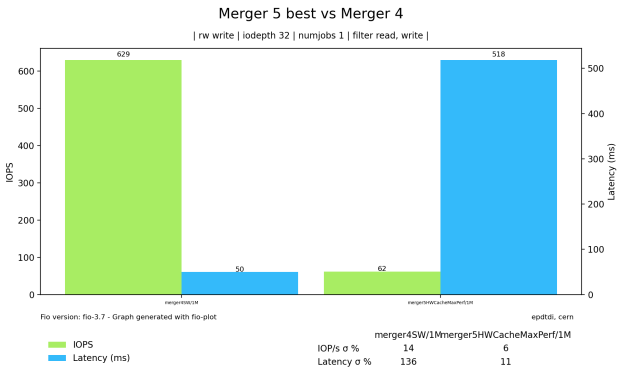

Figure D.11: Comparison between best HW RAID (right) implementation and software RAID (left), 1024 kB block size.

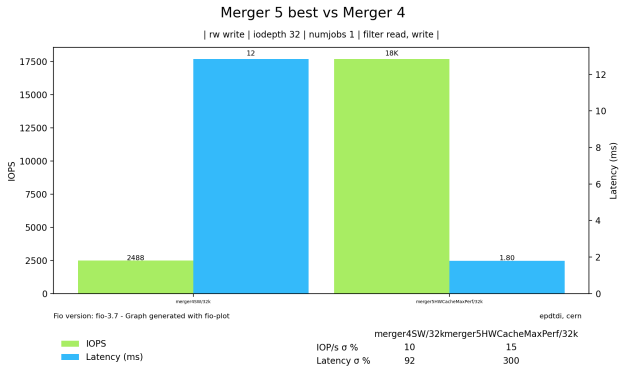

Figure D.12: Comparison between best HW RAID (right) implementation and software RAID (left), 32 kB block size.

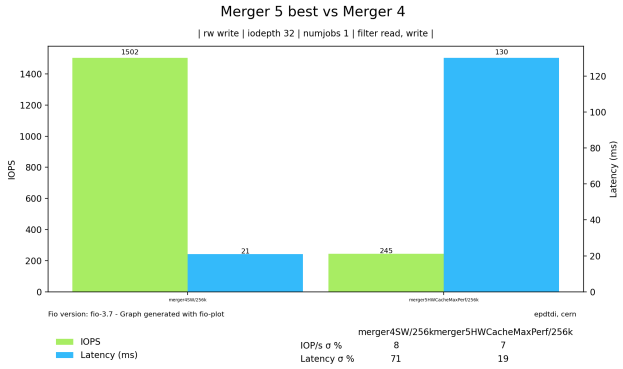

Figure D.13: Comparison between best HW RAID (right) implementation and software RAID (left), 256 kB block size.

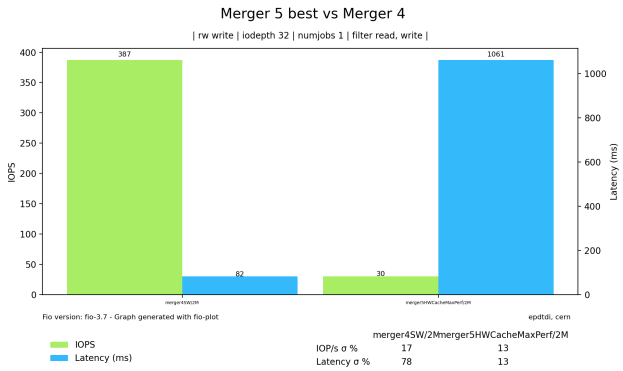

Figure D.14: Comparison between best HW RAID (right) implementation and software RAID (left), 2048 kB block size.

As a last study lets also compare the software RAID solution with one known working merger node, such as merger 2.

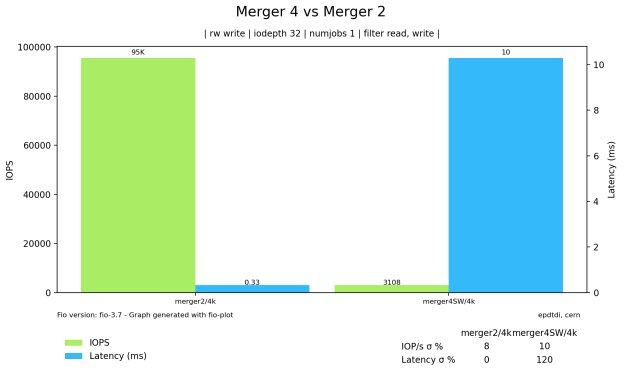

Figure D.15: Comparison between merger 2 hardware RAID (left) and merger 4 software RAID (right), 4 kB block size.

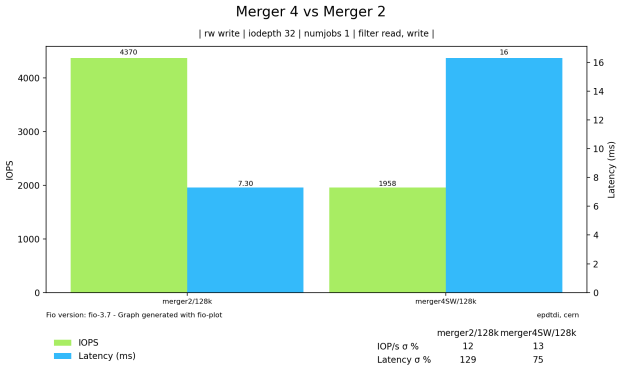

Figure D.16: Comparison between merger 2 hardware RAID (left) and merger 4 software RAID (right), 128 kB block size.

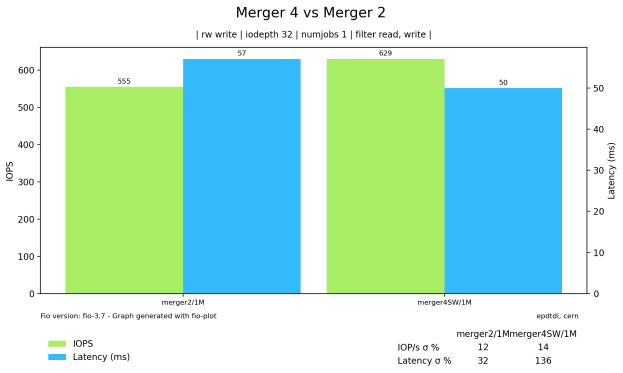

Figure D.17: Comparison between merger 2 hardware RAID (left) and merger 4 software RAID (right), 1024 kB block size.

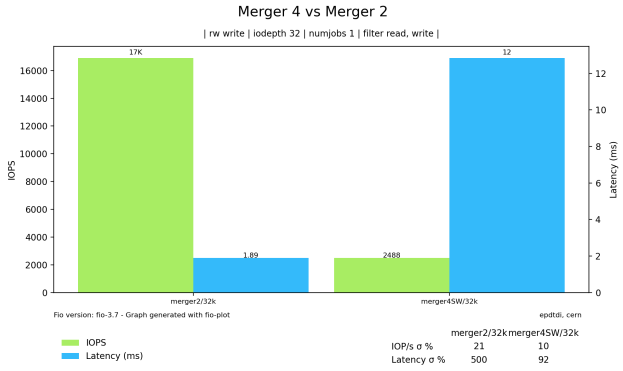

Figure D.18: Comparison between merger 2 hardware RAID (left) and merger 4 software RAID (right), 32 kB block size.

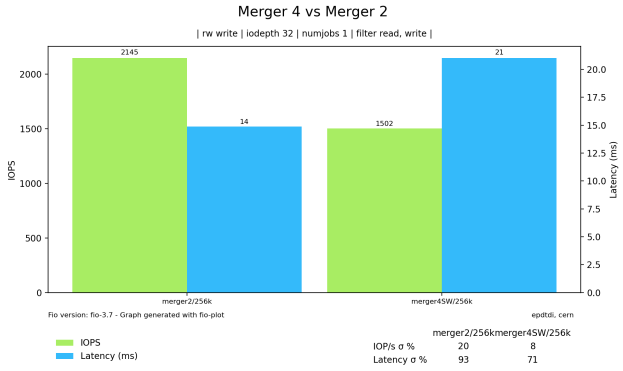

Figure D.19: Comparison between merger 2 hardware RAID (left) and merger 4 software RAID (right), 256 kB block size.

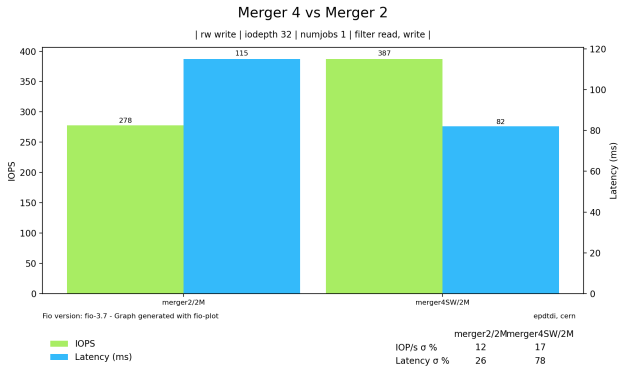

Figure D.20: Comparison between merger 2 hardware RAID (left) and merger 4 software RAID (right), 2048 kB block size.

As a final result we show in Figure D.22-D.21 the IOPS ratio between the different systems. R_{2-4} is the ratio between merger 2 and merger 4 IOPS and is related to the performance difference between the a working node and the new software RAID for merger 4-5; similarly R_{2-5} is the difference between the best hardware RAID configuration for merger 4-5 and merger 2. R_{4-5} is a direct comparison between software and hardware RAID on the merger 4 and 5 platform.

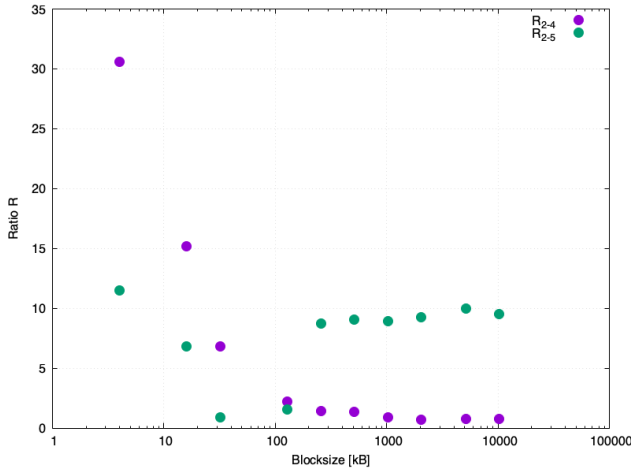

Figure D.21: IOPS ratios between a stable node (merger 2) and HW RAID (R_{2-5}), between merger 2 and SW RAID (R_{2-4}), on the merger 4-5 platform.

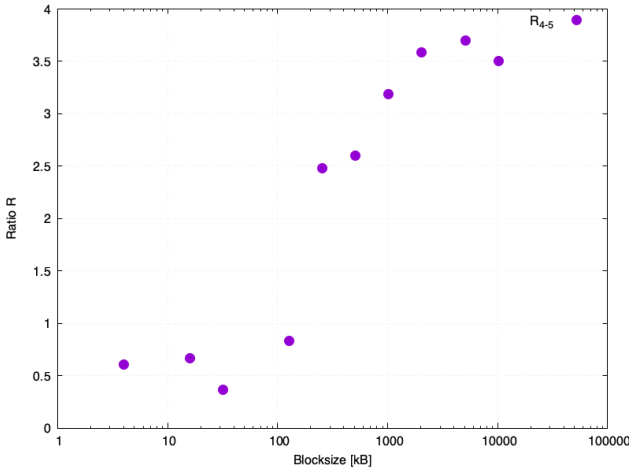

Figure D.22: IOPS ratios between HW and SW RAID on merger 4-5 platform.

D.6 Conclusion

It was shown that the Hardware RAID controller used in merger 4 and 5 does not have high enough performance to cope with the data taking conditions of NA62. A mitigation of the issue can be achieved in the form of a software RAID by setting up the controller in JBOD mode and using the `mdadm` software. The performance of this system, in the test performed appears better than merger 2 for blocksizes higher than 1 MB (up to 25% better), while being significantly lower for smaller block sizes. The improvement between software and hardware RAID on merger 4-5 is up to 4 times the IOPS for block sizes higher than 1 MB, hardware RAID on those nodes holds an advantage only for small block sizes (< 1 MB).

Nevertheless, the software RAID solution seems to be fit enough for coping with the intended workload, this was verified with some test runs with dummy data.

Bibliography

[1] E. Gamberini, *Development and Commissioning of the GigaTracker Data Acquisition and Control Systems for the NA62 Experiment at CERN*. PhD thesis, Ferrara U., 2017.

[2] G. A. et al, "Proposal to measure the rare decay $k \longrightarrow \pi\nu\bar{\nu}$ at the CERN SPS," tech. rep., CERN, 2005.

[3] A. Romano, "The $K^+ \to \pi^+\nu\bar{\nu}$ decay in the NA62 experiment at CERN," tech. rep., CERN, 2014.

[4] F. Ambrosino, A. Ceccucci, H. Danielsson, N. Doble, R. Fantechi, A. Kluge, C. Lazzeroni, M. Lenti, G. Ruggiero, M. Sozzi, P. Valente, and R. Wanke, *NA62 Technical Design Document*. NA62 collaboration, 2010.

[5] V. F. et al., "The beam and detector for the NA48 neutral kaon cp violation experiment at CERN," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 574, no. 3, pp. 433–471, 2007.

[6] *NA62 data format living note*, 2022.

[7] M. Boretto, *First observation of the rare decay mode $K \longrightarrow \pi\nu\bar{\nu}$ in the NA62 experiment at CERN SPS*. PhD thesis, Università degli studi di Torino, 2019.

[8] J. Christiansen, "HPTDC High Performance Time to Digital Converter," tech. rep., CERN, Geneva, 2004. Version 2.2 for HPTDC version 1.3.

[9] F. Spinella, B. Angelucci, G. Lamanna, M. Minuti, E. Pedreschi, J. Pinzino, R. Piandani, M. Sozzi, and S. Venditti, "The TEL62: A real-time board for the NA62 trigger and data acquisition. data flow and firmware design," in *2014 19th IEEE-NPSS Real Time Conference*, Institute of Electrical and Electronics Engineers Inc., 4 2015.

[10] A. Ceccucci, R. Fantechi, P. Farthouat, G. Lamanna, and V. Ryjov, "The na62 liquid krypton calorimeter readout module," *Journal of Instrumentation*, vol. 6, p. C12017, dec 2011.

[11] M. T. on behalf of ATLAS TDAQ Collaboration, "Felix: The new readout system for the atlas detector," tech. rep., CERN, 2019.

[12] A. F. Group, *FELIX User Manual*.

[13] M. Ceoletta, E. Gamberini, N. Lurkin, R. Marchevski, J. Swallow, and M. Zamkovsky, "The NA62 vetocounter detector." internal living note, 2023.

[14] P. Zyla, R. Barnett, J. Beringer, O. Dahl, D. Dwyer, D. Groom, C. Lin, K. Lugovsky, E. Pianori, D. Robinson, C. Wohl, W. Yao, K. Agashe, G. Aielli, B. Allanach, C. Amsler, M. Antonelli, E. Aschenauer, D. Asner, and W. Zheng, "Review of particle physics," *Progress of Theoretical and Experimental Physics*, vol. 2020, 08 2020.

[15] J. Laiho, E. Lunghi, and R. S. V. de Water, "Lattice QCD inputs to the CKM unitarity triangle analysis," *Physical Review D*, vol. 81, feb 2010.

[16] K. A. O. et al., "Review of particle physics," *Chinese Physics C*, vol. 38, 8 2014.

[17] J. Brod, M. Gorbahn, and E. Stamou, "Two-loop electroweak corrections for the $k \longrightarrow \pi\nu\bar{\nu}$," *Physical Review D*, vol. 83, feb 2011.

[18] G. Isidori, F. Mescia, P. Paradisi, C. Smith, and S. Trine, "Exploring the flavour structure of the MSSM with rare ik/i decays," *Journal of High Energy Physics*, vol. 2006, pp. 064–064, aug 2006.

[19] W. J. Marciano and Z. Parsa, "Rare kaon decays with "missing energy"," *Phys. Rev. D*, vol. 53, pp. R1–R5, Jan 1996.

[20] G. Buchalla and A. J. Buras, "The rare decays beyond leading logarithms," *Nuclear Physics B*, vol. 412, pp. 106–142, jan 1994.

[21] M. Misiak and J. Urban, "QCD corrections to FCNC decays mediated by z-penguins and w-boxes," *Physics Letters B*, vol. 451, pp. 161–169, apr 1999.

[22] L. S. Littenberg, "CP-violating decay $K_L^0 \to \pi^0\nu\bar{\nu}$," *Phys. Rev. D*, vol. 39, pp. 3322–3324, Jun 1989.

[23] A. J. Buras, S. Uhlig, and F. Schwab, "Waiting for precise measurements of $k \longrightarrow \pi\nu\bar{\nu}$," *Reviews of Modern Physics*, vol. 80, pp. 965–1007, aug 2008.

[24] SPS, "NA62 status report to the CERN SPSC," tech. rep., CERN, 2022.

[25] H. W. Atherton, C. Bovet, N. T. Doble, L. Piemontese, A. Placci, M. Placidi, D. E. Plane, M. Reinharz, E. Rossa, and G. Von Holtey, *Precise measurements of particle production by 400 GeV/c protons on beryllium targets*. CERN Yellow Reports: Monographs, Geneva: CERN, 1980.

[26] C. Bovet, R. Maleyran, L. Piemontese, A. Placci, and M. Placidi, "The cedar counters for particle identification in the SPS secondary beams: A description and an operation manual," tech. rep., CERN, 1982.

[27] SPS, "NA62 status report to the CERN SPSC," tech. rep., CERN, 2021.

[28] S. A. Fedotov, A. A. Kleymenova, and A. N. Khotjantsev, "New CHOD detector for the NA62 experiment at CERN," *Physics of Particles and Nuclei*, vol. 49, pp. 26–29, 1 2018.

[29] S. Kholodenko, "NA62 charged particle hodoscope. design and performance in 2016 run," *Journal of Instrumentation*, vol. 12, pp. C06042–C06042, jun 2017.

[30] G. Atoian, V. Issakov, O. Karavichev, T. Karavicheva, A. Poblaguev, and M. Zeller, "Development of Shashlyk calorimeter for KOPIO," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 531, pp. 467–480, oct 2004.

[31] G. Haefeli, A. Bay, F. Legger, and L. Locatelli, "TELL1: A common data acquisition board for LHCb," tech. rep., CERN, 2003.

[32] "URL: `www.caen.it/`."

[33] G. A. Rinella, M. Fiorini, P. Jarron, J. Kaplon, A. Kluge, E. Martin, M. Morel, M. Noy, L. Perktold, and K. Poltorak, "The tdcpix readout asic: A 75 ps resolution timing front-end for the gigatrackerof thena62 experiment," *Physics Procedia*, vol. 37, pp. 1608–1617, 2012. Proceedings of the 2nd International Conference on Technology and Instrumentation in Particle Physics (TIPP 2011).

[34] D. Moraes, W. Bonivento, N. Pelloux, and W. Riegler, "The CARIOCA Front End Chip for the LHCb muon chambers," tech. rep., CERN, 1 2003.

[35] "URL: `www.ansible.com/`."

[36] "User Datagram Protocol." RFC 768, Aug. 1980.

[37] W. Eddy, "Transmission Control Protocol (TCP)." RFC 9293, Aug. 2022.

[38] C. Gaspar, M. Dönszelmann, and P. Charpentier, "DIM, a portable, light weight package for information publishing, data transfer and inter-process communication," *Computer Physics Communications*, vol. 140, no. 1, pp. 102–109, 2001. CHEP2000.

[39] "URL: www.ntop.org/products/packet-capture/pf_ring/."

[40] "URL: www.dpdk.org/."

[41] "URL: http://zeromq.org/."

[42] L. Bician, M. Boretto, H. Danielsson, E. Gamberini, M. Koval, G. L. Miotto, P. Lichard, and J. Morant, "Proposal to implement new experimental DAQ system for NA62 readout NA62 trigger and data acquisition system (TDAQ)," tech. rep., CERN, April 2020.

[43] "URL: www.cdrdv2.intel.com/v1/dl/getContent/670531?fileName=//c10gx-51001-683485-670531.pd

[44] "URL: www.boost.org/."

[45] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series, Pearson Education, 1994.

[46] C. NA62, "2020 NA62 Status Report to the CERN SPSC," tech. rep., CERN, Geneva, 2020.

[47] "URL: www.na62-sw.web.cern.ch/index.html."

[48] "URL: www.globus.org/."

[49] "URL: www.fts.web.cern.ch/."

[50] "URL: www.mariadb.org/."

[51] P. Achenbach, C. Ayerbe Gayoso, J. Bernauer, R. Böhm, M. Distler, L. Doria, J. Friedrich, H. Merkel, U. Müller, L. Nungesser, J. Pochodzalla, S. Majos, S. Schlimme, T. Walcher, and M. Weinriefer, "Measurement of propagation time dispersion in a scintillator," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 578, pp. 253–260, 03 2007.

[52] "URL: www.crystals.saint-gobain.com/."

[53] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," *SIGMOD Rec.*, vol. 17, p. 109–116, jun 1988.

[54] "URL: www.linux.die.net/man/8/mdadm."

[55] "URL: www.dell.com/support/kbdoc/en-us/000139333/perc-performance-concerns-for-raid-controllers-without-cache-h330-h310-s130-s110-s300-s100-h200-sas-6-ir-sas-5-ir."

[56] "URL: www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf."

[57] "URL: www.fio.readthedocs.io/en/latest/fio_doc.html."

[58] "URL: www.thomas-krenn.com/en/wiki/Processor_P-states_and_C-states."