The data acquisition for the BGO-OD experiment

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Daniel Hammann
aus
Kirchen

Bonn, März 2016

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Hartmut Schmieden
2. Gutachter:	Prof. Jochen Dingfelder

Tag der Promotion:	31.5.2016
Erscheinungsjahr:	2016

Abstract

The BGO-OD experiment at ELSA was constructed to study meson-photoproduction. A main focus of the experiment is on associated strangeness and vector meson production. As these reactions are dominated by t-channel contributions their cross sections peak in forward direction. This motivates the two part detector design. While most of the angular range is covered by an electromagnetic calorimeter, the forward region is equipped with an open dipole spectrometer. This combination makes BGO-OD an unique setup. As the cross sections for interesting reaction channels are low compared to other hadronic reactions, the experiment was designed to operate at the highest possible rates.

For efficient operation this requires a sophisticated data acquisition system. This system was developed within this thesis. While the basic structure of the DAQ could be adapted from the Crystal Barrel experiment, significant modifications and extensions were necessary. To allow for an efficient data acquisition, the readout is distributed across several readout crates. A synchronization system is used to distribute the global trigger signal to all crates. A busy logic is used to ensure no new triggers are accepted until all crates are ready for the next event. To minimize the resulting dead time most of the readout is performed after the busy is reset. This allows to achieve dead times of approximately 50 µs. After readout the data is sent to a central event saver which assembles the data from all crates and saves them to disk. A dedicated control program manages the interactions between the readout and saver programs as well as the user interface.

With this setup the first data was taken in 2014. The data acquisition system operated at an event rate of approximately 850 Hz with a life time above 90 %. The data from the various detector components was investigated and found to agree with the expectations. On this basis hadronic reactions were reconstructed. The reactions $\gamma\mathrm{p} \to \mathrm{p}\pi^0$ and $\gamma\mathrm{p} \to \mathrm{p}\eta$ could be observed using only the BGO electromagnetic calorimeter. The reaction $\gamma\mathrm{p} \to \Lambda\mathrm{K}^+$ is more difficult to observe due to its significantly lower cross section. However, using combined data from all detector components, a kaon signal can be easily identified.

The BGO-OD experiment is now pursuing its physics program. The DAQ system introduced in this thesis allows this to be done efficiently while providing the flexibility to adapt to future developments.

Contents

1 Introduction

With the advent of particle accelerators in the 1950s many previously unknown particles were discovered. These particles where first classified according to their decays. It was found that some particles could be produced in strong interactions but decayed only via the weak interaction. This led to the introduction of a new additive quantum number, the strangeness. However the relationships between the discovered particles remained unclear until the introduction of the quark model by Gell-Mann[GM64] and independently Zweig[Zwe64]. The basic constituents in this model are the up, down and strange quarks following an SU(3) symmetry. Within this model all baryons are composed of three quarks while mesons are quark-antiquark pairs. The baryon ground states can be organized into a spin 1/2 octet and a spin 3/2 decouplet, shown in figure 1.1. The mesons can be grouped similarly.

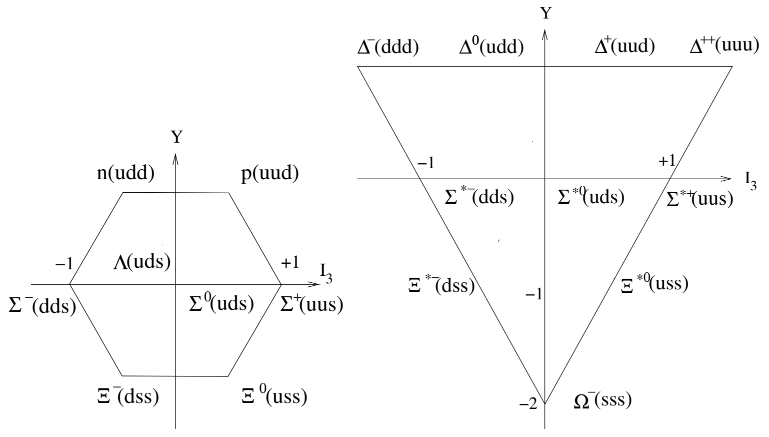


Figure 1.1: Baryon ground state spin 1/2 octet and spin 3/2 decouplet

With the exception of protons, all free hadrons are unstable. Baryons decay into lighter baryons emitting mesons, photons as well as leptons. Mesons also decay into lighter mesons, photons and leptons.

Photoproduction of mesons can be used to study the properties and interactions of these hadrons. Figure 1.2 shows the total photoabsorption cross section as well as the total cross sections for some possible final states. Several peaks are visible in the cross sections. These peaks are associated with the production of higher mass baryons, e.g. the Δ(1232) or excited states such as the $S_{11}(1535)$.

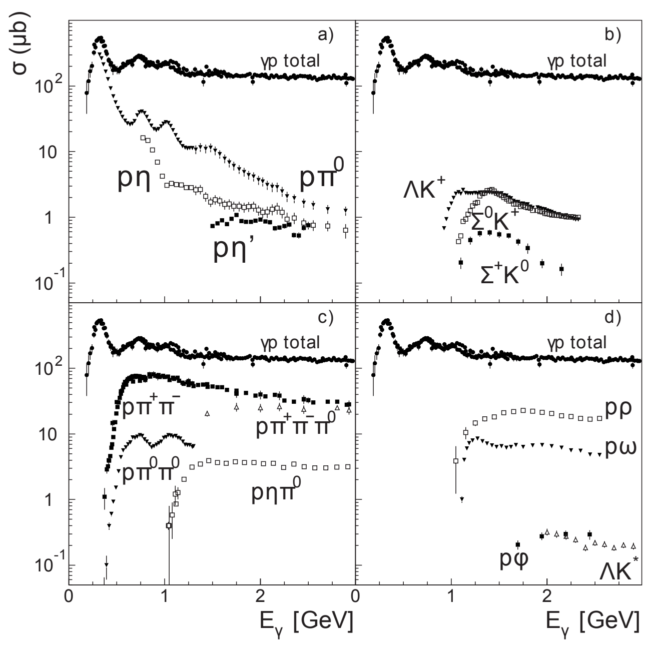


Figure 1.2: Total photoabsorption cross section and exclusive cross sections for single- and multimeson photoproduction off the proton[KR10]

Mapping the spectrum of excited baryons can provide information on the binding of the quarks inside hadrons. However, the separation between the excited states is similar to their widths. This causes the states to overlap and interfere with each other, depending on their quantum numbers. To disentangle these overlapping states additional information is necessary. This can be provided by using polarization degrees of freedom. Combinations of polarized targets, polarized photon beams and determination of recoil polarizations can be utilized.

Theoretical calculations of the excitation spectrum have proven difficult as well. Quark models [LMP01] as well as Lattice QCD calculations [EDRW11] predict many more states than have been observed so far.

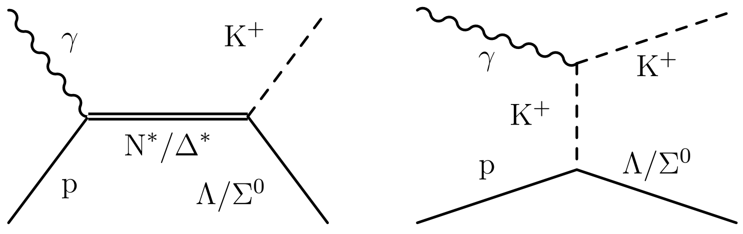


Figure 1.3: s-channel (left) and t-channel (right) diagrams for $\gamma \mathrm{p} \to \Lambda/\Sigma^0\mathrm{K}^+$

The situation is further complicated by the fact that the initial and final states of the reaction are not a unique tag of the intermediate state. Figure 1.3 illustrates two of the possible reaction mechanisms using the example of K^+ production. In the s-channel process the initial photon couples directly to the proton. This creates an intermediate state with an invariant mass equal to the square root of the Mandelstam variable s, providing the name for the mechanism. The intermediate state can be an excited baryon. This is different for the t-channel mechanism. In this case the momentum is transferred between the photon and the proton by exchange of a virtual particle. The square of this momentum is equal to the Mandelstam variable t. While excited baryons do not contribute in the t-channel mechanism it is possible to have contributions of dynamically generated resonances. These dynamically generated resonances arise from the interaction between baryon and meson[OR10]. They are typically expected just below the production threshold for the free baryon meson pair. Such a resonance might then be observed in the production of lighter mesons. For example a ΣK^* dynamic resonance might be detectable in ΣK close to the threshold of ΣK^*. Due to the involved t-channel mechanism small transverse momentum components are favored in this case. An effect of this may be visible in the production of K^0 at the K^* threshold where a step in the cross section was observed[E+12].

The BGO-OD experiment was designed to study these t-channel processes. This is reflected in the two part detector setup of the BGO-OD setup. As BGO-OD is a fixed target experiment, small transverse momentum components of the final state particles imply that at least one of the particles must be emitted in forward direction in the laboratory frame. To provide good resolution for charged particles an open dipole spectrometer covers this forward region. The spectrometer is accompanied by a BGO electromagnetic calorimeter covering most of the remaining solid angle, providing a high probability to detect all final state particles. This combination makes BGO-OD an unique setup.

From the wide range of reactions investigated at BGO-OD $\gamma p \rightarrow \Lambda K^+$ will be used to showcase the capabilities of this setup.

Measurements of the $\gamma p \rightarrow \Lambda K^+$ differential cross section exist from several experiments. Figure 1.4 provides an overview of the existing measurements. For polar angles of the K^+ in the center of mass system (CMS) between 25° and 155° measurements exist over a wide range of energies. However for the more extreme angles almost no data points are available. The range below angles of 25° is especially interesting for BGO-OD.

Figure 1.5 shows the differential cross section as a function of the K^+ angle in the center of mass system at two different photon energies. The cross section rises towards forward angles. Various model calculations are also shown in the figure. While they are in good agreement over most of the angular range they diverge for forward angles.

The BGO-OD setup is very well suited to study ΛK^+ in this kinematic region. Within the accessible energy range the CMS angle of 25° for the kaon is transformed to an angle of about 10° in the laboratory frame due to the Lorentz boost of the fixed target setup. This is within the acceptance of the forward spectrometer. However the detection of the K^+ is not enough to uniquely identify the reaction. The reaction $\gamma p \rightarrow \Sigma^0 K^+$ has very similar kinematics. To distinguish these two reactions the decay products of the Λ or Σ^0 need to be identified. The central electromagnetic calorimeter of the BGO-OD setup

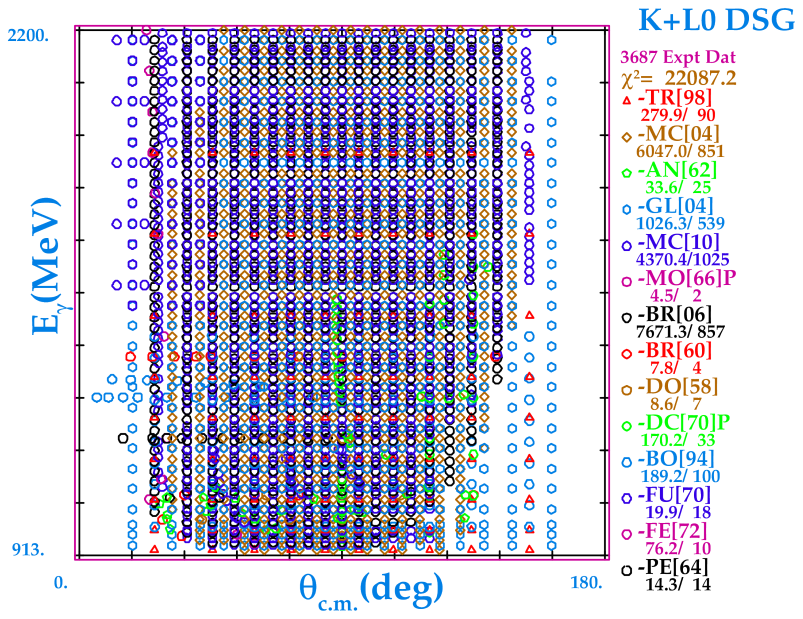


Figure 1.4: Differential cross section measurements used by the SAID[SAI] group for $\gamma p \to \Lambda K^+$ as a function of photon energy $E\gamma$ and K^+ polar angle Θ The different symbols, listed on the right, identify the experiments providing the data. For details see [SAI].

allows to detect neutral as well as charged particles over most of the angular region. This allows, for example, to distinguish the Σ^0 by detection of the additional photon from its decay $\Sigma^0 \to \Lambda\gamma$.

While the reaction $\gamma p \to \Lambda K^+$ is an interesting topic for the BGO-OD experiment, it is not the only subject studied. The setup is well suited for the investigation of other strangeness channels, including production of K^0 and excited hyperons, like the $\Lambda(1405)$, as well as the production of vector mesons. All of these reaction typically have complex final states due to the decays of the mesons or the hyperons. As these final states can contain both charged and neutral particles, the combination of the open dipole forward spectrometer and the BGO calorimeter is well suited to study them.

An additional topic pursued by BGO-OD focuses on the production of mesons off nuclear targets. Of specific interest here is the possibility to form bound states between the meson and the nucleus. This is planned to be studied using the reaction $\gamma^{12}C \to {}^{11}Bp\eta'$. However to form a bound state between the η' and the ^{11}B nucleus the momentum difference between the two must be small. This is only possible if the proton takes most of the photon momentum. The proton is then emitted within the acceptance of the forward spectrometer. To identify the η' its decay products can then be detected by the BGO calorimeter. This makes the BGO-OD setup uniquely qualified to search for these bound states.

Common to all these topics planned to be investigated by the BGO-OD experiment is that the cross sections are low compared to other processes. For this reason the BGO-OD experiment was designed to operate at the highest rates possible at the accelerator

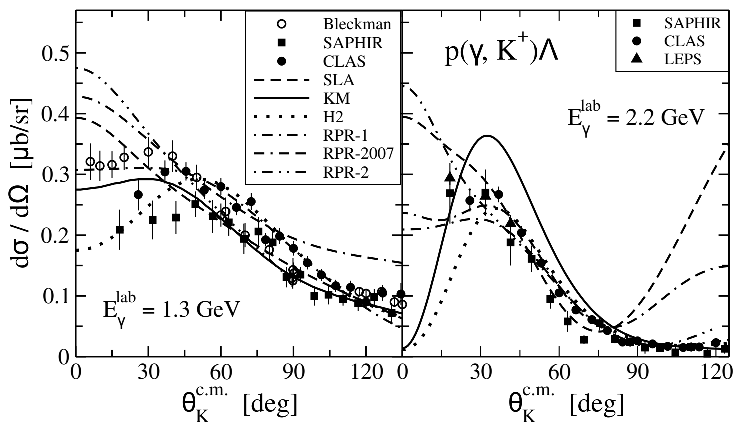


Figure 1.5: $K^+\Lambda$ cross section as function of the meson CMS angle[BS13]
Points indicate measurements by different experiments, see inset. The lines represent different model calculations. For details see [BS13].

facility ELSA. This high rate operation in turn requires a sophisticated data acquisition and trigger system to make efficient use of these rates. This data acquisition system is the main subject of this thesis.

Following the detailed description of the BGO-OD experiment provided in the following chapter, a brief section concerning general properties of any data acquisition system (DAQ) is given in chapter 3. Building on this the BGO-OD data acquisition and its performance are described in chapter 4. Preliminary data from the experimental setup are showcased in chapter 5. A summary and outlook is provided in the final chapter, 6.

2 The BGO-OD experiment

2.1 Overview

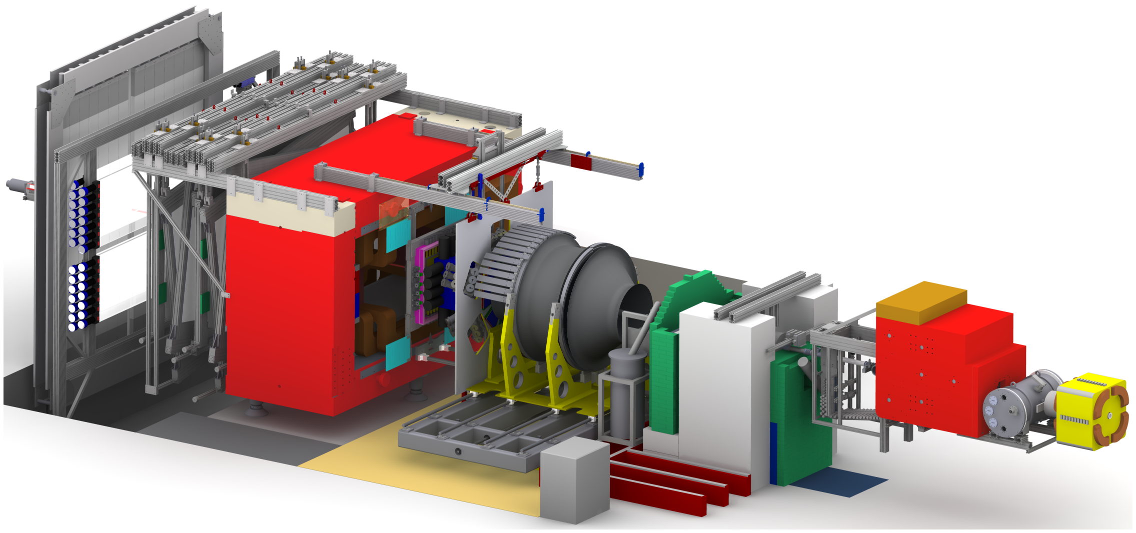


Figure 2.1: Overview of the BGO-OD setup

The detector setup of the BGO-OD experiment is used to investigate hadronic reactions induced by photons. An overview of the setup is shown in Figure 2.1. The setup can be grouped into three parts: the photon beam instrumentation, the central detector and the forward spectrometer.

The parts related to creating and monitoring the photon beam, as well as the accelerator ELSA, are discussed in chapter 2.2. The main components of this detector part are the tagging system where the photons are produced via bremsstrahlung on a radiator target and their energy is determined, the GIM and FluMo detectors for monitoring of the photon beam intensity and the liquid hydrogen target where the hadronic reactions take place.

The central detector surrounding this target is described in chapter 2.3. This detector part is used for measuring neutral as well as charged particles emitted at polar angles between 25° and 155° in the laboratory system. An MWPC and a barrel of thin plastic scintillators are used to identify charged particles and measure their directions. A crystal calorimeter, the Rugby Ball, is used to measure the energy as well as the direction of charged and neutral particles.

The forward direction is covered by a spectrometer setup (chapter 2.4) providing identification and momentum determination for charged particles. Momentum reconstruction is facilitated by a large acceptance open dipole magnet and tracking detectors. Particle identification can be achieved using time-of-flight measurements. The time-of-flight walls can provide additional detection efficiency for neutral particles.

Chapter 2.5 details how the measurements for the experiment are performed and which parameters limit the performance of the experiment.

2.2 Photon beam

2.2.1 ELSA

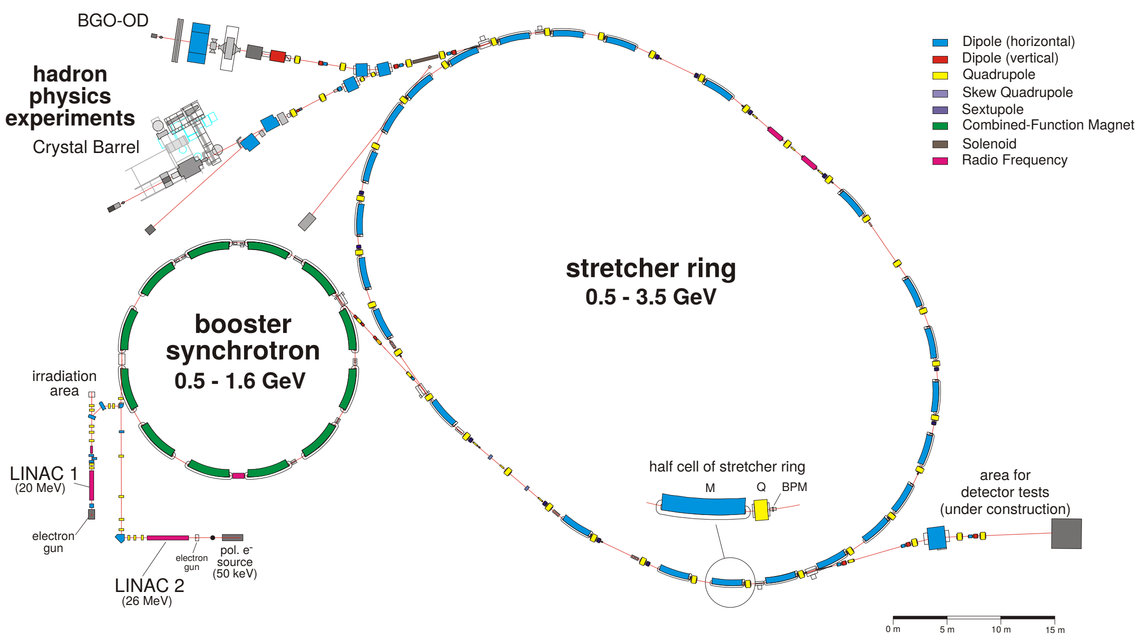


Figure 2.2: Overview of the ELSA accelelerator facility [F+13]

The ELSA[1] facility in Bonn is a three stage electron accelerator. Figure 2.2 provides an overview of the facility. The first stage of the accelerator is a LINAC[2] structure. For hadron physics experiments LINAC2 is used. It injects the electrons into the booster synchrotron with an energy of 26 MeV. The electrons are then injected into

[1]**El**ektronen **S**tretcher **A**nlage
[2]**lin**ear **ac**celerator

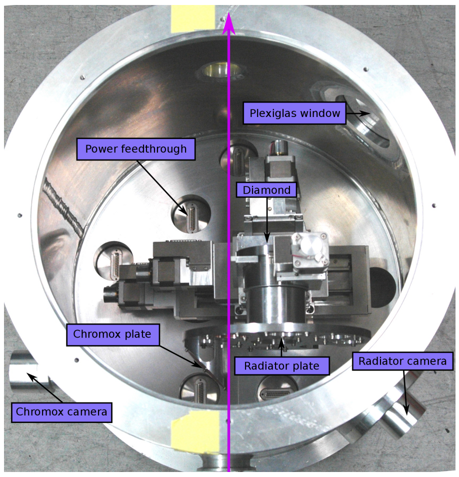


Figure 2.3: The goniometer with the radiator targets inside the vacuum tank [Bel11]

the synchrotron. After reaching an energy of 1.2 GeV the electrons are extracted from the synchrotron and injected into the ELSA stretcher ring. As the stretcher ring has a larger circumference than the synchrotron this step is repeated several times. When the stretcher ring is completely filled the electrons are accelerated to their final energy. Depending on the requirements of the experiment, energies up to 3.2 GeV are typically used. After reaching the desired energy the stored electrons are extracted to the experiment over several seconds. This cycle is then repeated when the stretcher ring is emptied.

This creates a so called spill structure. The experiment receives electrons for several seconds followed by a short break. Typical values for the duration of a complete cycle are around 5.5 s. Of these 5.5 s, 1.5 s are used for filling the stretcher ring and the acceleration of the electrons. The remaining time is available for extraction to the experiment.

2.2.2 Goniometer

As the BGO-OD experiment is investigating photoinduced reactions, the electron beam extracted from ELSA is sent to a radiator target. On this target some of the electrons produce photons via the process of bremsstrahlung.

Figure 2.3 shows the different available radiators and the goniometer to position them in the electron beam. The copper and diamond targets are used for experiments with unpolarized and linear polarized photons. For measurements with linear polarization the lattice structure of the diamond crystal has to be oriented with a defined angle to the beam axis to allow the recoil momentum of the bremsstrahlung process to be transferred to the lattice. To achieve linear polarization at any chosen photon energy the crystal needs to be rotated precisely around all three rotation axises, as described in [Bel11]. For this reason the crystal is positioned at the intersection of all three rotational axes of the goniometer system.

The chromox screen allows for a visual check of the beamspot at the radiator. For

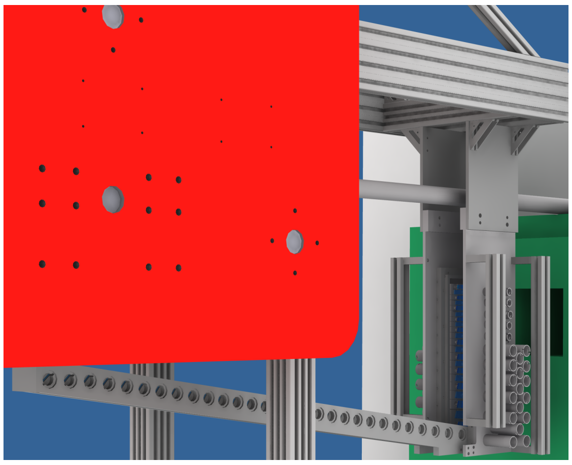


Figure 2.4: The photon tagging setup
The electrons, entering from the left, are deflected by the tagging magnet. Electrons which have created a photon have lost energy and are deflected into the tagging hodoscope, for energy determination. The remaining electrons are stopped in the beam dump.

a quantitative determination of the electron beam size and position, two 250 µm thin wires are available. These can be moved through the beam in vertical and horizontal directions. Measuring the detector rates during this process allows determination of the beam parameters.

2.2.3 Tagger

After passing the radiator target the electrons are deflected in a dipole magnet. The electrons that did not produce a photon are deflected into the beam dump. Electrons that produced a bremsstrahlung photon have lost energy in this process and are deflected more strongly. By measuring the deflection of those electrons it is possible to determine the energy of the photon by the simple relation:

$$E_\gamma = E_{\text{beam}} - E_{\text{e}^-}$$

The energy of the incoming electron beam, E_{beam}, is precisely known from the accelerator settings. The deflection of the electrons, and hence their energy E_{e^-}, is determined by an arrangement of scintillator detectors, the tagging hodoscope.

This detector array, visible in figure 2.4, consists of 120 individual scintillators. It covers a range of 10 % to 90 % of the incoming electron energy. The resulting energy resolution varies within the detector. For the highest detectable electron energies the resolution is 0.63 % of the primary beam energy, while for the lowest electron energies it is 1.56 % of the primary beam energy. For a beam energy of 3.2 GeV this gives an energy resolution ranging from 20 MeV to 50 MeV [Sie11, Bel12].

Due to the $1/E_\gamma$ shape of the bremsstrahlung spectrum, the scintillators corresponding to the lower photon energies need to detect a significantly higher rate of electrons than the ones for larger energies. This effect is further enhanced by the larger energy ranges for the individual detectors in the vertical part of the tagging hodoscope. Thus a high time resolution and rate capability is required to minimize ambiguities in the identification of the electron corresponding to a photon.

To suppress background from photomultiplier noise and particles not originating from the bremsstrahlung process at the radiator, e.g. neutrons, neighboring scintillators have an overlap of 55 %. A coincidence of two or three adjacent scintillators is therefore required for the identification of a bremsstrahl electron.

2.2.4 Target

To study hadronic reactions the photons impinge on a liquid hydrogen target at the center of the detector setup. The liquid hydrogen is contained in a cylindrical target cell of 6 cm length and 3 cm diameter[Rom12]. This cell is made of an aluminum tube and thin Mylar windows at the front and back. To further minimize reactions of the photons outside the target volume, the photon beam line from the bremsstrahl radiator to the hydrogen target is evacuated.

2.2.5 Photon beam monitoring

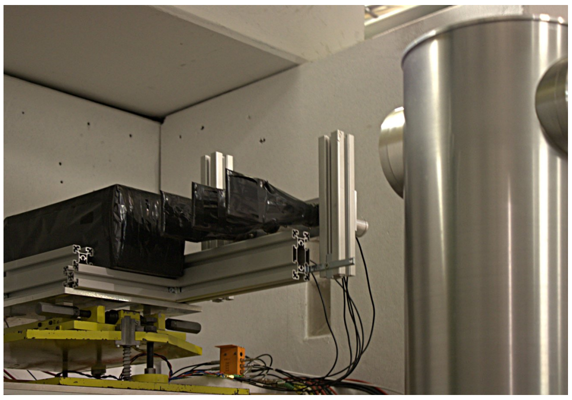


Figure 2.5: GIM, FluMo and Photon Camera [Zim12]
The picture shows the GIM to the left and the three FluMo scintillators in front of it. The Photon Camera housing is seen on the right side of the picture.

Detectors for monitoring the photon beam are located at the end of the photon beam line. The photon camera provides an online photon beam position measurement at the end of the experiment. This is achieved with a special scintillating sheet that is placed into the photon beam at a 45° angle. This sheet is observed with a CCD camera. The light distribution allows a measurement of the position of the photon beam in the horizontal and vertical direction.

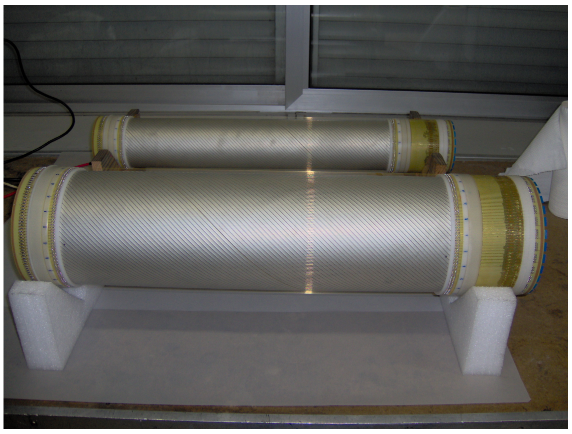


Figure 2.6: Photograph of the MWPC chambers during construction

The gamma intensity monitor, GIM, is a fully absorbing lead glass detector equipped with a single photo multiplier tube. The purpose of this detector is to count all photons passing through the experiment. This allows the determination of the photon flux, which is necessary to calculate the total cross sections of hadronic reactions.

At photon rates larger than 4 MHz the GIM detector is no longer able to detect all photons due to deadtime effects. To circumvent this problem an additional detector, called FluMo[3], is installed directly in front of the GIM, as shown in figure 2.5. The FluMo consists of three 5 mm thin scintillators. The purpose of this setup is to identify electron-positron pairs produced between the first two of the FluMo scintillators. This provides a significantly lower count rate compared to the GIM which still scales linearly with the total photon rate, allowing this detector to work without significant deadtime effects.

2.3 Central detector

2.3.1 MWPC

Surrounding the liquid hydrogen target are two cylindrical multi wire proportional chambers (MWPC). The two chambers, shown in figure 2.6, are used for the tracking of charged particles. While the wires run parallel to the beam axis, the cathode layers are segmented into strips tilted with respect to the beam axis. The inner and outer strips are tilted in opposing directions, resulting in two or three crossing points. This ambiguity is then resolved by the wire signals. Collecting the charge information of the cathode strips further increases the position resolution. An angular resolution of 2° is expected.

[3]**Fl**uss **Mo**nitor

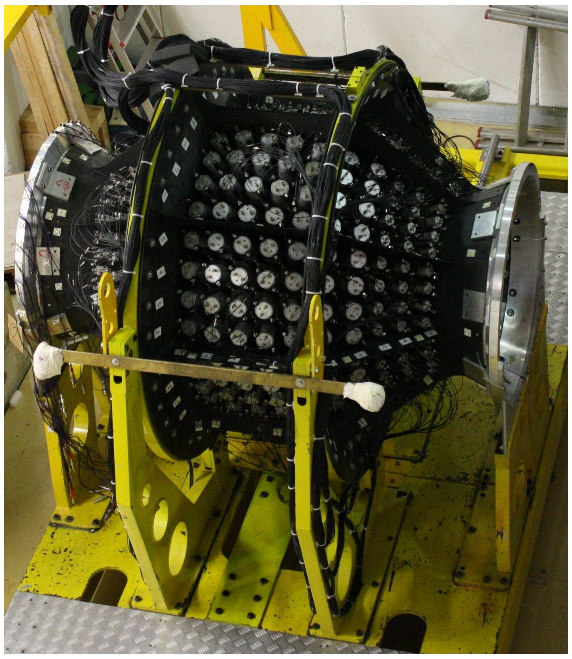


Figure 2.7: Photograph of the BGO Rugby Ball before cabling

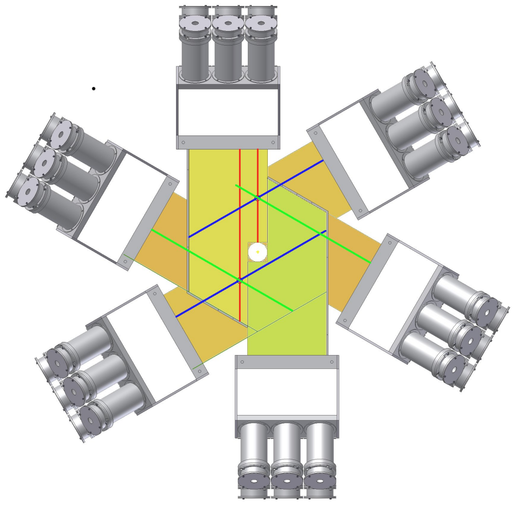
2.3.2 Barrel

A barrel of 32 scintillating bars is placed between the MWPC and the Rugby Ball. These bars provide energy loss information $\mathrm{d}E/\mathrm{d}x$. In combination with the total energy measured by the Rugby Ball it allows discrimination between protons and charged pions up to energies of 350 MeV. Additionally it provides the possibility to trigger on charged particles.

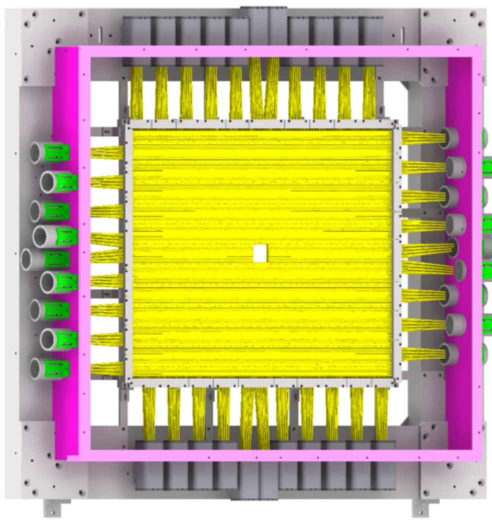
2.3.3 Rugby Ball

The main part of the central detector is the Rugby Ball crystal calorimeter. It consist of 15 crowns of BGO crystals. Each crown covers the complete azimuthal angle ϕ with 32 identical crystals. The polar angle θ is covered from 25° in forward direction to 155°. The crystals have a length of 24 cm, corresponding to approximately 21 radiation lengths. As for photons the electromagnetic shower spans several crystals, clusters of adjacent crystals are formed in the analysis. This allows to achieve a better angular resolution, as a weighted mean over the crystal angles inside the cluster can be used to calculate the photon angle. From simulations an angular resolution of $\approx 2°$ is expected for photons. The energy resolution of the detector scales with the energy deposited into the crystals. At an energy of 1 GeV it is $\sigma(E)/E = 0.94\,\%$ [Z+92].

Radioactive sources are used for the initial energy calibration and equalization of the photomultiplier gains. The ^{22}Na sources used emit, in addition to positrons from β^+ decay, photons with an energy of 1.275 MeV from the subsequent electromagnetic decay. The high voltages for the individual crystals are set, so that the photon peaks in the



(a) MOMO



(b) SciFi2 [Bö15]

Figure 2.8: Technical drawings of the MOMO and SciFi2 detectors.

ADC spectra appear at the same position. A first energy calibration for the ADCs is provided with this technique. To achieve the dynamic range required for hadron physics experiments, the signals are attenuated by a constant factor during data taking.

2.4 Forward spectrometer

The forward spectrometer covers the forward angular range up to $\pm 8°$ in the vertical direction and $\pm 12°$ in the horizontal direction. It is an open dipole spectrometer with scintillation fiber tracking in front of the magnet, and drift chambers behind the magnet. Particle identification is provided by a comparison of the time of flight, provided by scintillator walls at the end of the setup, and the measured momentum.

2.4.1 Magnet

The spectrometer magnet is a large open dipole. It has an opening of 84 cm in vertical direction and 150 cm in horizontal direction. At the maximum current of 1340 A, the magnetic field reaches values larger than 0.5 T in the central region. It has a bending power of approximately $\int Bdl \approx 0.71\,\text{Tm}$.

2.4.2 MOMO and SciFi2

Tracking in front of the magnet is provided by two scintillating fiber detectors. The first of the tracking detectors, MOMO, is placed 113 cm behind the target. It consists of three layers of 224 fibers each, with a 60° angle between the layers. The fibers have a diameter of 2.5 mm. This allows for an unambiguous reconstruction of particle tracks

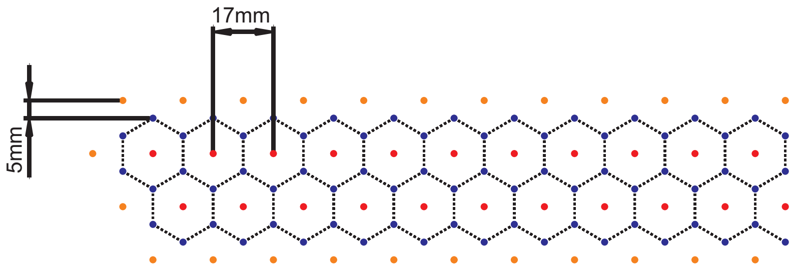


Figure 2.9: Drift cell geometry
The hexagonal drift cells (dotted lines) are arranged in a double layer structure. While the field wires (blue) are at negative high voltage, the sensitive wires (red) are at ground potential. Surrounding the drift cells additional field wires (orange) at ground potential are placed to ensure a well defined field configuration inside the drift cells.

for two particles passing the detector. The overlapping region of the fibers provides a hexagonal sensitive area with a diameter of 44 cm. A 4.5 cm diameter hole in the center of the detector allows the photon beam to pass through the detector[Joo96].

The SciFi2 detector has rectangular sensitive area of 66 cm by 51 cm. SciFi2 has one plane of 288 horizontal fibers and one of 352 vertical fibers. In the center a hole of 4 cm by 4 cm is left for the photon beam. The fibers have diameter of 3 mm. The overlapping arrangement of the fibers guarantees a minimum path length of 2 mm through the sensitive material. The Hamamatsu H6568 16 channel photo multipliers used to read out the detector are extensively shielded against the magnetic field of the open dipole.

2.4.3 Drift chambers

Behind the magnet, tracking is performed by eight planar drift chambers. Figure 2.9 shows the double layer structure of each drift chamber. To provide 3-dimensional track reconstruction the chambers are installed with different wire orientations. Two of the chambers have vertical wires and two chambers have horizontal wires. The remaining four chambers are tilted by $\pm 9°$.

The drift chambers cover a sensitive area of 2.4 m times 1.2 m. The central six wires of each chamber have an insensitive spot of 5 cm length at the position of the photon beam. This is realized by additional gold plating of the wires up to a thickness of 200 µm.

2.4.4 ToF

Placed at the end of the spectrometer setup are two large scintillator walls. These detectors are used for time-of-flight measurements. This allows for the identification of charged particles by measuring their velocity.

The first wall, called ToF3 is constructed of 8 horizontal scintillator bars, measuring 340 cm $\times$ 21 cm $\times$ 6 cm. The second wall, ToF4, is made of 14 vertical scintillator bars, measuring 270 cm × 20 cm × 4.5 cm[Mei13]. The bars are read out on both sides, compensating for the light propagation time inside the detector.

2.5 Measurement conditions

To study meson production reactions with the BGO-OD setup it is necessary to identify and measure the particles in the final state of the reaction. For the extraction of reaction properties, such as cross sections, angular distributions and polarization observables, a large number of reactions need to be observed. As the reconstruction of the reaction from the data provided by the detectors is very complex the full data needs to be stored for each reaction.

This is accomplished by using a trigger logic to detect reactions of interest and start the readout of the detectors. The set of data recorded for a trigger pulse is called an event. Each event contains all data of the reaction that caused the trigger logic to initiate the readout. Due to the high rates in the detectors, a single event can contain additional hits not related to the reaction which caused the trigger. These hits have to be filtered in the data analysis.

To estimate the expected event rate for the experiment, the total rate of hadronic reactions within the target can be used. This rate is determined by the rate of photons reaching the liquid hydrogen target and the target size. The maximum photon rate is determined by the tagger design value of 50 MHz for the full tagged range. As the hadronic cross section is energy dependent, the energy dependence of the bremsstrahlung photons has to be taken into account. It can be approximated by the formula

$$k\frac{\mathrm{d}\sigma_i}{\mathrm{d}k} = \left[1-(1-k/E_0)^2\right]\psi_1^i(\delta) - \frac{2}{3}(1-k/E_0)\,\psi_2^i(\delta) \tag{2.1}$$

given for the Bethe-Heitler cross section in [Tim69] dependent on the photon energy k and the electron beam energy $E_0 = 3200\,\mathrm{MeV}$, where the functions $\psi_1^i(\delta)$ and $\psi_2^i(\delta)$ have been approximated by constant values. For the bremsstrahlung part on the nucleus the values $\psi_1^i = 14.1$ and $\psi_2^i = 13.2$, for the part on the shell electrons $\psi_1^i = 4.1$ and $\psi_2^i = 4.0$ have been used. The absolute values have then been normalized to provide a rate of 50 MHz in the tagged range.

As not all produced photons reach the hydrogen target a reduction factor of 0.7 has been used for the photon flux. Using this photon flux, the total γp hadronic cross section from [B$^+$12] and the target parameters, a total hadronic rate close to 2000 Hz has been estimated.

As the data acquisition system needs time to process each event, the so called dead time, it cannot process all trigger attempts. The percentage of the total run time the DAQ is able to accept trigger attempts, the so called life time l, can be calculated according to the formula

$$l = \frac{1}{1+fd} \tag{2.2}$$

where f is the trigger attempt rate and d the average dead time. Figure 2.10 shows the calculated life time for a fixed trigger attempt rate of 2000 Hz. This life time directly

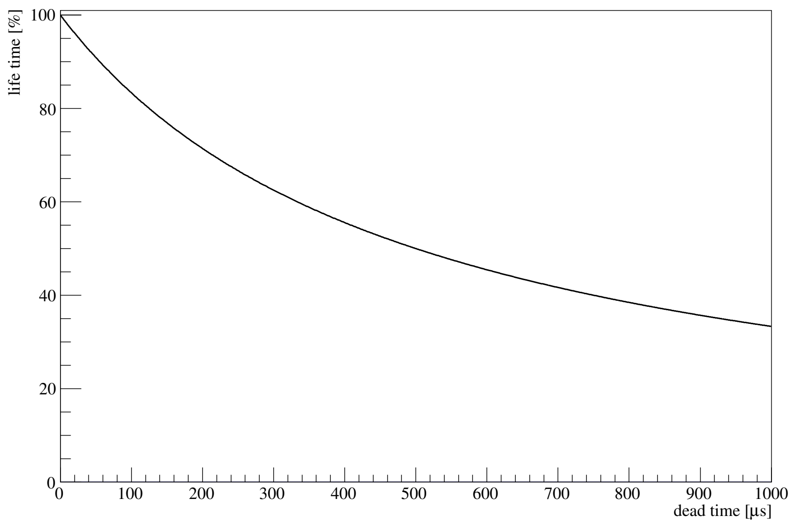


Figure 2.10: Calculated life time as function of dead time for a trigger attempt rate of 2000 Hz

affects the efficiency of the experiment. To achieve a life time better than 66 % the average dead time has to be below 250 µs.

3 Data acquisition systems

The task of a data acquisition system (DAQ) in particle physics experiments, such as BGO-OD, is to store the information about the investigated physical reactions provided by the detectors for a later analysis. The general data taking procedure used at the BGO-OD experiment is already briefly described in the previous section 2.5.

Section 3.1 discusses in more detail the modules used to digitize the detector signals. The considerations for the trigger system needed for the readout of most modules are given in section 3.2.

3.1 Digitization

To analyze the data provided by particle detectors using computers it is necessary to convert the analog pulses provided by the detectors to digital information. The shape of these pulses is typically defined by the characteristics of the used detector and electronics. At BGO-OD and similar experiments the most important information contained in the pulses is their arrival time and the charge integral. With this information it is possible to calculate the time a particle passed through the detector and the energy that was deposited. Additional characteristics such as the shape of the pulse may provide additional data, e.g. for particle identification. However this is difficult to analyze and used only for some detectors.

3.1.1 Scaler

The most basic type of digitization module is a scaler. This type of module simply counts the number of pulses within a known time interval. With this information pulse rates can be calculated. However the scaler does not provide event by event data, contrary to the other modules described below. This implies that it is not possible to require correlations between the pulses in the data analysis.

3.1.2 Time to digital converter, TDC

The time information is measured by a time to digital converter, TDC. As the flight times of particles through the detector system are several nanoseconds, it is often necessary to measure time differences with sub-nanosecond resolution.

These time resolutions can be achieved using a high frequency clock and a delay locked loop (DLL). When a signal is detected the state of the DLL and the number of the clock cycle are stored into a hit buffer. This allows to detect an arbitrary number of hits in short time frames. However, if two hits are too close together the new hit cannot be processed and will be missed by the logic. This time in which the TDC module is unable to process new hits is called dead time.

To compare the times measured with this method the used TDC modules must be synchronized. The most simple solution is to have a so called reference channel for each TDC module. On this channel the same signal is sent to all TDC modules. All measured times are then calculated as a difference to the arrival of this signal. This causes any jitter between the reference signals on the different modules to directly affect the time resolution. An additional disadvantage is that for each TDC module, one channel has to be used as reference channel, reducing the number of channels available for detector signals.

An alternative method of synchronization is to distribute a clock signal to all TDC modules providing a stable correlation between the clock states. Due to the difficulty of distributing a very high frequency clock over distances of several meters only a slower clock is distributed. The high frequency clock can then be regenerated on the TDC modules by using a phase locked loop (PLL). As this synchronization method depends on keeping the internal states of the TDC chips synchronized, it can only be used for synchronizing identical TDC modules.

A combination of the two described synchronization methods can also be used. For this the reference channel has to be provided only for one channel of each TDC type. TDCs of the same type are then synchronized by a common clock.

3.1.3 Charge to digital converter, QDC

In addition to the time a detector was hit, the deposited energy is often of interest. This information is typically contained in the charge of the analog pulses generated by the detectors. A traditional charge to digital converter (QDC) works by charging a capacitor during a time window called a gate. The capacitor is then discharged with a constant current to measure the accumulated charge.

A small offset current to the input signal may be introduced to ensure that the zero point of the module is always at a well defined position. This so called pedestal is determined by integrating a zero signal and measuring the response of the QDC. The offset current is then adjusted for a small positive pedestal value. This value is then subtracted from future measurements.

While this method allows the measurement of even very small amounts of charge precisely it has several drawbacks. First, the gate signal must have a fixed time correlation to the analog signal to guarantee the integration of the total charge. It is therefore derived from the analog signals. As this delays the gate signal, the detector signals have to be delayed as well. This requires either a very fast trigger logic or leads to very large amounts of delay cables. These delay cables can cause additional noise pickup and signal degradation.

Secondly, the QDC is very dependent on a constant baseline for the signal. Even very small variations can cause significant effects as they are integrated over the whole gate length.

Thirdly, at high rates additional pulses may occur during the time window of the gate, the so called pileup. As the total charge accumulated during the gate is the only information provided by a QDC it is very difficult to detect pileup and correct for it.

Finally, this type of QDC requires a dead time after the end of the gate for the charge conversion. This limits the usability of this type of module.

3.1.4 Sampling analog to digital converter, SADC

The sampling amplitude to digital converter (SADC) is a modern replacement for the QDC. The concept of this module type is to continuously sample the incoming analog signal height. The samples are then digitally stored and processed. Pulses are typically detected by applying a threshold to the samples. The charge information is then determined by summing up all samples belonging to a pulse. By fitting the leading edge of the pulse, the pulse start time can be reconstructed allowing the SADC to be used as a TDC as well. The possible resolution depends on the sampling frequency and the pulse shape. For sufficiently high sampling frequencies it can be similar or better than what is typically achieved using conventional TDCs. Further analysis of the pulse allows the detection of pileup pulses and to separate the two pulses. The samples obtained before and after a pulse contain information on the current baseline, which can then be subtracted. This removes most of the problems associated with a classic QDC.

Due to the high data rate produced by continuously sampling the input, the processing of the pulse is often done directly on the module. Only the computed pulse properties are then propagated to the data acquisition.

As the electronics necessary for continuous sampling of the input signals at a sufficient rate is complex, SADC modules are typically significantly more expensive than QDC modules.

3.2 Trigger system

The trigger system is used to decide which parts of the data provided by the detectors should be recorded. This data is split into events, which contain all data correlated in time with the trigger decision. For optimal data taking efficiency, each of these events should contain the information related to one reaction studied. In addition the number of interesting reactions missed should be minimized. Such a trigger system can be realized as a high level or a low level trigger.

A high level trigger system operates on the data continuously digitized from the detector signals. This allows for detailed analysis of the data in the trigger decision process. With this the event selection criteria can be defined very strictly, enabling efficient investigation of extremely rare reactions. The disadvantages of a high level trigger system are the requirement to use readout modules capable of continuous operation, e.g. preventing the use of standard QDC modules, and the bandwidth and processing capacity necessary to operate on the continuous data stream.

A low level trigger system operates on the analog detector data. For this the detector signals are typically split into two paths. One path is sent to the readout modules. The other path enters the trigger system. The trigger system then generates a trigger decision based on these signals. This is then sent to the readout modules to initiate the digitization and readout. The time required for a trigger decision has a direct impact on the amount of delays needed in the readout signal paths. As this delay may cause additional degradation of the analog signals it should be kept as short as possible. Typical low level trigger decision times are therefore well below 1 µs.

A combination of high and low level trigger systems can also be used. In this case the low level trigger is used to start the digitization process, while the high level trigger filters the events before they are recorded.

4 The BGO-OD data acquisition

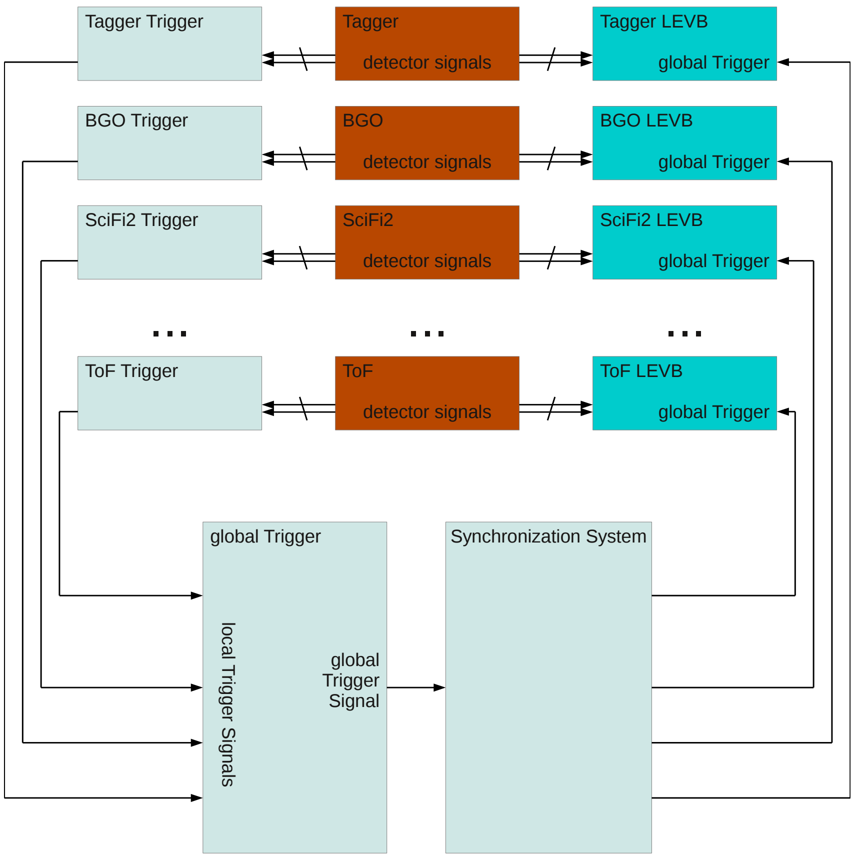


Figure 4.1: Data flow within the DAQ
The signals from the different subdetectors (brown) are split into a path to the readout modules (blue) and a path to the local trigger logics (light blue). The local trigger signals then are sent to the global trigger modules. The global trigger signal generated from the local triggers is distributed by the synchronization system to the readout modules initiating the data readout.

The data acquisition system or DAQ, for an experiment of the size of BGO-OD is necessarily complex. As such the BGO-OD DAQ was not developed from scratch, but derived from the existing DAQ of the Crystal Barrel experiment [Hof, Sch04].

Although it was significantly modified and extended it shares the same overall design. To adequately manage the data provided by the different subdetectors the system features a distributed readout, as well as a low level global trigger. As illustrated in figure 4.1 this global trigger receives inputs from local triggers which process the signals provided by a subdetector. The output of the global trigger is then distributed to the readout modules by a synchronization system. The readout hardware is located in several crates, each containing its own CPU. The crates, including the readout modules, as well as the acquisition software running on the CPUs are referred to as LEVBs.

Figure 4.2 schematically shows the LEVBs located in the experimental area and the components of the DAQ located outside. For storage, the data is transmitted from the LEVBs to the event saver evs. The interaction between the LEVBs and the event saver is coordinated by a central runcontrol program. This also provides the communication with the user interface daqUI and the run database runDb.

The following detailed description of the data acquisition system is organized according to the flow of the detector data. It starts with the description of the trigger system in section 4.1 and the synchronization system distributing the trigger in section 4.2. The LEVBs are discussed in section 4.3. The event saver is described in section 4.4. Section 4.5 discusses the runcontrol program and the run database. The description of the user interface in section 4.6, is followed by the description of the database runDb in section 4.7. The last section of this chapter, 4.8 is dedicated to the performance achieved in the BGO-OD setup.

4.1 The BGO-OD trigger system

The trigger logic for the BGO-OD experiment is split into two parts. First local trigger signals for each participating subdetector are created. The second step of the trigger logic is contained in the global trigger module. This module receives the local trigger signals of the subdetectors. If the trigger condition is met the trigger latch is set and the global busy is asserted. This prevents any further generation of trigger signals. When all LEVBs are ready for the next trigger, the latch is reset and the system can accept the next trigger.

4.1.1 The local triggers

A short overview of the subdetectors capable of providing a local trigger and their trigger conditions is given in table 4.1. These triggers indicate that a particle has been seen by the corresponding detector. The trigger conditions vary between the different subdetectors according to detector type and geometry.

Tagger The tagger trigger is implemented on two Spartan3 FPGA modules. These modules form the coincidences between adjacent scintillators, see figure 4.3. The modules allow for timing adjustment to compensate for differences in cable lengths and flight times. The trigger output is then determined by the `OR` of all coincidences. As the coincidences as well as the `OR` are implemented in clocked logic the output signal can

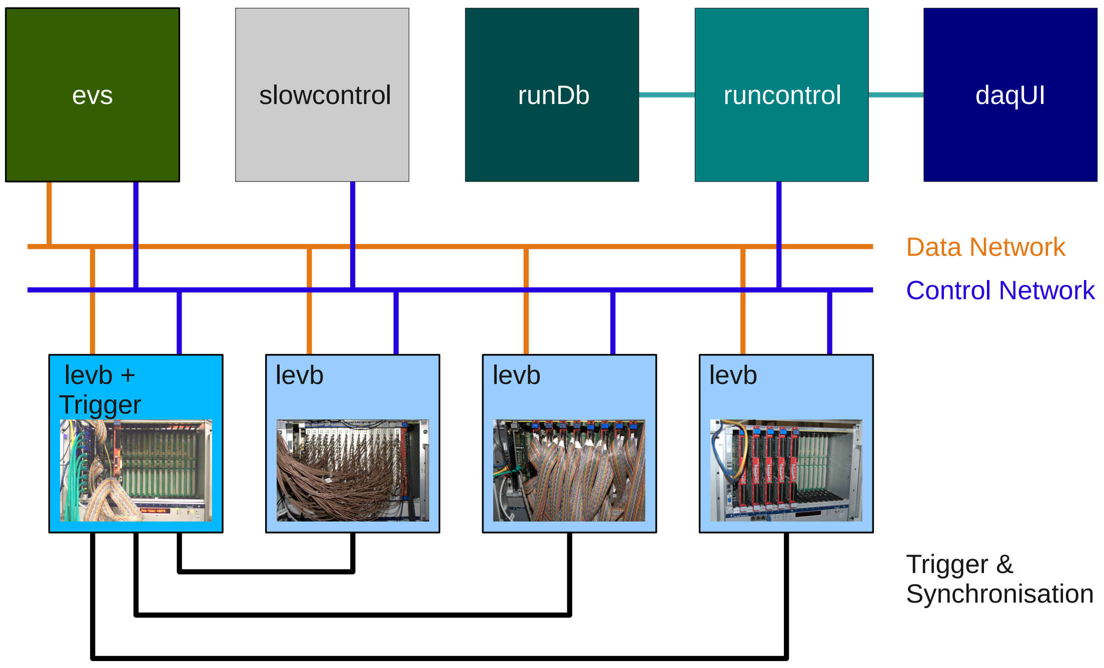


Figure 4.2: Overview of the DAQ components
The DAQ system is split into the event saver evs, the runcontrol, the user interface daqUI, the database runDb and the readout machines called LEVBs. The daqUI allows the user to control the DAQ while the runDb provides a log of all runs acquired. They have no direct connection to the readout machines, but only connect to the central runcontrol. The runcontrol and the remaining computers are connected using two Ethernet networks. The data network is used for the sending the detector data from the LEVBs to the event saver. The control network carries the control commands between the different components and provides an interface to the slowcontrol. The trigger information is sent using dedicated synchronization lines from the LEVB containing the global trigger modules to all other LEVBs.

only occur in fixed intervals. The output can be generated at six steps relative to the 200 MHz clock, providing a step with of ≈ 833 ps.

GIM As the GIM detector has only one single channel the trigger is simply implemented using a discriminator. A discriminated signal of one of the FluMo scintillators is used for tagger time alignment data.

Scintillator Barrel The trigger condition for the Scintillator Barrel inside the BGO is given by an `OR` of all 32 channels. This is implemented using the `OR` output of two LeCroy 3420 CAMAC constant fraction discriminators. For convenience, the outputs of both discriminators are sent to the global trigger module to be `OR`ed in the global trigger logic.

Subdetector	Trigger Condition	Implementation
Tagger	coincidence of adjacent channels	FPGA module
GIM	detector hit	NIM discriminator
Scint. Barrel	`OR` of all channels	NIM logic
SciFi2	`OR` of all channels	FPGA modules & NIM logic
Rugby Ball	Energy Sum	analog sum & NIM discriminator
ToF	mean-time of bar signals	FPGA module

Table 4.1: List of subdetectors providing local trigger signals

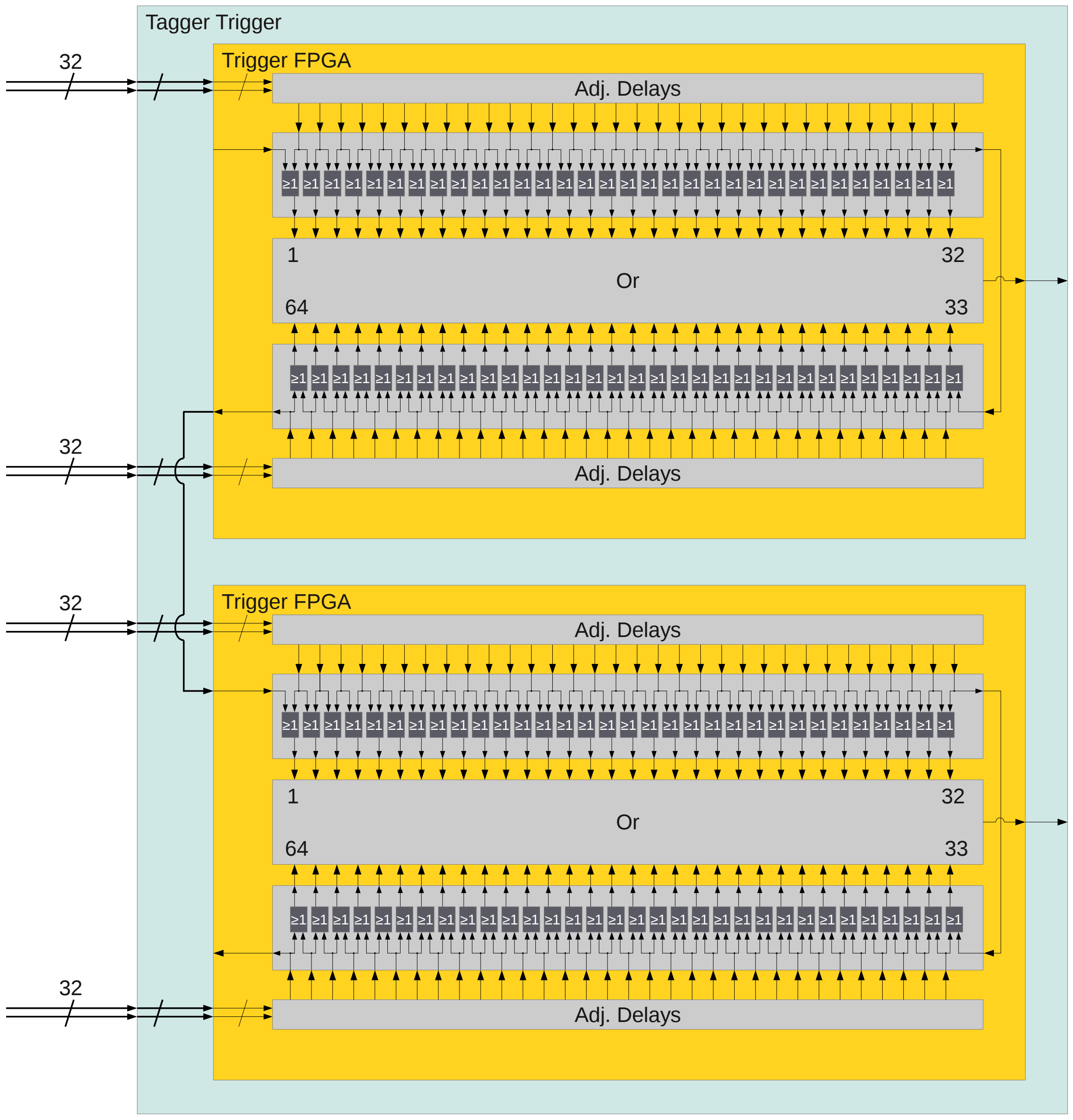


Figure 4.3: Schematic of the Tagger Local Trigger

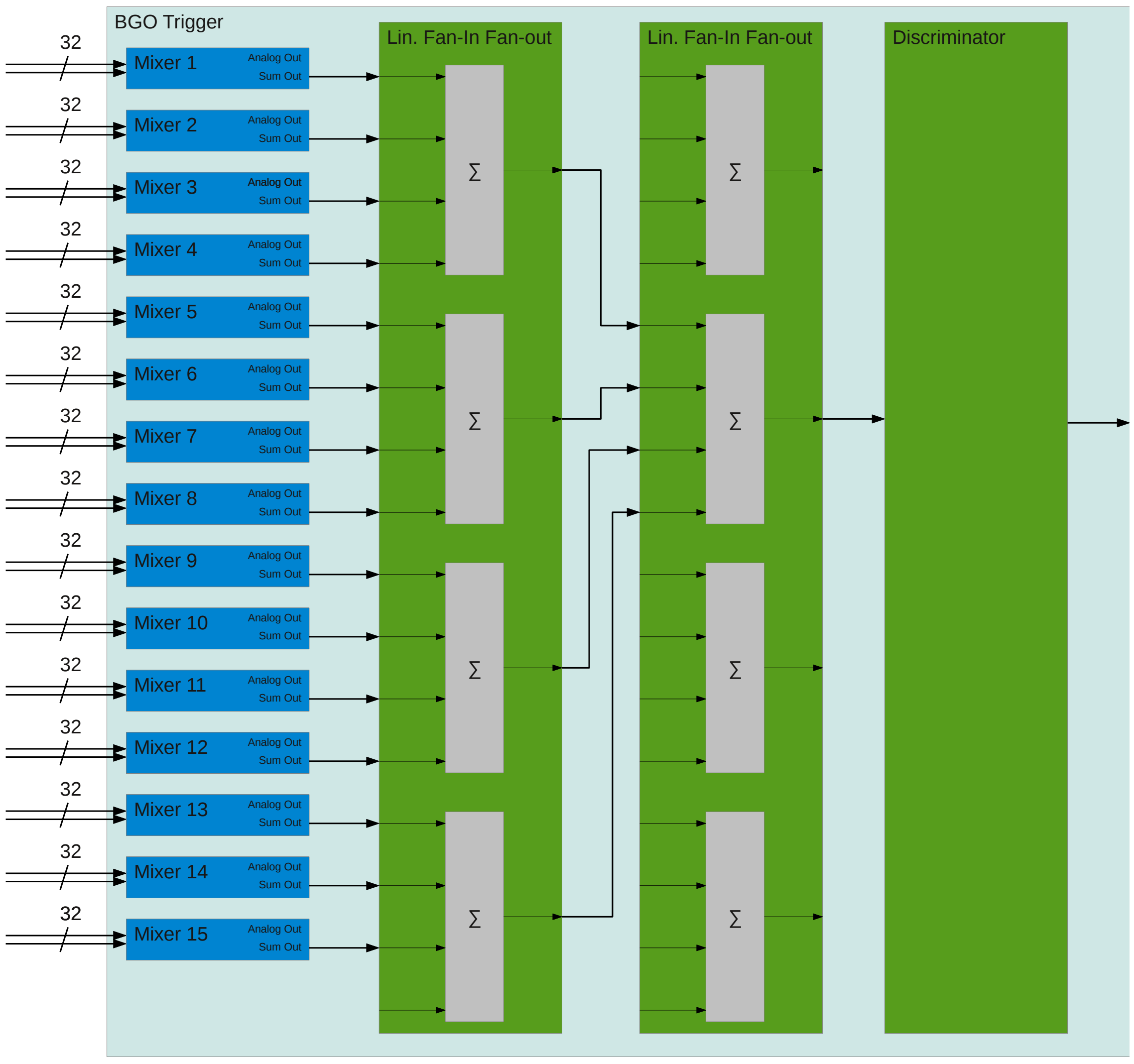


Figure 4.4: Schematic of the BGO Local Trigger

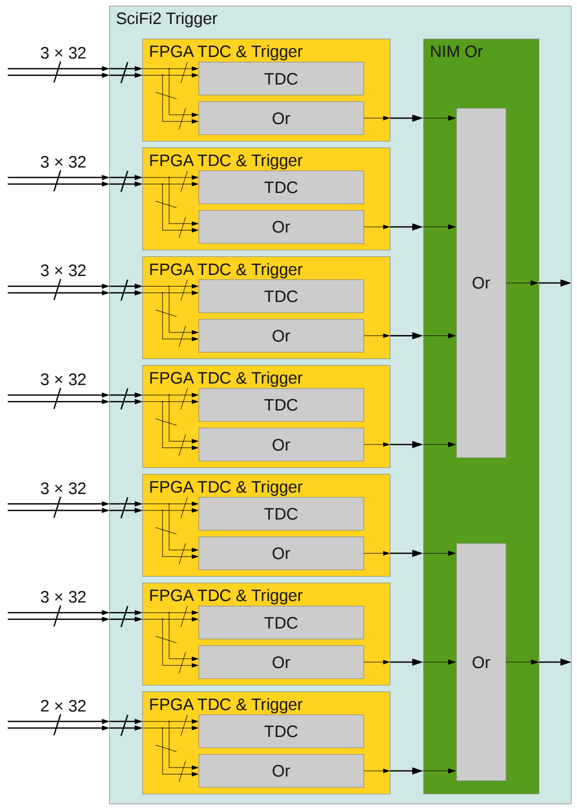


Figure 4.5: Schematic of the SciFi2 Local Trigger

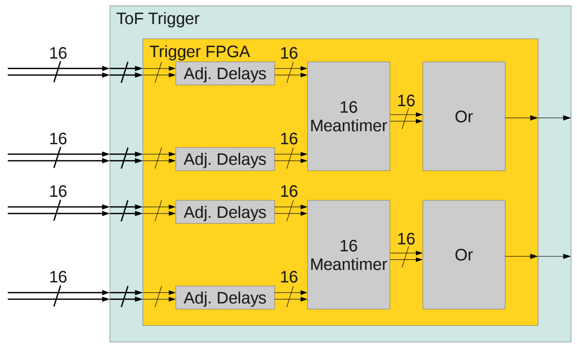


Figure 4.6: Schematic of the ToF Local Trigger

Rugby Ball The trigger condition for the Rugby Ball is given by a threshold on the total energy deposited in the calorimeter. This is realized by building the analog sum of all analog signals of the detector and discriminating on the amplitude, illustrated in figure 4.4. To create the analog sum first the signal of each crown of 32 channels is summed in the so called mixer modules CAEN SY493. The 15 sums are then combined using two stages of LeCroy 428F Linear Fan-In Fan-Out modules. The analog output of the second Fan-In Fan-Out module is then passed to a discriminator. The threshold of the discriminator is set according to the required energy deposit in the Rugby Ball, typically around 120 MeV.

SciFi2 A trigger on the SciFi2 scintillating fiber detector is implemented as an `OR` of all fibers. To avoid unnecessary signal splitting a trigger output was added to the FPGA module containing the TDC firmware, illustrated in figure 4.5. This output gives the `OR` of 96 channels. To create the `OR` of all 640 channels the signals are then `OR`ed using NIM modules.

The light pulser system of the SciFi2 can produce an additional trigger signal in coincidence with its LEDs.

ToF The trigger condition on the ToF detector is given by a hit in any of the scintillator bars. As the bars are read out on both sides, the meantime between the two signals is used to reduce noise and improve the trigger timing. Figure 4.6 shows a schematic of the trigger implementation inside a Spartan3 FPGA. Using the adjustable delay timing differences between the different channels can be compensated. For each of the ToF walls a separate output signal is generated and sent to the global trigger module.

4.1.2 The global trigger

The global trigger module combines the information provided by the individual local triggers. The global trigger module is implemented as a VME FPGA board. The board provides 32 LVDS input channels for the local triggers, as can be seen in figure 4.7. Before the input signals are passed to the logic part of the trigger module, the signals are sampled with a resolution of 800 ps. This is achieved by using regular structures on the FPGA, known as carry chains, as tapped delay lines. The sampled signals are then processed with clocked logic. Using shift registers the signals can be delayed by up to 375 ns in steps of the sampling resolution. In addition to the configurable delay, the gate length for all inputs can be set independently of the input signal length. This allows the adjustment of the trigger timings by software and removes the need for additional delay cables in the trigger signal paths.

The delayed signals are then analyzed in the logic part (CreateDataTrigger) of the trigger module. To allow for a flexible setup covering the interesting trigger combinations, the logic is separated into two steps. In the first step, the so called primary logic, 16 individual logic blocks can be configured. These blocks each take the 32 input signals and build the logical `AND` or the logical `OR` on any subset of the inputs. In addition, inputs may be declared as `VETO` inputs to suppress the output of the combination. A

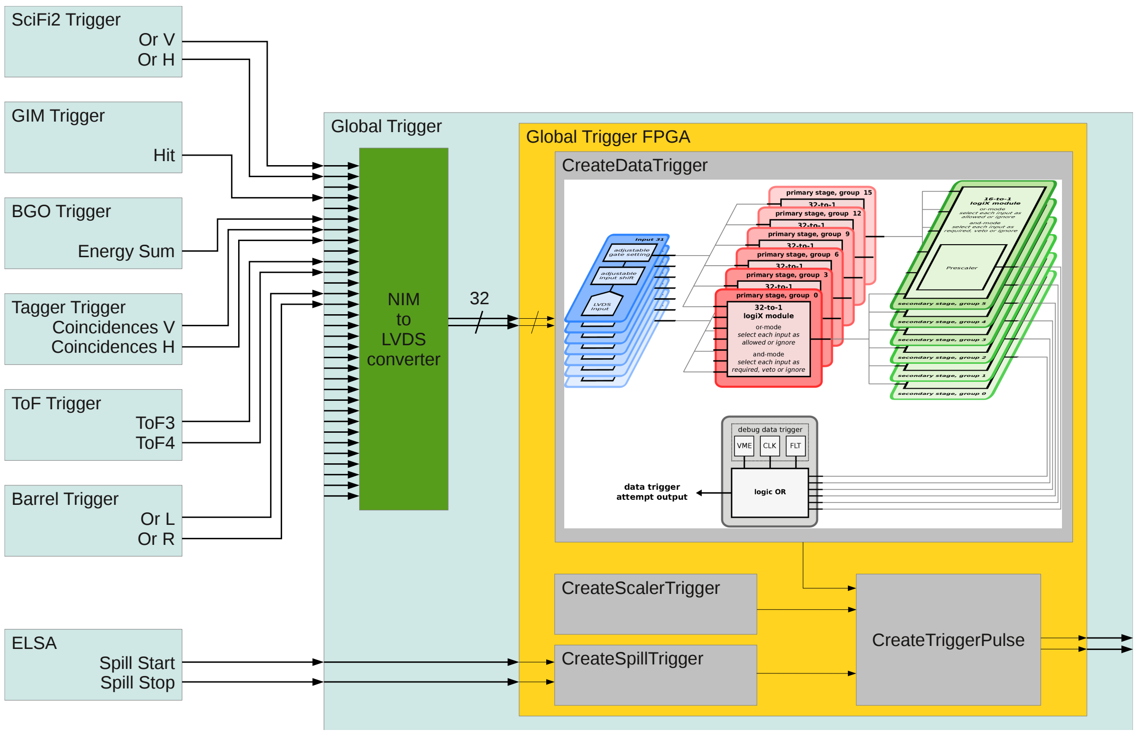


Figure 4.7: Overview of the trigger system and the global trigger module [Bie12]

second stage of logic blocks takes the output of the primary logic blocks and processes them in the same way as the primary logic processes the input signals. These 6 secondary logic blocks allow for the creation of independent, complex trigger conditions. The output of the secondary logic blocks is then prescaled and `OR`ed to generate the final trigger attempt.

Apart from these data trigger attempts, the global trigger module features several other trigger possibilities not based on the detectors. An integrated clock and fixed life time trigger provide minimum bias triggers for studying the behavior of the detectors as well as the data acquisition system. These are then included in the `OR` of the secondary logic blocks. A special scaler trigger configured to a rate of 20 Hz is used to initiate the readout of the scaler modules. The start and end of spill signals provided by the accelerator are also used to generate trigger signals. These are used for automatically changing parameters during special runs. These scaler and spill triggers are identified in the signal distributed through the synchronization system, allowing all LEVBs to act accordingly. As the special triggers must not be discarded they are queued instead of ignored if they occur during the dead time. They are then sent as soon as the DAQ can handle the next trigger.

The Global Trigger module provides built-in scalers for all input channels as well as the outputs of the secondary trigger logic blocks. This allows the continuous monitoring of the rates of the incoming signals and of the trigger attempts. In addition, the module has 64 LVDS outputs which are connected to a CAEN V1190 TDC. 32 of the outputs are a pass-through of the unmodified input signals, preserving the original timing. The remaining channels are used for the prescaled output of the secondary logic blocks as well as the final trigger decision.

4.2 The synchronization system

The synchronization system is responsible for ensuring a synchronized readout of all distributed readout machines. As such it is an integral element of the BGO-OD DAQ. The overall method and structure for achieving event synchronization is discussed in 4.2.1, and the individual components of the system are described in sections 4.2.2 to 4.2.4.

4.2.1 Synchronization structure

Figure 4.8 provides a full overview of the hardware components of the synchronization system. The first task of the synchronization system is the distribution of the trigger signal, the trigger number and the trigger type to all LEVBs. This is achieved by encoding this information into a serial signal, the trigger-link discussed in section 4.2.2. The encoded signal generated in the global trigger FPGA is then distributed by the sync master FPGA to all external sync client FPGAs as well as the internal sync client. The sync clients decode the trigger-link data and generate the trigger signals for the local readout modules. The trigger number and type is stored in the FPGA and read via the VME bus by the LEVB program.

To ensure no new trigger signal is generated before the LEVBs are ready, the global trigger asserts the trigger latch as soon as the trigger decision is made, setting the global busy to `one`. Upon receiving the trigger-link signal the sync client FPGAs each set their local busy to `one` as well, see figure 4.9. These local busy signals are reset by the LEVB program when all local readout modules are ready for the next event. The sync master module provides a VME accessible busy status register. The bits corresponding to the sync clients are set to `one` as soon as the trigger-link signal arrives at the sync master module. They are then reset to `zero` when the LEVBs reset their local busy signals. This ensures the busy status bits in the register are set to `zero` only after the LEVBs have seen and processed the trigger signal. The busy status register is monitored by the sync master LEVB program via the VME bus. As soon as all LEVBs participating in the current run have cleared the local busy, the sync master LEVB program resets the global trigger latch. This ends the global busy.

In addition to the local busy signal the sync master also monitors an `OK` status provided by the LEVBs. This enables the sync master LEVB to stop the acquisition if any of the LEVBs have an error condition.

While the monitoring of the `OK` and busy status of the sync clients ensures that no trigger signal is missed, it offers no protection against spurious trigger signals on the LEVBs. To guard against this, a 5 bit trigger number is sent with the trigger signal. This number is then included in the data sent by the LEVBs to the event saver as well as compared to a local event number. If the trigger number provided by the synchronization

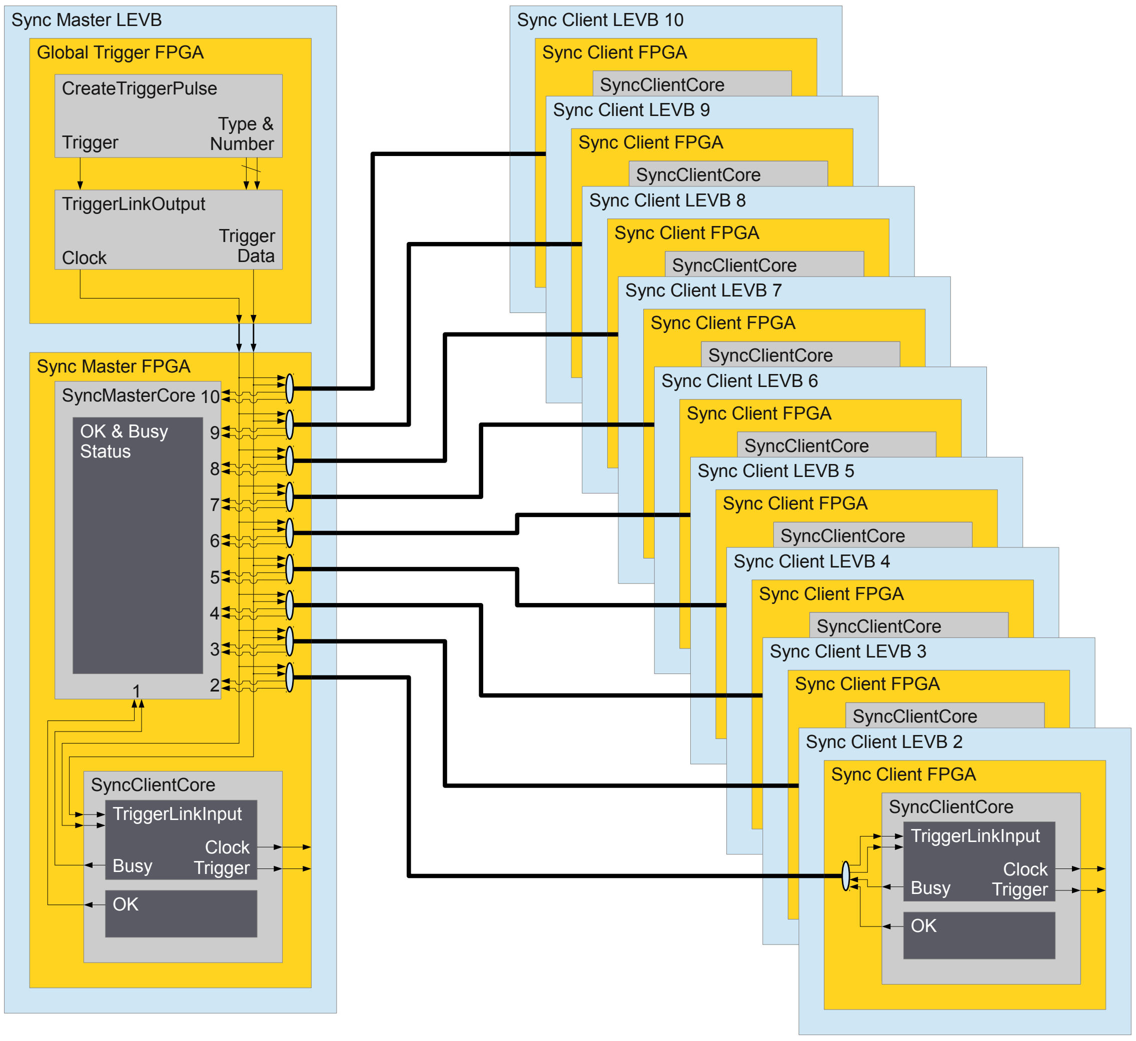


Figure 4.8: Schematic of the synchronization system
The synchronization system distributes the trigger information generated in the global trigger module through the sync master FPGA to the sync client FPGAs in the LEVBs. The client LEVBs send their busy and `OK` status back to sync master FPGA and provide the trigger signals for the local modules.

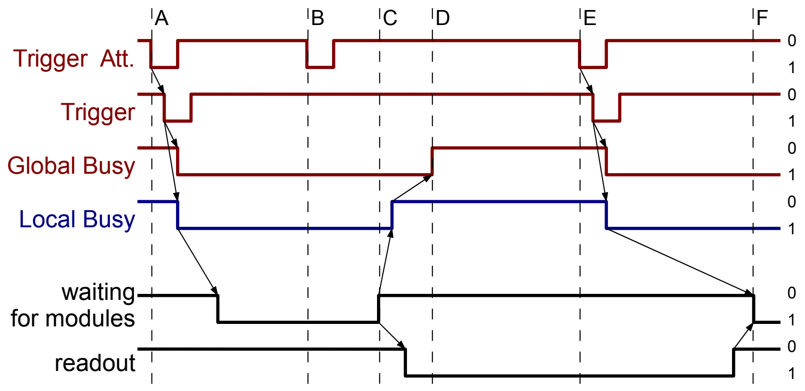


Figure 4.9: Readout sequence
The trigger logic contained in the global trigger module generates trigger attempts. If a trigger attempt occurs while the global busy is `zero` (A), a trigger is generated and the global busy asserted. Trigger attempts during the global busy (B) are ignored. When the trigger signal has been distributed to the sync client modules they assert their local busy signals. After the LEVB program detects the local busy it waits till the readout modules are ready for the next event (C). When the modules are ready the local busy is reset and the readout of the modules is started immediately. When all LEVB have reset their local busy the master LEVB resets the global busy (D). If a trigger is generated while a LEVB is still reading from the modules (E), the LEVB will detect the trigger only after the readout is finished, delaying the waiting phase (F).

system does not match the local event number, or the trigger numbers provided by the LEVBs to the event saver do not match, the event is detected as corrupt and discarded.

4.2.2 The trigger-link

The global trigger module generates the trigger decision and encodes this together with the trigger number and the trigger types in the trigger-link data sent to the sync master module. These signals are transmitted from the global trigger to the sync master using NIM signals and two coaxial cables. From the sync master module to the distributed sync clients PECL differential signals are used. Shielded Cat7 cables, each containing 4 twisted pairs, are used to transmit the PECL signals. Two pairs carry the trigger-link data while the remaining ones carry the local busy and `OK` status back to the sync master. The use of twisted pair cables and differential signals prevent the creation of unnecessary ground loops and provide better signal integrity. In addition the cabling is simplified due to the reduction in the number of individual cables necessary.

As figure 4.10 shows the trigger-link uses two signal lines. One is used for a 40 MHz reference clock, while the other carries the trigger information. During idle this line will be at logical `zero`. When the trigger condition is met and the trigger latch is set the signal on the line will become `one` synchronously with the trigger latch. This happens asynchronously to the 40 MHz clock. The signal then remains `one` for at least one clock cycle, assuring proper detection of the signal on the receiving end. This is followed by

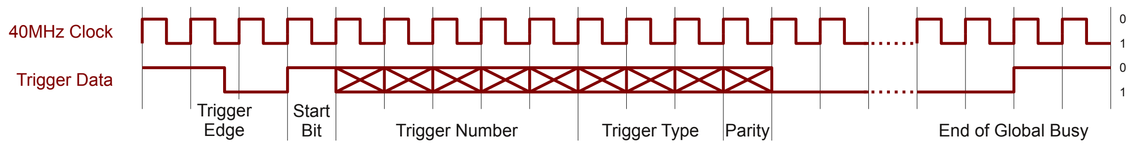


Figure 4.10: Serial transmission of Trigger Information
The start of the trigger data is marked by the asynchronous trigger edge. This is followed by the synchronous start bit after at least `one` complete 40 MHz clock cycle. The 5 bit trigger number and the 3 bit trigger type are transmitted synchronously to the clock. A parity bit allows for error checking. After the data transmission the data line remains `one` until the end of the global busy.

the serial transmission of the additional trigger information. To mark the start of the data transfer a zero bit is always sent first. Following this start bit, 9 data bits are transferred synchronously to the 40 MHz clock. 5 bits are used as event number, 3 bits identify the trigger type. The remaining bit is used as a parity bit. It is calculated by the `XOR` of the 8 data bits (odd parity). After the data transmission the line remains at logical `one` as long as the trigger latch is set. To avoid the problem of too small gaps between the reset of the trigger-link line and the next trigger, the trigger-link line is reset shortly before the trigger latch. This results in an additional dead time of 70 ns after the end of the signal on the trigger link.

The trigger-link provides several key advantages:

- The asynchronous leading edge of the trigger signal preserves the trigger timing. In addition it allows for a fast output of the trigger signal to the detectors before the following data has been decoded.
- The trigger latch signal allows the reconstruction of the life and dead times for use by readout modules, e.g. lifetime gated scalers.
- The transmission of the event number ensures the detection of accidental double triggers. It is also used to assemble the events of all LEVBs in the event saver.
- The trigger type distinguishes different trigger causes. The LEVBs can then perform different actions accordingly, e.g. read out scalers.
- The parity bit enables detection of transmission errors.
- The serial transmission of the data reduces the amount of cabling necessary with respect to a parallel transmission.

4.2.3 The sync client

The sync client modules receive the trigger-link signals distributed by the sync master and report back the local busy and `OK` signals. The `OK` signal is set and reset by the LEVB program. It is used to propagate errors detected by the software back to the sync master LEVB. The local busy is managed by a latch implemented in the sync client

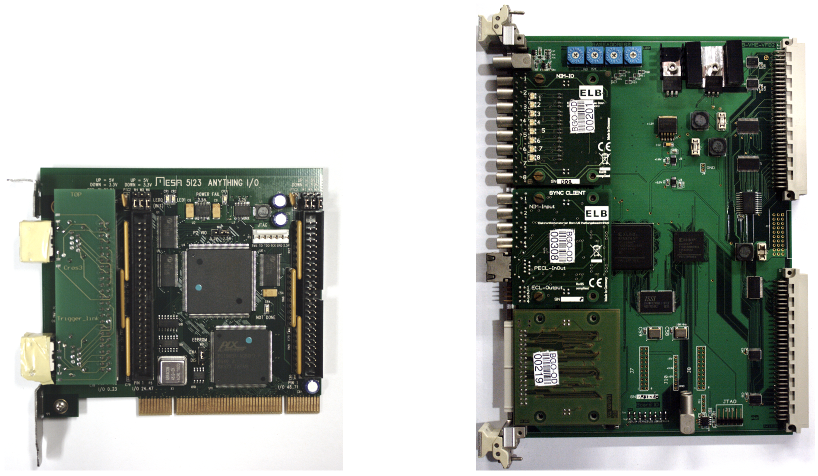


Figure 4.11: PCI (left) and VME (right) sync client module

module. This latch is set by the leading edge of the trigger-link. The reset of the local busy latch is then done by software.

The setting of the local busy latch also starts the state machine used to decode the trigger-link data. This state machine samples the trigger-link with a 200 MHz clock derived from the 40 MHz clock provided with the trigger link. As soon as it detects the start bit it enters the decoding state. The 9 data bits are sampled and stored in a shift register during this state. To ensure proper detection the sampled bits are taken close to their center. When the sampling is finished a parity check is performed and the state machine waits for the trigger-link signal to become `zero` to return to its idle state. The sampled data bits are stored in the sync client module to be accessed by the LEVB program until the next event.

A PCI and VME version of the sync client have been implemented. Both share the same internal logic but differ in the external connections. Figure 4.11 shows both modules.

The VME sync client uses an ELB VFB2 VME board using a Xilinx Spartan3 1500 FPGA. This boards provides two onboard NIM inputs and two NIM outputs. Additional inputs and outputs are provided by mezzanine boards. All VME sync clients use an ELB Sync Client mezzanine featuring a modular jack for the PECL in- and outputs used by the synchronization system. In addition it has four ECL outputs and four unused NIM inputs. Optionally a VME sync client can be equipped with an ELB NIM-IO and/or an ELB ECL out. These provide 8 NIM and 16 ECL outputs respectively.

The PCI sync client uses a MESA 5i23 Anything I/O PCI board. This board uses a Spartan3 400 FPGA. The on board pin headers are branched out to two modular plugs. One is used for the PECL in- and outputs, while the other provides two LVDS outputs.

To allow for flexible deployment of the sync clients and readout hardware, all outputs, except the ones used for the synchronization system itself, are individually configurable in software to any of the following signals.

- *trigger* default trigger signal, derived from the leading edge of the trigger-link

- *triggerSync* trigger signal synchronous to the 40 MHz clock, derived from the trigger-link start bit
- *gate* configurable length trigger signal, may jitter up to 5 ns with respect to *trigger*
- *busyGlobal* status of the DAQ global busy
- *busyLocal* status of the LEVB local busy
- *ok* status of the LEVB `OK`
- *clock* 40 MHz clock distributed by the trigger-link
- *triggerLink* pass-through of the trigger-link
- *triggerLinkSampled* output of the trigger-link data as sampled by the sync client, used for debugging
- *dcmLock* status of the sync client DLL lock, used for debugging
- *parity* result of the parity check of the trigger-link data, used for debugging
- *off* no output

The *trigger* signal is provided as a standard trigger signal for TDCs and SADCs. As its leading edge is directly derived from the leading edge of the trigger-link signal it preserves its timing. The trailing edge is generated at least 160 ns later on either the rising or falling edge of the internal 200 MHz clock. This leads to a jitter of 2.5 ns in the length of the generated trigger signal.

The CAEN V1190 TDCs sample their trigger signals only with the 40 MHz reference clock. This leads to 25 ns jumps between the different TDC modules if the trigger signal is not synchronized to this clock. If the trigger signal is synchronized to the 40 MHz clock within the sync client modules, this effect would be eliminated only within each LEVB. The TDCs in different LEVBs however would continue to show the timing jumps between each other. Therefore a trigger signal synchronized to the 40 MHz clock must be distributed to all sync clients. As the trigger-link data are already transmitted synchronously to this 40 MHz clock the start bit is used to generate the synchronous trigger *triggerSync*.

QDC modules require a *gate* signal of fixed, configurable length. As the *trigger* signal length can vary it is unsuitable for use as a gate. To generate a signal of constant length from the trigger-link data the *gate* signal is generated synchronously to the 200 MHz clock. As both clock edges are used the length is configurable in steps of 2.5 ns. However, the start of this signal relative to the leading edge of the trigger-link signal may jump by one clock cycle due the extra synchronization step required.

The *busyGlobal* signal provides an output of the status of the global busy as distributed by the trigger-link. It is used for gated scalers, allowing to count only events occurring during the life time of the DAQ.

The remaining output signals provided are either used within the synchronization system itself or used for debugging.

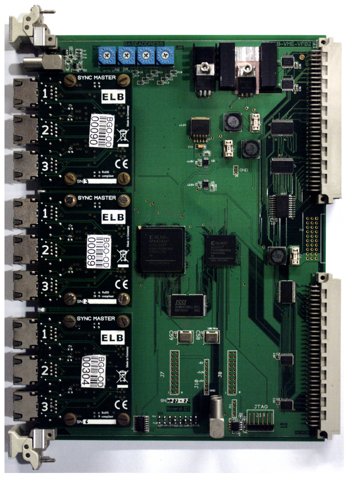


Figure 4.12: The sync master module

4.2.4 The sync master

The sync master module uses an ELB VFB2 board identical to the ones used for the VME sync client modules. It is equipped with three ELB Sync Master mezzanines. As shown in figure 4.12 each of the mezzanines has three modular jacks carrying the PECL signals for the synchronization system. As the sync master module has the sync client functionality integrated, a single master module allow for a distributed readout of 10 LEVBs. If more LEVBs are needed multiple sync master modules can be daisy-chained to provide the necessary connections. Each LEVB is assigned a synchronization id according to the jack on the sync master module to which it is connected. The sync master LEVB uses the sync id 1, while the clients are assigned the ids 2 to 10.

The trigger-link data and clock signals from the global trigger module are connected to the on-board NIM inputs. The trigger-link data is forwarded unmodified to the sync clients, while the clock is fed through a clock buffer to improve its duty cycle and stability. A VME accessible register stores the `OK` and busy status provided by the sync clients according to the synchronization ids. The `OK` signal stored is the direct signal from the LEVBs. The busy bits of this register are all set to `one` on receiving a trigger signal. They remain `one` until the trailing of the local busy signal of the corresponding LEVB is detected. The sync master module itself does not distinguish between sync clients participating in the readout and unused ones. Therefore the busy status of the unused LEVBs will remain `one` after the first trigger. These bits are then filtered in the sync master LEVB software when checking for the busy status of the LEVBs.

4.3 The local event readout, LEVBs

The local event builders, LEVBs, are the part of the data acquisition system performing the actual digitization and readout of the detector signals. The term LEVB refers both to the hardware used for readout, including the synchronization client, the CPU board and the readout modules, and the software program used to read the data from the modules and send it to the event saver. The overall structure of the LEVB software is discussed in section 4.3.1, the individual parts in sections 4.3.2 to 4.3.7. Section 4.3.8 discusses the specific LEVB implementations including the used hardware.

4.3.1 The LEVB software structure

The LEVB programs are capsuled in two main classes, `CBaselevbSync` and `IReadout`. A class derived from the abstract base class `IReadout` is implemented for each LEVB. This class provides the access to the readout hardware by common functions defined in `IReadout`. These functions are called at various stages during the data acquisition.

The functionality common to all LEVBs is provided by the class `CBaselevbSync`. To further modularize the code much of the necessary functionality is split off from `CBaselevbSync` into separate classes. The instances of these classes are then managed by the `CBaselevbSync` class.

The class `CControlSession` handles the command connection to the runcontrol. For each connection a thread is started to process the incoming commands.

The class `CStatusThread` provides the status connection. Messages are sent to all connected programs when the LEVB status changes.

The `rawDataBuffer` and `pointerBuffer` classes implement buffers for the raw data and a buffer handler.

The class `ReadoutThread` implements the thread for the data readout and the synchronization handling, while the `DataThread` class provides the thread responsible for the data transfer to the event saver.

4.3.2 The readout interface class, `IReadout`

The class `IReadout` provides the interface between the hardware independent part of the LEVB program and the readout hardware. As the readout hardware differs between the LEVBs each LEVB implements a separate class which inherits from `IReadout`. The `IReadout` base class provides access to the sync module as well as a set of functions called at various points during the readout procedure. The functions provide an empty default implementation. The LEVB specific classes then provide implementations as needed for the specific LEVB.

The readout hardware is accessed using classes providing the module specific code. The classes inheriting from `IReadout` create objects of these classes and use them as required for the data acquisition. Due to the significant variations in the way the hardware is accessed, the module classes do not implement a significant common set of functions. However, three functions are typically implemented, `configure()`, `waitEvent()` and `readEvent()`. `configure()` parses the XML configuration file and sets the module configuration accordingly. `waitEvent()` blocks until either a timeout occurs or the module has processed the current trigger signal and is ready for the next event. `readEvent()` reads the module FIFO into a provided buffer space.

command	function
`INIT`	initiates the connection to the event saver, creates readout and data thread
`START`	enter the readout thread event loop
`STOP`	stop readout and data thread
`ABORT`	alias for `STOP`
`USER`	passed on to the readout interface class to provide special functionality
`SYNC SO`	set synchronization `OK` signal
`SYNC CO`	clear synchronization `OK` signal
`SYNC CB`	clear synchronization busy signal
`SCAL`	get trigger number or trigger rate
`PROG`	get status of threads
`COUNTER`	get event, lifetime and deadtime counter values
`DATARATE`	get the data rate
`DEADTIME`	get dead time
`KILLPROG`	stop threads, close data connection, call `Uninitialize()` function of readout interface
`STATUS`	get LEVB status information
`RESET`	call `Reset()` function of readout interface
`CONFIG`	read configuration file and execute `PrepareRun()` function of readout interface
`CMDLOGON`	start logging commands
`CMDLOGOFF`	stop logging commands
`RUNNR`	sets the current run number

Table 4.2: Commands processed by the LEVBs

4.3.3 The command thread class, `CControlSession`

The class `CControlSession` handles the communication between the runcontrol and the LEVB. When the runcontrol opens a network connection to the LEVB a `CControlSession` object is created. To avoid unwanted interference with other parts of the LEVB, a separate thread is started. The thread then waits for the text commands sent by the runcontrol. Table 4.2 gives an overview of the text based commands of the LEVB processes. Section 4.5.2 describes the usage of the commands during the operation of the data acquisition.

4.3.4 The status thread class, `CStatusThread`

The `CStatusThread` class provides a second network socket to which the runcontrol connects. Using this connection no commands are sent from the runcontrol to the LEVB. Instead the LEVB reports on changes of its internal status. This allows the runcontrol to monitor the LEVB without the requirement for polling. In addition it provides the

LEVB with the possibility to signal the runcontrol on error conditions, e.g. to stop the current run.

4.3.5 The buffer classes, `rawDataBuffer` and `pointerBuffer`

The `rawDataBuffer` class is used to store the raw data acquired from the readout modules and to send it to the event saver. The `rawDataBuffer` objects are created by the `CBaselevbSync` class and then provided to the `IReadout` class by the readout thread. The `IReadout` class then writes the raw data into the memory allocated by the `rawDataBuffer`. Inside the `rawDataBuffer` the data is organized into banks. These banks have a four letter identifier which is provided by the `IReadout` class. Care has to be taken to ensure this identifier is unique across all LEVBs.

To transfer the `rawDataBuffer` objects between the readout thread and the data thread a buffer handler is necessary. The `pointerBuffer` class implements this buffer handler. It stores a pointer to each `rawDataBuffer` object created by the `CBaselevbSync` class. The readout thread can then request one of these unused buffers using the `waitFreeBuffer()` method. After this the raw data is read from the readout modules into this buffer. `writeBuffer()` is used to mark this buffer as containing new data. The data thread calls `waitFullBuffer()` to request a buffer to be send to the event saver. `releaseFreeBuffer()` marks this buffer as unused again, allowing it to be reused by the readout thread. The ordering of the buffers remains unchanged, keeping the event sequence intact. To avoid delaying the readout thread due to the unavailability of free buffers, 6000 buffer objects are kept in the `pointerBuffer`. At the expected event rates this allows for a buffering of several seconds.

To send the raw data to the event saver the `rawDataBuffer` class implements a function to stream the used part of the allocated memory to a tcp connection. A second function, used in the event saver, is implemented to read this stream back from the network socket and reconstruct the `rawDataBuffer` object. As only the used part of the allocated memory is transferred, this allows the allocation of a large memory area in the buffer without sacrificing performance.

4.3.6 The readout thread class, `ReadoutThread`

The `ReadoutThread` class is responsible for the readout sequence and the handling of the synchronization system. The `ReadoutThread` object is created by the command thread on receiving the `INIT` command. A thread implementing the readout procedure is then started.

Figure 4.13 shows a sequence diagram of this readout thread. A run can be split into three phases. The first phase is the initialization, in which all modules are set to a known state. After this the data loop which performs the readout event by event is executed. Lastly, the system is reset to an idle state at the end of the run.

In the initialization phase the readout hardware is initialized to a defined configuration. This configuration is provided by an XML configuration file and can be changed for each run, allowing for different data taking conditions. After the initialization of the

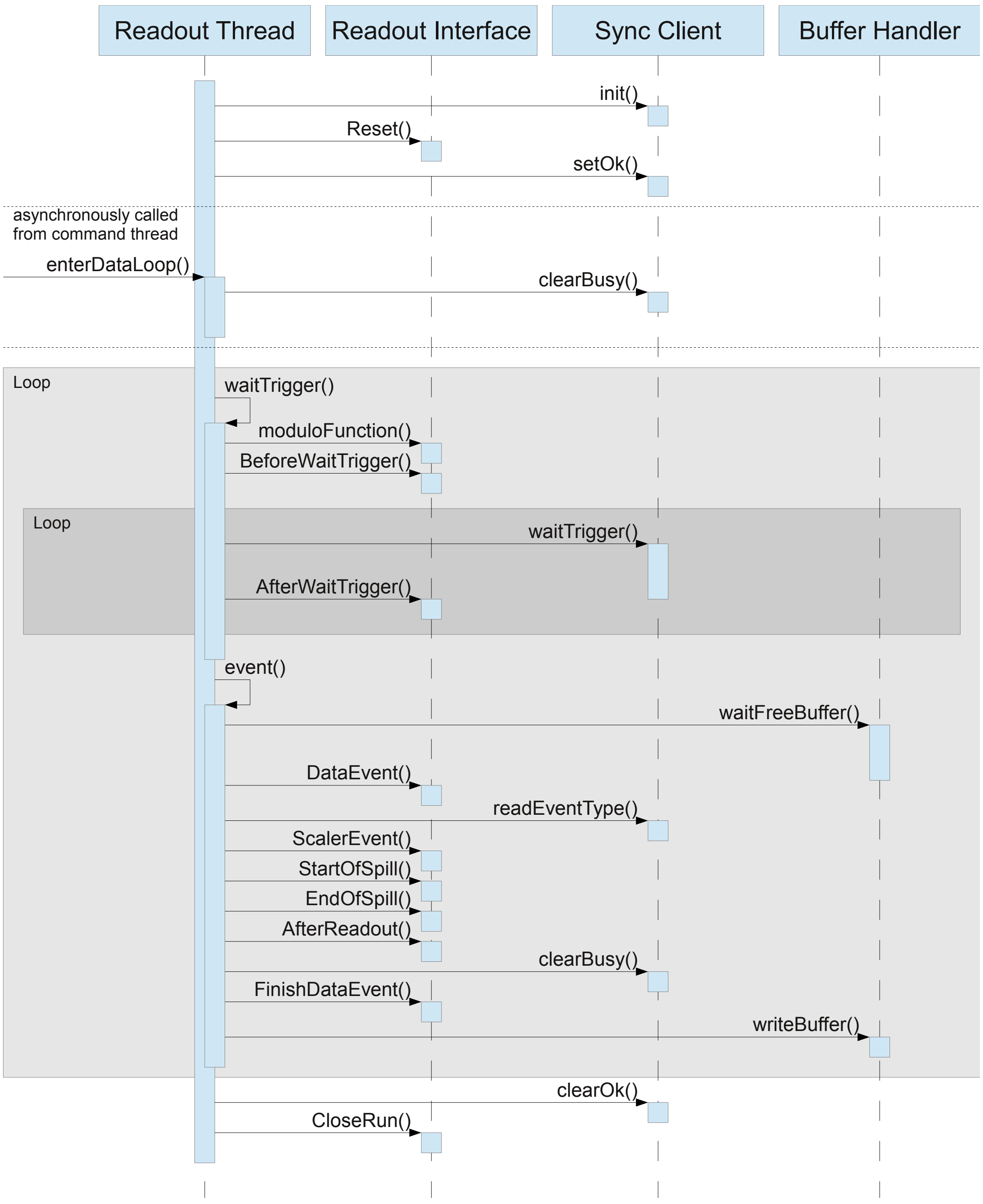


Figure 4.13: Sequence diagram for the readout thread described in section 4.3.6

readout hardware the synchronization client is initialized. At this point the local busy of the sync client is set, indicating that the LEVB is not yet ready for the first event.

After this initialization the readout waits before entering the data loop. When all LEVBs have reached this stage of their initialization, the runcontrol gives the command to proceed into the data loop and to clear the local busy signals. This is done last for the LEVB containing the global trigger, ensuring all LEVBs are ready for the first trigger.

The data loop itself consist of two parts, waiting for a trigger and reading the event data. At the start of the waiting part, the `moduloFunction()` of the readout interface is called if the event number matches the specified modulo. This provides the possibility to perform periodic checks. Before the synchronization module is checked for the busy status the `BeforeWaitTrigger()` function is called. The busy status is then repeatedly queried until either a timeout of 10 ms occurs or a trigger has been received. This is performed inside the `waitTrigger()` function of the synchronization module class. Before calling the `waitTrigger()` function again after a timeout, the `AfterWaitTrigger()` function is called with the return value from `waitTrigger()`. This allows for error checking in the sync master LEVB.

After `waitTrigger()` was successful `AfterWaitTrigger()` is called a final time and the readout thread proceeds to the `event()` function. This function handles the readout of all data relevant to the event. The first step in the `event()` function is to acquire a `rawDataBuffer` object from the buffer handler `pointerBuffer`.

After acquiring the data buffer the `DataEvent()` function is called. This function waits for for the readout modules to be ready for the next event. For modern modules with an internal FIFO this is usually implemented by polling a FIFO status register. Older modules without internal FIFO need to be read out before the next trigger can accepted. For these modules the full data readout is already performed at this stage.

Additional readout functions are called depending on the event type transmitted by the synchronization system. As an event may belong to more than one event category, e.g. scaler and start of spill event, more than one of these functions may be called for the same event. These functions read additional data or perform other special actions during the run.

After these functions, the `AfterReadout()` function is always called just before the local busy signal for the synchronization system is to be cleared. From this point onwards the next trigger signal may be sent via the synchronization system. The `FinishDataEvent()` function is used to read the internal FIFOs containing the digitized data of modern readout hardware. Doing this after clearing the busy signal minimizes the dead time of the DAQ system, as explained in section 4.2. When all data has been written to the readout buffer, the buffer is returned to the buffer handler by calling `writeBuffer()`.

At the end of a run, after the event loop has finished, the `OK` signal for the synchronization system is cleared and the `CloseRun()` function is called. The readout thread then terminates and the `ReadoutThread` object is deleted.

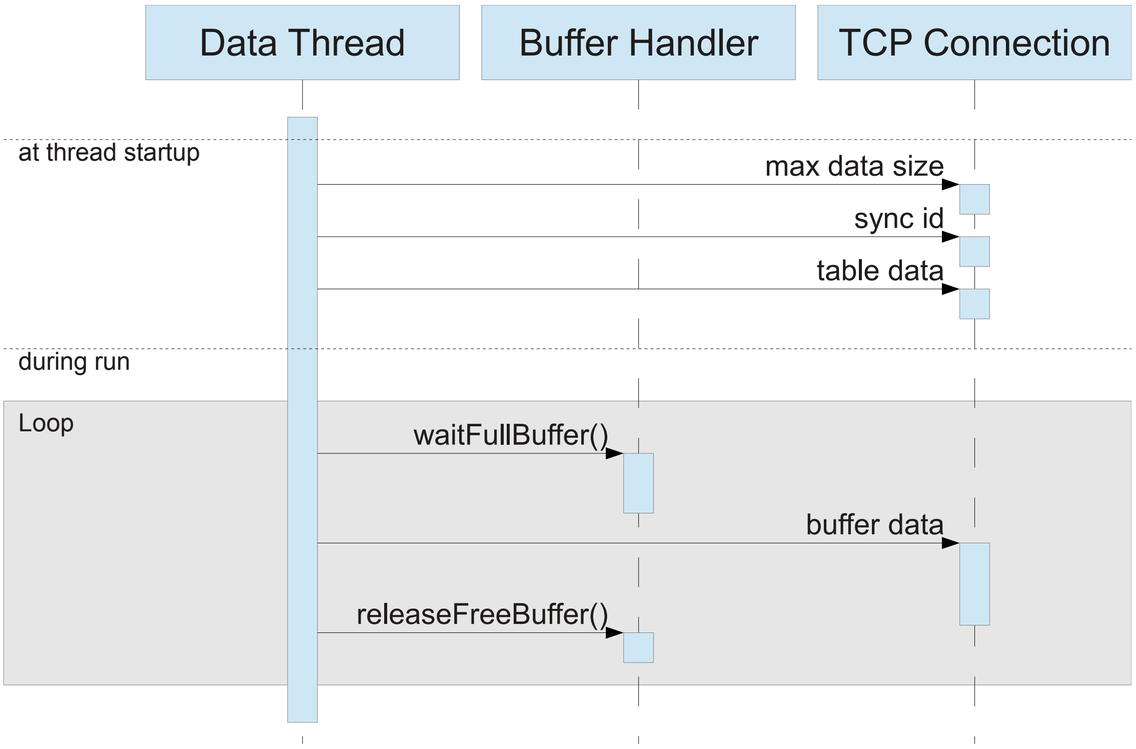


Figure 4.14: Sequence diagram for the data thread described in 4.3.7

4.3.7 The data thread class, `DataThread`

The `DataThread` object is created immediately before the `ReadoutThread` object by the same `INIT` command. The `DataThread` class then starts the thread used to transfer the data to the event saver.

This thread then opens and maintains the connection to the event saver as shown in figure 4.14. After sending the maximum data size and the synchronization id, the `TableEvent()` function of the readout interface is executed. This provides the possibility for the readout interface to provide data that is to be stored once per run, the so called table event. This is typically calibration data, such as pedestal values. The table event data is then transmitted to the event saver as the first event.

During the event loop the readout buffers filled by the readout thread are requested from the buffer handler. The contained data is sent to the event saver and the buffers are marked as free to be reused.

At the end of the run the data thread is stopped and the object deleted.

4.3.8 The LEVB implementations

Table 4.3 gives an overview of the LEVBs as they have been used in the BGO-OD experiment during the November 2014 data taking period.

LEVB	id	detectors	readout modules	raw data banks
Trigger	1	Tagger	Global Trigger, Sync master, Tagger Trigger FPGA, CAEN V1190 TDC, jTDCv6s	GSMS, TN_D, TRIT, TRIG, TN_S, TA0T, TA1T, TAGS, JTDC, JTDS, GONI
ToF	3	ToF, GIM	ToF Trigger FPGA, CAEN V1190 TDC, CAEN V1742 SADC	TOFT, TOSA, TOFS
MOMO	4	MOMO & Barrel (TDC)	CAEN V1190 TDCs	MOMT
SciFi2	5	SciFi2	jTDCv6s	SCIT, SCIS
bgo1	6	left half of BGO & Barrel (ADC)	Wiener AVM16 SADCs, CAEN V792 QDC	BG6A, BA6A
bgo2	7	right half of BGO & Barrel (ADC)	Wiener AVM16 SADCs, CAEN V792 QDC	BG7A, BA7A
Drift	10	Driftchamber	Cros3 DC readout system	DCDA, DCTB

Table 4.3: List of LEVBs as used in the BGO-OD experiment in November 2014

command	function
`USER SETCPU`	define LEVBs taking part in the readout
`USER OK`	check `OK` status of all LEVBs
`USER BUSY`	check busy status of all LEVBs
`USER TRIGGER`	set the configuration file for the global trigger

Table 4.4: Additional commands processed by the synchronization master LEVB, Trigger

The Trigger LEVB

The Trigger LEVB contains the global trigger module and the synchronization master module. To allow the runcontrol access to these modules, the LEVB provides the additional commands given in table 4.4. The values of the internal scalers of the synchronization master module, measuring the global life and dead time, are copied into a raw data bank named GSMS. The LVDS outputs of the global trigger module containing the pass-through of the local trigger signals and the trigger attempts are read by a CAEN V1190 TDC module. The data of this module is stored in the TN_D bank. As the global trigger provides also the final trigger decision this is used as the reference time for all V1190 TDC modules. The global trigger module provides an additional internal TDC with $\approx 830\,\text{ps}$ resolution for the local trigger signals. Contrary to the pass through signals on the CAEN TDC the delays of the trigger logic are applied to the internal TDC. This internal TDC is used to monitor the trigger alignment. As the readout of this TDC is slow compared to the other modules it is read only every ten events into the TRIT bank. Information concerning the trigger type, time in spill and other supplementary information from the trigger module is saved into the TRIG bank. The global trigger module also provides several counters which are read into the TN_S bank on scaler events.

The tagger local trigger FPGAs are also contained in the trigger LEVB. They provide internal TDCs identical to the one of the global trigger module. Also these TDCs are read only every ten events into the banks TA0T and TA1T. Their scalers are read on scaler events into the bank TAGS. The normal readout of the tagger signals is done using two jTDCv6 modules. The TDC data is provided in the JTDC bank while on scaler events the JTDS bank is filled with the scaler data.

In addition the trigger LEVB also implements an interface to the slowcontrol of the experiment. This allows the LEVB to set and read slow control parameters at the start and end as well as during a run. This is used to move the goniometer to defined positions at the start of the run, as well as within the spill pauses during special runs performed for alignment of the electron beam. On scaler events the goniometer position is saved into the GONI bank.

The ToF LEVB

The ToF LEVB contains the readout hardware for the ToF, GIM and FluMo detectors. For normal operation only time information is required, and the three detectors are connected to a single CAEN V1190 module. This data is saved into the TOFT bank. A CAEN V1742 SADC is used for the ToF walls as well providing the TOSA bank. However this SADC does not provide zero suppression or feature extraction, resulting in very large raw data blocks. For this reason, the SADC module is only used in dedicated runs for adjusting the high voltages and thresholds for the ToF walls.

In addition to these readout modules, the ToF LEVB contains the ToF trigger FPGA. Apart from configuring this module, the LEVB reads the integrated scalers into the TOFS bank on scaler events.

The MOMO LEVB

The MOMO LEVB uses six CAEN V1190 TDC modules to read the signals from the MOMO detector into the MOMT bank. As each of the TDC modules has 128 channels the discriminated signals from the scintillator barrel are connected to 32 of the otherwise unused channels. As they are stored in the same bank as the data for the MOMO detector the data has to be separated by the analysis software according to the channel numbers.

The SciFi2 LEVB

The readout of the SciFi2 detector is performed using seven jTDCv6 FPGA based TDC modules. These modules also provide the SciFi2 local trigger signal. The time information provided is stored in the SCIT bank, while the data from the integrated scaler modules is read into the SCIS bank on scaler events.

The bgo1 and bgo2 LEVBs

The bgo1 and bgo2 LEVBs only have minor differences. Both use 15 Wiener AVM16 SADC modules to read half of the Rugby Ball and one CAEN V792 QDC to read the corresponding half of the scintillator barrel. The data is stored in the BG6A and BA6A, respectively in the BG7A and BA7A banks. The bgo1 LEVB has a CAMAC branch interface in addition. Via this interface the CAMAC crate containing the constant fraction discriminators used for the scintillator barrel is accessed. As the discriminators are only set, no additional data banks are provided.

The Drift LEVB

The Drift LEVB differs from all other LEVBs in that it uses the PCI bus instead of the VME bus to access the readout modules. The drift chamber readout is performed using the CROS-3B readout system developed by the PNPI Gatchina[GUUY10]. This system uses frontend electronics directly attached to the drift chambers. The digitized data is then sent to a single PCI card for readout. The data of the full set of drift chambers is then stored in the DCDA bank. An automated threshold scan is performed during the initialization phase. The results of this scan are stored in the table event bank, DCTB.

4.4 The event saver, evs

The event saver, evs, is the program responsible for writing the experiment data to disk. The evs receives the data from the LEVBs and assembles them to complete events. In addition to storing the data on disk, the evs also provides an interface for monitoring the data online.

The evs is implemented using the ROOT[BR97] framework developed at CERN for storing the data. This allows the analysis software, also based on ROOT, to directly use the created data files.

Section 4.4.1 provides an overview of the event saver structure, while sections 4.4.2 to 4.4.8 detail the components of the evs.

4.4.1 The evs software structure

The communication between the event saver and the runcontrol is handled by the `CControlSession` (section 4.4.2) and `CStatusThread` (section 4.4.3) classes in the same way as the communication between the LEVBs and the runcontrol.

Objects of the `ClevbSession` class (section 4.4.4), each implementing a separate thread, are used to read the data sent by the LEVBs. As illustrated in figure 4.15, the data is then transferred to a single instance of the `ClevbSessionThread` class (section 4.4.5). For a thread-safe transfer of the data, the data storage class `rawDataBuffer` and buffer handler class `pointerBuffer`, already described in section 4.3.5, are used.

The `ClevbSessionThread` class packages the raw data into `CEvent` objects (4.4.6) which are then transferred to the `EventProcessThread` object (section 4.4.7). The

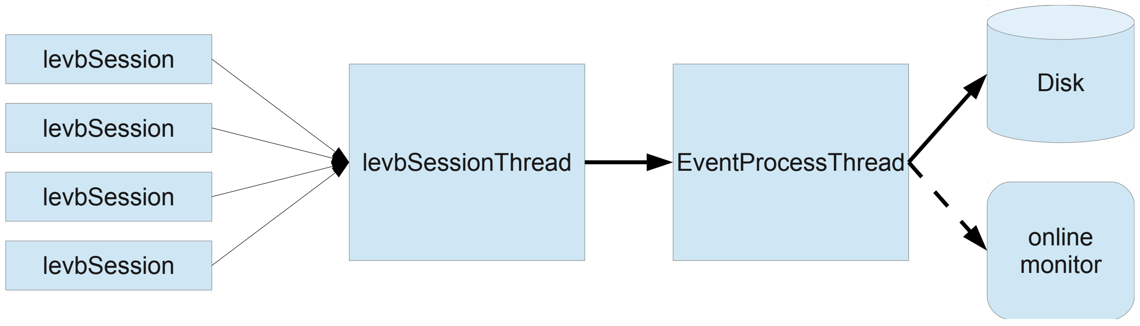


Figure 4.15: Data flow in the event saver

Command	Function
`RUN NR`	set run number
`RUN GETNR`	report current run number
`DB ROOT`	create and open output file
`DB CLOSE`	close output file
`DB WRITTENBYTES`	report number of bytes written to output file
`DB STATUS`	report if a file is open
`CONNECT`	initiate LEVB connection
`START`	start `ClevbSessionThread`, create and start `EventProcessThread` and `EVMProcessThread`
`STOP`	stop threads, delete thread objects
`DATARATE`	report data rate
`STATUS`	report status

Table 4.5: Commands processed by the event saver

`EventProcessThread` writes the data to a ROOT file and generates copies for the online monitor interface class `EVMProcessThread` (section 4.4.8) as needed.

4.4.2 The evs command thread class, `CControlSession`

A dedicated thread, implemented by the `CControlSession` class, is used to process the commands sent from the runcontrol to the event saver. This thread is started when the TCP connection is opened. The handling of the connection and the processing of commands is identical to the case of the LEVBs. The commands processed are listed in table 4.5 with a short description. Section 4.5.2 shows the usage during normal operation of the data acquisition.

4.4.3 The evs status thread class, `CStatusThread`

The `CStatusThread` class used by the evs is the same as the one used by the LEVBs. In addition to signaling errors and state changes, the status thread is used to signal the current number of events and the total size of the data taken for the run. This information is used by the runcontrol to monitor the progress of the run.

4.4.4 The LEVB interface class, `ClevbSession`

An object of the `ClevbSession` class is created for each LEVB connecting to the event saver during the run start. This object then starts a thread to read the data provided by the LEVB. As already shown in figure 4.14 first the maximum data size is received from the LEVB, followed by the synchronization id and table data. During the run the data sent by the LEVB is read back into `rawDataBuffer` objects. As done in the LEVBs the `rawDataBuffer` objects are managed by an instance of the `pointerBuffer` class. As an individual set of buffers for each LEVB is used, only 3000 buffers are provided per LEVB to conserve main memory.

4.4.5 The event building class, `ClevbSessionThread`

A single instance of the `ClevbSessionThread` class is created at run start to receive and assemble the data provided by the `ClevbSession` objects. This class implements a thread to receive the `rawDataBuffer` objects from the `ClevbSession` objects via the `pointerBuffer` objects. A `CEvent` object (described in the next section) is created for each event. Copies of the `rawDataBuffer` objects are then stored in the `CEvent` object before the original `rawDataBuffer` objects are marked as free in the `pointerBuffer` objects.

The `ClevbSessionThread` class then checks the event for integrity. This check tests if the `CEvent` object contains data for all LEVBs and that the event numbers given in the `rawDataBuffer` objects match. If this fails the corrupt event is discarded and an error message logged. The good `CEvent` objects are then provided to the `CEventBuffer` buffer handler.

In regular intervals a status message concerning the number of events and the number of bytes processed is sent to the runcontrol using the `CStatusThread`.

4.4.6 The event buffer classes, `CEvent` and `CEventBuffer`

The classes `CEvent` and `CEventBuffer` provide functionality similar to `rawDataBuffer` and `pointerBuffer` respectively. The `CEvent` class is used to store all data belonging to a single event. This is done by saving copies of `rawDataBuffer` objects. In addition to simply storing a collection of `rawDataBuffer` objects, the `CEvent` class provides functions to check the event for integrity and to stream the event data to the online monitor. The function `writeData` is used to extract the data banks and provide them to the file interface class `CRootDB`.

The `CEventBuffer` class provides a way to buffer the `CEvent` objects passed from the `ClevbSessionThread` to the `EventProcessThread`. Contrary to the `rawDataBuffer` object handled by the `pointerBuffer`, the `CEvent` objects are not reused. They are created in the `ClevbSessionThread` and deleted in the `EventProcessThread`.

4.4.7 The event storage classes, `EventProcessThread` and `CRootDB`

The `EventProcessThread` and `CRootDB` classes are responsible for writing the event data to a ROOT file on disk. The `EventProcessThread` class provides the thread to access the `CEvent` objects, and the `CRootDB` class handles the file interface.

For each event, the data banks provided by the LEVBs are extracted from the `CEvent` container. A loop over all data banks of all `rawDataBuffer` contained in the `CEvent` object is performed to write the event data to the file. For each bank, a `CBTByteArray` object named rawdata.XXXX, where XXXX is replaced by the bank identifier, is created. The `CBTByteArray` class is the same class used by the analysis software for handling of raw data.

The `CBTByteArray` objects are created in a container, a `TClonesArray`. This container class provides storage for objects of the same class with efficient memory management. To organize the data by events this `TClonesArray` is contained in a `TTree` structure. For each event the `Fill()` function of the `TTree` is called, storing the `TClonesArray` and its contents in the file. This output file is created during run start, when the `DB ROOT` command is received from the runcontrol.

In addition to the rawdata `TClonesArray` another `TClonesArray` called *tabledata* is created outside of the `TTree` to store the table events provided by the LEVBs. A `TNamed` Object is used to identify the version of this data format, currently 1.

This organization of the data in the file is convenient as it mirrors the handling of the raw data in the analysis software.

To provide data to the `EVMProcessThread` class interfacing to the online monitor however, a copy of the `CEvent` may be created. To prevent the `EventProcessThread` thread being delayed by the online monitor, this copy is only created if the previous copy has been processed. In case of scaler events a copy is always created. If the previous copy has not been processed it is deleted.

4.4.8 The online monitor interface class, `EVMProcessThread`

The `EVMProcessThread` takes the `CEvent` objects copied by the `EventProcessThread` and sends the data banks contained within to the online monitor interface program explorad via a network connection. The explorad program then forwards this data to the analysis software. As new copies of the event data are only created after the data has been sent to the online monitor it does not interfere with writing the data to disk, even when the rate processed by the explorad is below the event rate written.

4.5 The data acquisition control program, runcontrol

The runcontrol program manages the state of, and the interaction between, the LEVBs and the event saver. It also processes the user input provided by the user interface, see section 4.6. As such the runcontrol is responsible for the configuration of the readout system, as well as the starting and stopping of runs. This configuration is stored in a database, the runDb. This database also logs the runs taken and any comments provided.

Command	Function
`STARTRUN`	start a new run
`STOPRUN`	stop the current run
`ABORT`	abort a failed run start
`PAUSE`	pause data taking
`CONTINUE`	resume data taking
`RESTART`	kill and restart one of the LEVBs or the event saver
`LEVB`	list known LEVBs
`RUNINFO`	print information to current run
`NEXTRUN`	specify new run list
`GETSTATUS`	print runcontrol status
`EXIT`	disconnect
`RUNTYPES`	list available run types
`TYPEINFO`	print information to specified run type
`AUTOPILOT`	enable or disable AUTOPILOT function
`DUMPDATA`	enable or disable discarding data
`NEWTYPE`	define a new run type
`REMOVETYPE`	remove an existing run type
`DEFAULTPATH`	specify default file output path
`COMMENT`	add a comment to the run database
`CATEGORIES`	list available comment categories
`UISTATUS`	set status message for the daqUI connection

Table 4.6: Commands processed by the runcontrol program.

In addition to coordinating the interaction with the LEVBs and the evs, the runcontrol monitors their status. This allows the runcontrol to provide the daqUI with feedback concerning the status of DAQ processes. Status information concerning the current run is collected from the event saver and also forwarded to the daqUI.

The following detailed description of the runcontrol is split into three parts. The configuration of the system is discussed in section 4.5.1, the operation of the DAQ in section 4.5.2, and the monitoring of the system in section 4.5.3.

4.5.1 System configuration

The runcontrol maintains the configuration of the DAQ setup in the runDb database. A list of DAQ programs excluding the runcontrol itself is stored in the LEVBs table. This table contains the hostnames, command lines and the synchronization id. A description of the programs is also provided. To keep a record of past configurations, each row of the table contains timestamps indicating the time period for which it is valid. When the configuration is changed these timestamps are updated and a new row is inserted. This list is read during startup of the runcontrol. The runcontrol then tries to open command and status connections to all listed processes. The listed processes can be restarted by the runcontrol upon receiving the appropriate `RESTART` command, see table

4.6. This is handled by spawning an ssh process to log in to the corresponding host, where a previous instance of the program is killed and a new instance started with the command line provided from the database. The command and status connection are reestablished automatically after being closed.

A list of run types is stored in the database in addition to the list of programs. For access to previous configurations, a unique id is provided in addition to timestamps used in the same way as for the list of programs. The configuration for a run type contains a name for the type, which LEVBs are to participate and the name of the trigger configuration file. A file name template and the maximum number of events and bytes to take for a run are also stored. The names of the configuration files for each LEVB are kept in a separate table, indexed by the unique id of the run type.

The runcontrol provides a list of runs to facilitate taking different types of run periodically during a beamtime. The runs in this list are then sequentially acquired. When the end of the list is reached the runcontrol restarts from the beginning. This list is kept in the database to preserve it during a restart of the runcontrol. As the run types in this list are identified by their names instead of the unique ids, changes to a run type do not require a change to the run list to take effect. The AUTOPILOT function of the runcontrol is implemented using this run list. It automatically stops runs when either the maximum number of events or the maximum size is reached. The next run from the run list is then started, allowing for minimal user interaction. The AUTOPILOT is automatically disabled if the acquisition is stopped by the user or an error condition was encountered. A special run type called -----STOP----- indicates that the acquisition should be stopped and no new run started automatically.

4.5.2 DAQ operation

The runcontrol initiates the start of a new run upon receiving a `STARTRUN` command from the daqUI or after the AUTOPILOT function has stopped a run. The first step of this procedure is to check the status of all DAQ programs. If the reported state of the overall system is not idle, the start run procedure is aborted. The next step is to retrieve the run type information from the database. At this point all programs to participate in this run are checked again for their status.

If no error is detected the run about to be started is entered into the runDb before the start up sequence illustrated in figure 4.16 is initiated. After resetting the synchronization system with the `SYNC CO` and `SYNC CB` commands, the run number is distributed to the event saver and all LEVBs. The trigger file name and the configuration file names are sent to the LEVBs. These files are then read by the LEVBs and the readout modules are configured accordingly. The files are also read by the runcontrol to preserve past configurations. The runcontrol then saves the files in the runDb where they are associated with the current run. An MD5 hash of the file content is used to avoid saving unnecessary duplicates. After the LEVB configuration is finished the synchronization ids of the participating LEVBs are sent to the sync master LEVB. The file name for the output ROOT file of this run is generated from the template provided in the run type description and sent to the event saver. The connections between the event saver and the LEVBs are established using the `CONNECT` and `INIT` commands. The master

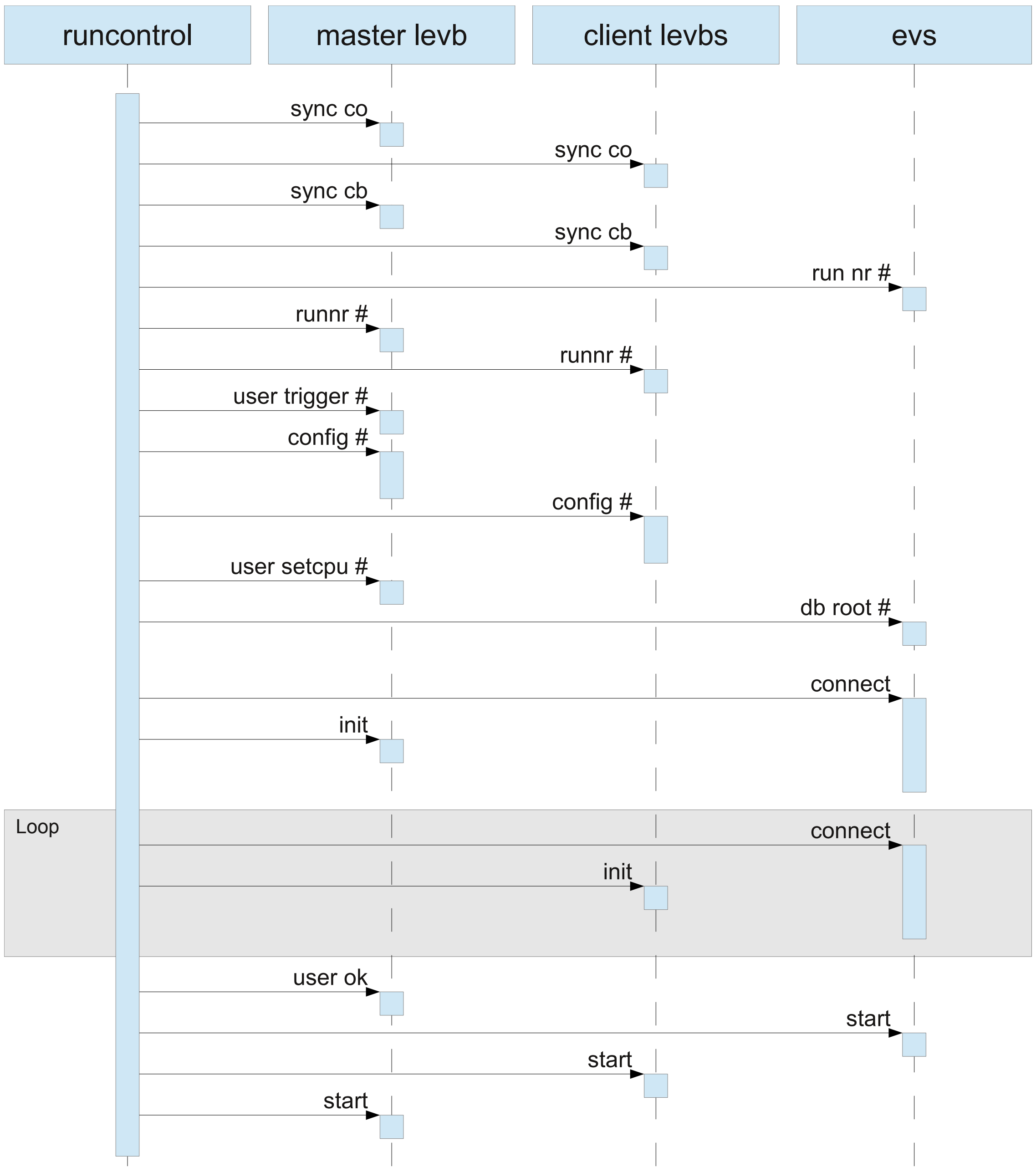


Figure 4.16: Sequence diagram for the start of a run
For a description of the commands sent by the runcontrol see tables 4.2 (all LEVBs), 4.4 (master LEVB) and 4.5 (evs).

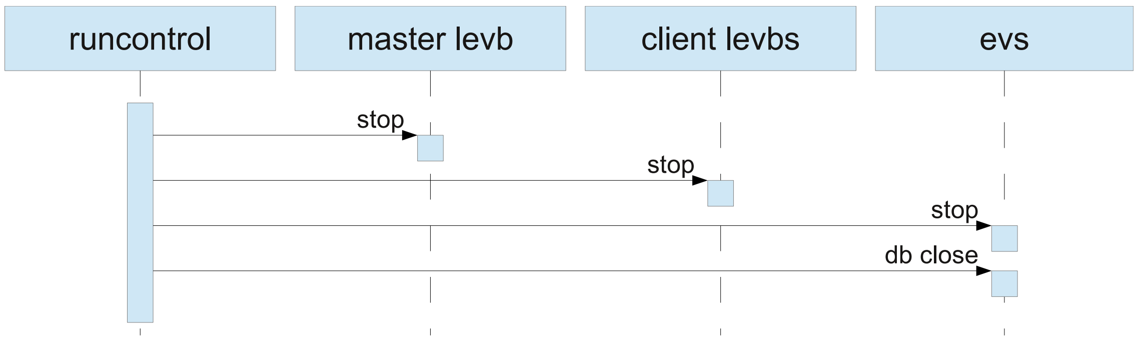


Figure 4.17: Sequence diagram for the stop of a run
For a description of the commands sent by the runcontrol see tables 4.2 (LEVBs) and 4.5 (evs).

LEVB is ensured to be the first connecting LEVB.

When all used LEVBs are connected to the event saver the `OK` status transmitted through the synchronization system is checked. As soon as all LEVBs report `OK` the event and readout loops are started. The loops on the event saver are started first, then the readout loops on the client LEVBs and lastly the readout loop on the master LEVB. This ensures that no trigger signals are accepted before all LEVBs have entered the readout loop.

After the run is started, the runDb entry created at the beginning of the start run procedure is updated with the actual output file name and the contents of the trigger configuration file. As with the LEVB configuration files the trigger configuration is indexed by its MD5 hash.

The procedure to stop the current run is initiated when either the `STOPRUN` command received from the daqUI, the maximum data size is reached, the maximum event count is reached while the AUTOPILOT is active or when one of the LEVBs sends a `STOPRUN` message using the status connection. The stop run procedure first sends the `STOP` command to the master LEVB, then to the client LEVBs. The LEVBs acknowledge the `STOP` command when they have transmitted all their data to the event saver. After this the `STOP` command is sent to the event saver. This sequence, shown in figure 4.17, ensures that all remaining data reaches the event saver before the output file is closed with the `DB CLOSE` command. After closing the file, the runDb entry for the finished run is updated with the final number of events, the data size and the stop time.

In addition to managing starting and stopping runs, the runcontrol provides the daqUI with the possibility to insert comments into the runDb. These comments can either be associated to a specific run, or only indexed by the time they are inserted. To avoid misinterpreting the comment as a separate command, the comment is sent base64 encoded. Files may be included in the comment. These are sent base64 encoded as well and stored in the runDb according to their MD5 hash.

4.5.3 System monitoring

The runcontrol continuously monitors the status of the DAQ system. Dedicated connections are used to implement the monitoring without unnecessary polling of the readout programs. This allows the LEVBs and the event saver to send messages about their status asynchronously with respect to the commands sent by the runcontrol.

The messages sent by the LEVBs are:

- *LEVBSTATUS* followed by a number indicates the status of the various parts of the LEVB. This is sent on each change of the status. Besides the overall status of the LEVB (starting, stopping, running, idle, error) it also contains information on which of the internal threads are currently running.
- *STOPRUN* indicates that the LEVB encountered a condition that requires the current run to be stopped. This is either sent either as a consequence of an error during data taking or in case the specific run has reached a defined end condition.

In addition to the messages used by the LEVBs the event saver can send:

- *BYTES* followed by the number of data bytes received from the LEVBs and the data rate.
- *EVENTS* followed by the number of events received and the event rate.

The status information provided by these messages is cached by the runcontrol and forwarded to the running user interfaces. The status of the runcontrol itself is sent to the daqUI in the same way.

4.6 The user interface, daqUI

The daqUI provides the user interface for the data acquisition system. The interaction with the acquisition system is mediated only by two network connections to the runcontrol. This allows several instances of the daqUI to run in parallel without harmful interference. All changes made in one daqUI instance are transmitted to the runcontrol over the command connection. The runcontrol then updates the status of all daqUI instances by sending the information asynchronously via the status connection.

A screenshot of the main window is shown in figure 4.18. The daqUI interface consists of several regions dedicated to different aspects of the data acquisition.

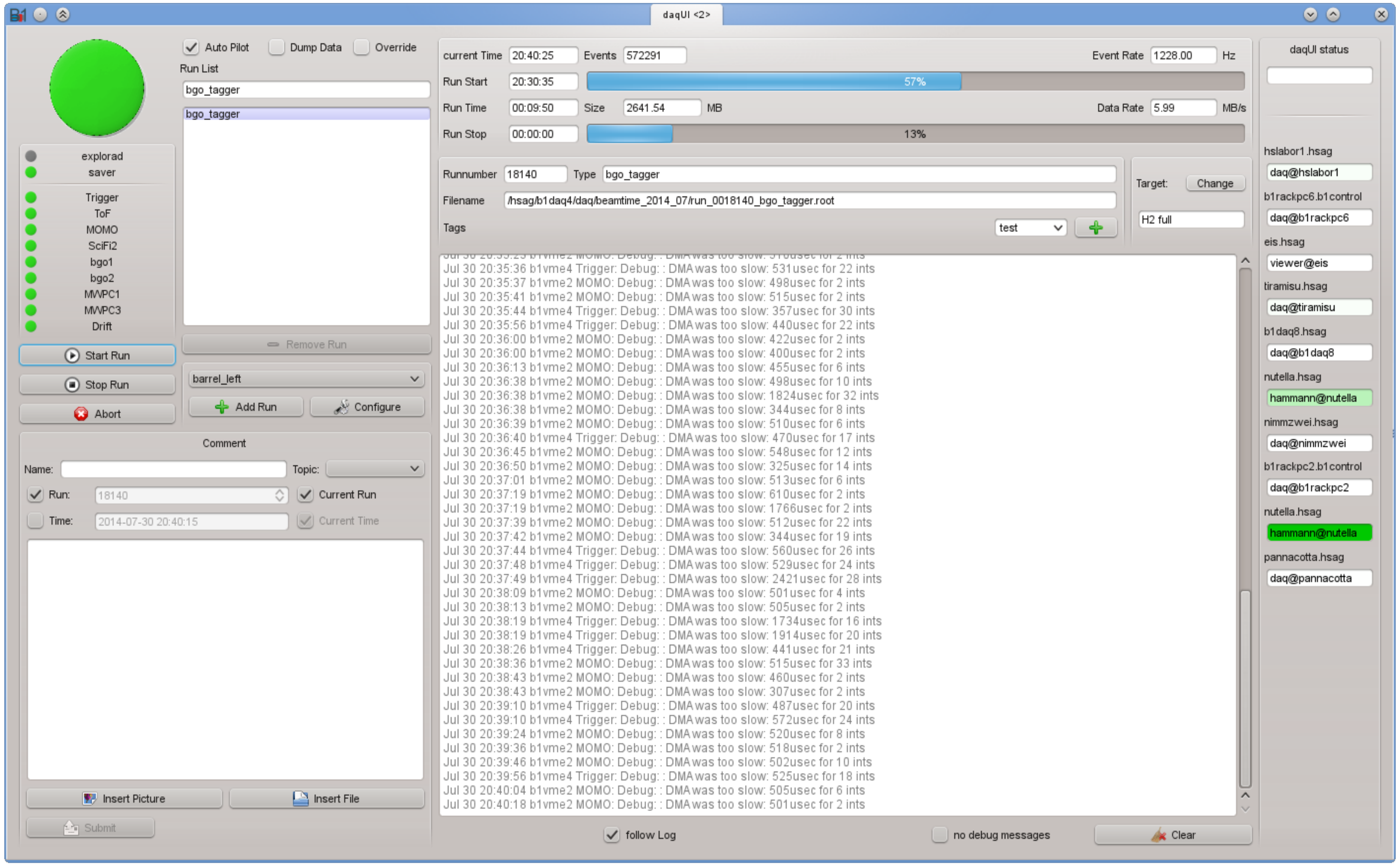
- *Process status indication.* The top left part of the main window provides a list of the LEVBs and other processes in the DAQ system. A tool-tip provides additional information on the process. Each of the names is accompanied by a colored bullet indicating the current status. The possible colors are gray, blue, green and red indicating idle, starting or stopping, running and error status respectively. A more detailed status information is given as a tool-tip to the bullet points. A context menu allows restarting of the processes as well as opening a configuration dialog for changing the LEVB settings.

- *Run List.* To the right of the *Process status indication* there is an area for the list of runs to be performed. This list can be edited either by the provided buttons or by a context menu and drag and drop. The Auto Pilot feature allows the DAQ system to automatically cycle through all runs present in the run list. A configuration dialog for the run types is provided here.
- *Comment section.* Below the *Process status indication* and the *Run List* comments can be entered which will be saved into the run database. This gives the shift crew the possibility to report on special occurrences and errors encountered during data acquisition. Pictures and other files can be inserted by drag and drop or by use of the buttons provided.
- *Current run status.* Information about the current run is collected in the top center region. This includes current running time data and event rates and totals as well as the name of the output file.
- *Log viewer.* Log messages of all DAQ processes are displayed below the *Current run status.* They are colored according to their severity.
- *daqUI status.* The rightmost part of the main window is used for a status display of all currently running daqUI processes. It displays the host names and the user name or a message given by the user. Recent activity of individual daqUI processes is indicated by a colored background.

4.7 The run information database, runDb

The runDb stores information on the runs in a central PostgreSQL database. The database contains several tables for the different data stored.

- *rundb* Contains the list of all runs with the corresponding information. The table contains run number, file name, file path, run type, start time, stop time, number of events, data size and information from the slowcontrol. At the end of each run, an automatic script analyzes the slowcontrol data for the run period and extracts the required information. Added to the runDb database are the average, minimum, maximum and standard deviation of the values for the magnetic field of the spectrometer and tagger magnets, electron beam energy and current as well as the position of the goniometer system. These values can then be accessed in the same way as the other run parameters.
- *comments* Contains comments inserted by users via the daqUI or the web front end. The comments can either correspond to specific runs or be sorted by the insertion time.
- *beamtimes* Contains a list of all beam times and other periods. The total size and number of events are automatically calculated and updated. This is achieved by a database trigger on the rundb table.



☑ Auto Pilot ☐ Dump Data ☐ Override

Run List: bgo_tagger

barrel_left

Name:
Topic:
☑ Run: 18140 ☑ Current Run
☐ Time: 2014-07-30 20:40:15 ☑ Current Time

current Time 20:40:25 Events 572291 Event Rate 1228.00 Hz
Run Start 20:30:35 57%
Run Time 00:09:50 Size 2641.54 MB Data Rate 5.99 MB/s
Run Stop 00:00:00 13%

Runnumber 18140 Type bgo_tagger
Filename /hsag/b1daq4/daq/beamtime_2014_07/run_0018140_bgo_tagger.root
Tags test
Target: H2 full

```
Jul 30 20:35:36 b1vme4 Trigger: Debug: : DMA was too slow: 531usec for 22 ints
Jul 30 20:35:37 b1vme2 MOMO: Debug: : DMA was too slow: 498usec for 2 ints
Jul 30 20:35:41 b1vme2 MOMO: Debug: : DMA was too slow: 515usec for 2 ints
Jul 30 20:35:44 b1vme4 Trigger: Debug: : DMA was too slow: 357usec for 30 ints
Jul 30 20:35:56 b1vme4 Trigger: Debug: : DMA was too slow: 440usec for 22 ints
Jul 30 20:36:00 b1vme2 MOMO: Debug: : DMA was too slow: 422usec for 2 ints
Jul 30 20:36:00 b1vme2 MOMO: Debug: : DMA was too slow: 400usec for 2 ints
Jul 30 20:36:13 b1vme2 MOMO: Debug: : DMA was too slow: 455usec for 6 ints
Jul 30 20:36:38 b1vme2 MOMO: Debug: : DMA was too slow: 498usec for 10 ints
Jul 30 20:36:38 b1vme2 MOMO: Debug: : DMA was too slow: 1824usec for 32 ints
Jul 30 20:36:38 b1vme2 MOMO: Debug: : DMA was too slow: 344usec for 8 ints
Jul 30 20:36:39 b1vme2 MOMO: Debug: : DMA was too slow: 510usec for 6 ints
Jul 30 20:36:40 b1vme4 Trigger: Debug: : DMA was too slow: 470usec for 17 ints
Jul 30 20:36:45 b1vme2 MOMO: Debug: : DMA was too slow: 548usec for 12 ints
Jul 30 20:36:50 b1vme2 MOMO: Debug: : DMA was too slow: 325usec for 14 ints
Jul 30 20:37:01 b1vme2 MOMO: Debug: : DMA was too slow: 513usec for 6 ints
Jul 30 20:37:19 b1vme2 MOMO: Debug: : DMA was too slow: 610usec for 2 ints
Jul 30 20:37:19 b1vme2 MOMO: Debug: : DMA was too slow: 1766usec for 2 ints
Jul 30 20:37:39 b1vme2 MOMO: Debug: : DMA was too slow: 512usec for 22 ints
Jul 30 20:37:42 b1vme2 MOMO: Debug: : DMA was too slow: 344usec for 19 ints
Jul 30 20:37:44 b1vme4 Trigger: Debug: : DMA was too slow: 560usec for 26 ints
Jul 30 20:37:48 b1vme4 Trigger: Debug: : DMA was too slow: 529usec for 24 ints
Jul 30 20:37:49 b1vme4 Trigger: Debug: : DMA was too slow: 2421usec for 28 ints
Jul 30 20:38:09 b1vme2 MOMO: Debug: : DMA was too slow: 501usec for 4 ints
Jul 30 20:38:13 b1vme2 MOMO: Debug: : DMA was too slow: 505usec for 2 ints
Jul 30 20:38:19 b1vme4 Trigger: Debug: : DMA was too slow: 1734usec for 16 ints
Jul 30 20:38:19 b1vme4 Trigger: Debug: : DMA was too slow: 1914usec for 20 ints
Jul 30 20:38:26 b1vme4 Trigger: Debug: : DMA was too slow: 441usec for 21 ints
Jul 30 20:38:36 b1vme2 MOMO: Debug: : DMA was too slow: 515usec for 33 ints
Jul 30 20:38:43 b1vme2 MOMO: Debug: : DMA was too slow: 460usec for 2 ints
Jul 30 20:38:43 b1vme2 MOMO: Debug: : DMA was too slow: 307usec for 2 ints
Jul 30 20:39:10 b1vme4 Trigger: Debug: : DMA was too slow: 487usec for 20 ints
Jul 30 20:39:10 b1vme4 Trigger: Debug: : DMA was too slow: 572usec for 24 ints
Jul 30 20:39:24 b1vme2 MOMO: Debug: : DMA was too slow: 520usec for 8 ints
Jul 30 20:39:36 b1vme2 MOMO: Debug: : DMA was too slow: 518usec for 2 ints
Jul 30 20:39:46 b1vme2 MOMO: Debug: : DMA was too slow: 502usec for 10 ints
Jul 30 20:39:56 b1vme4 Trigger: Debug: : DMA was too slow: 525usec for 18 ints
Jul 30 20:40:04 b1vme2 MOMO: Debug: : DMA was too slow: 505usec for 6 ints
Jul 30 20:40:18 b1vme2 MOMO: Debug: : DMA was too slow: 501usec for 2 ints
```

hslabor1.hsag: daq@hslabor1
b1rackpc6.b1control: daq@b1rackpc6
eis.hsag: viewer@eis
tiramisu.hsag: daq@tiramisu
b1daq8.hsag: daq@b1daq8
nutella.hsag: hammann@nutella
nimmzwei.hsag: daq@nimmzwei
b1rackpc2.b1control: daq@b1rackpc2
nutella.hsag: hammann@nutella
pannacotta.hsag: daq@pannacotta

Figure 4.18: Main window of the daqUI during a run

- *run_types* Contains a list of all run types with periods for which they are valid. Specified in the table are the file name template, name of the trigger configuration file, maximum number of events, maximum data size and the ids of the participating LEVBs. The configuration file names for the LEVBs are contained in the *configs* table, linked by the run type uid.
- *files* Contains a list of all files stored in the database with original file name, mime type and MD5 hash.
- *used_configs* Contains the LEVB config file hashes for each run.
- *runcontrol_settings* Contains permanently stored configuration options of the runcontrol.
- *runlist* Contains the list of runs currently in the runlist.

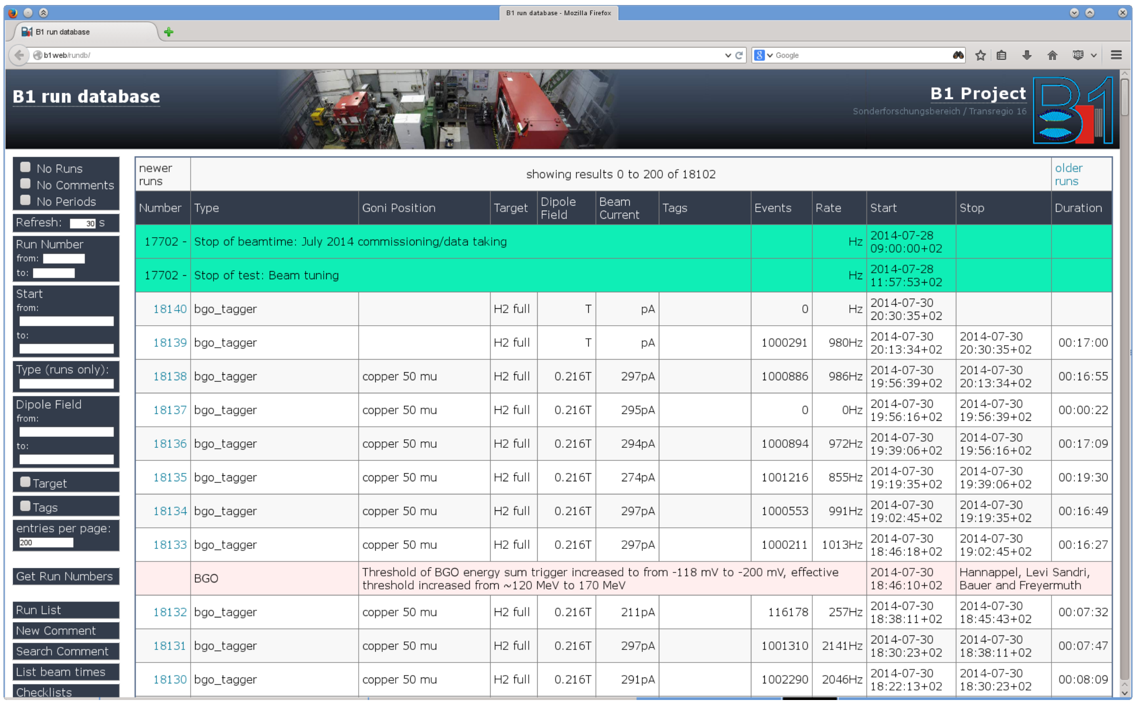
These tables allow a full reconstruction of the DAQ settings for each run. In combination with the user provided comments and the data provided by the slowcontrol the experimental conditions can be logged.

The runDb database can either be accessed directly by the analysis software or with the help of a web front end. As shown in figure 4.19, the web front end lists the runs sorted by run number together with the comments. This overview can be filtered by run numbers, time spans and/or run types. In addition to the general information provided by the overview, more detailed information is available for each run by clicking on the run number.

4.8 Performance of the BGO-OD DAQ

A complex system like the BGO-OD DAQ has several parameters related to its performance. These characteristics are however not independent. Therefore it is necessary to investigate several quantities to be able to correctly characterize the DAQ system. The most important types of performance measurements are:

- *Buffer occupancy* illustrates the amount of events buffered in the various layers of the DAQ system. If the number of used buffers approaches the number of available buffers this indicates a performance bottleneck in the DAQ system.
- *Readout time* is the time, typically given in µs, required to read all data for a given event from the readout hardware. This limits the maximum possible rate at the LEVB level.
- *Dead time* is the time during which the DAQ cannot accept new triggers. This is either given in µs similar to the readout time or as a percentage of the total time.
- *Life time* is the time during which the DAQ may accept new triggers. This is given in µs or as a percentage of the total time in the same way as the dead time. On the assumption of no time correlation between trigger attempts this can be used to calculate the fraction of accepted triggers.



newer runs	showing results 0 to 200 of 18102										older runs
Number	Type	Goni Position	Target	Dipole Field	Beam Current	Tags	Events	Rate	Start	Stop	Duration
17702 -	Stop of beamtime: July 2014 commissioning/data taking							Hz	2014-07-28 09:00:00+02		
17702 -	Stop of test: Beam tuning							Hz	2014-07-28 11:57:53+02		
18140	bgo_tagger		H2 full	T	pA		0	Hz	2014-07-30 20:30:35+02		
18139	bgo_tagger		H2 full	T	pA		1000291	980Hz	2014-07-30 20:13:34+02	2014-07-30 20:30:35+02	00:17:00
18138	bgo_tagger	copper 50 mu	H2 full	0.216T	297pA		1000886	986Hz	2014-07-30 19:56:39+02	2014-07-30 20:13:34+02	00:16:55
18137	bgo_tagger	copper 50 mu	H2 full	0.216T	295pA		0	0Hz	2014-07-30 19:56:16+02	2014-07-30 19:56:39+02	00:00:22
18136	bgo_tagger	copper 50 mu	H2 full	0.216T	294pA		1000894	972Hz	2014-07-30 19:39:06+02	2014-07-30 19:56:16+02	00:17:09
18135	bgo_tagger	copper 50 mu	H2 full	0.216T	274pA		1001216	855Hz	2014-07-30 19:19:35+02	2014-07-30 19:39:06+02	00:19:30
18134	bgo_tagger	copper 50 mu	H2 full	0.216T	297pA		1000553	991Hz	2014-07-30 19:02:45+02	2014-07-30 19:19:35+02	00:16:49
18133	bgo_tagger	copper 50 mu	H2 full	0.216T	297pA		1000211	1013Hz	2014-07-30 18:46:18+02	2014-07-30 19:02:45+02	00:16:27
	BGO	Threshold of BGO energy sum trigger increased to from -118 mV to -200 mV, effective threshold increased from ~120 MeV to 170 MeV							2014-07-30 18:46:10+02	Hannappel, Levi Sandri, Bauer and Freyermuth	
18132	bgo_tagger	copper 50 mu	H2 full	0.216T	211pA		116178	257Hz	2014-07-30 18:38:11+02	2014-07-30 18:45:43+02	00:07:32
18131	bgo_tagger	copper 50 mu	H2 full	0.216T	297pA		1001310	2141Hz	2014-07-30 18:30:23+02	2014-07-30 18:38:11+02	00:07:47
18130	bgo_tagger	copper 50 mu	H2 full	0.216T	291pA		1002290	2046Hz	2014-07-30 18:22:13+02	2014-07-30 18:30:23+02	00:08:09

Figure 4.19: The runDb website

The performance may be limited at various stages of the acquisition. Possible limitations are:

- *Slow modules* may increase the needed readout and dead time.
- *Insufficient network bandwidth* can cause the buffers in one LEVB to fill, leading to longer dead times once no free buffers are available.
- *Slow writing to hard disk* causes the buffers in the event saver to fill. When all buffers in the event saver are full, the LEVB buffers are filled.
- *Slow processing* in either the event saver or any LEVBs also results in full buffers.

The performance characteristics of the DAQ depend strongly on the experimental conditions. Section 4.8.1 provides an overview of these for the existing data as well as expectations for future data taking runs. Section 4.8.2 details the performance characteristics related to the event saver, while section 4.8.3 focuses on those of the LEVBs.

4.8.1 Experimental conditions

Conditions for test acquisitions

For tests of the DAQ system, data taking was performed without beam. These runs typically acquire data from all available detectors. The absence of the beam leads to less

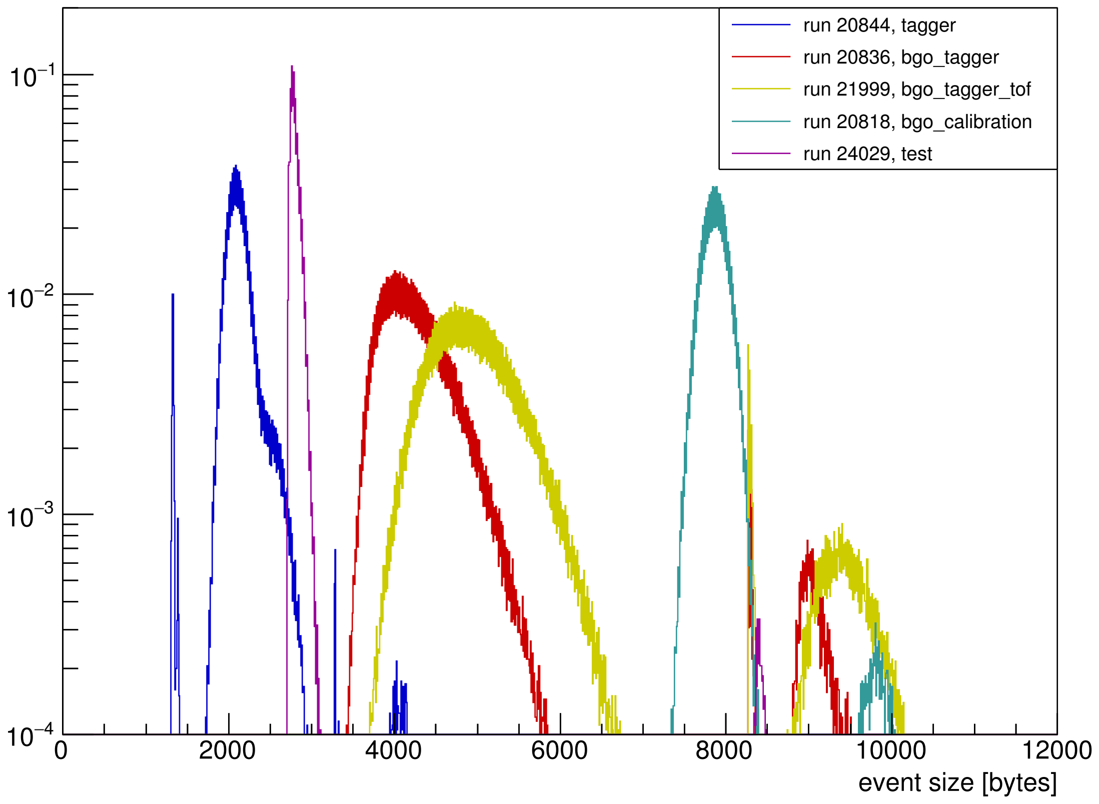


Figure 4.20: Event sizes for different data taking conditions
For better comparison the integral over the distributions has been normalized to 1.

particles depositing energy in the detectors, as well as less noise on the signal lines. This results in a significantly smaller event size than for data taking runs, e.g. *bgo_tagger* or calibrations, as illustrated in figure 4.20. The *test* run type uses the fixed life time trigger with a minimal life time, allowing for the maximum event rate possible reading the full setup. Due to this high event rate, the data rate for the *test* runs is comparable to *bgo_tagger* runs.

Conditions during beam time

During beam times the conditions vary according to the requirements of the experiment. The extracted electron beam energy only has a minor influence on the data acquisition, whereas variations in the extracted beam intensity and the used radiator target have significant influences on the rates in the detectors. Figure 4.21 shows the rates for several of the local triggers during the November 2014 beamtime.

The trigger logic used during a beam time is chosen according to the final states of the investigated reactions. Usually a hit in the tagging system is required in coincidence with one or more hits in the central detector and/or in the forward spectrometer. During the November 2014 beam time the data taking run type used was *bgo_tagger* with either liquid hydrogen or deuterium as target. The trigger condition imposed by these runs

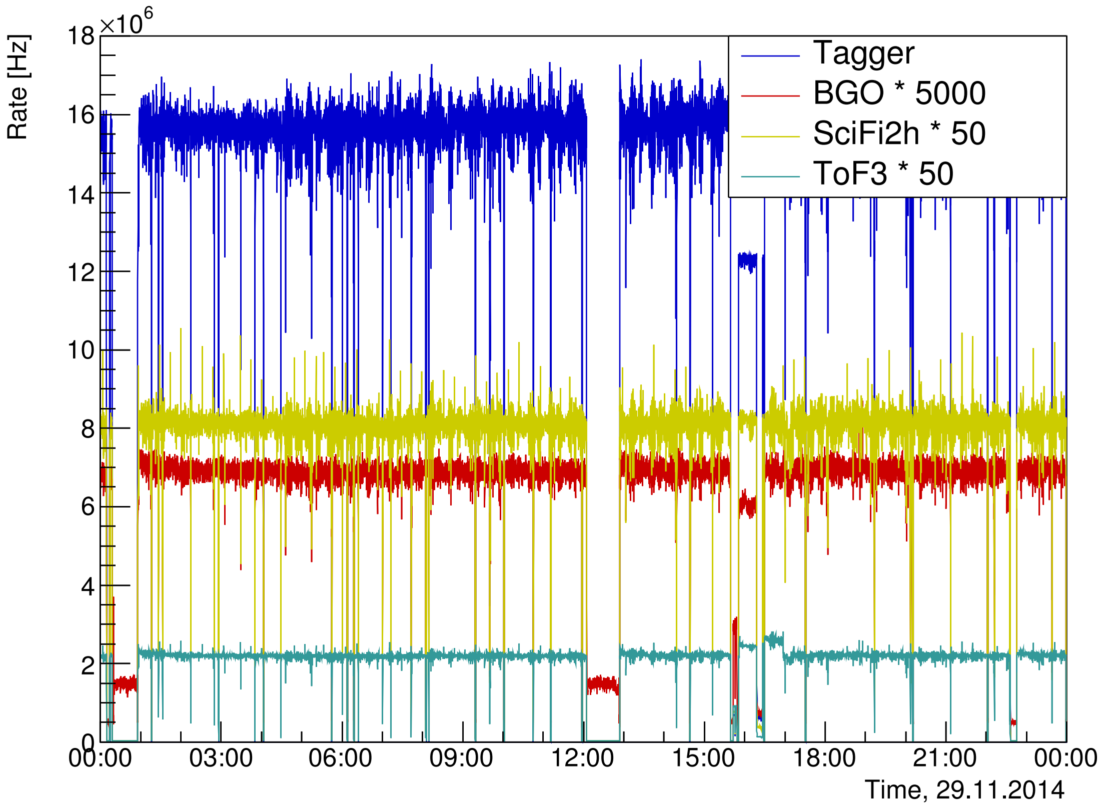


Figure 4.21: Rates of the Tagger, BGO, SciFi2h and ToF3 local triggers. The local trigger rates are shown for one day of the November 2014 beamtime. No beam was extracted from the accelerator in the periods shortly after midnight and twelve o'clock as BGO calibrations were performed. At sixteen o'clock runs were performed using a different bremsstrahl radiator. Short drops in the rate are mostly due to the accelerator not being filled correctly for a single spill.

requires a signal from one of the tagger local triggers in coincidence with a local trigger signal from the Rugby Ball. For the *bgo_tagger* condition trigger attempt rates of more than 1200 Hz have been reached,

For data taking with a carbon target a more restrictive trigger condition was used. The run type *bgo_tagger_tof* requires a signal from the ToF local trigger in addition to the coincidence between the tagger and the Rugby Ball. Due to this additional requirement and the use of the carbon target, the event size for a *bgo_tagger_tof* run are slightly larger than for *bgo_tagger* runs.

To study systematic effects in the determination of the absolute photon flux and the linear polarization of the photon beam dedicated runs were taken on a regular basis. These *tagger* runs require only signal from one of the tagger local triggers for their trigger condition. In addition only the readout of the ToF and Trigger LEVBs has been enabled for this run type. This leads to significantly smaller event sizes, as seen in 4.20.

Conditions for BGO calibration runs

The special run type *bgo_calibration* has been created for the calibration of the Rugby Ball calorimeter. As explained in the detector description in section 2.3.3, ^{22}Na sources are used for this. This run type uses the same fixed life time trigger with minimal delay as the test runs, but differs in the readout configuration from all other run types. To be able to detect the small signals from the 1.27 MeV line of the ^{22}Na sources the internal thresholds of the BGO SADCs are reduced and the attenuation in the mixer modules switched off. In addition, the allowed time window is significantly increased, as the trigger is uncorrelated to the signals from the sources. As shown in figure 4.20, this produces significantly larger events.

4.8.2 Performance of the event saver

To investigate the performance of the event saver it is useful to look at the buffer occupancy. As long as the event saver has enough free buffers it is able to accept data from the LEVBs. Figure 4.22 shows the buffer usage for a test run. The numbers of free buffers on the LEVBs is shown as a solid line. The dashed lines are the number of free `rawDataBuffer` objects for each levb on the event saver. The number of free buffers are limited to the total number of buffers allocated, 6000 and 3000 respectively. The number of free `CEvent` buffers is shown in black, labeled event buffers. This is limited to a maximum of 1000. The time needed to execute the `Fill()` function is shown in magenta, labeled write times.

For the *test* run type the number of available buffers on the LEVBs and on the event saver never reaches zero. Drops in the number of free event buffers can be seen. Figure 4.23 shows this regular structure in more detail. As this is the last level of buffers this structure can be directly correlated to longer write times. These varying write times are caused by file management done internally by ROOT. The data for several events is first stored to a ROOT internal buffer. This buffer is regularly compressed and written to disk, causing delays. As the number of free buffers never reaches zero and fully recovers

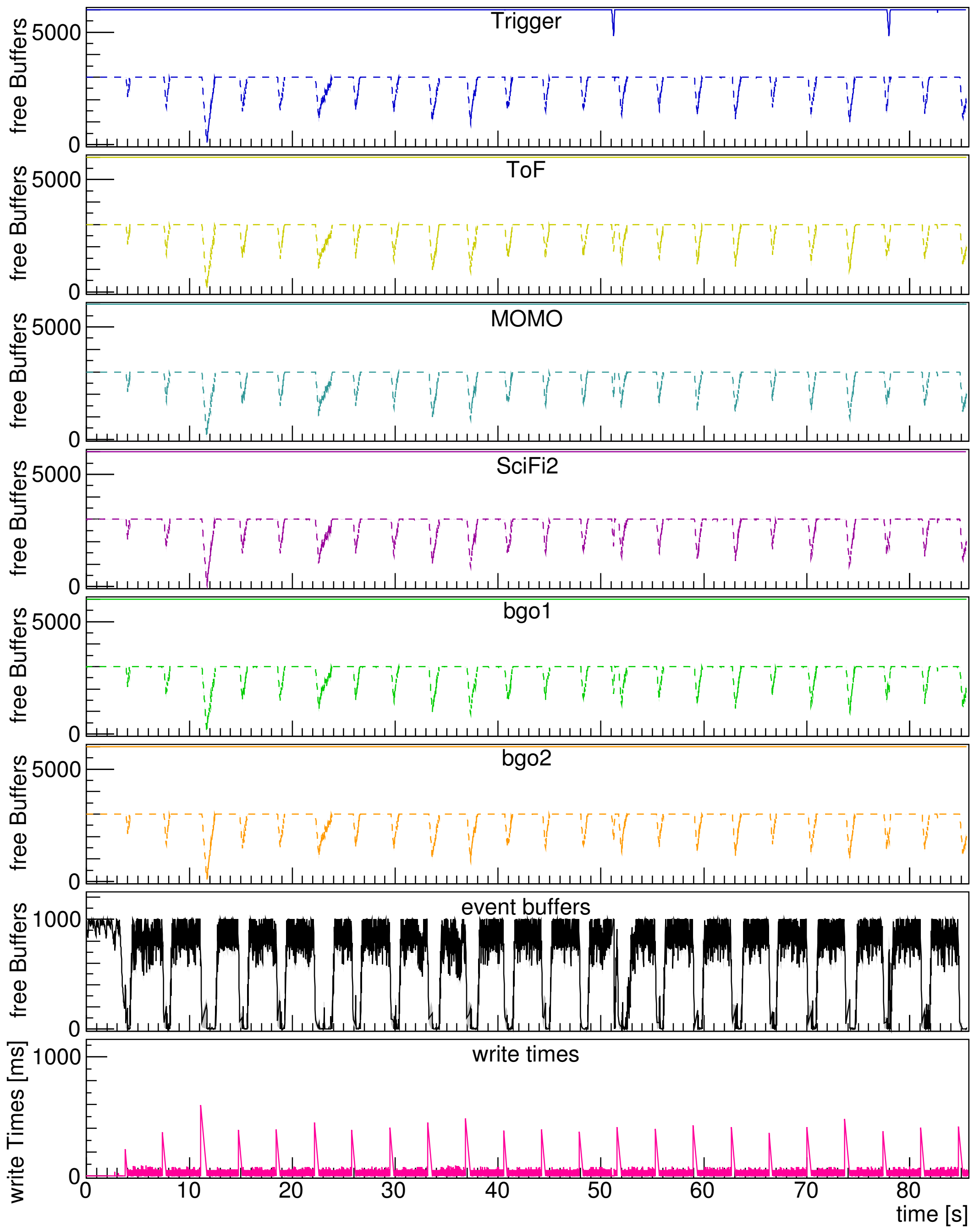


Figure 4.22: Buffer occupancy as a function of time for the *test* run 24029

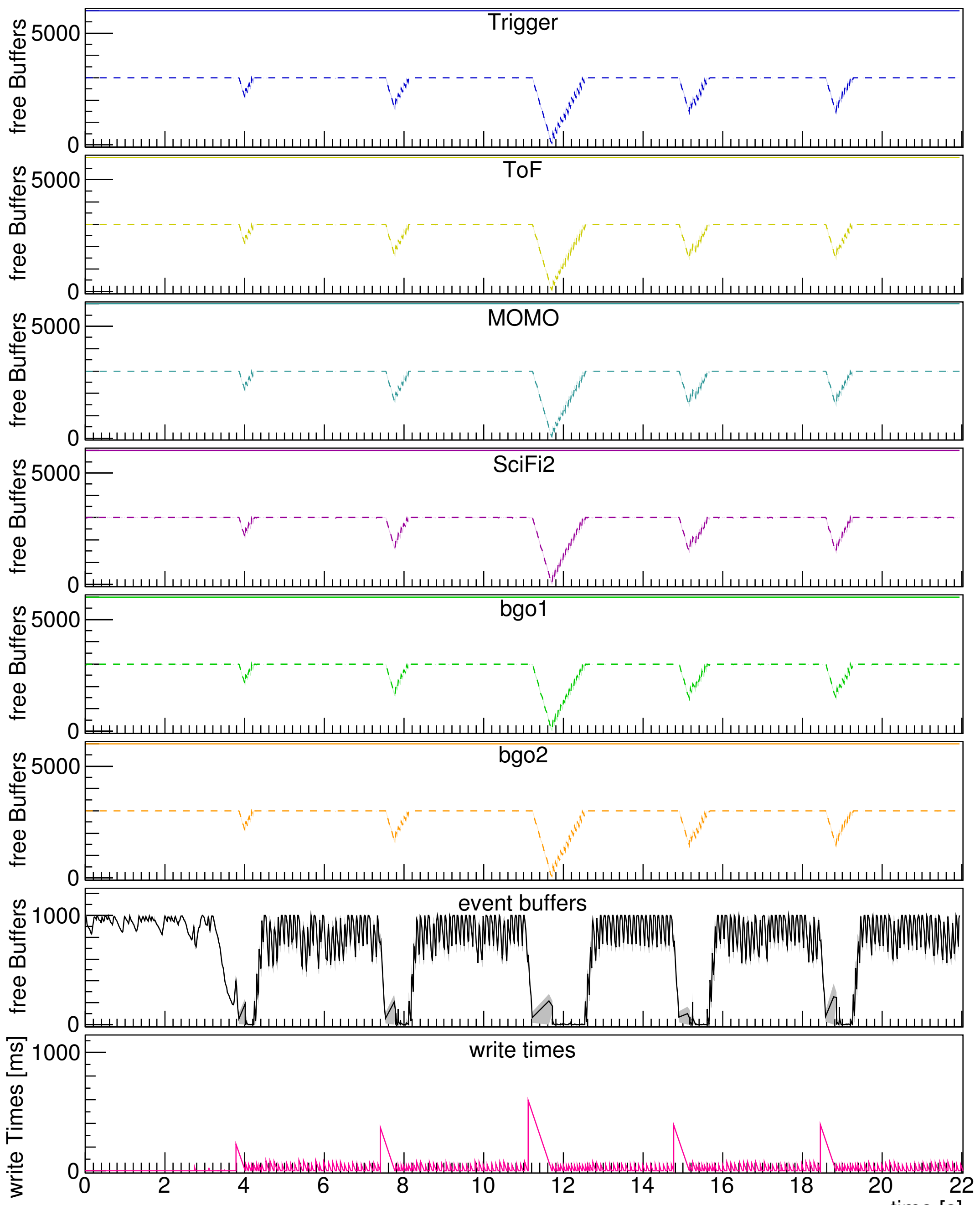


Figure 4.23: Buffer occupancy as a function of time for the first 22 s of the *test* run 24029

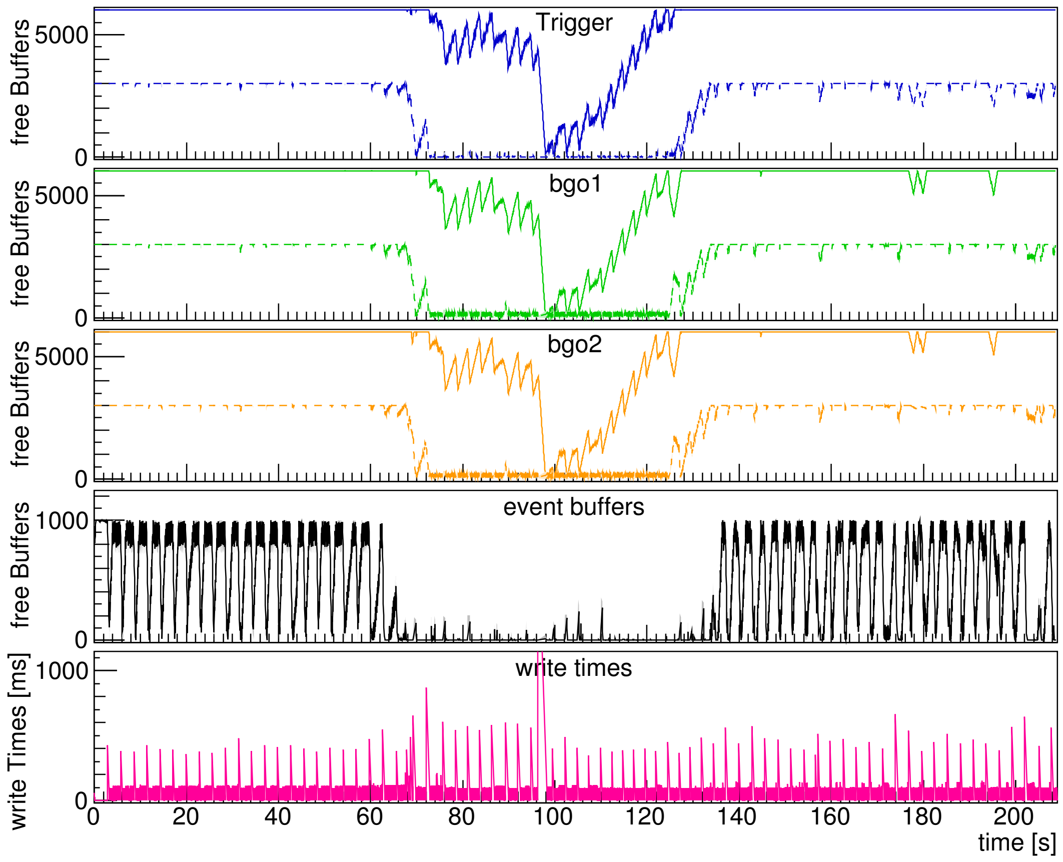


Figure 4.24: Buffer occupancy as a function of time for the *bgo_calibration* run 20818

after each of the delays, the total performance of the DAQ system is not influenced by this.

For the BGO calibration runs, the situation is similar as shown in figures 4.24 and 4.25. However the average write times are larger. As visible in the region around 100 s, this cannot always be compensated by the buffers in the event saver. In this case the number of free buffers in the LEVBs drops. Only when this number reaches zero the readout is delayed.

The examples demonstrate a sufficient performance of the event saver. During normal conditions the event saver is able to write all data to disk without causing additional delays. The number of buffers provided is sufficient to allow for the variations in write times introduced by the ROOT framework. Only in conditions where a high event rate is combined with very large event sizes the evs can introduce short delays into the readout. As these conditions do not occur during regular data taking runs, such as *bgo_tagger*, the performance of the evs can be regarded as sufficient.

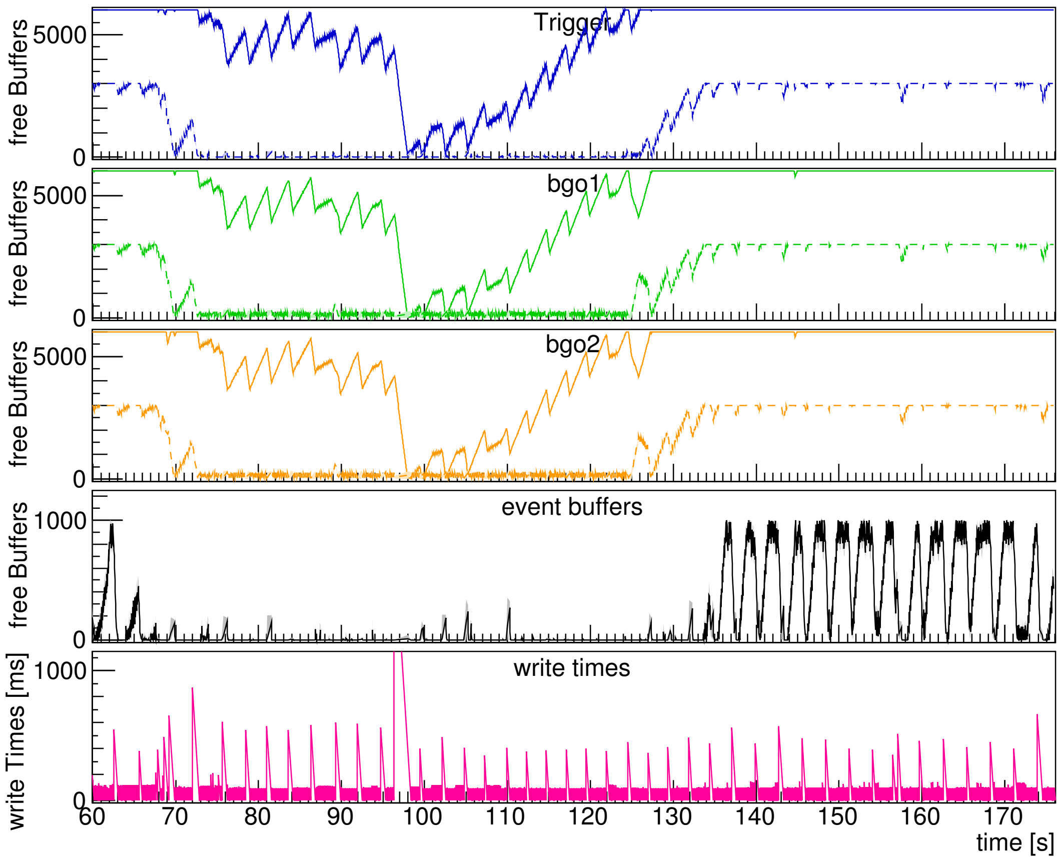


Figure 4.25: Buffer occupancy as a function of time for 50 s to 110 s of the *bgo_calibration* run 20818

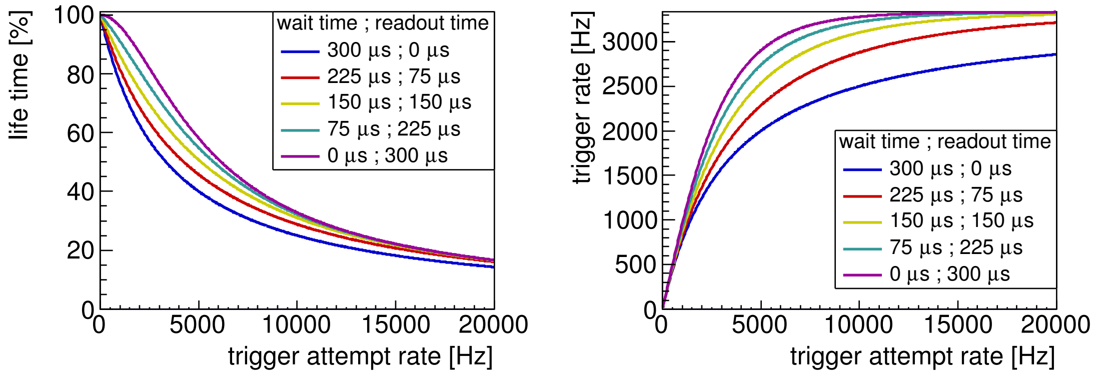


Figure 4.26: Calculated life time and accepted trigger rate as function of the trigger attempt rate

4.8.3 Performance of the LEVBs

The most important performance figure for the DAQ system is the achieved life time. This life time is however dependent on the trigger attempt rate. In the case of a dead time d that is independent on the trigger attempt rate f, the life time l can be calculated according to equation 2.2 (given in section 2.5).

$$l = \frac{1}{1 + fd}$$

However the dead time may be increased if the readout of the previous event is not finished when a new trigger is accepted. As the probability for this to occur depends on the trigger attempt rate, the dead time becomes rate dependent. Using the time w waiting for modules and the time r needed to finish their readout the dead time can be calculated as

$$d(r) = w + \left(r + \frac{e^{-rf} - 1}{f} \right) \tag{4.1}$$

This leads to a life time of

$$l = \frac{1}{wf + rf + e^{-rf}} \tag{4.2}$$

The accepted trigger rate f_a is given by

$$f_a = l \cdot f = \frac{1}{w + r + e^{-rf}/f} \tag{4.3}$$

Figure 4.26 illustrates the life time and accepted trigger rate achieved for a fixed total time of $w + r = 300\,\mu s$. The calculation has been performed for different splits between w and r. This shows a clear advantage in minimizing the time waiting for modules at moderate trigger attempt rates. However, for very high trigger attempt rates the curves approach the same limit.

Figure 4.27 shows the dead time distribution for a *bgo_tagger* run, separated for

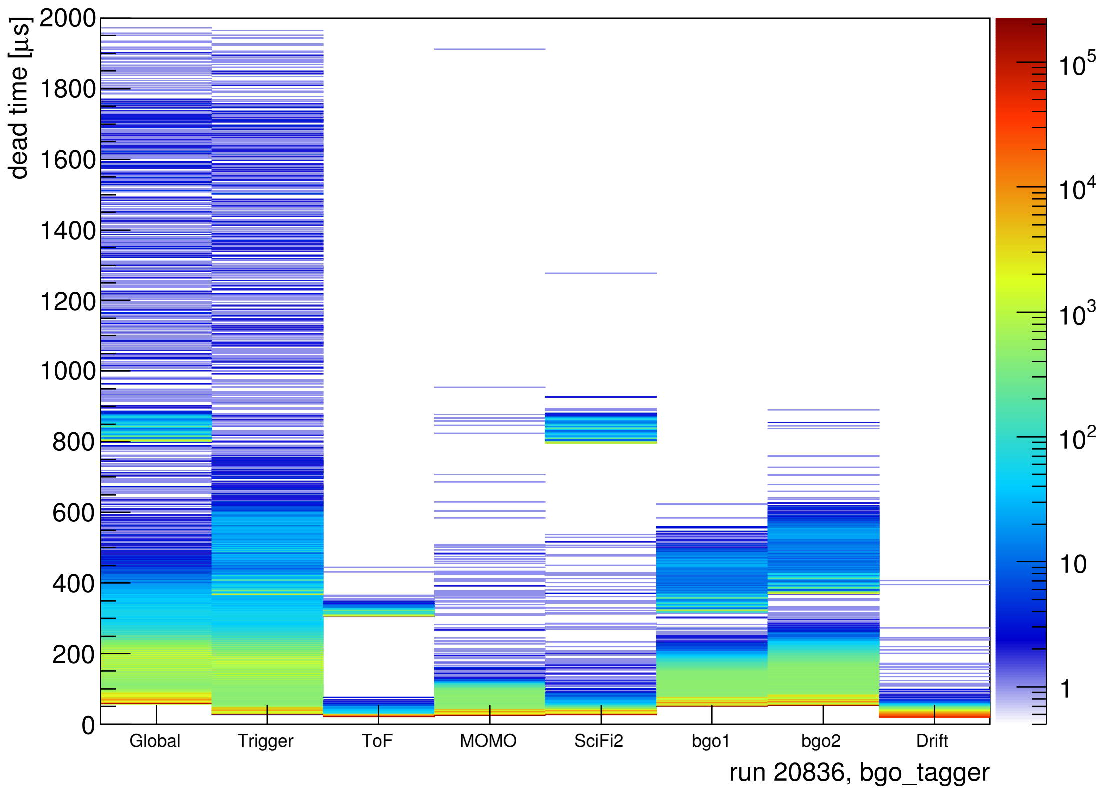


Figure 4.27: Dead time distribution per LEVB

the individual LEVBs. The dead time distribution is similar for most LEVBs. These distributions peak at their minimum value. This minimum is below 50 µs for Trigger, ToF, MOMO, SciFi2 and Drift, and around 50 µs for bgo1 and bgo2. This time can be identified with the wait time w. The Tof, SciFi2 and bgo LEVBs show an additional peak at 300 µs, 800 µs and 500 µs respectively. These peaks are caused by the readout of the included scaler modules during the wait time. The structure around 500 µs in the dead time of the trigger LEVB is caused by the readout of the trigger calibration TDCs every ten events. As the global dead time is given by the slowest LEVB it follows the distribution of the bgo or trigger LEVBs for normal events and the SciFi2 distribution for scaler events.

As already seen in figure 4.26, the performance of the DAQ system is not only dependent on the wait time w but also on the time r needed to read the module data. Figure 4.28 shows these readout times for a *bgo_tagger* run. The Drift LEVB has only a minimal readout time, as most of the readout is already performed within the dead time. While the readout times for the MOMO and bgo LEVBs are typically at 100 µs, the ToF and SciFi2 LEVBs have readout times between 30 µs and 40 µs. The trigger LEVB has the largest distribution of readout times. It is centered at $\approx$ 175 µs but extends up to several hundred µs for a small fraction of the events.

During this run a life time of 90.8 % at a trigger rate of 846 Hz was achieved. This is

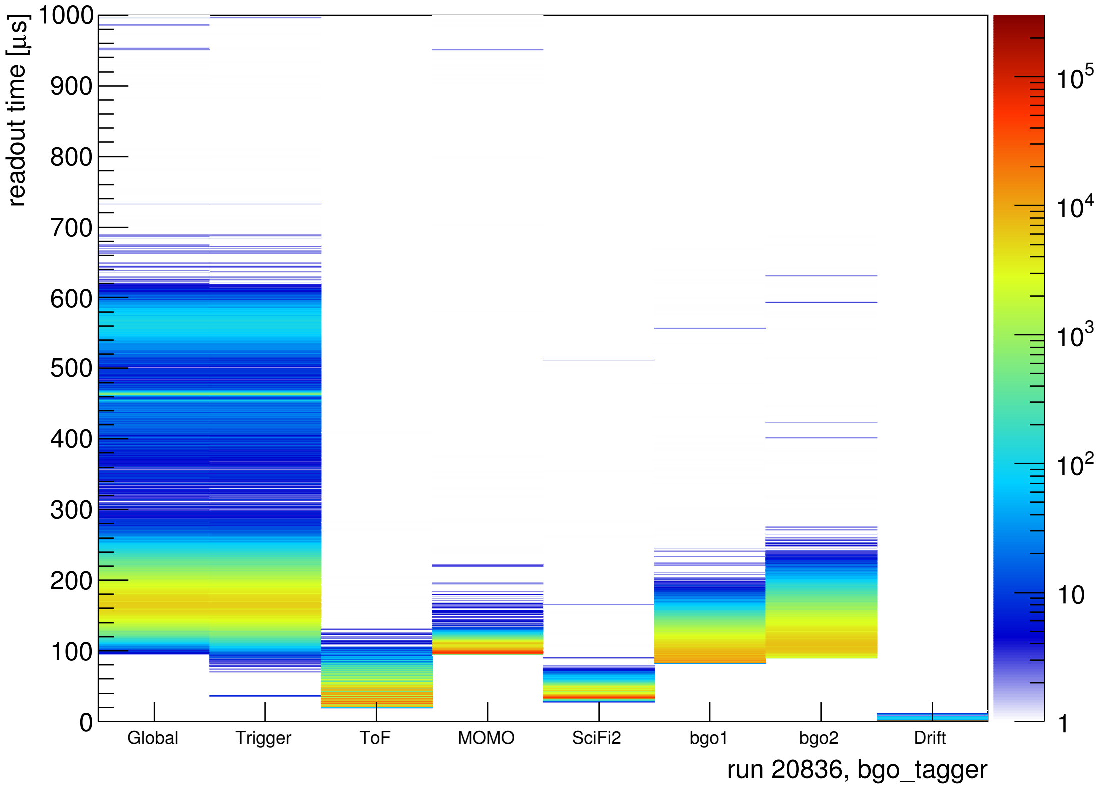


Figure 4.28: Readout time distribution for different LEVBs

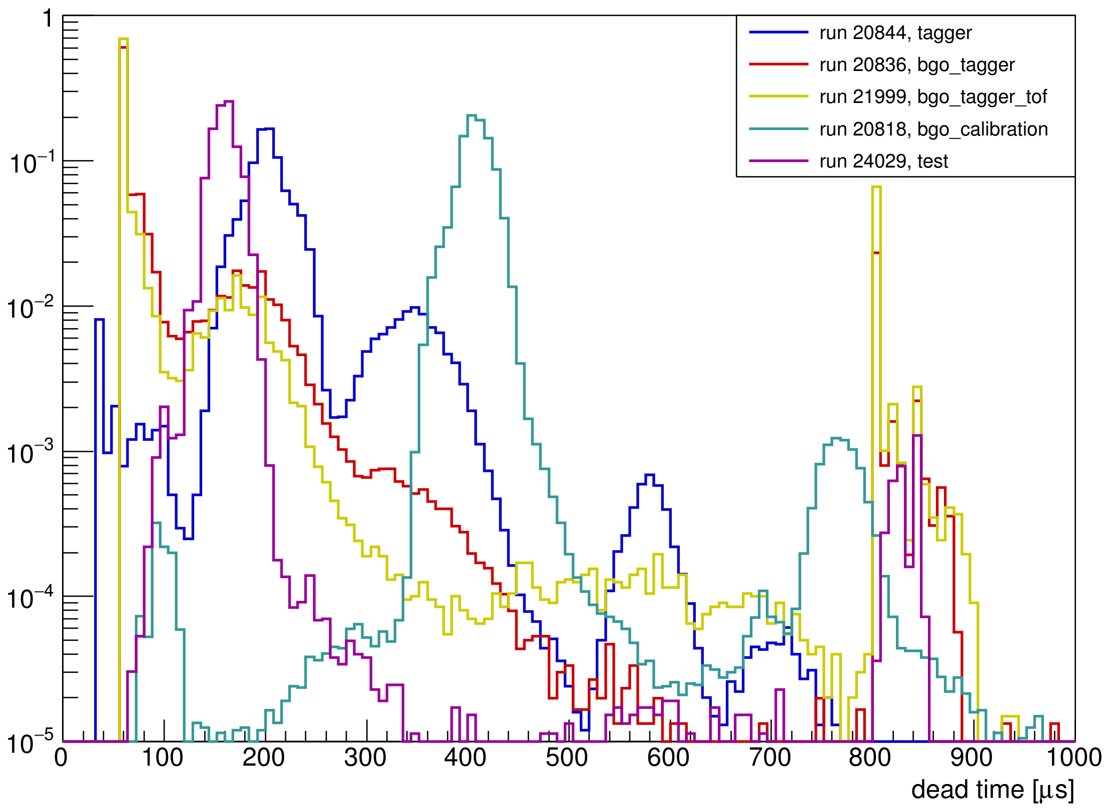


Figure 4.29: Dead time distribution for different run types
For better comparison the integral over the distributions has been normalized to 1.

compatible with the expectations from calculations using the wait and readout times. For the full set of 220 runs with comparable conditions these values vary from 89.5 % to 91.2 % and 817 Hz to 876 Hz respectively. These variations are mostly due to variations in the extracted electron beam.

The achieved values however depend strongly on the experimental conditions. Figure 4.29 shows the distribution of the total dead time per event for different run types. The distribution for the *bgo_tagger_tof* run is similar to the *bgo_tagger* run. Due to the slightly larger events the dead times of the *bgo_tagger_tof* run extend to higher values for data events. The *tagger* run shows a different pattern. In this case the minimum dead time is lower since only the Trigger and ToF LEVBs are included in the readout. However, this minimum dead time occurs less frequently as the trigger attempt rate is much higher. This effect is even more pronounced for the *bgo_calibration* and *test* runs. These run types use the fixed life time trigger with a minimum life time to achieve the maximum event rate possible. The main peak of the distribution for the *test* run is located around 175 µs, corresponding to the readout times required. For the *bgo_calibration* runs the required readout times are larger resulting in a peak around 400 µs.

5 First data of the BGO-OD experiment

This chapter shows data from the first beamtimes of the BGO-OD experiment. This data, acquired using the DAQ software developed within this thesis, shows the successful commissioning of the BGO-OD experiment and of its data acquisition in particular.

To illustrate the effect of the trigger logic on the acquired data, section 5.1 shows timing spectra for some local trigger signals. Section 5.2 follows with the data acquired for the individual subdetectors. This is followed in section 5.3 by the first steps toward an extraction of hadronic reactions from this data.

Plots showing positions or angles use the global experiment coordinate system. The origin of this system is located in the target center. The z-axis points along the photon beam direction, the y-axis upwards and the x-axis horizontal providing a right handed coordinate system. The polar angle θ is the angle between a particle trajectory and the beam axis. The azimuthal angle ϕ gives the rotation around the beam axis, with $\phi = 0$ pointing in positive x direction.

5.1 Trigger

5.1.1 Single local trigger

To understand and correctly interpret the data provided by the data acquisition system it is necessary to know how this data is influenced by the data acquisition itself. As the trigger decision made by the data acquisition is the reference time for all performed time measurements it has a significant impact on all data.

Figure 5.1 shows the measured time distribution of the local trigger signal of the vertical tagger part. The global trigger condition for this run was an `OR` of the vertical and horizontal tagger local trigger signals. As the rate in the vertical part of the tagger is much larger than in the horizontal part, the vertical part dominates the trigger condition.

The time distribution shown in figure 5.1 has several structures. The most prominent of these is the so called prompt peak located at 0 ns. This peak contains the local trigger signals that caused the global trigger signal. As the absolute values of the measured times are dependent on arbitrary delays, e.g. cable length, all measured times are shifted by a fixed, channel dependent offset. These additional offsets are calculated to shift the prompt peaks to the zero position. As the output signal of the global trigger is generated synchronously to a 200 MHz clock, a jitter of up to 5 ns between the local and the global trigger signal is introduced. This broadens the prompt peak to slightly more than 5 ns.

Gaps in the time distribution are seen in front and behind the prompt peak. These are caused by the dead time of the local trigger signal. As all events contain a signal in the prompt peak, the distance of the last signal before the prompt peak to the prompt peak must be at least the dead time of the local trigger module. The gap after the prompt peak follows with the same reasoning. After the signal in the prompt peak, a minimum of the dead time of the local trigger module must pass before another signal can occur. In figure 5.1 the dead time gaps around the prompt peak are not completely empty. This is due to the fact that the selected trigger condition is the `OR` of the two tagger local triggers. When the global trigger was caused by the horizontal tagger part, no hit in the prompt peak of the vertical tagger part is required. In this case there may be uncorrelated hits within the dead time gaps.

In front of the prompt peak an exponential falloff of the background is observed. Hits occurring at this time would have caused a trigger signal if the global busy was not set. These hits would then be found in the prompt peak instead of the background in front of it. The closer one gets to the prompt peak the more likely it is that the global trigger is no longer busy. This causes the falloff to follow an exponential decrease with a time constant determined by the average rate.

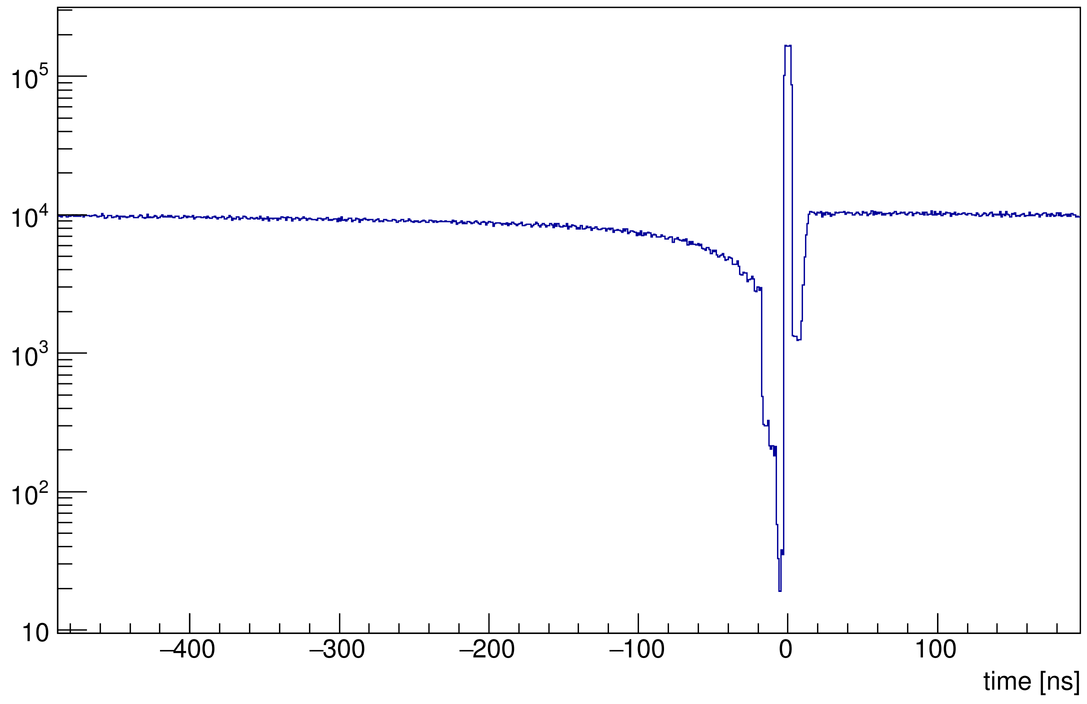


Figure 5.1: Time distribution of the vertical tagger local trigger signal

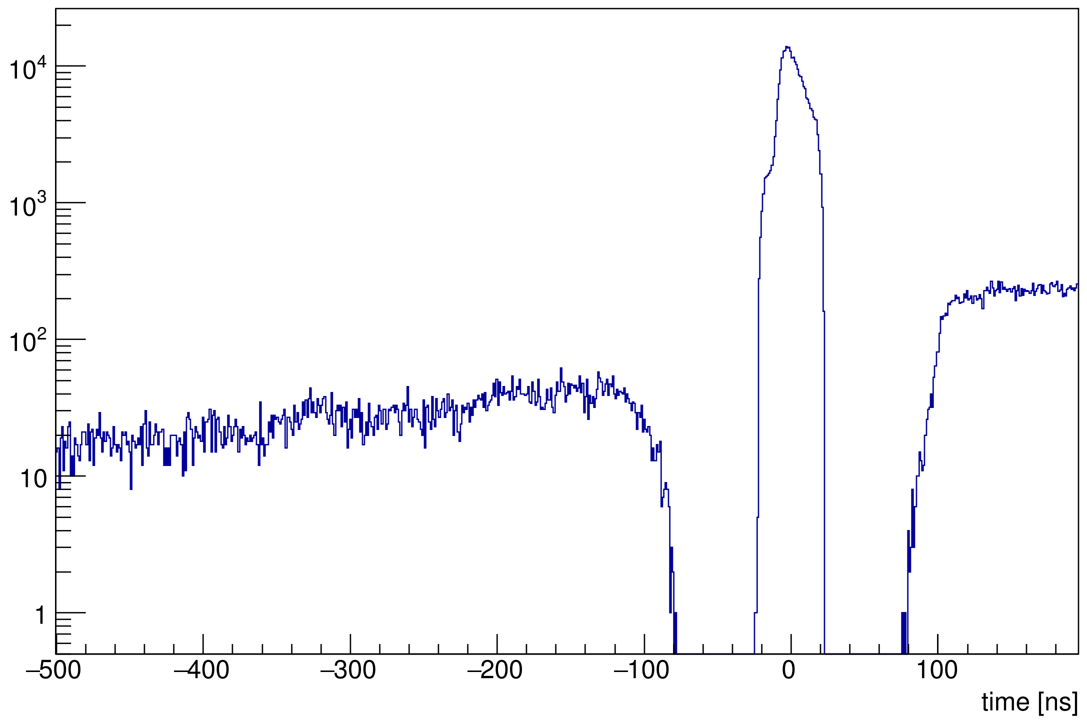


Figure 5.2: Time distribution of the BGO local trigger signal

5.1.2 Combination trigger

When using combination triggers, additional modifications to the local trigger time distributions can be seen. Figure 5.2 shows the time distribution of the BGO local trigger signal during a run triggering on the coincidence of the tagger local trigger and the BGO local trigger. Similarly to the single trigger case, the most prominent feature of the spectrum is the prompt peak surrounded by dead time gaps. However the width of the prompt peak is now determined by the length of the coincidence gate, in this case 40 ns. The peak also features structures not present in the single local trigger case. A plateau of full peak width is caused by the accidental coincidences between tagger and BGO local trigger signals. On top of these accidental coincidences a peak formed by the real coincidences can be seen.

5.1.3 Trigger time correction

As seen already in figure 5.1, the global trigger introduces a 5 ns jitter to the trigger time. This jitter can be removed as the times of the different local triggers signals is known in the analysis. The straightforward way to accomplish this is to search for the first hit in the 5 ns wide prompt peak of the local trigger signal that defines the timing of the global trigger logic. The tagger local trigger is normally used for this, as it has a good time resolution and is included in most run configurations. Figure 5.3 shows the result of this correction on the data presented in figure 5.1. However, due to accidental coincidences the timing of the local trigger signal used for the correction is not necessarily the timing of an interesting reaction contained in the event.

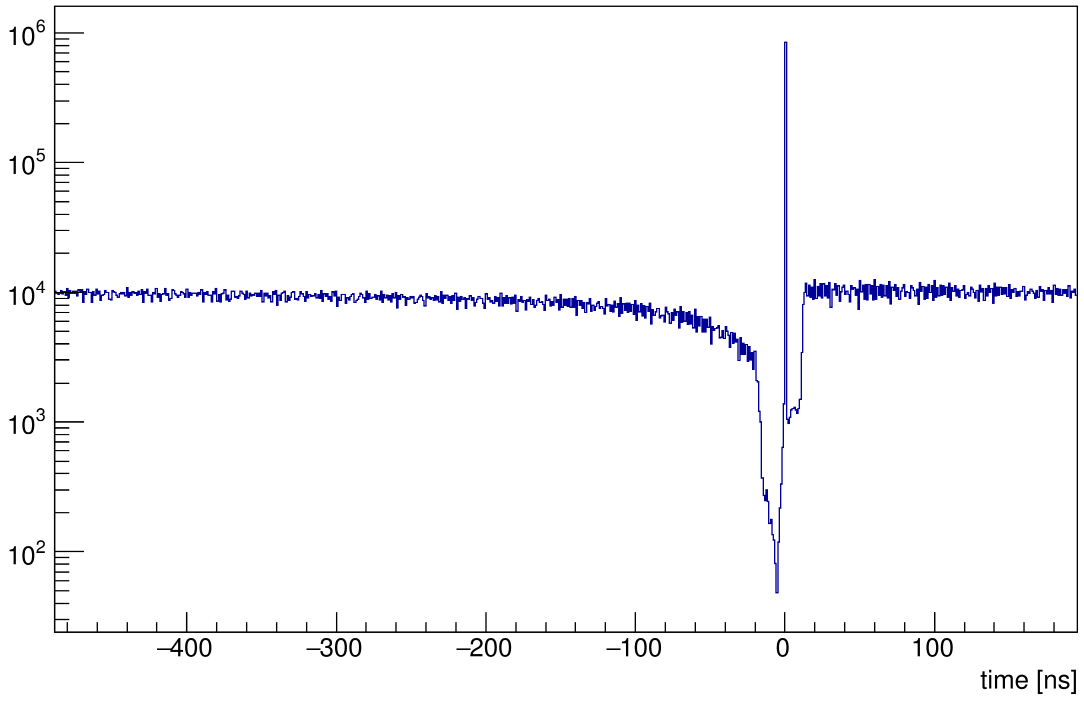


Figure 5.3: Time distribution of the vertical tagger local trigger signal after trigger time correction

5.2 Detector spectra

5.2.1 Tagger

As described in chapter 2.2, the tagger detector is used to determine the energy of each individual beam photon. To avoid misidentification of photons it is essential to understand the timing provided by the detector. Figure 5.4 shows the time distribution of the 120 tagger scintillators. As can be seen from the color code the number of entries is much higher for the channels with higher indexes. This is expected as these channels correspond to the low photon energies and the photon distribution follows an approximate $1/E_\gamma$ behavior. However, for the highest three channels the number of entries drops. For these channels the rate is so high that a significant fraction of the pulses are lost, resulting in the reduced number of entries. As the trigger time correction discussed in the previous section has been applied to this data, the width of

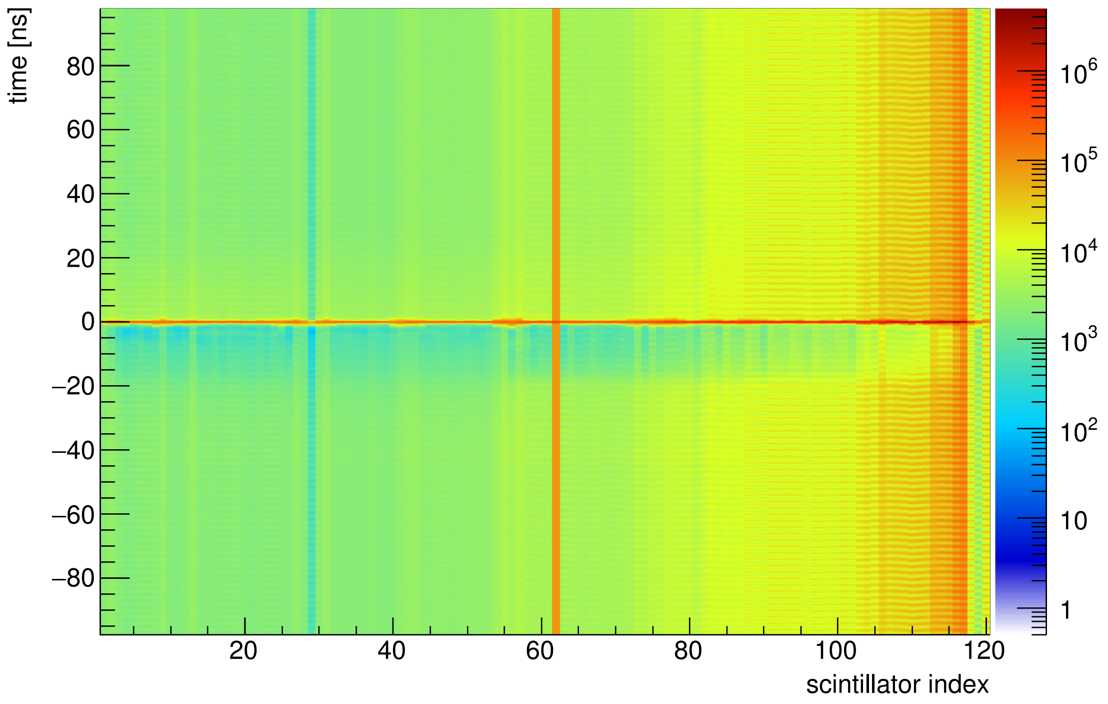


Figure 5.4: Tagger TDC times vs. Tagger scintillator indexes

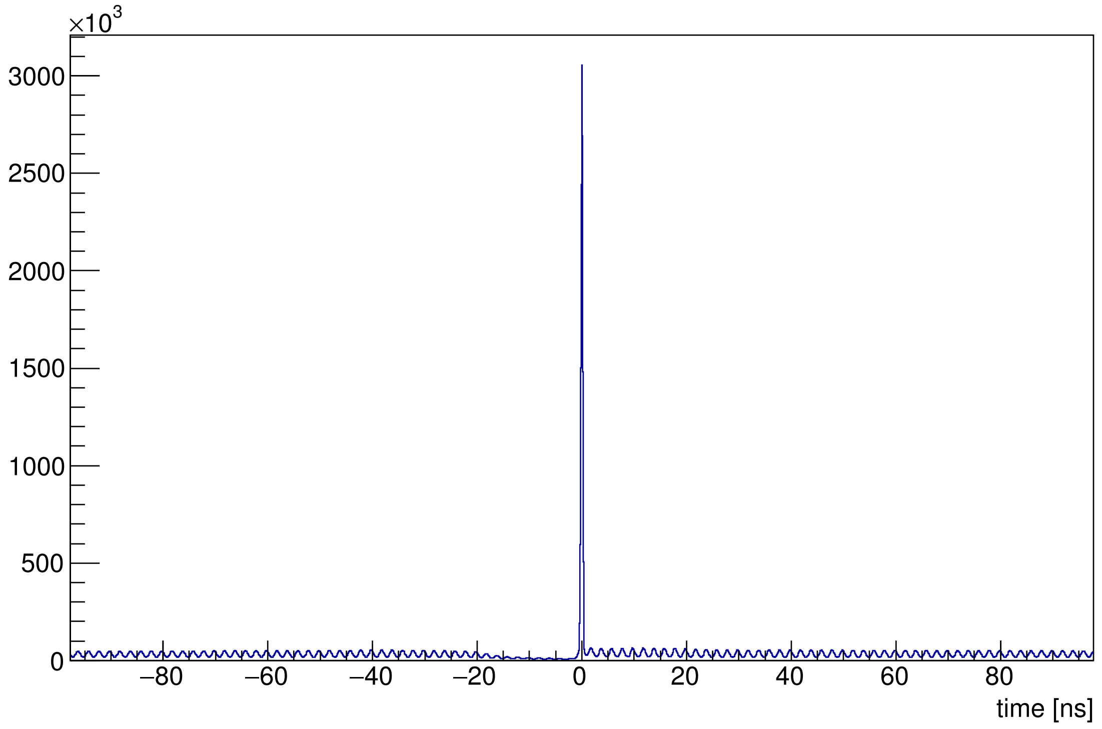


Figure 5.5: Tagger TDC times for Tagger scintillator 111

the prompt peak has been reduced to the intrinsic resolution of the tagger local trigger. The calibration offsets for the individual channels have been adjusted so that the prompt peak is at the same position for all channels.

In the one dimensional plot of a single channel, figure 5.5, the features of the time distribution are more easily visible. In front of the prompt peak the dead time gap is visible. However the dead time gap after the prompt peak is not visible in contrast to the trigger signal. This is due to the high probability of the trigger being caused by a different tagger channel. In the background distribution the 2 ns bunch structure induced by the accelerator on the electron beam is visible, due to the fixed flight path of the electrons and the good time resolution of the detector.

5.2.2 Rugby Ball

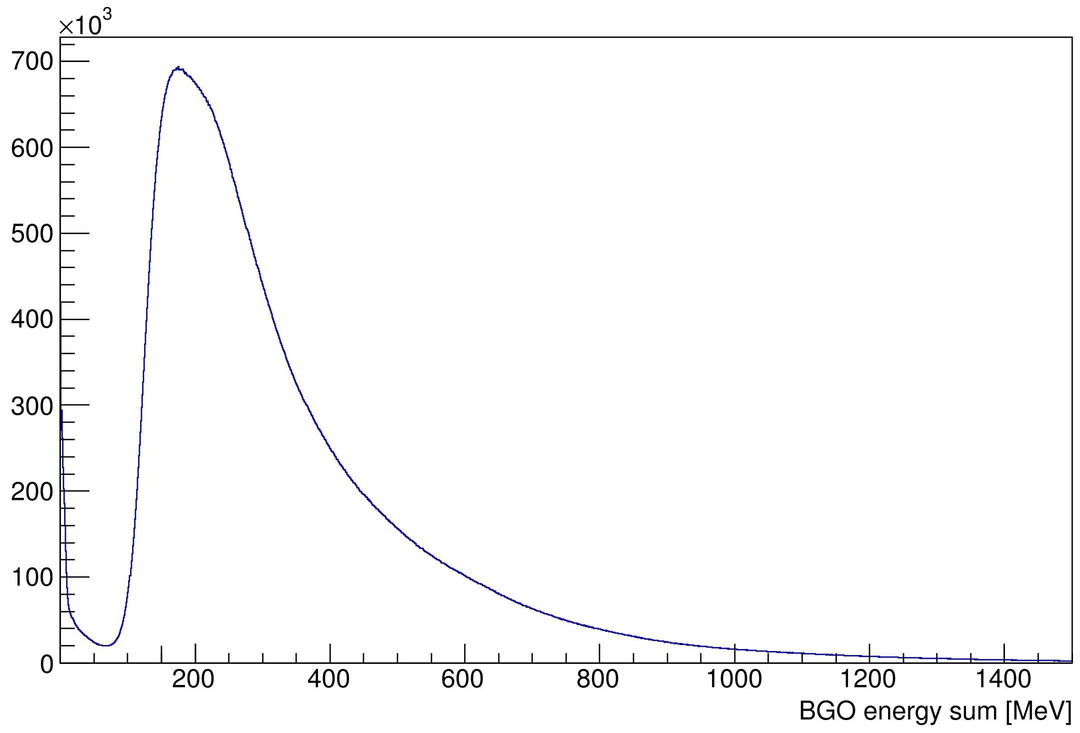


Figure 5.6: Reconstructed energy sum of the Rugby Ball calorimeter

Figure 5.6 shows the sum of the energy deposited in the Rugby Ball calorimeter. Almost no entries are present below 100 MeV. This is the result of the threshold on the analog sum of crystals signal in the BGO local trigger. However the trigger threshold is applied to the amplitude of the summed signal while the deposited energy is proportional to the integral. This leads to an imperfect correlation between trigger signal and energy sum.

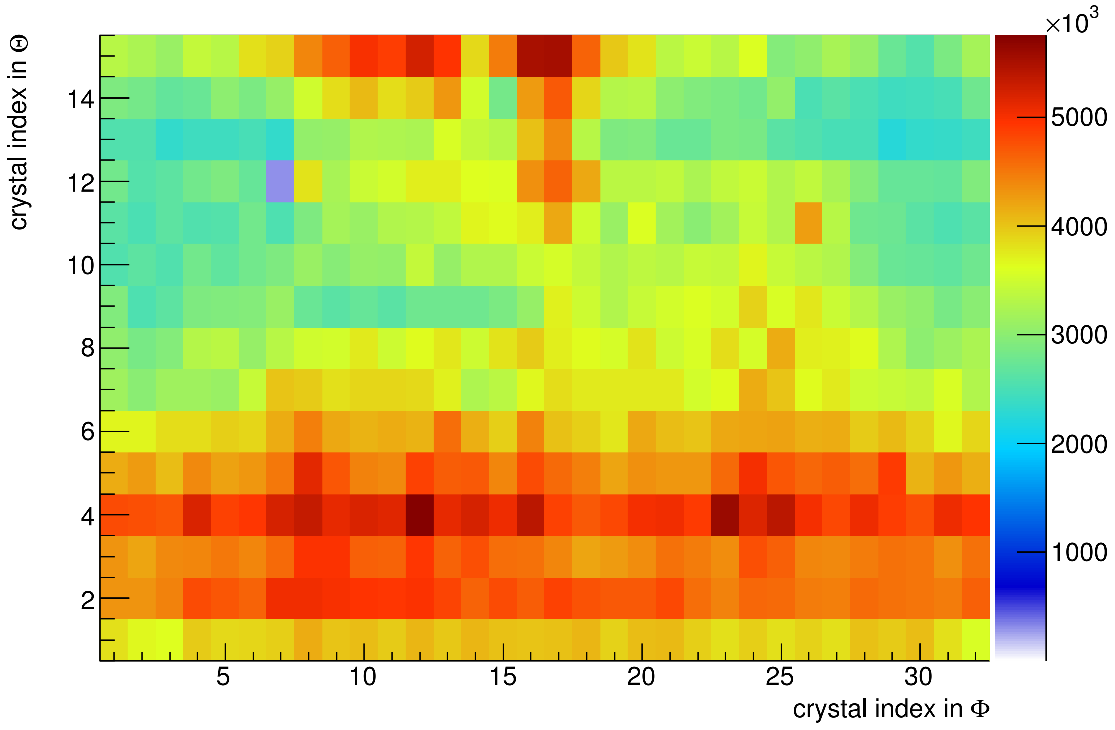


Figure 5.7: Rugby Ball hit distribution

The distribution of the BGO crystal hits is plotted in figure 5.7. As the detector setup is symmetric in the Φ angle, the hit distribution is homogeneous with respect to this angle. In Θ however the experiment is not symmetric due to the Lorentz boost of the particles and the varying crystal sizes.

5.2.3 MOMO and SciFi2

The primary function of the fiber detectors, MOMO and SciFi2, is the determination of the hit position for charged particles. The hit distributions for the different detector layers are shown in figures 5.8 and 5.9. The structures seen in those distributions are mainly due to two effects. First, since all cross sections are forward peaked in the laboratory system due to the Lorentz boost, the rate of charged particles drops for fibers further away from the beam axis. This causes the continuously falling slope towards high fiber indexes in MOMO and to both low and high fiber indexes in SciFi2. Secondly, the detectors have a central hole around the beam axis. Due to this the fibers around this hole are shorter causing the steep drop below fiber index 32 in the MOMO layers and at the center of the SciFi2 layers.

Figures 5.10 and 5.11 show the reconstructed hit positions in the two fiber detectors. Using the hit positions in both detectors it is possible to reconstruct the particle trajectories in front of the magnet.

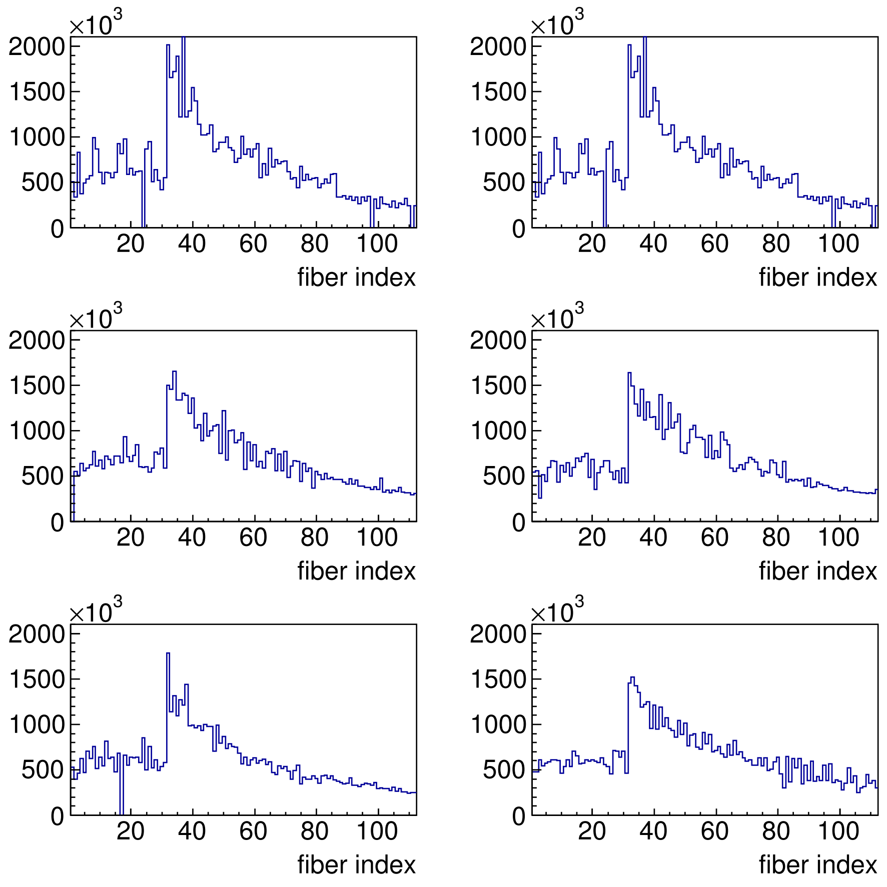


Figure 5.8: Hit distributions of the six MOMO layers

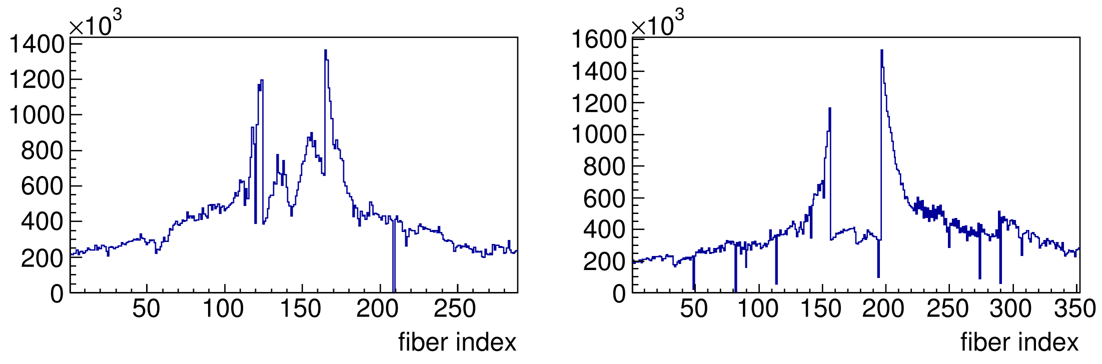


Figure 5.9: Hit distributions of the horizontal (left) and vertical (right) SciFi2 layers

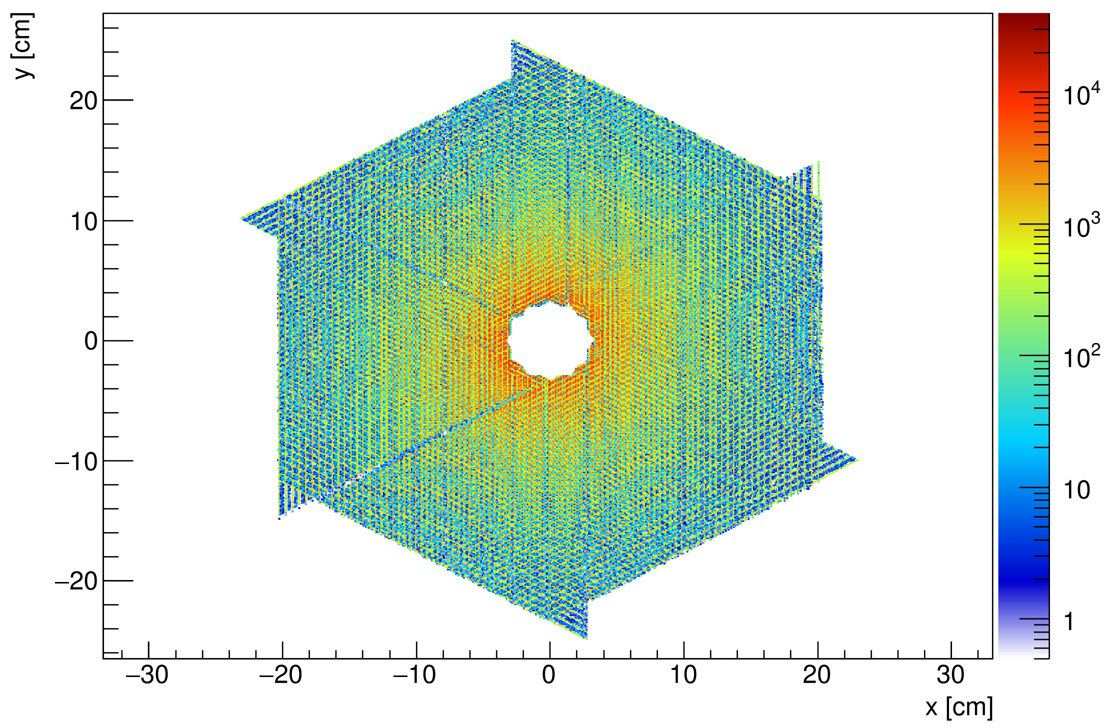


Figure 5.10: MOMO cluster positions

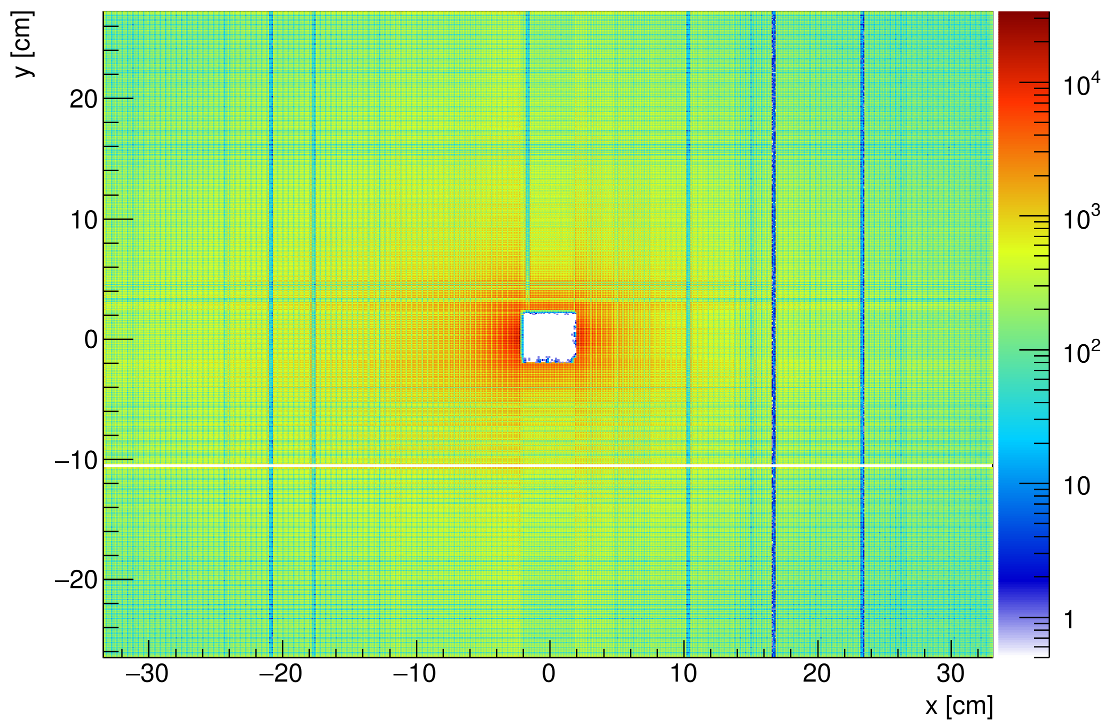


Figure 5.11: SciFi2 cluster positions

5.2.4 Drift chambers

The hit distribution in the drift chambers is similar to those seen in the fiber detectors as shown for the first X drift chamber in figure 5.12. One can see a similar drop towards the outer channels and a decreased rate in the center due to the insensitive spot. However this is true only as long as the magnet is switched off. With magnetic field the particles are deflected by the magnetic field and cover a much larger range in the horizontal plane. This effect is shown in figure 5.13.

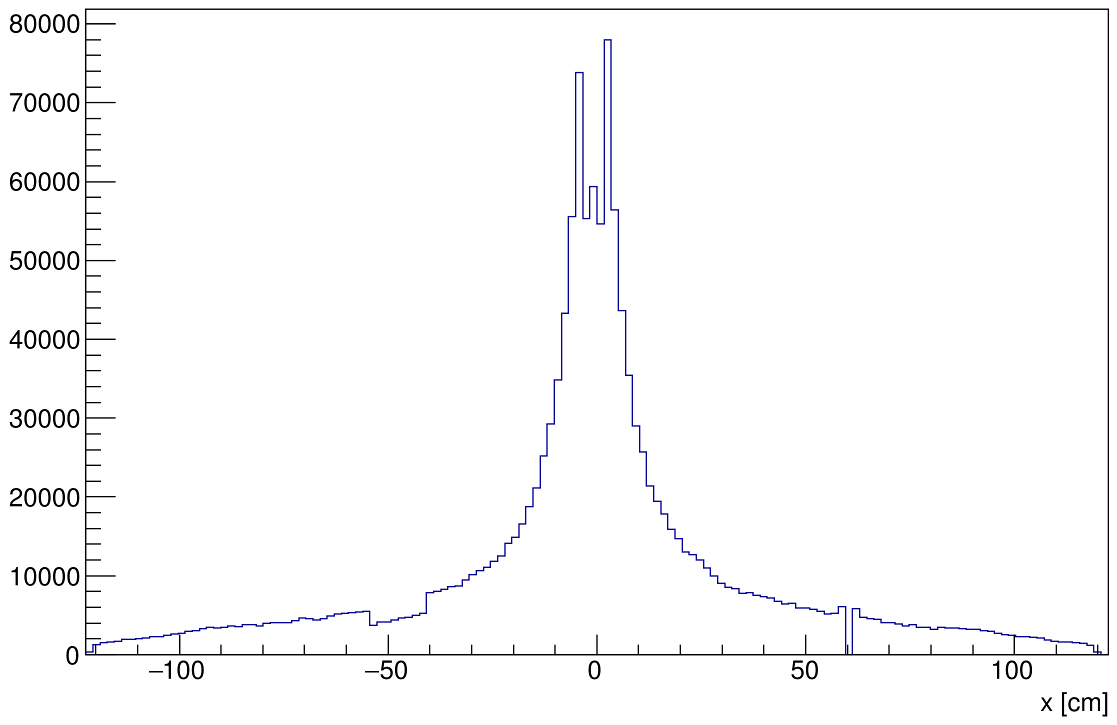


Figure 5.12: Drift chamber hit distributions without magnetic field

As the drift times within the gas volume are large it is not possible to use the times provided by the drift chamber readout to correlate the hits in the drift chambers with other hits. Instead the drift time can be used to calculate the distance from the particle track to the sensitive wire that provided the signal. As figure 5.14 shows, this drift time has a distinct distribution. On top of the uncorrelated background the correlated events have a wide time distribution, spanning more than 400 ns. This distribution has two peaks. The first around 50 ns is created by tracks passing close to a sensitive wire, while the second at 350 ns is created by tracks passing close to one of the field wires. A more detailed explanation of the correlation between the drift times and the distance of the particle track to the sensitive wire can be found in [Sch10] and [Ham08].

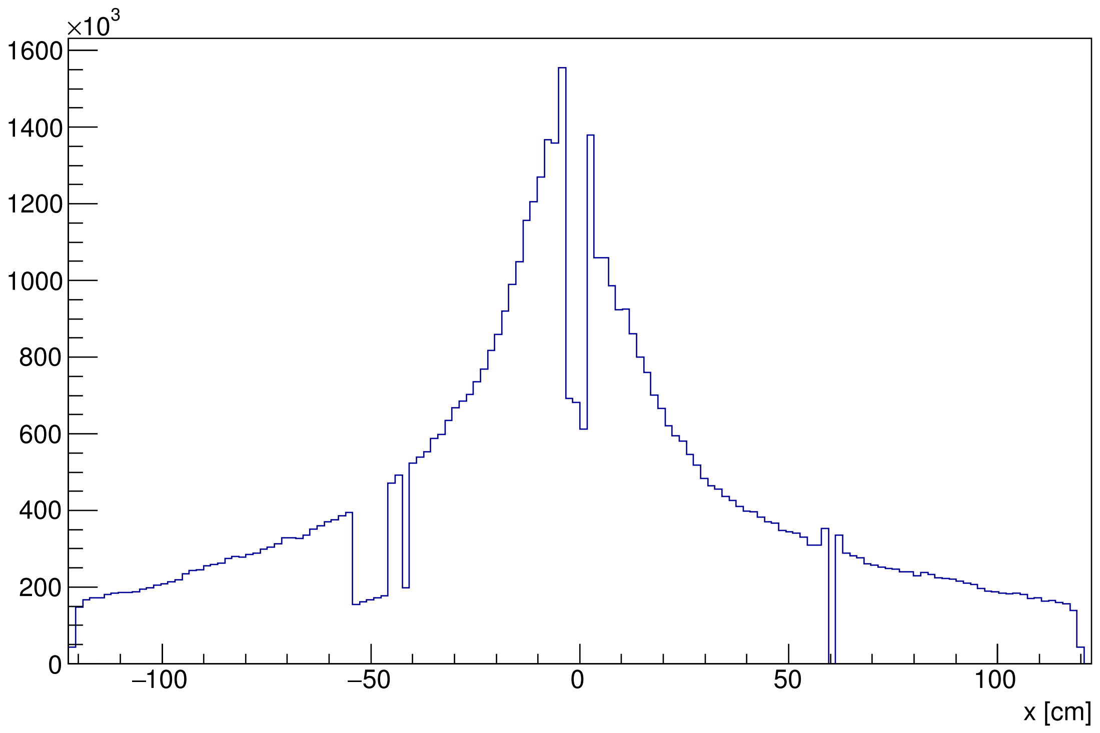


Figure 5.13: Drift chamber hit distributions with magnetic field

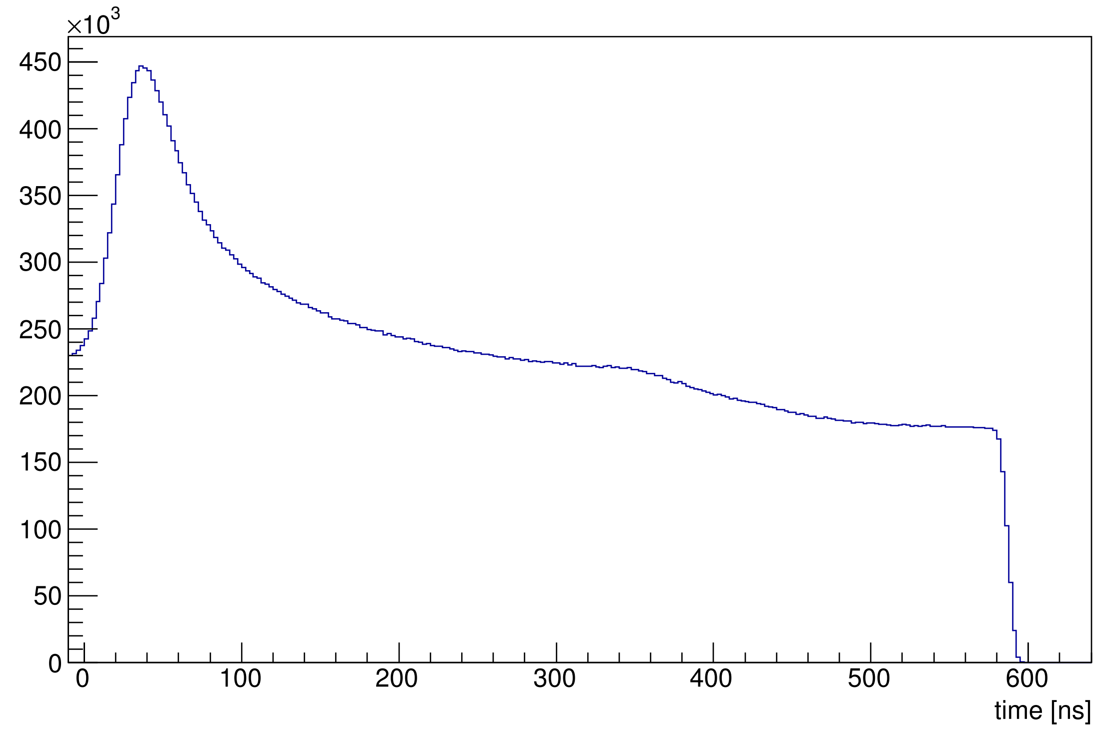


Figure 5.14: Drift time distribution

5.2.5 ToF

The time of flight walls at the end of the experimental setup are used to identify particles by their speed. To achieve the necessary time resolution with a detector as large as the ToF walls, the scintillators must be read out on both sides. Using the mean time between the signals from the two sides the light propagation time inside the scintillator drops out. Figure 5.15 shows the time measured at the ToF walls. To calculate an accurate speed value, not only the time the particle reaches the ToF wall must be measured, but also the length of the flight path must be considered. However, already without correcting for different flight paths a tail caused by slow particles is visible next to the prompt peak.

To correlate the time information with a particle track seen in the drift chambers a position information is necessary. The vertical component is provided by the index of the scintillator. The horizontal component can be calculated using the time difference between the signals on both sides of the scintillator. With the known effective speed of light inside the scintillator a position resolution around 8 cm can be achieved. Figure 5.16 shows the reconstructed positions for the ToF3 wall. Clearly visible are the horizontal bars with the central bar missing to allow for the photon beam to pass trough.

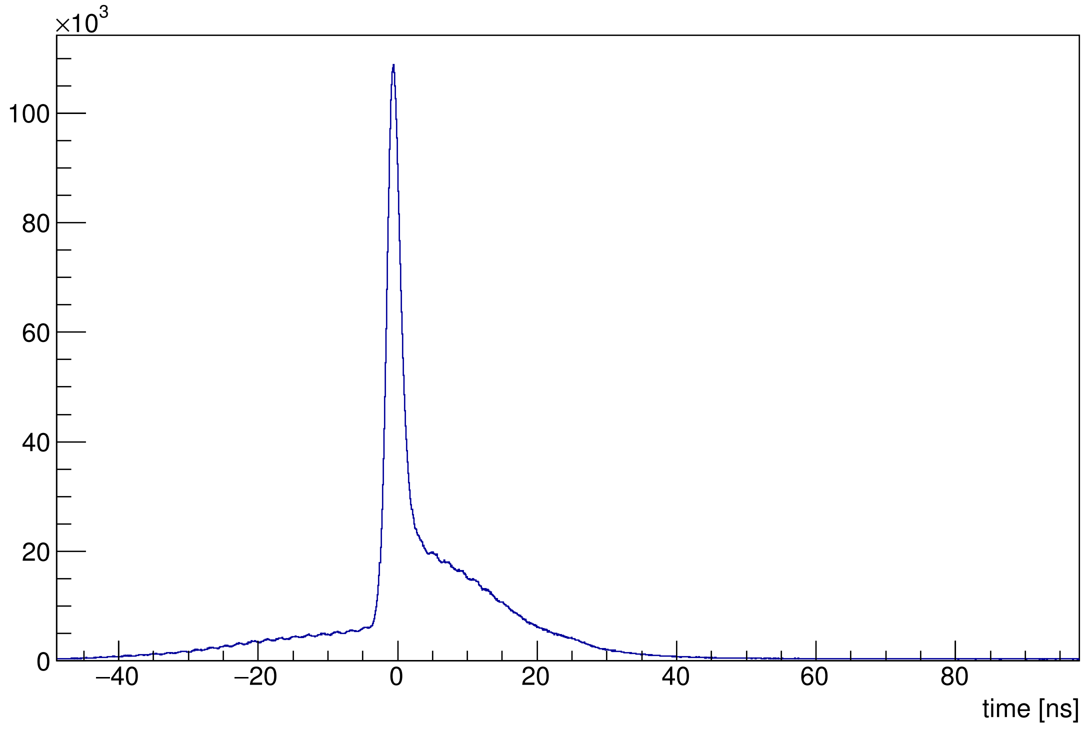


Figure 5.15: ToF scintillator TDC time distributions

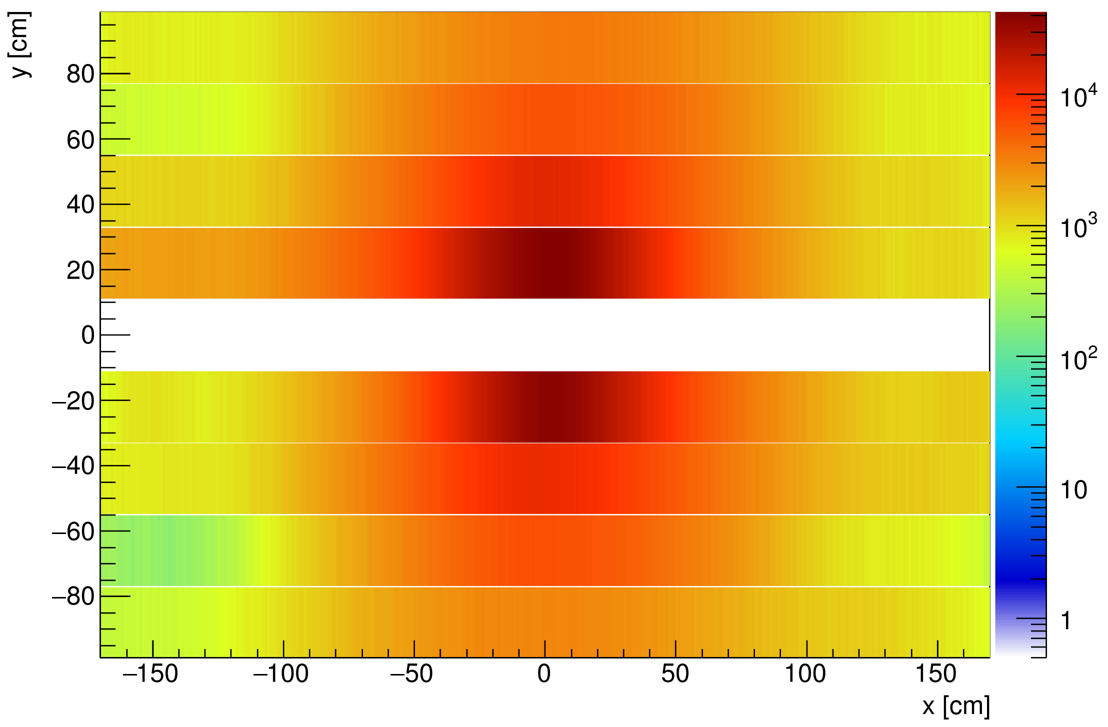


Figure 5.16: ToF hit distribution

5.3 Reconstruction of hadronic reactions

To study hadronic reactions, the particle four-momenta at the reaction vertex need to be reconstructed and the particle types need to be identified. This is achieved by combining the data of all sub detectors. To ensure only data belonging to the same reaction are combined, cuts on the time differences are used. After this selection the four-momenta of the final state particles are calculated and used for further analysis.

As the selection on particle type and multiplicity does not uniquely identify a final state, a background of events not containing the investigated reaction remains, for example

$$\begin{aligned} &\gamma + \mathrm{p} \to \mathrm{p} + \eta \to \mathrm{p} + 3\pi^0 \to \mathrm{p} + 6\gamma \qquad \text{and} \\ &\gamma + \mathrm{p} \to \Sigma + \mathrm{K} \to \mathrm{p} + 3\pi^0 \to \mathrm{p} + 6\gamma . \end{aligned}$$

This background can then be further reduced by requiring the particle momenta to fulfill the kinematics of the desired reaction. The two most important kinematic variables used in the analysis are the invariant mass and the missing mass. The invariant mass m, or rest mass, of a particle is given by the square root of its four momentum $\boldsymbol{p}$ squared. For particles decaying into multiple particles with momenta $\boldsymbol{p}_i$ the invariant mass can be calculated by first summing the four momenta to get the four momentum $\boldsymbol{p}$ of the decayed particle.

$$m = \sqrt{\boldsymbol{p}^2} = \sqrt{\left(\sum_i \boldsymbol{p}_i\right)^2}$$

The missing mass m_{miss} is useful in cases where the four momentum $\boldsymbol{p}$ of a particle could not be fully determined, either because it was not detected or the detector could not measure a reliable energy signature. It is calculated as the square root of the four momentum difference between the initial and final states.

$$m_{\text{miss}} = \sqrt{(\boldsymbol{p}_{\text{initial}} - \boldsymbol{p}_{\text{final}})^2}$$

Here, $\boldsymbol{p}_{\text{final}}$ is the sum of the four momenta of the detected final state particles and $\boldsymbol{p}_{\text{initial}}$ the sum of the target proton and incident photon four momenta. In the case of only one missing particle, the missing mass is the invariant mass of that particle.

As the reconstruction steps and detector efficiencies and acceptances can introduce a bias into the data, a simulation of the setup is essential for understanding the system. This simulation allows the determination of the reconstruction acceptances and efficiencies of the analysis.

Data from November 2014 beamtime was used to investigate the performance of the setup. Sections 5.3.1 and 5.3.2 show reconstructions for selected final states using the central calorimeter and the forward spectrometer, respectively. Section 5.3.3 then uses the combined data from both parts of the setup.

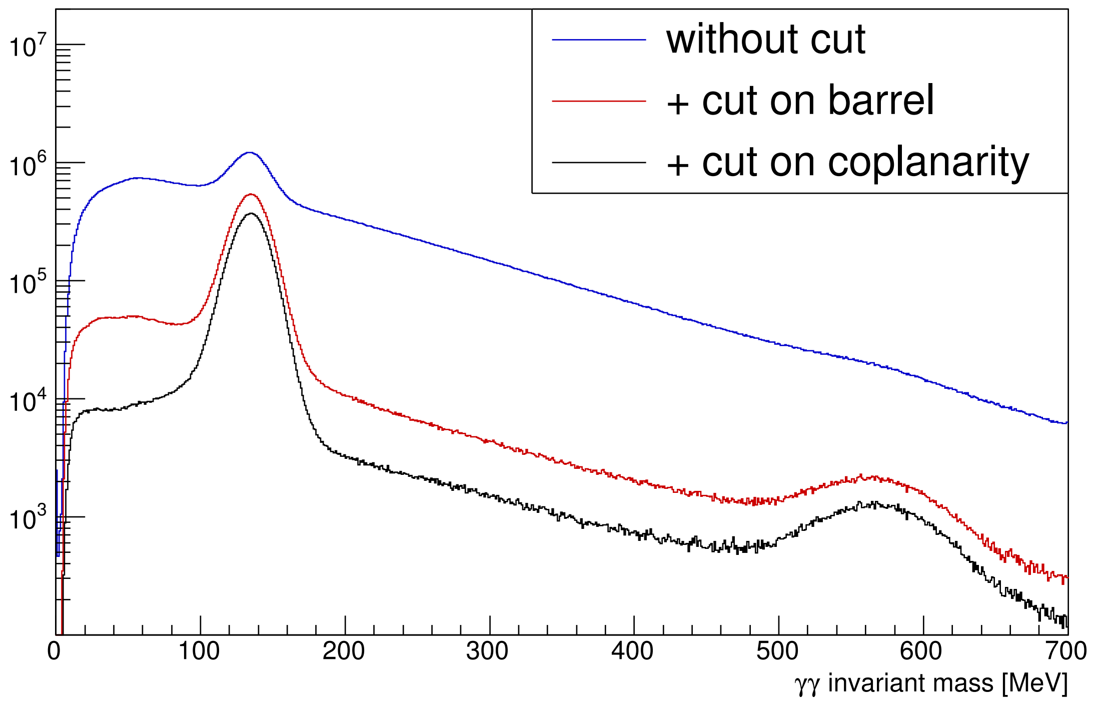


Figure 5.17: Two photon invariant mass

5.3.1 Reconstruction using the BGO calorimeter

The reaction

$$\gamma + \mathrm{p} \to \mathrm{p} + \pi^0 \to \mathrm{p} + \gamma + \gamma$$

is ideally suited to investigate the performance of the central detector of the BGO-OD experiment due to its high cross section and the neutral decay of the π^0. To identify events containing a reaction of this type several selection steps are performed. First, events containing three clusters in the Rugby Ball are selected. All three possibilities to assign the proton to one of the clusters are considered. Figure 5.17 shows the invariant mass calculated using the two photon energy deposits. However, a large amount of background from other reaction channels and misidentification between the proton and photons is still present.

To reduce this background the scintillator barrel can be used to identify the proton. Protons deposit energy in the thin scintillator bars through ionization. However, photons are not seen as the cross section for pair production within the plastic scintillators is sufficiently low. Requiring a coincidence between the proton cluster and a hit in the scintillator barrel allows to discriminate between the two particle types.

The proton and pion are produced back to back in the center of mass frame. Due to the Lorentz boost the Θ angles change when transforming to the laboratory system. As the Φ angles remain unaffected, their difference is fixed to 180°. A cut on this difference, called coplanarity cut, can then be used to further reduce the background.

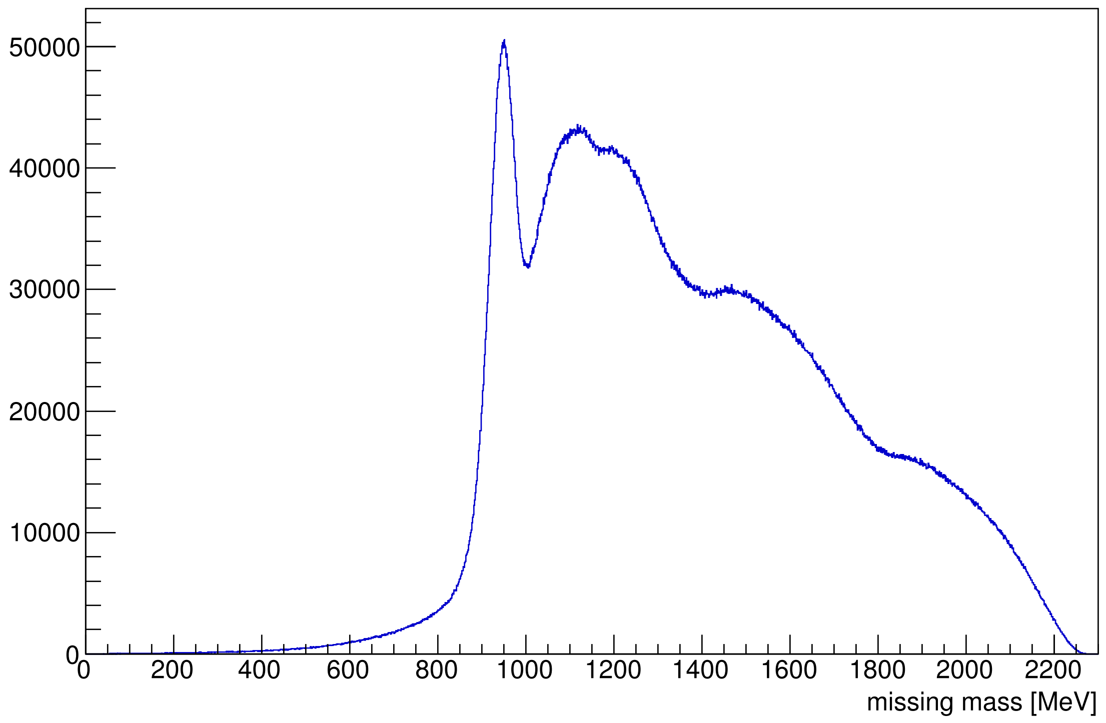


Figure 5.18: Missing mass to the two detected photons

The effect of these cuts is also shown in figure 5.17. While for the calculation without any cuts only a small peak at the pion mass of 135 MeV is visible, the peak becomes significantly more pronounced when the cuts are used. In addition to this peak, a second peak around 550 MeV appears. This is the η invariant mass from the reaction

$$\gamma + \mathrm{p} \rightarrow \mathrm{p} + \eta \rightarrow \mathrm{p} + \gamma + \gamma$$

which has an identical final state.

Including the information provided by the photon tagger into the analysis, it is possible to fully reconstruct events in which the proton was not detected in the calorimeter. In this case events containing only two clusters in the Rugby Ball are selected. These clusters are then assumed to be created by two photons. The invariant mass of the two photons can be calculated as in the three cluster case. Using a time coincidence with the tagger detector, candidates for initial state photons are selected. As the target proton is at rest, the four momentum of the initial state can be calculated. With this the initial state is known and the missing mass can be determined. Figure 5.18 shows its distribution. It has the expected peak at the proton mass. The width of the distribution is limited by the precision in the reconstruction of the initial and final state vectors. The structure at higher masses is mostly caused by misidentified reactions containing more missing particles, for example from the reaction $\gamma + \mathrm{p} \rightarrow \mathrm{p} + 2\pi^0 \rightarrow \mathrm{p} + 4\gamma$.

5.3.2 Reconstruction using the forward spectrometer

The reconstruction of particle tracks using the forward spectrometer is performed in several steps. First the track segments in front of the magnet are identified. Rear tracking, behind the magnet, is then performed using the drift chamber data. The time of flight information provided by the ToF detector is used to discriminate particle types.

Front tracking

The front tracking is performed using the two scintillating fiber detectors MOMO and SciFi2. 3D cluster positions, already shown in figures 5.10 and 5.11 are used as input. All possible combinations of clusters in MOMO and SciFi2 are used to form track segments. These segments are then extrapolated back to the target position as straight lines. Figure 5.19 shows the position of the extrapolated tracks at the target z-coordinate. All tracks not intersecting the target volume are then discarded.

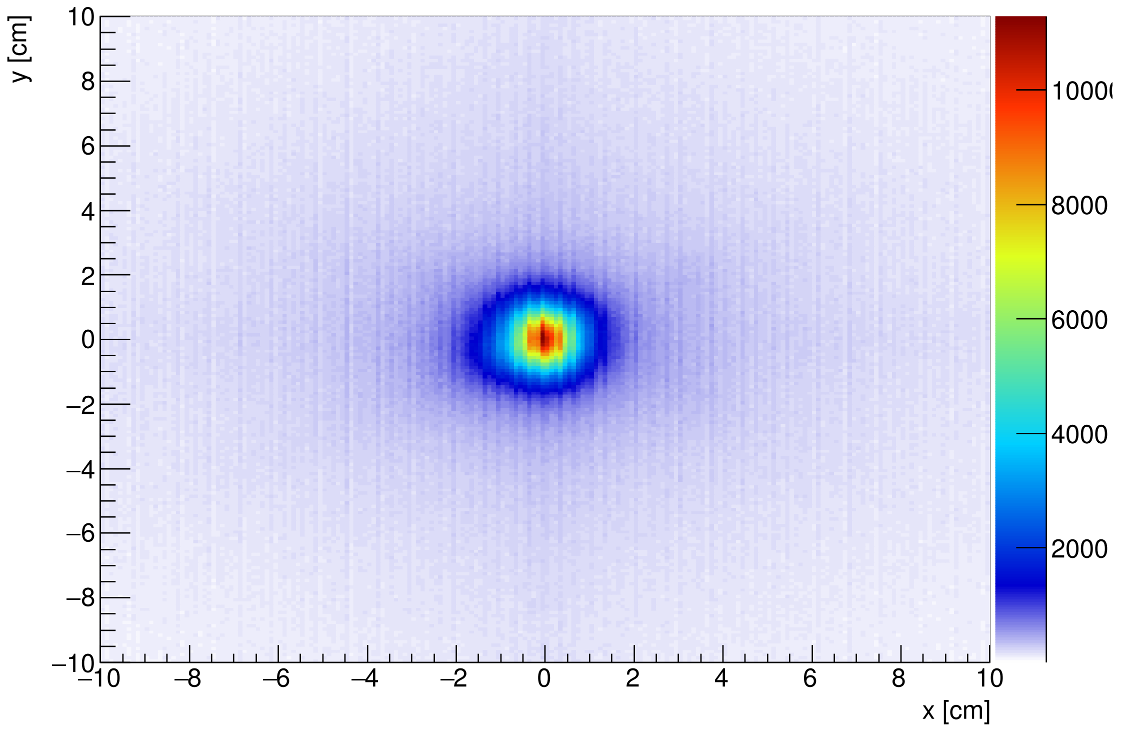


Figure 5.19: Front track coordinates, extrapolated to the target z-position

Rear tracking

For the drift chamber clusters only the coordinate perpendicular to the wires and the z-position of each cluster is known. Due to this the rear track finding needs to consider the information provided by the chambers separately. To reduce the number of candidates a cut on the extrapolated position at the target is applied on the y-component. This is possible as this component is only minimally affected by the magnetic field. Figure 5.20 shows a reconstructed track segment together with the hits in the various detectors for a single event.

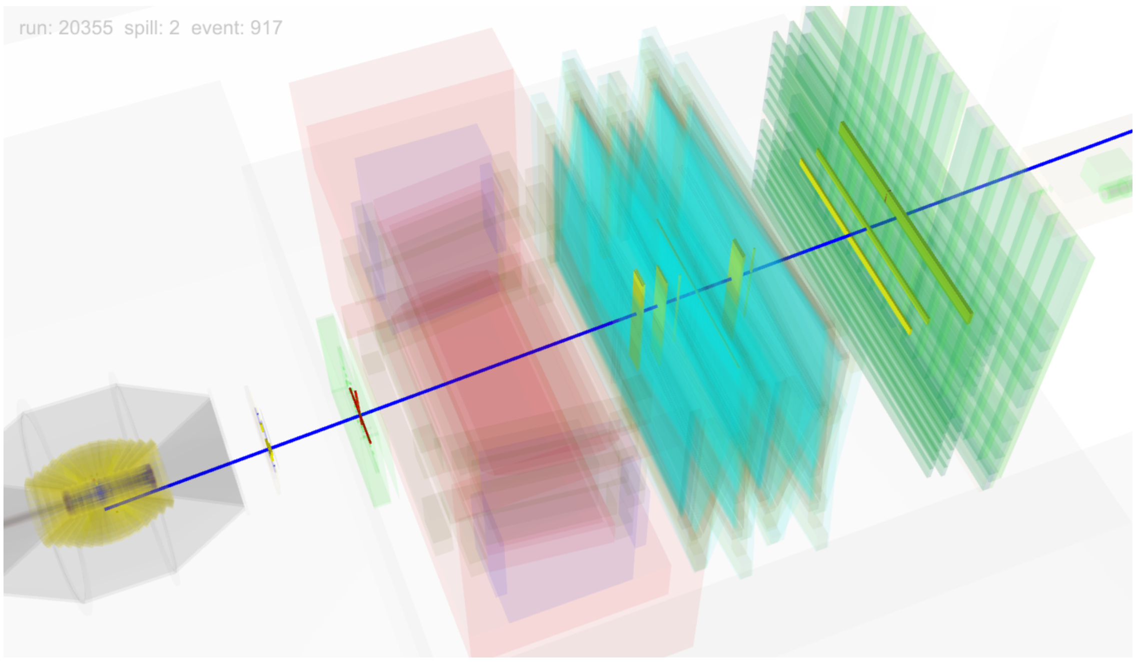


Figure 5.20: Event display showing detector hits and reconstructed rear track segment for event 917 of run 20355

Track combination

The combination of front and rear tracks is performed by matching the horizontal and vertical components of the tracks independently. For the vertical component the influence of the magnetic field is only minor and selected cuts ensure front and rear track combinations form straight lines. For the horizontal component both tracks are extrapolated towards the center of the magnet. The intersection of these two dimensional track components is then required to be in the central region of the magnet, as would be expected for a homogeneous magnetic field. These few requirements for the combination of front and rear tracks are sufficient to ensure both a good reconstruction efficiency as well as suppression of accidental combinations.

Momentum reconstruction

The first estimate for the momentum of the particle is calculated from the angle between the front and rear track segments using an uniform magnetic field. A second estimate is calculated using the full field information and taking the energy loss along the track into account. The expected resolution is determined by simulation, taking into account effects such as multiple scattering and energy loss along the track. The resolution depends on the chosen value for the magnetic field strength. According to simulations for maximum field strength a resolution of 1.5 % to 2 % can be reached. For half field strength the expected resolution is about 3 %.

Time of flight

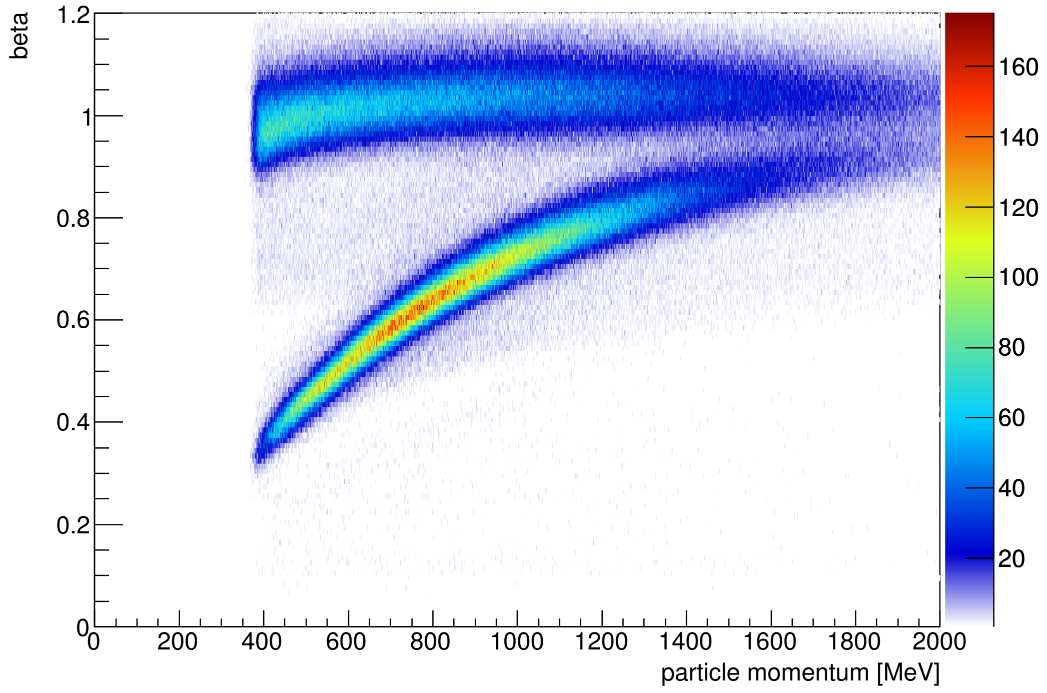


Figure 5.21: Beta vs. momentum for particles reconstructed using the forward spectrometer

After determination of the track momentum the rear part of the track is extrapolated towards the ToF detector as a straight line. If a cluster in the ToF wall matching the extrapolated impact point is found, the cluster time is associated to the track. This time is used together with the track length to calculate the beta of the particle. Figure 5.21 shows a plot of beta versus the reconstructed momentum of the track. Two bands are visible in this plot. The topmost band corresponds to the charged pions. For momenta above 600 MeV their beta is close to 1. The lower band corresponds to protons. It follows the behavior expected according to the mass. The two bands can be separated for the highest momenta achievable with ELSA.

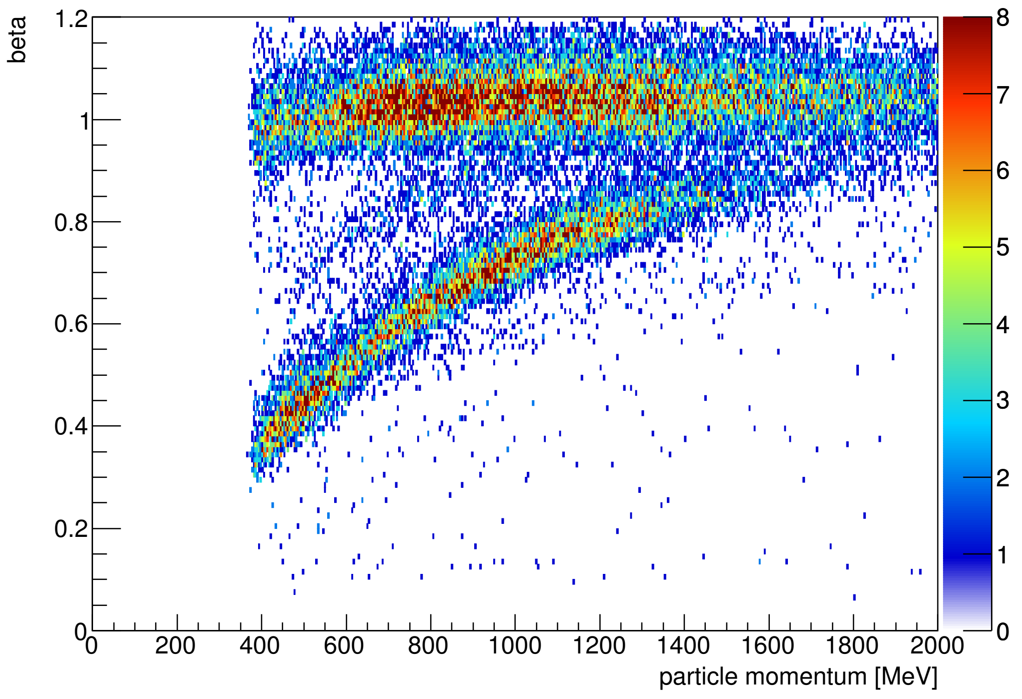


Figure 5.22: Beta vs. momentum for particles reconstructed using the forward spectrometer and the selection criteria given in section 5.3.3

5.3.3 Reconstruction using the full setup

While in figure 5.21 charged pions and protons could be easily identified, charged kaons were not visible. This is due to their significantly lower production cross section. To enhance the ratio of kaons with respect to protons and pions in the forward spectrometer, cuts can be made on the particles identified in the other detectors. First, a cut on the incoming photon energy, provided by the tagger, is used. This removes reactions where the center of mass energy is not sufficient to produce kaons. A requirement on a minimum photon energy of 900 MeV is used since this energy is just below the threshold for $\gamma p \to K^+\Lambda$. Secondly, a requirement that the total energy deposited in the Rugby Ball is less than 250 MeV has been applied, as much of the energy provided by the photon is carried away by the forward kaon. The last cut applied is on the identification of a neutral pion in the Rugby Ball via its invariant mass. This pion is created by the decay of the Λ produced in coincidence with the kaon.

$$\gamma + \mathrm{p} \rightarrow \Lambda + \mathrm{K}^{+} \rightarrow \mathrm{n} + \pi^{0} + \mathrm{K}^{+} \rightarrow \mathrm{n} + 2\gamma + \mathrm{K}^{+}$$

The identification of this π^0 suppresses the charged background in forward direction. The effect of these cuts on the beta versus momentum plot can be seen by comparing figures 5.21 and 5.22. Due to the restrictive cuts, the total number of entries in the plot has been reduced drastically. However, instead of mostly uniform background below the pion band, a structure is now visible. This coincides with the expected kaon band. This illustrates the unique capabilities of the BGO-OD setup for the investigation of reactions with complex final states and forward charged particles.

6 Conclusion

The BGO-OD experiment has been set up at the accelerator facility ELSA in Bonn. It is pursuing its physics program since November 2014. The data acquisition system developed within this thesis is paramount to this success.

The distributed design of the DAQ allows for an efficient readout of all detector parts. In addition, this design provides modularity as it allows to add or remove complete readout crates without affecting the remaining system. The software design of the LEVBs is also modular. The readout module specific code is encapsulated within classes sharing similar structure. This allows to easily add new module types and to move modules between LEVBs.

The readout crates need to be synchronized to allow the operation of the DAQ as an integrated system. For this purpose an FPGA based synchronization system was set up. This system distributes the trigger and event information from a synchronization master module to the client modules contained in each readout crate. This trigger signal not only initiates the readout across all readout modules, but also provides a common time reference. The clients transmit their status back to the synchronization master.

The distributed nature of the DAQ also benefits the performance of the system. The dead time, in which no events can be acquired, is reduced as all crates can read the data from their modules simultaneously. To further minimize this dead time the readout process was split into two parts. In the first part the DAQ system waits until the modules have transferred their data into an internal buffer and are ready for the next event. In the second part the data is read from the internal buffers and transmitted to the event saver. This split allows the second part to take place during the life time. With this method, typical dead times of 50 µs are reached. This minimal dead time is limited by the time required until the modules are ready for the next event. Larger dead times occur if the LEVB has not finished reading the previous event or is otherwise not able to immediately process the new event. The event saver, evs, needs to be able to receive the events of the LEVBs and write them to disk. If it was not able to do this sufficiently fast, the LEVBs would need to wait for free buffers before continuing with the readout. When investigating the evs performance it was found that the time needed to write events shows large fluctuations. These fluctuations are due to the compression of the data performed by the ROOT library. Large buffers in the event saver as well as in the LEVBs have been used to mitigate this effect. As the average write times are sufficiently short, the number of free buffers never reaches zero during normal runs.

With this setup first data was taken in 2014. During this time the data acquisition system operated at an event rate of approximately 850 Hz with a life time above 90 %. The data from the various detector components was investigated and found to agree with the expectations. Structures in the raw detector spectra could be explained by either general effects introduced by the event based data acquisition or the specific detector characteristics.

On this basis the reconstruction of hadronic reactions has been performed. Due to their simple final states, the reactions $\gamma p \rightarrow p\pi^0$ and $\gamma p \rightarrow p\eta$ are easily observed using the BGO calorimeter. The use of kinematic cuts improves the signal to background ratio in the detection significantly. Reactions with charged particles in forward angles profit from the forward spectrometer. The reaction $\gamma p \rightarrow \Lambda K^+$, as discussed in the introduction, is one of those. However, the spectrometer data is dominated by protons and charged pions as the cross sections for producing strange particles is comparatively low. This hides the signature of the kaons in the beta versus momentum plot. This situation can be improved by using the full BGO-OD setup. Cuts on the energy of the incoming photon and the particles detected in the BGO calorimeter allow to sufficiently reject the number of reactions with forward going protons and pions compared to kaons. When these cuts are applied the kaon signal becomes visible.

The BGO-OD experiment is now pursuing its physics program. The DAQ system introduced in this thesis allows this to be done efficiently while providing the flexibility to adapt to future developments.

Glossary

ADC Analog to digital converter, see sections 3.1.3 and 3.1.4.

base64 Data encoding format, This format allows the encoding of arbitrary binary data into ASCII symbols.

CAMAC Computer Automated Measurement And Control, bus system developed in 1972 for readout and control of nuclear and particle physics experiments..

CPU Central processing unit, main processing unit of a computer.

DAQ Data acquisition system, see section 3.

daqUI DAQ user interface, see section 4.6.

ECL Emitter coupled logic, differential electric signaling standard.

evs Event saver, see section 4.4.

FIFO First in, first out, data buffer organization scheme where the buffered data is retrieved in the same order as written.

FPGA field programmable gate array, electronic chip containing programmable logic.

LEVB Local event builder, see section 4.3.

LVDS Low voltage differential signal, differential electric signaling standard.

MD5 hash MD5 message-digest algorithm, cryptographic hash function used to ensure data integrity.

mime type standardized identifier for file formats, e.g. text/plain for not formatted text.

NIM nuclear instrument module, mechanical and electrical standard for modules used in nuclear physics.

PCI Peripheral Component Interconnect, bus system commonly used in personal computers.

PECL Positive emitter coupled logic, differential electric signaling standard.

pedestal Value recorded by a QDC in the absence of a signal, see section 3.1.3.

PostgreSQL SQL database server application.

QDC Charge to digital converter, see section 3.1.3.

ROOT Object orient data analysis framework, developed at CERN.

runcontrol DAQ control program, see section 4.5.

runDb Run database, see section 4.7.

SADC Sampling analog to digital converter, see section 3.1.4.

slowcontrol System to set and monitor slowly or infrequently changing parameters, e.g. temperatures or thresholds, of the experiment..

TCP Transmission control protocol, networking protocol used on top of IP.

TDC Time to digital converter, see section 3.1.2.

VME Versa Module Eurocard, bus system used for control and readout hardware.

XML Extensible Markup Language, human and machine readable text based document format.

Bibliography

[B$^+$12] J. Beringer et al. (Particle Data Group), Review of Particle Physics. *Phys. Rev. D*, 86:010001, Jul 2012. 2.5

[Bel11] Andreas Bella. *Setup of a Goniometer System for the Production of Linearly Polarised Photons for the BGO-OD Experiment at ELSA*. diploma thesis, University Bonn, 2011. 2.3, 2.2.2

[Bel12] Andreas Bella. *Tagger Status.* Internal note, BGO-OD Collaboration, 2012. 2.2.3

[Bie12] John Bieling. FPGA module and firmware descriptions. *BGO-OD internal note*, BGO-OD-008-2011, 2012. 4.7

[BR97] Rene Brun and Fons Rademakers. ROOT - An Object Oriented Data Analysis Framework. *Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. and Meth. in Phys. Res. A*, 389:81–86, 1997. See also `http://root.cern.ch/`. 4.4

[BS13] Petr Bydžovský and Dalibor Skoupil. Present status of isobar models for elementary kaon photoproduction. *Genshikaku Kenkyu*, 57:86 – 92, 2013. arXiv:1211.2684 [nucl-th]. 1.5, 6

[Bö15] Sabine Böse. *Aufbau und Test eines neuen Szintillationsfaser-Detektors für das neue Vorwärtsspektrometer an ELSA*. doctoral thesis, University Bonn, 2015. 2.8(b)

[E$^+$12] R. Ewald et al. (Crystal-Barrel/TAPS Collaboration), Anomaly in the K^0 Σ^+ photoproduction cross section off the proton at the K* threshold. *Physics Letters B*, 713(3):180 – 185, 2012. 1

[EDRW11] Robert G. Edwards, Jozef J. Dudek, David G. Richards, and Stephen J. Wallace. Excited state baryon spectroscopy from lattice qcd. *Phys. Rev. D*, 84:074508, Oct 2011. 1

[F$^+$13] Frommberger F. et al. Electron Stretcher Accelerator (ELSA). `http://www-elsa.physik.uni-bonn.de/index_en.html`, 2013. retrieved 2015-07-15. 2.2, 6

[GM64] M. Gell-Mann. A schematic model of baryons and mesons. *Physics Letters*, 8(3):214 – 215, 1964. 1

[GUUY10] V. Golotsov, L. Uvarov, S. Uvarov, and V. Yatsyura. *CROS-3B Reference Guide*. St. Petersburg Nuclear Physics Institute, 2010. 4.3.8

[Ham08] Daniel Hammann. *Test und Inbetriebnahme der Prototyp-Driftkammer für das B1-Spektrometer*. diploma thesis, University Bonn, 2008. 5.2.4

[Hof] Phillip Hoffmeister. *The Crystal-Barrel Data acquisition*. doctoral thesis, in preparation, University Bonn. 4

[Joo96] Rainer Joosten. *Aufbau und Inbetriebnahme eines hochgranularen Vertexdetektors aus szintillierenden Fasern fuer das Experiment MOMO an COSY - Erste Ergebnisse zur Reaktion p + d –> $^3He + \pi^+ + \pi^-$*. doctoral thesis, University Bonn, 1996. 2.4.2

[KR10] Eberhard Klempt and Jean-Marc Richard. Baryon spectroscopy. *Rev. Mod. Phys.*, 82:1095–1153, Apr 2010. 1.2, 6

[LMP01] U. Löring, B.Ch. Metsch, and H.R. Petry. The light-baryon spectrum in a relativistic quark model with instanton-induced quark forces. *The European Physical Journal A - Hadrons and Nuclei*, 10(4):395–446, 2001. 1

[Mei13] Peter Meiß. *The Time Of Flight Spectrometer of the BGO-OD Experiment*. diploma thesis, University Bonn, 2013. 2.4.4

[OR10] Oset, E. and Ramos, A. Dynamically generated resonances from the vector octet-baryon octet interaction. *Eur. Phys. J. A*, 44(3):445–454, 2010. 1

[Rom12] M. Romaniuk. Target system. *BGO-OD internal note*, BGO-OD-016-2012, 2012. 2.2.4

[SAI] SAID group. Kaon Photoproduction Data Base. http://gwdac.phys.gwu.edu/analysis/prk_analysis.html. retrieved 2016-01-16. 1.4, 6

[Sch04] Christoph Schmidt. *Entwicklung eines neuen Datenakquisitionssystems für das CB-ELSA-Experiment*. doctoral thesis, University Bonn, 2004. 4

[Sch10] Timothy Schwan. *Test und Inbetriebnahme der Driftkammern für das BGO-OD-Spektrometer*. diploma thesis, University Bonn, 2010. 5.2.4

[Sie11] Georg Siebke. *Design of the BGO-OD Tagging System and Test of a Detector Prototype*. diploma thesis, University Bonn, 2011. 2.2.3

[Tim69] U. Timm. Coherent Bremsstrahlung of Electrons in Crystals. *Fortschritte der Physik*, 17(12):765–808, 1969. 2.5

[Z$^+$92] A. Zucchiatti et al. Optimization of response of BGO sectors for a 4π electromagnetic calorimeter. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 317(3):492 – 497, 1992. 2.3.3

[Zim12] Thomas Zimmermann. *Photon Flux Monitor for the BGO-OD experiment.* diploma thesis, University Bonn, 2012. 2.5

[Zwe64] G Zweig. An SU_3 model for strong interaction symmetry and its breaking; Version 2. (CERN-TH-412):80 p, Feb 1964. Version 1 is CERN preprint 8182/TH.401, Jan. 17, 1964. 1

List of Figures

List of Tables