GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Fakultät für Physik

Bachelor's Thesis

Zeitgenaue Messung einer radioaktiven Quelle mit einer Trigger Logic Unit

Time-resolved Measurement of a radioactive Source using a Trigger Logic Unit

prepared by

Björn Klaas

from Detmold

at the II. Physikalisches Institut

Thesis number:	II.Physik-UniGö-BSc-2013/06
Thesis period:	15th April 2013 until 15th July 2013
First referee:	Priv.Doz. Dr. Jörn Große-Knetter
Second referee:	Prof. Dr. Arnulf Quadt

Abstract

To discover the Higgs boson and interesting new physics the *Large Hadron Collider* (Lhc) was built at Cern. It began physics operation in 2010 with reduced centre-of-mass energy and is upgraded during a shut-down in 2013-14 to reach its nominal centre-of-mass energy and luminosity, resulting in increased irradiation and hit occupancy of the physics experiments at the Lhc. To prepare the Pixel detector of the Atlas general-purpose detector for operation under these tougher conditions the *Insertable B-Layer* (Ibl) project was established, developing a fourth and innermost layer of the Pixel detector. This new layer consists of improved n-in-n planar silicon sensors and newly developed 3D silicon sensors, and as well newly developed Front-End read-out chips (FE-I4). The characteristics of these chips have to be carefully analysed in testbeams (TB) and labs to ensure precision and accuracy of the measurements once installed in Atlas.

This bachelor thesis examines the *timewalk* in the FE-I4, which is the time difference between registration of differently sized charges by the FE-I4, corresponding to different amounts of energy deposited in the sensor. Measurements are done using a lab set-up resembling a testbeam, containing a radioactive source and a *Trigger Logic Unit* (TLU).

Keywords: Atlas, Ibl, Front-End, FE-I4, timewalk, TLU, USBpix, EUDAQ

Contents

1. Introduction

The scientific search for the fundamental particles of our universe and their interactions has led physicists to the *Standard Model of Particle Physics*. Even though this well tested model describes all particles observed to date as well as three fundamental interactions between them many unanswered questions, and for decades untested predictions, remain. To continue the search with higher precision and greater scope the *Large Hadron Collider* (Lhc) was built at the *European Centre for Nuclear Research* (Cern). It is the highest energy particle collider ever built and designed to reach very high luminosities to generate a large amount of data in a short time. Four large experiments are located at the Lhc. One of these is the Atlas experiment, a large general-purpose particle detector for proton-proton collisions described in Chapter 2.

After a first long shut-down in 2013–14 the Lhc will operate at its nominal centre-of-mass energy and luminosity, increasing the irradiation and hit occupancy of the detectors. To prepare the Atlas Pixel detector for these tougher conditions the *Insertable B-Layer* Project (Ibl) was established, designing a fourth and innermost pixel layer inserted in between the current Pixel detector and a smaller beampipe. The modules developed for the Ibl are described in Chapter 3.

These new modules have to be closely examined and tested in testbeams and labs. The lab set-up described in Chapter 4 was designed to closely resemble the data acquisition systems used in testbeams. Chapter 5 describes the analysis framework developed to evaluate the data produced by the data acquisition.

The property examined is the *timewalk*, a characteristic of the read-out electronics. It causes hits to be registered with a small delay, whose size depends on the energy of the particle detected by the sensor. The method used to examine the timewalk and the measurements are explained and presented in Chapter 6.

2. The Large Hadron Collider and the ATLAS Experiment

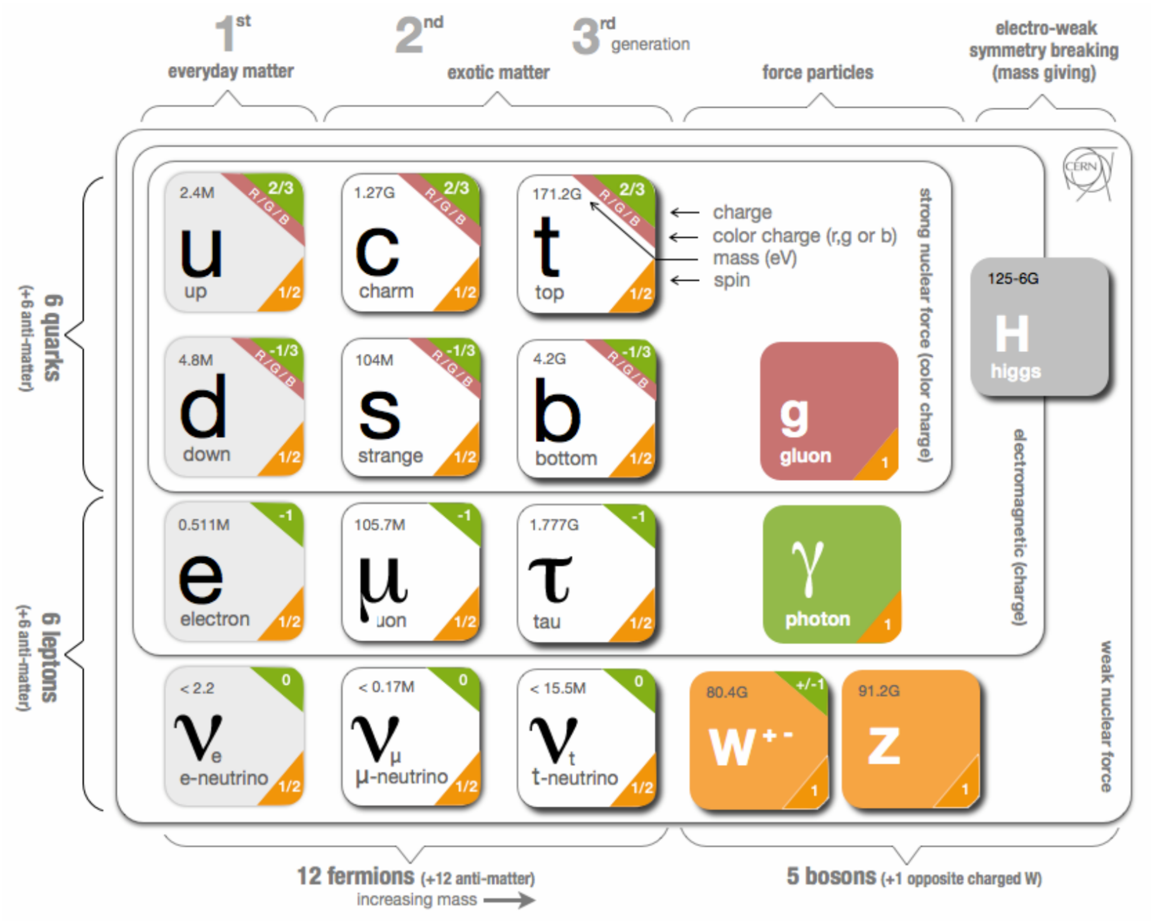
This chapter summarizes the current state of particle physics as described by the *Standard Model of Particle Physics* (SM). It also gives a short introduction to the Large Hadron Collider (LHC) and the ATLAS Experiment, designed to test the predictions made by the SM and to possibly extend it by discovering new physics.

2.1. The Standard Model of Particle Physics

The *Standard Model of Particle Physics* (SM, Figure 2.1) is to date the most comprehensive and best tested model of the elementary physics at work in our universe, describing all known elementary particles and interactions between them. It contains twelve spin-1/2 matter particles called *fermions*, subdivided into two groups of six particles each, *leptons* and *quarks* [1]. The fermions are also organized in three generations combining two particles of each subgroup. The particles of the first generation have the lowest mass and cannot decay further, therefore called stable, They make up almost all observable matter. Each particle has a partner with equal mass but opposite charge signs, called an antiparticle.

The four spin-1 particles, called *gauge bosons*, mediate the three interactions described by the SM. These are the *electromagnetic force* mediated by photons, the *weak force* mediated by the vector bosons Z and $W^{\pm}$ and the *strong force* mediated by eight different gluons. Which particles are affected by a certain force depends on the respective gauge boson's characteristics. Photons couple only to electrically charged particles, gluons to particles carrying a colour charge, and the Z and $W^{\pm}$ to all fermions. Since the photon is massless the range of the electromagnetic interaction is infinite, while the range of the strong interaction, even though the gluons are also massless, is strongly limited due to colour confinement. The range of the weak force is limited by the decay of the Z and $W^{\pm}$, occurring because they are massive particles, which results in a problem for the SM.

In theory the four gauge bosons should be massless to ensure local gauge invariance,



1st everyday matter
2nd exotic matter
3rd generation
force particles
electro-weak symmetry breaking (mass giving)
2.4M u up 2/3 R/G/B 1/2
1.27G c charm 2/3 R/G/B 1/2
171.2G t top 2/3 R/G/B 1/2
charge
color charge (r,g or b)
mass (eV)
spin
4.8M d down -1/3 R/G/B 1/2
104M s strange -1/3 R/G/B 1/2
4.2G b bottom -1/3 R/G/B 1/2
g gluon 1
125-6G H higgs
0.511M e electron -1 1/2
105.7M μ μon -1 1/2
1.777G τ tau -1 1/2
γ photon 1
< 2.2 νe e-neutrino 0 1/2
< 0.17M νμ μ-neutrino 0 1/2
< 15.5M ντ t-neutrino 0 1/2
80.4G W+- +/-1 1
91.2G Z 1
strong nuclear force (color charge)
electromagnetic (charge)
weak nuclear force
6 quarks (+6 anti-matter)
6 leptons (+6 anti-matter)
12 fermions (+12 anti-matter)
increasing mass
5 bosons (+1 opposite charged W)

Figure 2.1.: Overview of the particles in the Standard Model of Particle Physics [2]

but experimentally the Z was shown to have a mass of $m_Z = (91.1876 \pm 0.0021)\,\text{GeV}$ and the $W^{\pm}$ to have a mass of $m_W = (80.385 \pm 0.015)\,\text{GeV}$ [3]. To maintain local gauge invariance the spin-0 Higgs boson, as part of the Higgs mechanism, was added to the SM as the most probable way of keeping the model consistent. Finding the Higgs is one of the main goals of the LHC and ATLAS and a complicated task, since the model does not predict the Higgs' mass and no decay exclusive to the Higgs. So far a Higgs-like boson was found at a mass of approximately 125 GeV, announced in July 2012 [4].

Multiple extensions of the SM, like *Supersymmetry* (SUSY), have been proposed, trying to answer several questions left open by the current version of the SM. These extension will be searched for and possibly examined using the LHC as well.

2.2. The Large Hadron Collider

The *Large Hadron Collider* (LHC) is the world's largest circular particle collider, built in the former LEP tunnel at CERN, Geneva, Switzerland. It was designed to explore new physics and discover new particles by reaching an unprecedented centre-of-mass collision energy of $\sqrt{s} = 14\,\text{TeV}$, while maintaining a high luminosity of $10^{34}\,\text{cm}^{-2}\,\text{s}^{-1}$ to provide the necessary statistical significance in the shortest time frame possible [5].

The target luminosity required a proton-proton collider, since the currently achievable production rate of anti-protons is too small to sustain the targeted collision rate of 40 MHz of bunches containing $1.1 \cdot 10^{11}$ particles each. To reach this rate at full luminosity, 2808 bunches will be circulated in two separate beampipes, required to circulate the equally charged particles in opposite directions. The number of particles per bunch and the achieved beam focusing result in more than 25 inelastic interactions per bunch crossing, generating more than 1000 charged particles.

At the four interaction points the large LHC experiments ATLAS, CMS, ALICE and LHCB are located. ATLAS and CMS are general-purpose experiments, designed to gather information on all physics examined at the LHC, while LHCB focuses on b-physics and ALICE on heavy ion collisions. It thus mainly operates when the LHC circulates lead ions in heavy ion mode.

So far the LHC operated at a maximal centre-of-mass energy of 8 TeV, due to technical issues. It delivered an integrated luminosity of approximately $\mathcal{L} = 25\,\text{fb}^{-1}$ [6]. In early 2013 it was shut down for upgrade and consolidation work, meant to enable operation at nominal centre-of-mass energy and luminosity when resuming operation in 2015. Another upgrade called High Luminosity-LHC (HL-LHC) is scheduled for 2022, further increasing the LHC's instantaneous luminosity by a factor of ten.

2.3. The ATLAS Experiment

The ATLAS experiment was designed as a general-purpose proton-proton experiment for the LHC [7]. The discovery of the origin of mass at the electroweak scale being a major focus of interest, the detector was optimized to be sensitive to the largest possible Higgs mass range. It was also optimized for detailed studies of top quark decays, SUSY searches and sensitivity to large compositeness scales. This wide range of abilities demonstrates the detector's potential to discover unexpected new physics.

Since most interesting physics questions at the LHC require high luminosity, the ATLAS experiment was designed to provide as many signatures of new physics as possible, using electron, gamma, muon, jet and missing transverse energy measurements. During the initial lower luminosity operation the experiment was meant to address more complex signatures, including tau detection and heavy flavour tags, while at higher luminosities a restricted set of signatures is to be studied.

2.3.1. Detector Design

To achieve the goals described above a number of basic design criteria were established. These include very good electromagnetic and full-coverage hadronic calorimetry, high-precision muon momentum measurements, efficient tracking for high-p_T lepton-momentum measurements, electron and photon identification, τ-lepton and heavy-flavour identification and full event reconstruction capability at lower luminosity. Other criteria were large acceptance in pseudo-rapidity with almost full azimuthal angle coverage everywhere, and triggering and measurements of particles at low-p_T thresholds.

This was intended to provide high efficiencies for most physics processes of interest at the LHC, highlighting the general-purpose nature and versatility of ATLAS.

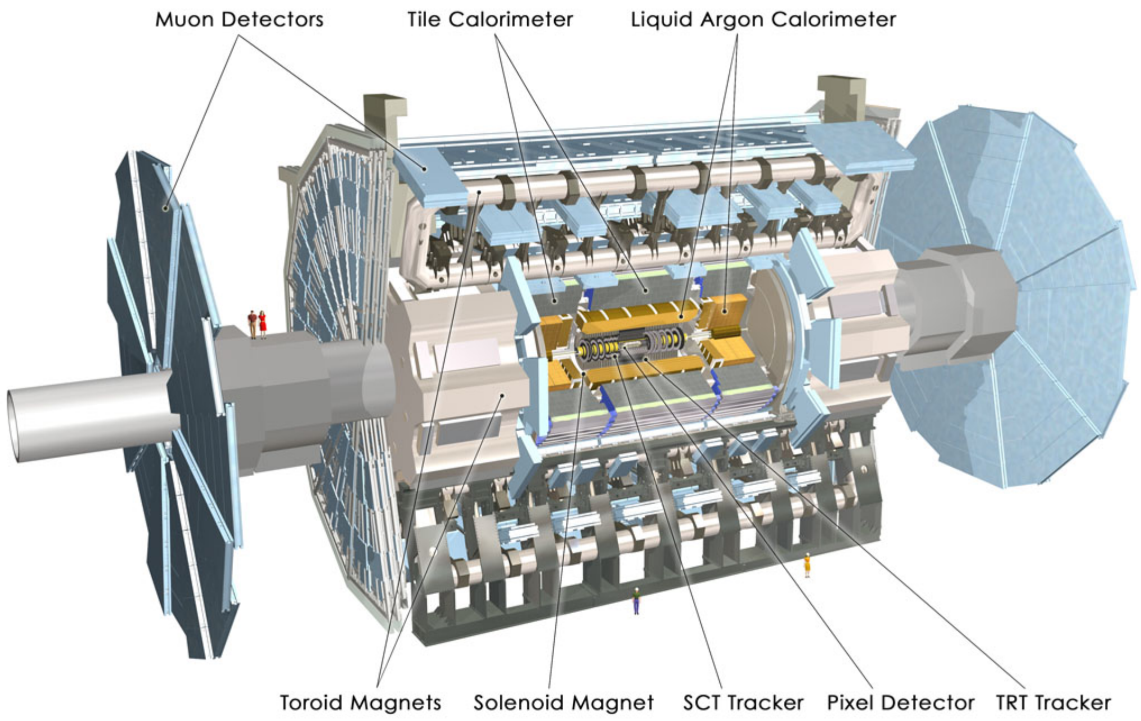


Figure 2.2.: Overall layout of the ATLAS general-purpose detector

The overall detector layout chosen to accomplish these goals is shown in Figure 2.2. It consists of an *Inner Detector* (AID) extending to 1.15 m from the beampipe and surrounded by a thin superconducting solenoid. The solenoid itself is surrounded by the calorimeter system, extending to 4.25 m from the beampipe. The calorimeters are enclosed by the muon system, containing a large superconducting air-core toroid consisting of independent coils arranged with an eight-fold symmetry. It provides the magnetic field for the muon system and extends to 11 m from the beampipe [8].

The Inner Detector

The AID surrounds the beampipe and interaction point with three subdetectors consisting of several barrel layers and endcaps, allowing very precise measurements of the interaction point's and secondary decay vertices' position. The innermost subdetector providing the best spatial resolution is the *Pixel detector* (APD). It is surrounded by the *Semiconductor Tracker* (SCT), itself encased by the *Transition Radiation Tracker* (TRT). Combined, the three subdetectors have a length of 7 m, a diameter of 2.3 m and provide a relative momentum resolution of $\sigma/p = (4.38 \pm 0.16) \cdot 10^{-4}\,\mathrm{GeV}^{-1} \cdot p_T$.

The APD (Figure 2.3) consists of three of the aforementioned barrel layers and two endcaps, consisting of three disks each, on either side of the interaction point in beampipe direction. It provides three space points per particle on average, The main components are 1744 identical sensor-chip-hybrid modules, totalling approximately $8 \cdot 10^7$ pixels. Each module contains a silicon sensor subdivided into 47 232 pixels, mostly $50\,\mu\mathrm{m} \times 400\,\mu\mathrm{m}$, individually connected to 16 Front-End read-out chips (FE-I3). The FEs contain 2880 analogue pixels each, organized in 18 columns by 160 rows.

During the LHC shut-down in 2013-14 the APD will be upgraded by inserting a fourth barrel layer, called the *Insertable B-Layer* (IBL, Chapter 3).

The SCT consists of four barrel layers and two endcaps with nine disks each. The layers and disks are made of single sided p-in-n microstrip sensors glued back to back and rotated by 40 mRad to provide three-dimensional hit information, read out via $6.2 \cdot 10^6$ channels. A resolution of 16 µm is provided in the $R\phi$ plane, and 580 µm in direction of the beampipe.

The TRT is made of straw tubes, arranged parallel to the beampipe in the barrel layers and perpendicular in the endcaps, providing a resolution of 170 µm per straw. Each tube contains a sense wire and is filled with a gas mixture. It detects the photons emitted when a particle traverses the wall of a tube. The amount of radiation depends on the ratio of energy to mass of the particle, so e.g. electrons emit more radiation and can be distinguished from heavier particles. Approximately 36 hits are recorded per particle,

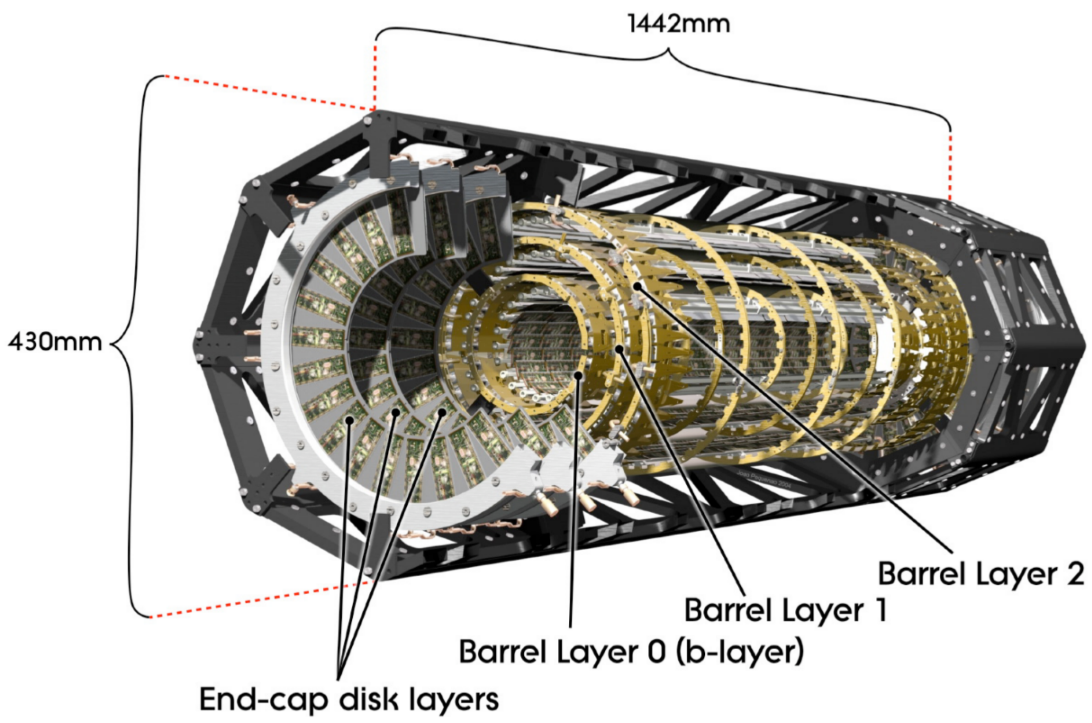


Figure 2.3.: Engineering drawing of the ATLAS Pixel detector in its global support frame

read out via 420 000 channels.

The AID sits in a 2 T magnetic field created by a superconducting solenoid magnet, measuring 5.6 m long and 2.4 m across. It is supplied with a current of 7600 A and creates an axial field of 2.6 T peak strength.

The Calorimeter System

The Calorimeter system is a two part system designed to precisely measure the energy of particles leaving the AID, and to stop all particles except for muons and neutrinos.

The inner part of the Calorimeter system is the *electromagnetic calorimeter*, a sampling calorimeter made of lead absorbers and liquid Argon. The outer part is the *hadronic calorimeter*, also a sampling calorimeter, but made of steel absorbers and plastic scintillators. To stop almost all particles both calorimeters are made approximately 25 radiation length thick.

The Muon System

Since muons can leave the calorimeter a third detector, the *Muon system*, was built surrounding the calorimeter. To measure the muon energy a magnetic field is required, created by a toroid magnet consisting of eight superconducting air-coils and two endcaps. The coils are 25.3 m long and extend from a radius of 9.4 m to a radius of 20.1 m, producing a magnetic field of 3.9 T peak strength, while the endcaps extend from a radius of 1.65 m to 10.7 m and produce a magnetic field of 4.1 T peak strength.

The muon system is a four part system consisting of *Resistive Plate Chambers* (RPC) in avalanche mode in the barrel region and *Thin Gap Chambers* (TGC) in the endcap regions for triggering. Precision measurements are made by *Drift Tubes* (MDT) made of aluminium and filled with a gas mixture, designed to withstand high radiation doses, and *Cathode Strip Chambers* (CSC) made of multi-wire proportional chambers. They provide a spatial resolution of approximately 50 µm.

3. The *Insertable B-Layer* Project

This chapter describes the *Insertable B-Layer* (IBL) Project, an effort to prepare the ATLAS Pixel detector (APD) for the increased irradiation and pixel occupancy resulting from current and future LHC upgrades. The IBL itself is a fourth and innermost pixel layer, inserted into the existing APD during the LHC shut-down in 2013-14. During this time consolidation and upgrade work is done on the LHC to allow operation at the nominal centre-of-mass energy of $\sqrt{s} = 14\,\mathrm{TeV}$ and high luminosities. Even higher luminosities will be reached after the *High Luminosity*-LHC (HL-LHC) upgrade in 2022, making the IBL project essential for the future of the APD [9].

3.1. The IBL Design

In between the current APD and beampipe exists an 8.5 mm radial gap, which will be enlarged to 12.5 mm by reducing the radius of the beampipe [10]. The reduction became possible because the observed shifting of the ATLAS cavern floor was smaller than suspected. The enlarged free space allows for installation of a fourth pixel layer with full coverage in the $R\phi$ plane and only small gaps in z direction (along the beampipe). These gaps cannot be avoided, since no space is available to tilt the modules in z direction to create overlaps. Therefore, measures were taken to reduce the inactive edges of the modules and the size of the guard rings to a minimum.

The design of the IBL foresees the installation of 14 support structures, called *staves*, around the beampipe, each holding 32 modules. They will be enclosed by an IBL Support Tube (IST), supporting the IBL during installation and operation. All parts are thinned as much as possible to reach the target radiation length of $0.015\,X_0$ [11]. A technical drawing of the IBL can be seen in Figure 3.1, showing three of the 14 staves.

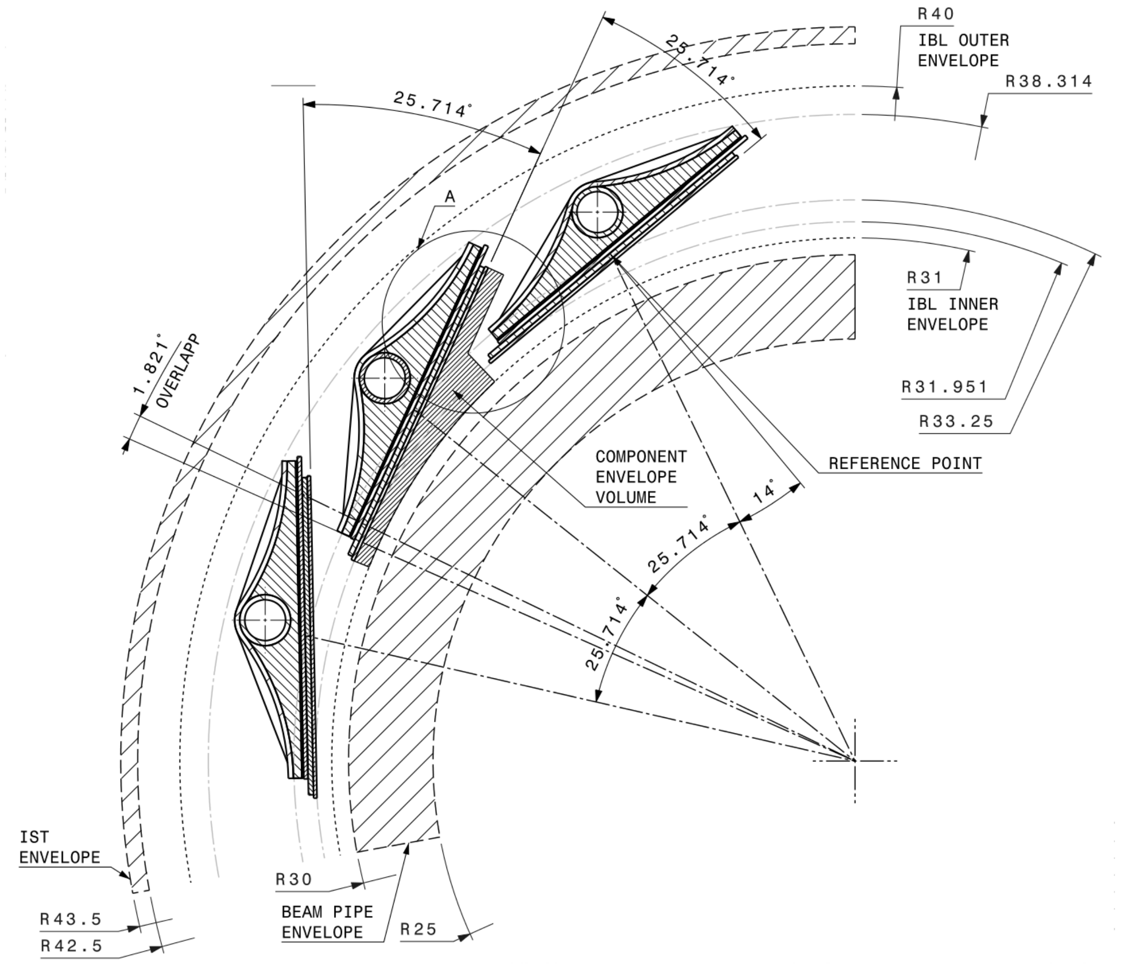


Figure 3.1.: Technical drawing of a part of the IBL, showing the placement of three staves in between the beampipe and the current ATLAS Pixel detector [10]

3.2. The IBL Module Design

Two types of modules were developed for the IBL, consisting of either n-in-n planar silicon or 3D silicon sensors and Front-End I4 read-out chips (FE-I4). The planar sensor modules consist of one sensor and two FEs (double-chip modules), while the 3D sensor modules consist of one sensor and one FE (single-chip modules). Both provide the increased radiation hardness and hit processing capabilities required by the IBL specification as well as individual advantages.

The planar sensors are based on a very mature technology and can be produced by many vendors, resulting in foreseeable high yield, low cost and most likely quick optimization based on previous experiences [12].

The 3D sensors provide faster charge collection, need lower bias voltage after irradiation and are generally more radiation hard than the planar sensors. They are challenged by higher capacitance, efficiency loss under certain circumstances and manufacturability concerns. The production of enough 3D sensors to cover 50% of the IBL was started and 25% of the IBL will actually be covered with 3D sensors, especially the high η (pseudo-rapidity) regions.

Below the n-in-n planar silicon sensor and the FE-I4 read-out chip are described in greater detail.

3.2.1. The n-in-n Planar Silicon Sensors

The IBL sensors are made from silicon, a semiconductor. Therefore the behaviour of particles in matter is described and the signal generation in semiconductors explained, before describing the sensor itself.

Particles in Matter

Energy loss and scattering are the main effects on a particle travelling through matter. The magnitudes of these effects differ depending on the charge and mass of the considered particle, requiring a differentiated approach to the topic. To describe the energy loss of heavy charged particles the *Bethe-Bloch formula* [13] with density effect correction δ and shell correction C

$$-\frac{\mathrm{d}E}{\mathrm{d}x} = 2\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A}\frac{z^2}{\beta^2}\left[\ln\left(\frac{2m_e\gamma^2 v^2 W_{\max}}{I^2}\right) - 2\beta^2 - \delta - 2\frac{C}{Z}\right]$$

is used, with the classical electron radius $r_e = 2.817 \cdot 10^{-15}\,\mathrm{m}$, the charge z of the incident particle in units of the elementary charge e, the electron mass m_e, the density ρ of the absorbing material, Avogrado's number $N_a = 6.022 \cdot 10^{23}\,\mathrm{mol}^{-1}$, $\beta = v/c$ of the incident particle, the Lorentz factor $\gamma = 1/\sqrt{1-\beta^2}$, the mean excitation potential I, the atomic number Z of the absorbing material, the atomic weight A of the absorbing material, and the maximum energy transfer in a single collision $W_{\max}$.

Regarding electrons and positrons the assumption of an undeflected projectile has to be dropped and the indistinguishability of the projectile and target considered, leading to the modified Bethe-Bloch formula

$$-\frac{\mathrm{d}E}{\mathrm{d}x} = 2\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A}\frac{z^2}{\beta^2}\left[\ln\left(\frac{\tau^2(\tau+2)}{2(I/m_e c^2)^2}\right) + F(\tau) - \delta - 2\frac{C}{Z}\right],$$

where τ is the kinetic energy of the projectile in units of $m_e c^2$. Additionally the energy loss by *bremsstrahlung* has to be considered, which amounts to

$$-\frac{\mathrm{d}E}{\mathrm{d}x}_{\text{brems}} = 2\alpha N_a r_e^2 \frac{Z(Z+1)}{A} E \ln\left(\frac{183}{Z^2}\right) ,$$

where α is the fine-structure constant and E the energy of the incident particle. This formula is also used to define the radiation length X_0 as

$$X_0 \coloneqq \frac{A}{2\alpha N_a r_e^2 Z(Z+1)\ln(183/Z^2)} ,$$

which is the mean distance a high energy electron travels through a specific material until it retains only $1/e$ th of its original energy.

Signal Generation in Semiconductors

The energy lost by the projectile ionizes the material it traverses, creating free positive and negative charges collectable by applying an external electric field. While drifting towards the respective electrode the moving charges instantaneously induce a charge on read-out electrodes attached to the material. The amplitude of the induced charge can be calculated as a function of time by applying the *Shockley-Ramo theorem* [14].

Since the induced charge depends on the amount of charges drifting towards the read-out electrode it is beneficial to choose a material with low ionisation energy, high density and high mobility of charges to gain large signals, good energy resolution, and high detection rates. This is provided by semiconductors like silicon, used for the current APD as well as the IBL.

The desired low ionisation energy has the side effect of producing a large noise signal from thermal electrons, easily excited to the conducting band. To counter this effect atoms with an atomic number higher or lower than silicon are introduced into the crystal, called either *n*- or *p*-doping. Applying a voltage to a *p*-*n* junction creates a depletion zone at the boundary of the implantations, removing all free charge carriers when fully depleted, and thus minimizing noise.

IBL Sensor Design

The IBL planar *n*-in-*n* silicon sensor consists of a 200 µm thick, lightly *n*-doped substrate containing highly n^+-doped implantations on the front side and a highly p^+-doped implantation on the back side [11].

The p^+ implantation opposite of the pixel matrix is designed as a single large high voltage pad separated from the cutting edge by a 350 µm wide strip consisting of 13 guard ring implantations, providing a controlled voltage drop from high voltage to ground. The guard ring area is overlapped by the pixels as shown in Figure 3.2, determined to be the design providing the slimmest edge possible while still allowing safe depletion before irradiation.

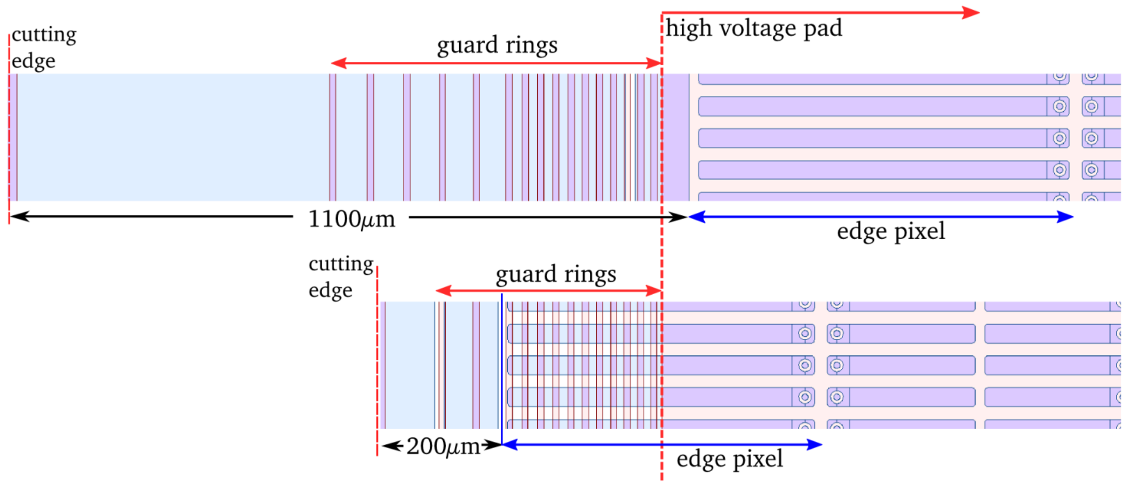


Figure 3.2.: Schematic comparing the current ATLAS Pixel detector's (top) and IBL's (bottom) sensor layout. In the IBL design the pixels overlap the guard rings to minimize the inactive edge [11]

The n^+ implantation on the front side of the sensor is segmented into a pixel matrix of 80 columns by 336 rows of mostly $250\,\mu m \times 50\,\mu m$ pixels, matching the design of the FE-I4 read-out chip (Section 3.2.2). Only the outermost columns contain long pixels of 500 μm length, which overlap the guard ring structure by 250 μm. The pixels are separated from each other following the moderated p-spray concept. The entire matrix is surrounded by an inactive edge region required to be $(210 \pm 10)\,\mu m$ wide, containing a 90 μm wide bias grid ring and an edge implant (outer guard), ensuring an equal potential on the surface outside the pixel matrix and the cutting edge. The bias ring is connected to the bias grid, a structure prohibiting a floating potential on pixels with an open bump-bond connection. In a fully functional module all pixels, the bias grid, and the outer guard are bump-bonded to the FE-I4.

The sensor was shown to have good hit efficiency and remain operational after type inversion, due to the depletion zone growing from the n^+ implantation.

3.2.2. The Read-out Electronics

Described is the FE-I4A prototype IC (FE), changes planed for the FE-I4B production IC are described at the end of the Section. First the specifications of the FE are discussed, followed by descriptions of the analogue and digital architecture and the periphery embedded in the hybrid read-out IC.

Specifications

The FE (Figure 3.3) is a pixel array consisting of 26 880 pixels in 80 columns in direction of the beampipe and 336 rows in $R\phi$ direction [11]. Measuring 18.8 mm × 20.2 mm when diced, the FE is the largest ASIC designed for a HEP application so far, providing several size related advantages. Most important are an improved active over total area ratio and a reduced material budget, due to a highly integrated design. The pixel and feature size of the FE were shrunk to a 250 µm × 50 µm pixel size and a 130 nm feature size bulk CMOS process. This increased the digital circuitry density, allowing for an increased complexity of the on chip digital circuits, enabling them to cope with higher hit rates. The smaller feature size also increased the total ionizing dose (TID) radiation hardness to a 250 MRad rating, fitting the IBL environment.

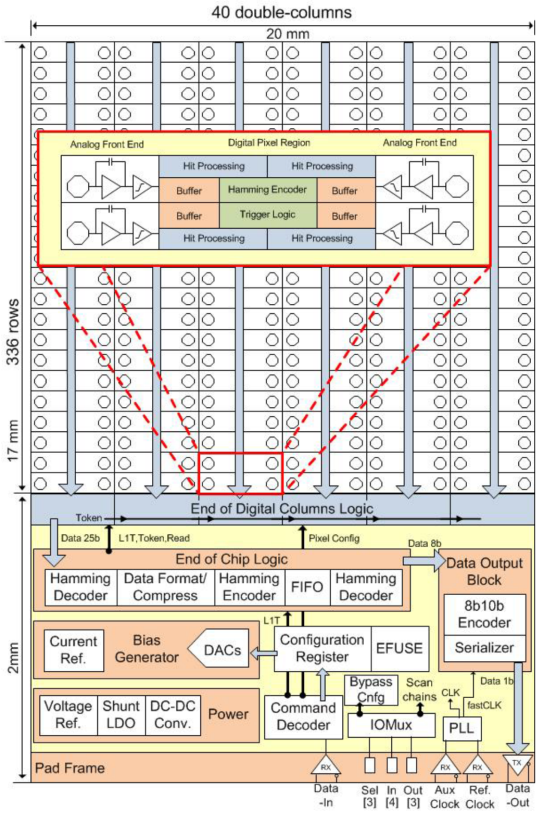


Figure 3.3.: Schematic of the FE-I4 read-out chip layout, enlarged are four analogue pixels connected to a 4-Pixel Digital Region [11]

Most significant is the new pixel matrix architecture, storing data locally at the pixel level until triggering and subsequent propagation of the trigger inside the pixel array.

ToT_{Code}	0-12	13	14	15
ToT [25 ns]	$\text{ToT}_{\text{Code}} + 1$	> 13	delayed hit	no hit

Table 3.1.: Data encoded in the time-over-threshold as read out from the FE-I4 (ToT_{Code}) [2]

This ensures the FE read-out to be efficient up to a luminosity of $3 \cdot 10^{34}\,\text{cm}^{-2}\,\text{s}^{-1}$ at the IBL radius.

Analogue Electronics

The 26 880 pixels are organized in columns of 80 analogue pixels. Each pair of columns has a digital double-column centred in between. Each pixel consists of an independent analogue section (Figure 3.4) taking up about 60% of the total pixel area. Using a two-stage amplifier, the collected charge from the bump-bonded sensor is amplified and passed on to the discriminator. Once the collected charge reaches the adjustable threshold of a comparator, the hit is discriminated and the time over threshold (ToT) recording starts. The ToT is later translated to collected charge via a proportionality factor adjustable by changing the return to baseline behaviour of the pixel. Linearity of the ToT signal is achieved by using a continuous current-source reset on the preamplifier. The ToT read out from the FE is an encoded value ToT_{Code}, the translation to real ToT is shown in Table 3.1.

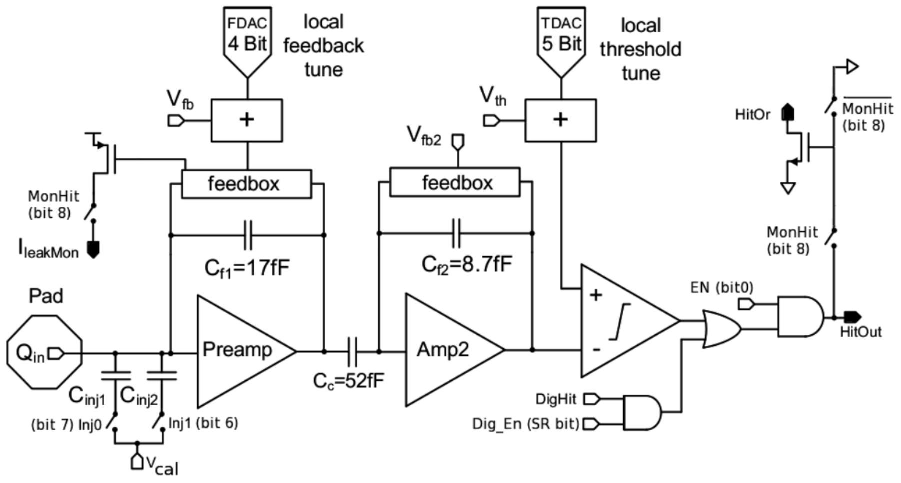


Figure 3.4.: Schematic of the FE-I4 read-out chip's analogue pixels, showing the two-stage amplifier and discriminator [11]

For testing and calibration of the analogue section charges can be injected at the preamplifier input, and for testing the trailing digital section after the discriminator.

Each pixel stores 13 bit locally to control the injection switches, tune the feedback current determining the charge to ToT proportionality factor, tune the discriminator threshold, switch on the MonHit and HitOR output and for masking off the pixel.

Timewalk Depending on the energy deposited in the sensor the rise time of the preamplifier output differs. The maximum output is always reached after the same time, therefore the delay between the beginning of charge collection and reaching the discriminator threshold differs. It is shortest for large charges and gets longer for smaller charges. The difference between delays is called *timewalk* (Figure 3.5) and can be measured as a function of collected charge.

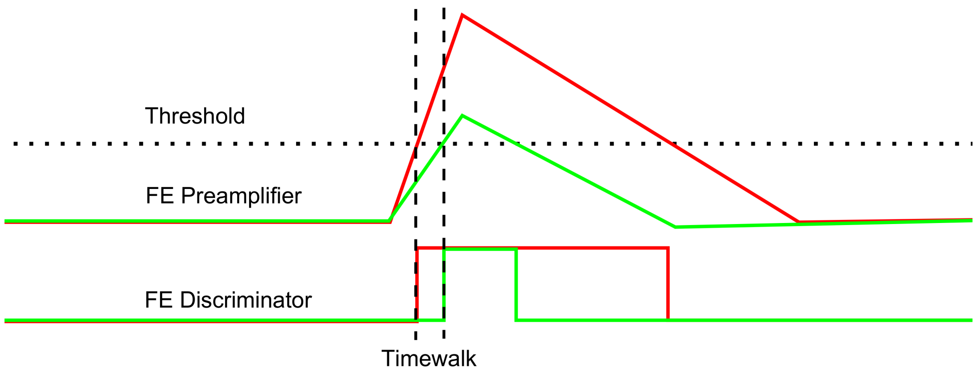


Figure 3.5.: Sketch showing an analogue pixel's preamplifier and discriminator output, illustrating the timewalk. The red line represents a large charge, the green line a small charge

Digital Electronics

The digital circuitry is organized in digital double-columns centred in between pairs of analogue columns (Figure 3.6). Inside the digital double-column four analogue pixels are tied to a *4-Pixel Digital Region* (4-PDR). Communication is handled inside the double-columns and coordinated by peripheral logic. Hit information is stored locally in the 4-PDRs in 20 buffer memories, five for each analogue pixel, holding the ToTs during trigger latency until triggering or erasing.

Inside the 4-PDRs a separate hit processing unit exists for each of the four discriminator outputs, which provide timestamping and encode the ToT. An additional level of digital discrimination can be used to distinguish small from large collected charges. When a large charge is detected by one or more of the four hit processing units the central latency counter is booked and four ToT buffers, one from each of the four hit processing units, are associated to the event.

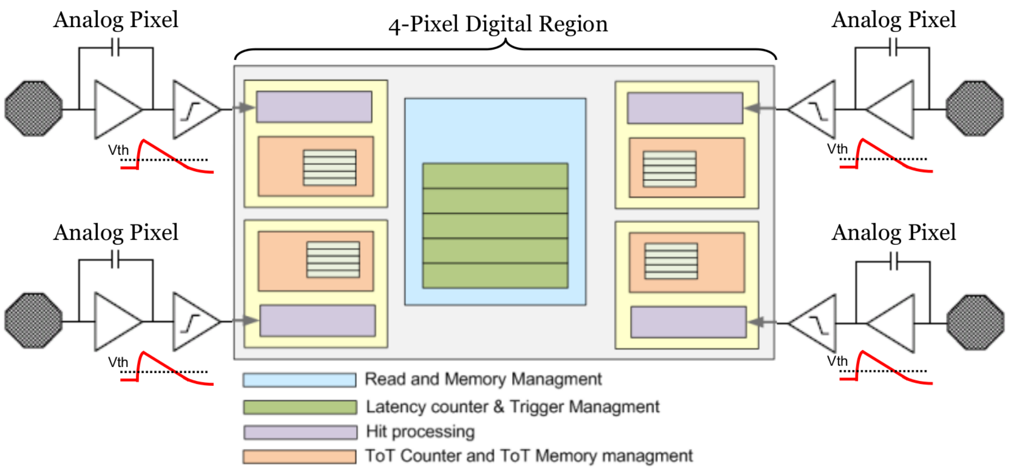

Analog Pixel
4-Pixel Digital Region
Analog Pixel
Vth
Analog Pixel
Analog Pixel
Vth
Read and Memory Managment
Latency counter & Trigger Managment
Hit processing
ToT Counter and ToT Memory managment

Figure 3.6.: The 4-Pixel Digital Region (4-PDR) of the FE-I4, connected to four analogue pixels

This processing scheme is advantageous in several ways. Resources are shared within the 4-PDR, pixel-level storage reduces power consumption, digital discrimination of large and small hits can be used for timewalk compensation and the active fraction is increased by removing circuitry from the periphery.

Periphery

The periphery contains blocks for communication and operational tuning, data read-back organization, fast data output serialization, extra testing capabilities, and prototype blocks. For proper operation, the clock and the command Data-In need to be provided via two LVDS inputs. Received trigger signals are decoded and sent to the 4-PDRs and to the End of Chip Logic (EoCL). When a trigger passed to a 4-PDR confirms a hit, the four ToTs associated with the hit are sent to the EoCL as well, where they are encoded and saved in a FIFO to be read out.

Production IC

For the FE-I4B production IC one of multiple double-column flavours used on the FE-I4A prototype IC was chosen and thus the chip made uniform. A few DAC ranges were extended and an issue with leakage current on the pulse generator addressed. The Shift Register read back needed to be made more reliable and the skipped trigger counter properly implemented. Some new features, like a temperature sensing circuit and some DAC related features, were implemented.

4. The Experimental Set-up

This chapter describes the experimental set-up designed to most closely resemble the environment faced by the device-under-test (DUT) in testbeams (TB). To achieve this a *Data Acquisition System* (DAQ) similar to existing testbeam set-ups was constructed.

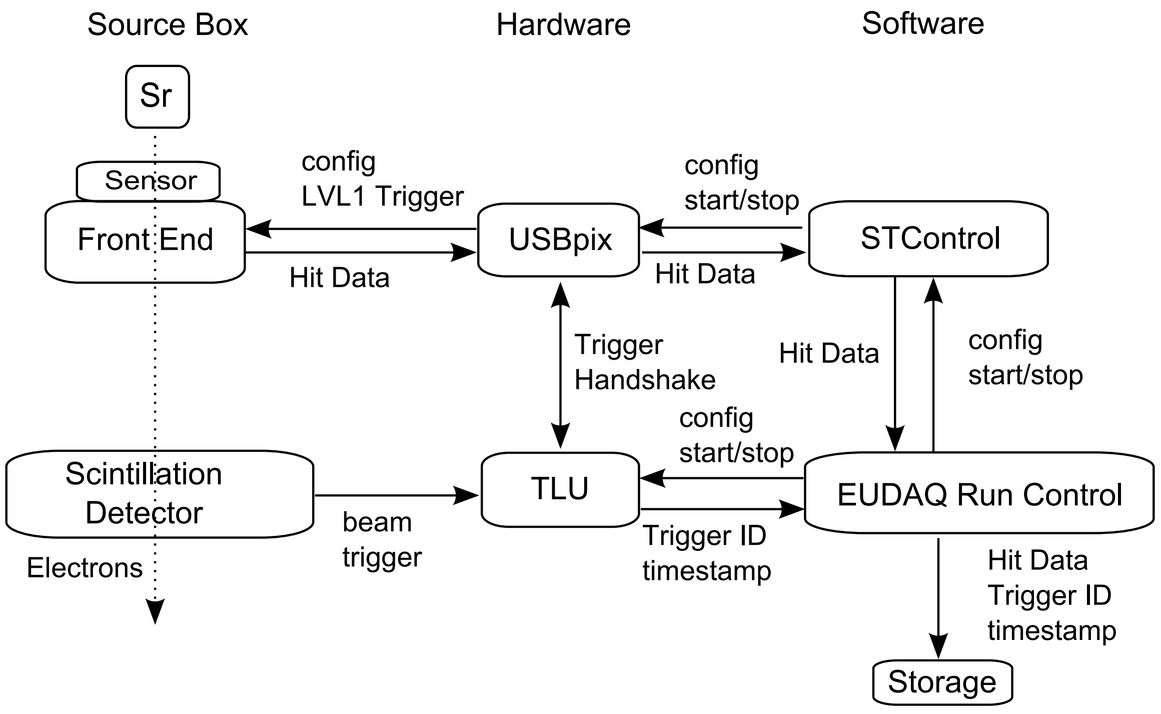


Figure 4.1.: Sketch of the Data Acquisition system

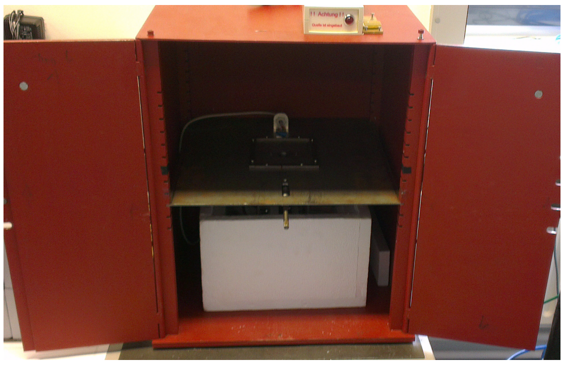
4.1. Overview

The experimental set-up is divided into three parts. First is the *Source Box* (SB), containing a Strontium source, a sensor (Section 3.2.1), a Front-End read-out chip (FE, Section 3.2.2) and a scintillation detector. Next are a *Trigger Logic Unit* (TLU) and a USBpix board, used to generate the trigger signal and communicate with the FE. Last is the software controlling the set-up and writing the data (EUDAQ Run Control), and configuring the FE via the USBpix board (STControl).

The entire system including the communication links is shown in Figure 4.1 and described below.

4.2. The Source Box

The Source Box (SB, Figure 4.2a) is a metal box housing and shielding the radioactive source, sensor, FE and scintillation detector. The inside of the SB is divided horizontally by a metal plate with a centred shutter, closed by a movable lead block a few centimetres thick. On top of the shutter a lead box with a second shutter can be placed, containing the radioactive source. Once this box is firmly in place its shutter is opened. The shutter in the metal divider is only opened from outside the SB, simultaneously locking its door.


(a) The SB with metal divider and Styrofoam box in place

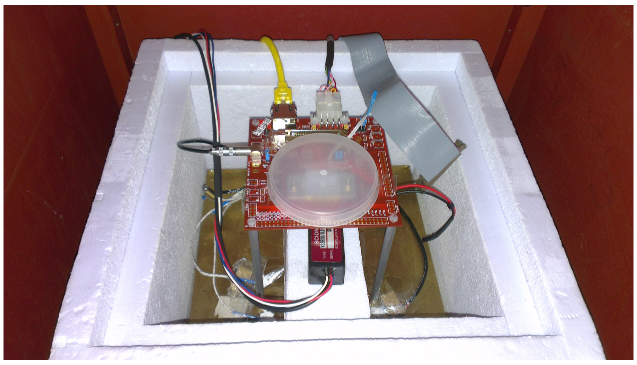

(b) The Front-End and scintillation detector mounted inside the SB

Figure 4.2.: The Source Box (SB)

The lower part of the SB contains a Styrofoam box, part of a cooling system for irradiated sensors not used in this set-up. The box provided the base for mounting the scintillation detector and FE as seen in Figure 4.2b. Using metal spacers the FE was placed approximately 2 cm below the source and the scintillation detector just below the FE.

Strontium Source

A particle beam was provided by the decay chain of the Strontium isotope ^{90}Sr, decaying to ^{90}Y and further decaying to ^{90}Zr, both via β^--decay. The first isotope has a half-life of 28.78 a and emits electrons with an energy of 0.546 MeV. The first decay product has a half-life of 64.1 h and emits 2.282 MeV electrons. The source used provided an activity of 18 MBq.

Scintillation Detector

A V22B20/0.3-E1-PX scintillation detector by Scionix containing an EJ-204 plastic scintillator by Eljen Technology, combined with a PMT type H10721-110, was used. It was operated with an input voltage of 5 V and a control voltage of 0.9 V, experimentally determined to produce an approximately 1 kHz output frequency with the source in place, and negligible noise when the source was removed.

4.3. The USBpix System

4.3.1. Hardware

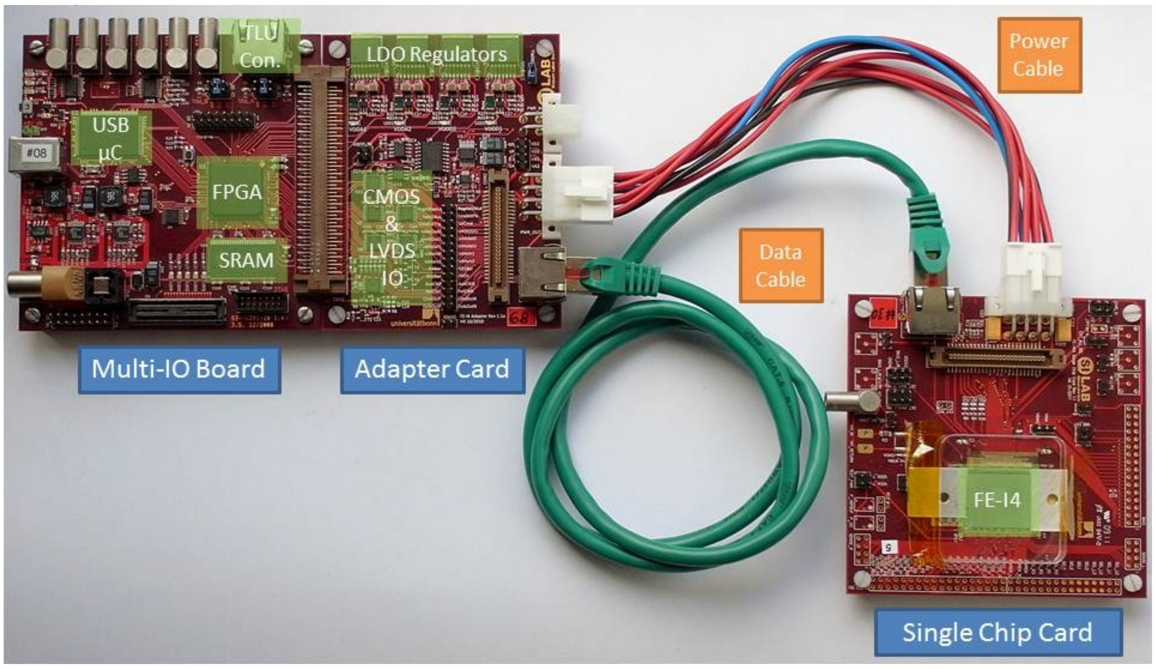


Figure 4.3.: USBpix board with FE-I4, the power to the Single Chip Card was actually provided by a dedicated power supply[1]

The *USBpix system* [15] was developed to provide a lightweight and modular read-out system for the FE-I3 and FE-I4 read-out chips. It consists of a *Multi-IO board*, an *Adapter card* and *Single chip card* specific to the flavour of FE under test. The system is connected to a PC via USB 2.0 and configured and controlled by the *STControl* software.

All hardware parts of the system can be seen in Figure 4.3 and are described below.

[1] `http://icwiki.physik.uni-bonn.de/twiki/pub/Systems/UsbPix/USBpixFE-I4.jpg`, 4th July 2013

The Multi-IO Board

The S3 Multi-IO board version 1.03 [16] contains a USB 2.0 micro-controller, an FPGA, a 2 MB SRAM, and a versatile arrangement of connectors. The micro-controller handles the communication between the PC and the board, while the FPGA handles the communication between the board and the FE. It also stores the information sent by the FE in the SRAM.

Communication to and from the PC is transmitted via the USB 2.0 port, to and from the TLU via an RJ45 port. The board is connected to the FE specific adapter card via a 100 pin connector, the three LEMO in- and three LEMO outputs were not used.

FE-I4 Adapter Card

The Adapter Card rev. 1.1a is connected to the Multi-IO board via the 100 pin connector and to the Single Chip Card holding the FE via an RJ45 port. The card provides LVDS signal buffers, contains the USB regulators and offers several headers for functionality tests.

Single Chip Card

The Single Chip Card rev. 1.1 holds a wire-bonded FE and optionally a sensor bump-bonded to the FE. Power for the FE is supplied by a dedicated power supply, while data transfer between cards is accomplished using an RJ45 port. The bias voltage for the sensor is supplied via a LEMO connector.

4.3.2. STControl

The USBpix system was controlled by the STControl software version 4.3 with EUDAQ [17] integration, running on Windows XP. The application is based on the ATLAS PixLib package, a collection of C++ classes used to access the ATLAS Pixel detector's RODs. It was adapted to communicate with an FE via the USB/FPGA interface and given a graphical user interface (GUI) using ROOT and Qt.

The Main Window

The STControl main window (Figure 4.4) consists of a left and a right panel. The left panel displays the modules and DCS objects stored in the loaded configuration, controlled via GPIB, and provides interfaces for modification of the configuration and control of supply voltages.

When operating STControl in standalone mode the right panel is used for sending configuration commands to the modules, start scans, and view received data and logs. This is done by choosing between several right panels, the most important of which are listed below. When connected to a EUDAQ Run Control (Section 4.5) all these tasks are handled by the Run Control.

Tool Panel Modules and DCS objects are initialized via the Tool panel.

PixScan Panel The PixScan panel offers a variety of scans in a drop-down menu, the most important of which are the

RX_DELAY_SCAN Used to determine the proper timing for faultless read-out of the FE, has to be done first.

ANALOG_SCAN Injects a predefined number of pulses at the FE's preamplifiers and records the number of replies per pixel to test the analogue pixels.

DIGITAL_SCAN Injects pulses after the analog pixels' discriminators, used to test the digital section.

SOURCE_SCAN Starts a data run recording a previously defined number of events triggered by one of several possible trigger sources.

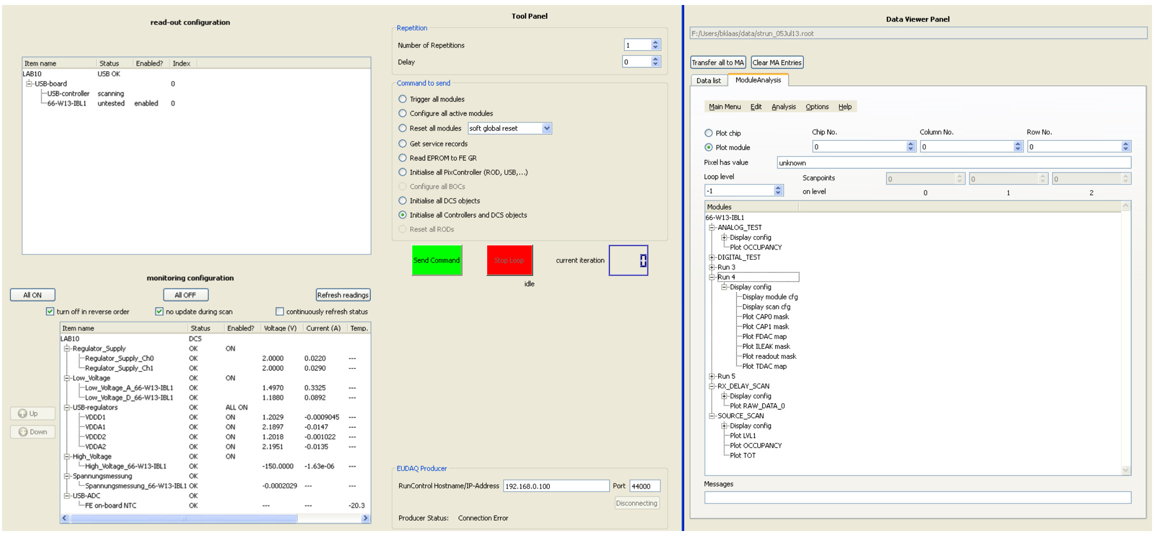
THRESHOLD_SCAN Injects different charges into the analogue pixel to precisly determine the discriminator threshold.

Data Panel Examination of the scan results using ROOT histograms is done using the Data Panel.

EUDAQ Integration

To be able to connect to the EUDAQ Run Control STControl is compiled in testbeam mode, providing the USBpixI4 Producer (Section 4.5.1) and creating a second STControl executable offering the `-r [IP-Address]` parameter to connect to a Run Control, running at the specified IP, via the TCP/IP protocol. Communication between STControl and the Run Control is managed by the Producer, allowing the Run Control to perform all necessary operations for data recording.

When starting a run via the RunControl the `TESTBEAM_EUDAQ` scan is automatically selected within STControl, preconfigured to listen on the Multi IO board's RJ45 port for



read-out configuration

Item name	Status	Enabled?	Index
LAB10	USB OK		
USB-board			0
USB-controller	scanning		
66-W13-IBL1	untested	enabled	0

monitoring configuration

[✓] turn off in reverse order [✓] no update during scan [ ] continuously refresh status

Item name	Status	Enabled?	Voltage (V)	Current (A)	Temp.
LAB10	DCS				
Regulator_Supply	OK	ON			
Regulator_Supply_Ch0	OK		2.0000	0.0220	---
Regulator_Supply_Ch1	OK		2.0000	0.0290	---
Low_Voltage	OK	ON			
Low_Voltage_A_66-W13-IBL1	OK		1.4970	0.3325	---
Low_Voltage_D_66-W13-IBL1	OK		1.1880	0.0892	---
USB-regulators	OK	ALL ON			
VDDD1	OK	ON	1.2029	-0.0009045	---
VDDA1	OK	ON	2.1897	-0.0147	---
VDDD2	OK	ON	1.2018	-0.001022	---
VDDA2	OK	ON	2.1951	-0.0135	---
High_Voltage	OK	ON			
High_Voltage_66-W13-IBL1	OK		-150.0000	-1.63e-06	---
Spannungsmessung	OK				
Spannungsmessung_66-W13-IBL1	OK		-0.0002029	---	---
USB-ADC	OK				
FE on-board NTC	OK		---	---	-20.3

Tool Panel

Repetition
Number of Repetitions: 1
Delay: 0

Command to send
- ( ) Trigger all modules
- ( ) Configure all active modules
- ( ) Reset all modules: soft global reset
- ( ) Get service records
- ( ) Read EPROM to FE GR
- ( ) Initialise all PixController (ROD, USB,...)
- ( ) Configure all BOCs
- ( ) Initialise all DCS objects
- [✓ Initialise all Controllers and DCS objects]
- ( ) Reset all RODs

current iteration: 0
idle

EUDAQ Producer
RunControl Hostname/IP-Address: 192.168.0.100 Port: 44000
Producer Status: Connection Error

Data Viewer Panel

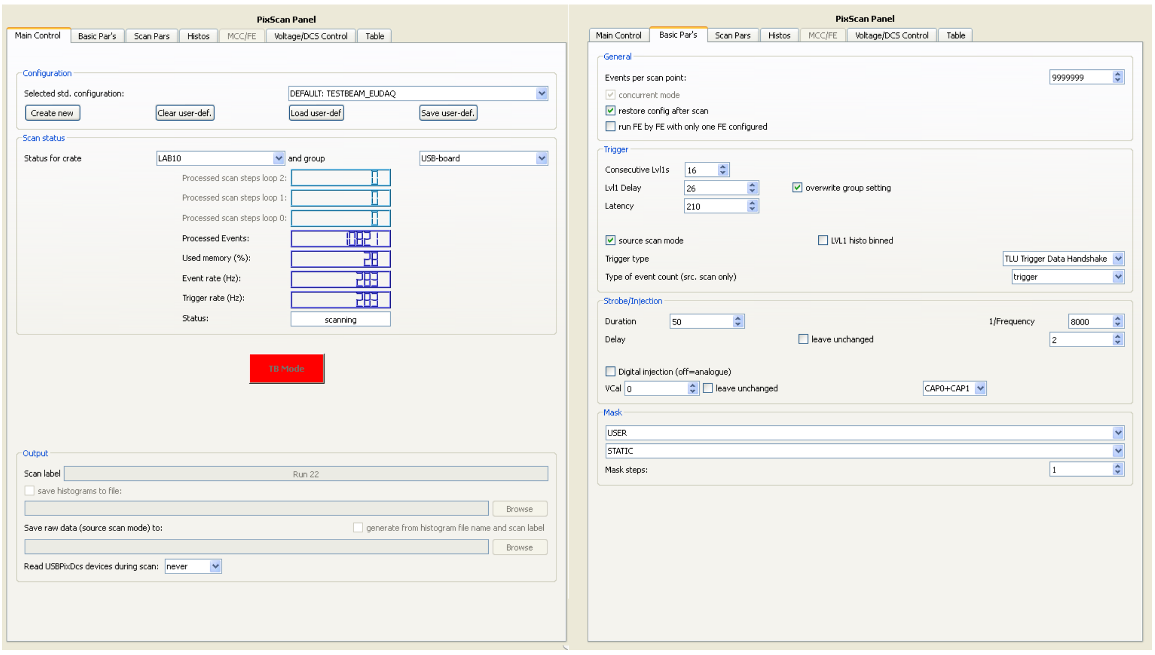
F:/Users/bklaas/data/strun_05Jul13.root

( ) Plot chip [✓ Plot module]
Chip No.: 0 Column No.: 0 Row No.: 0
Pixel has value: unknown
Loop level: -1
Scanpoints on level 0: 0, 1: 0, 2: 0

```
Modules
66-W13-IBL1
 ANALOG_TEST
   Display config
   Plot OCCUPANCY
 DIGITAL_TEST
 Run 3
 Run 4
   Display config
     Display module cfg
     Display scan cfg
     Plot CAP0 mask
     Plot CAP1 mask
     Plot FDAC map
     Plot ILEAK mask
     Plot readout mask
     Plot TDAC map
 Run 5
 RX_DELAY_SCAN
   Display config
   Plot RAW_DATA_0
 SOURCE_SCAN
   Display config
   Plot LVL1
   Plot OCCUPANCY
   Plot TOT
```

Messages

(a) Left Panel, Tool Panel and Data Panel



PixScan Panel – Main Control

Configuration
Selected std. configuration: DEFAULT: TESTBEAM_EUDAQ

Scan status
Status for crate: LAB10 and group: USB-board
Processed scan steps loop 2: 0
Processed scan steps loop 1: 0
Processed scan steps loop 0: 0
Processed Events: 10821
Used memory (%): 28
Event rate (Hz): 283
Trigger rate (Hz): 283
Status: scanning

Output
Scan label: Run 22
[ ] save histograms to file:
Save raw data (source scan mode) to: [ ] generate from histogram file name and scan label
Read USBPixDcs devices during scan: never

PixScan Panel – Basic Par's

General
Events per scan point: 9999999
[✓] concurrent mode
[✓] restore config after scan
[ ] run FE by FE with only one FE configured

Trigger
Consecutive Lvl1s: 16
Lvl1 Delay: 26 [✓] overwrite group setting
Latency: 210
[✓] source scan mode [ ] LVL1 histo binned
Trigger type: TLU Trigger Data Handshake
Type of event count (src. scan only): trigger

Strobe/Injection
Duration: 50 1/Frequency: 8000
Delay: [ ] leave unchanged 2
[ ] Digital injection (off=analogue)
VCal: 0 [ ] leave unchanged CAP0+CAP1

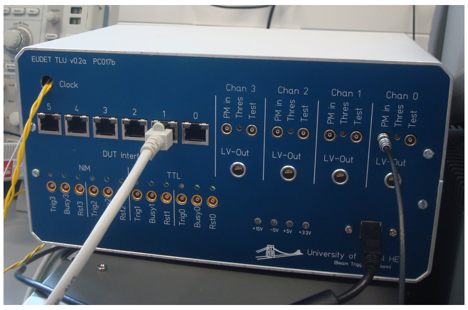
Mask
USER
STATIC
Mask steps: 1

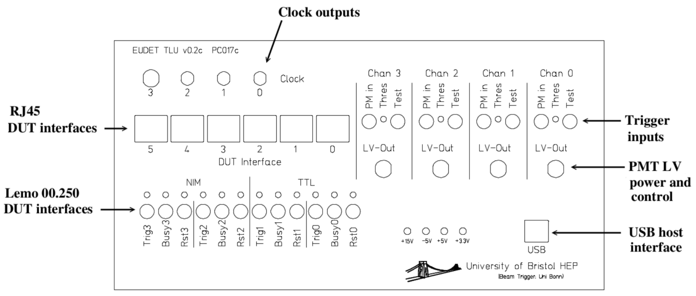
(b) The *Main Control* and *Basic Par's* tabs of the PixScan Panel

Figure 4.4.: Pictures of various panels of the STControl application, showing interfaces for configuration of modules and DCS objects, choosing scans and trigger modes, and data analysis

triggers from a TLU and respond in *Trigger Data Handshake* mode. It is also set up to record the maximum number of 9 999 999 events before quitting, so the run can be stopped from the Run Control and is not previously force stopped by STControl. All data is passed to the Run Control's Data Collector by the USBpixI4 Producer.

4.4. The Trigger Logic Unit


(a) A TLU v0.2a connected to the beam-trigger, DUT and DAQ



(b) Sketch of the front panel of a TLU v0.2c [18]

Figure 4.5.: The Trigger Logic Unit, shown are a picture of a TLU v0.2a and a sketch of a TLU v0.2c

To trigger the FE read-out via the USBpix board based on the scintillation detector's output a *EUDET JRA1 Trigger Logic Unit v0.2a* (TLU, Figure 4.5) [19] was used. It was developed at Bristol University to communicate trigger information between the beam-trigger, DAQ and DUT in testbeam environments. Below the used TLU v0.2a and the improved TLU v0.2c [18] are described, both of which provide multiple trigger in- and outputs, clock I/O and a USB 2.0 interface to the Run Control. If no version is specified the description applies to both.

4.4.1. Trigger IO

The TLU provides four channels for trigger input, which can be logically linked via AND- and OR-masks. The TLU creates an output trigger signal on a previously defined arrangement of ports, when a combination of triggers matching the defined mask is received. Available are six RJ45 ports transmitting LVDS signals, two LEMO ports transmitting TTL signals and two LEMO ports transmitting NIM signals. Only the RJ45 ports can be operated in all three of the TLU's trigger handshake modes, since only those transmit a `TRIGGER_CLOCK` line in addition to the `TRIGGER`, `BUSY` and `RESET` lines. Therefore they were chosen to convey the trigger to the DUT. The mentioned trigger handshake modes are:

No-Handshake: The TLU raises the `TRIGGER` line for a fixed period of time whenever a trigger is supposed to be issued.

Simple Handshake When a trigger is issued the TLU asserts `TRIGGER` and only de-asserts it when the DUT raises the `BUSY` line in response. After `BUSY` goes low again the TLU is sensitive to new triggers.

Trigger Data Handshake The TLU asserts `TRIGGER`, in response the DUT asserts `BUSY`. Once the TLU recognizes `BUSY` going high it de-asserts `TRIGGER` and switches the `TRIGGER` line to the output of a shift register holding the TriggerID. In response the DUT clocks out the ID by toggling `TRIGGER_CLOCK`. Once 15 cycles are completed the TLU returns `TRIGGER` to low, the DUT de-asserts `BUSY` and after 16 cycles the system is sensitive to new Triggers (Figure 4.6).

Since the TriggerID can be used to match the triggers issued by the TLU to events recorded by the DAQ later on the *Trigger Data Handshake* was chosen.

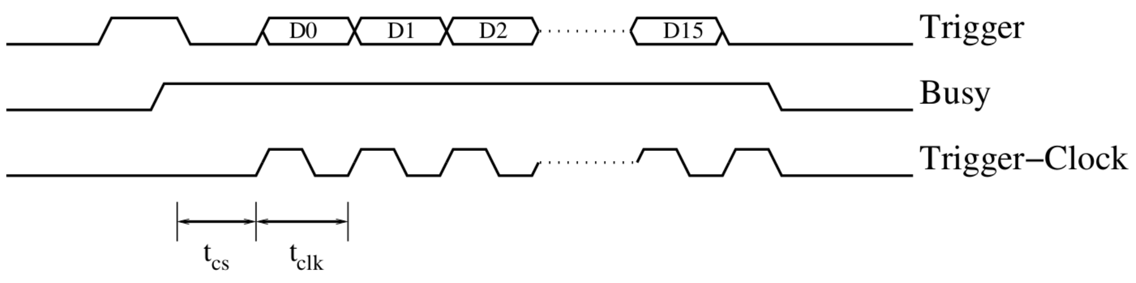


Figure 4.6.: Timing of signals in *Trigger Data Handshake* mode [18]

4.4.2. Timing

By default the TLU uses a 48 MHz clock on the internal USB-FPGA-board for timing. Using this clock trigger information and commands are processed and timestamps for the issued triggers generated. The clock source for timestamp generation can supposedly be changed to an external clock, in this case taken from the `RefClk` headers on the USBpix board's Adapter card. This signal needed to be boosted in order to minimize the power drawn from the Adapter card, achieved by building a *LVDS Booster board* (Figure 4.7).

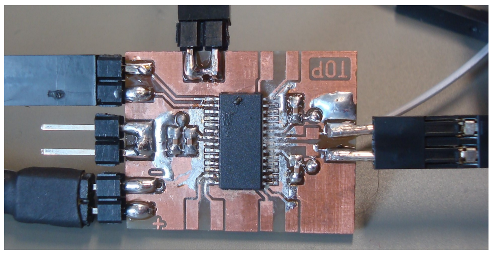


Figure 4.7.: LVDS Booster board built to boost the clock signal taken from the USBpix system and passed to the TLU

Timestamps

Based on the selected clock a 64 bit timestamp is generated and recorded for each trigger issued. It is obtained by the DAQ via writing a specific sequence of registers on the TLU's FPGA and saved per event.

TLU v0.2a The timestamps generated by this version are increased by a value of 8 once per clock cycle, thus providing a resolution of 25 ns when using the LHC's 40 MHz clock.

Using this TLU version's default firmware v241 an external clock should be selectable for timestamp generation by setting the `CLOCK_SOURCE_SELECT_ADDRESS` to `1`. This failed because the TLU became unoperational when this firmware was loaded, so an internally generated 40 MHz clock had to be used for timestamp generation, provided by the firmware v65_40MHz.

TLU v0.2c The timestamps generated by this version are incremented eight times per clock cycle, thus providing a precision of 3.125 ns when using the LHC's 40 MHz clock.

This version is outfitted with a clock board containing a CDE929 PLL chip, capable of generating a large variety of clocks. Either from an external clock or a quartz crystal, depending on links on a header on the clock board. The source of the clock used by the FPGA can be set by writing the `CLOCK_SOURCE_SELECT_ADDRESS` register, the same as for the TLU v0.2a. When using the TLU v0.2c's default firmware v253 writing `0` to the register selects the output from the clock board, and thus an external clock when also correctly configuring the clock board.

Trigger Latency

Due to the wiring inside the TLU and processing time of the FPGA a (27.3 ± 3.0) ns delay exists between a trigger arriving at the TLU and a trigger being issued. The uncertainty reflects a wide range of factors, e.g. temperature variations and differences in supply voltage, which may vary from day to day but can be assumed constant for a single run, therefore greatly reducing the uncertainty to a jitter of 31 ps RMS according to lab measurements [18]. Since this timespan is negligible compared to the 3.125 ns resolution of the timestamp the trigger latency is assumed constant.

4.4.3. FPGA Registers and Address Map

To configure the TLU a specific firmware file is loaded and several registers specified by the firmware are written by the *TLU Producer*, part of the EUDAQ Run Control (Section 4.5). The most important registers are listed below.

TRIG_INHIBIT_ADDRESS Enables and disables triggers.

STATE_CAPTURE_ADDRESS Copies the current `BUFFER_POINTER`, `TIMESTAMP`, `TRIGGER_COUNTER` and `TRIGGER_IN` counters to the read-out registers.

INITIATE_READOUT_ADDRESS Prepares the data transfer from the TLU.

CLOCK_SOURCE_SELECT_ADDRESS Switches the clock source for the timestamp between internal and either front panel or clock board clock

BEAM_TRIGGER_AMASK_ADDRESS Contains the AND mask for input triggers.

DUT_MASK_ADDRESS Sets the active DUT sockets.

DUT_TRIGGER_ADDRESS Asserts `TRIGGER` for one clock cycle.

DUT_BUSY_ADDRESS Value of the `BUSY` line coming from the DUT.

TIMESTAMP_ADDRESS Holds the current timestamp, copied to `REGISTERED_TIMESTAMP_ADDRESS` when writing to `STATE_CAPTURE_ADDRESS`

4.5. The EUDAQ Run Control

The EUDAQ software consists of a collection of individual processes, designed for portability and modularity. The central process is the Run Control [17], receiving information from and sending commands to all other running processes. Its GUI shows all registered processes and their status and provides an interface for configuration and run control. The processes needed for proper operation of the DAQ system and any modifications are described below, a schematic of the software's architecture is shown in Figure 4.8.

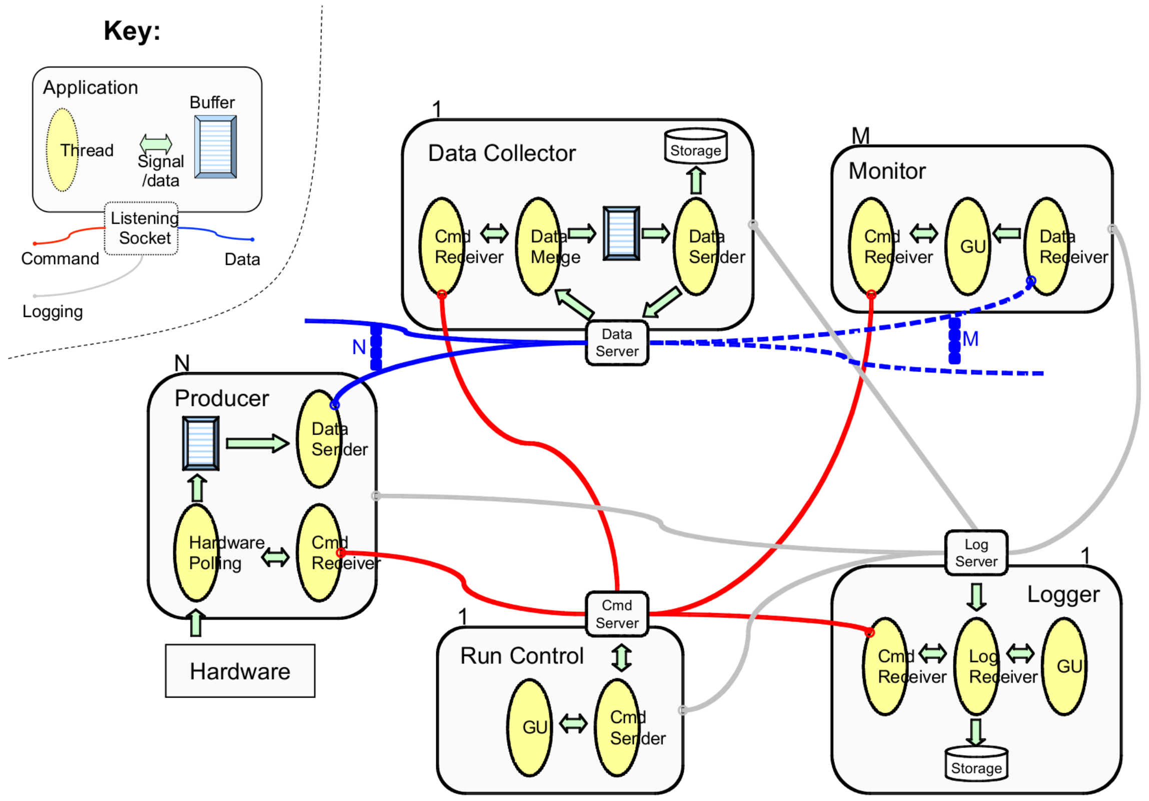


Figure 4.8.: Schematic of the EUDAQ architecture [17]

4.5.1. Producer

All components of the DAQ system producing data need a process called *Producer* for operation. It passes the data generated by the component, properly encoded for later analysis, to the *Data Collector* and relays all commands issued by the Run Control to the hardware. This is achieved through a specific set of functions matching the options offered to the user by the Run Control, called when making the corresponding user input.

For this set-up two Producers were used, the TLU Producer, part of EUDAQ, and USBpixI4 Producer, part of STControl with EUDAQ integration. The TLU Producer uses functions from the `TLUController` class, extended for setting the clock for timestamp generation to an external source. The method `SetClockSourceSelect` (Listing A.1) was implemented as a public member function of the `TLUController` class and the corresponding function call added to the `OnConfigure` method of the TLU Producer. Depending on the firmware loaded during configuration a specific value to be written to `SELECT_CLOCK_SOURCE_ADDRESS` has to be set in the TLU Producer and the software recompiled. The function's output, visible in the TLU Producer's GUI, showed the function being called correctly.

4.5.2. Other Processes

Data Collector

All data streams are transmitted to the Data Collector to be combined and stored on disk. Using the `Serializer` class the structured data objects are serialized and passed on to the `FileWriter` class to be written to binary raw files.

Log Collector

Similar to the Data Collector combining and storing all data streams the Log Collector receives all log messages, generated by the DAQ or entered by the user, and stores them in a single file. This approach offers easy access to all log messages for faster troubleshooting.

Online Monitor

The data file written by the Data Collector is read once per second by the Online Monitor, immediately displaying its contents using a set of ROOT histograms. This offers close monitoring of the DAQ's operation and provides almost real-time access to the recorded data.

4.5.3. Configuration

From the Run Control GUI a configuration file has to be chosen and applied to the registered processes using the *Config* button. The file consists of an individual section for each Producer, containing a set of `key` and `value` pairs modifying the Producer's default settings. The config file used is shown in Listing A.2.

5. The Analysis Framework

This chapter describes the framework used to analyse the data collected by the DAQ system described in Chapter 4. It consists of the parts of the EUDAQ software used to extract the relevant data from the binary raw files written by the DAQ, and the tools developed to analyse this data and calculate the timewalk.

5.1. EUDAQ Analysis Software

The EUDAQ software contains a `Converter.exe` executable for converting the binary raw files used for data storage to certain other formats. This is achieved by calling a specified extension of the `FileWriter` class for writing the data to a new file. The File Writer calls the `FileReader` class and the needed *Data Converter Plugins* to gain access to the data to be converted.

5.1.1. Data Converter

The binary raw files written by the DAQ part of the EUDAQ software are read by the `FileReader` class using the `Deserializer` class. Blocks of binary data are copied from a file to a data object, to be interpreted by the Data Converter Plugins matching the Producers used when storing the data. They transform the data to multiple structured data objects, each holding a specific kind of information.

This analysis uses the `TLUConverterPlugin` to retrieve the TriggerID and the timestamp, and the `USBpixI4ConverterPlugin` to retrieve the Time-over-Threshold (ToT) and LVL1 Trigger ID (LVL1ID) from the stored data.

All information gathered is combined on a per event basis to reconstruct the individual events. An event is identified by the run number and unique event number and corresponding timestamp. All additional information provided by the sensors is attached to the event in sensor specific *planes*, identified by a unique ID for each sensor. From these reconstructed events the relevant data was extracted by a specifically designed extension of the `FileWriter` class, called `FileWriterTwAnalysis`.

5.1.2. File Writer

The `FileWriterTwAnalysis` class (Listing A.3) is adapted from the existing `FileWriter` classes provided by the EUDAQ package. It creates an `ofstream` object associated with a file matching the current run number, which is passed to the `PrintTsLvl1ToT` method. This method is a public member function added to the `StandardEvent` class to extract the desired information from the reconstructed events and print them to the `ofstream` object passed as a parameter.

The method (Listing A.4) first checks if the current event contains any sensor planes, avoiding errors when called from a *Begin of Run Event* (BORE) or *End of Run Event* (EORE). It then cycles through the 16 frames attached to the first and only plane (`USBPIXI4`), containing the data from the FE. Each frame corresponds to a LVL1ID and may contain one or more hits, determined using the `HitPixels` method of the `StandardPlane` class. For each hit the timestamp of the event, the LVL1ID (equal to the frame number), the ToT and, for consistency checks, the TLU Trigger ID recorded by USBpix and the event number recorded via the TLU Producer are written to the `ofstream` object passed to the function.

5.2. Code Development

The files created by the `FileWriterTwAnalysis` class contain three mandatory columns holding the timestamp, ToT and LVL1ID, and may contain additional columns containing optional information. To read these files, calculate the timewalk based on the method described in Section 6.1, and save the obtained data to file a `C++` class was developed.

The Data Class

The `Data` class (Listings A.5 – A.6) was designed to read, store and operate on variable sets of data with variable numbers of integer values. An object of the `Data` class consists of a two dimensional integer array storing the data, and two additional integer values holding the array's number of rows and columns.

Two special Constructors are provided, one taking a data file and its number of columns as parameters and creating a `Data` object from it, another creating a `Data` object with numbers of rows and columns given as parameters, initialized with an optional third parameter, zero by default.

To construct a `Data` object from a file the `readData` method was designed, also called by the Constructor taking the same parameters. Using the `writeData` method `Data` objects can be stored in files of given name as a space-separated matrix.

Operations on `Data` objects can be performed using specific methods for setting or getting the value of a specified element. Two more methods are provided to respectively return the number of columns or rows of a `Data` object.

The TwAnalysis Tool

To calculate the timewalk using the methods provided by the `Data` class the `TwAnalysis` application (Listing A.7) was developed. It has to be called with at least two command-line arguments. The first one is the number of columns in the files created by the `FileWriterTwAnalysis` plugin, the remaining the file names.

First the tool constructs a `Data` object from each file and does in-place calculations on the objects (Section 6.1), replacing the previous values in the objects but leaving the data files untouched. Then another `Data` object is created containing an element for each combination of the previously calculated delay value and ToT possible. This object is used to determine the number of times each combination occurred and then stored in a new file for plotting using ROOT.

6. Timewalk Measurement

This chapter explains the method used to determine the timewalk's dependence on the charge collected by the FE (Section 3.2.2), and presents the acquired results.

6.1. Measurement Method

The measurement method is described in two parts, since the timing of the individual components' responses is best illustrated at a per hit level, while the computation of the timewalk is based on statistics, thus needing to be presented at a level of at least a few thousand events.

6.1.1. Timing

An overview of the relevant timing information for timewalk calculation is shown in Figure 6.1. The baseline for all timings is the LHC/USBpix 40 MHz clock, to which the TLU's clock was supposed to be synchronized. Synchronization of the USBpix and TLU clocks is necessary to ensure consistency across all components and events, but could not be achieved, due to issues with the TLU v0.2a used.

TLU

When an electron is emitted by the Strontium source it travels through the sensor and FE, and shortly thereafter through the scintillation detector. The TLU's discriminator then registers the signal coming from the scintillation detector and issues a trigger signal, at the same time copying the current timestamp to the read-out register. This process takes the same amount of time for all electron energies. Therefore the timestamp provides an independent marker of the time the hit occurred in the sensor, with a precision of 3.125 ns and biased by a constant offset[1].

[1]Only using a TLU v0.2c, using a TLU v0.2a with a precision of 25 ns

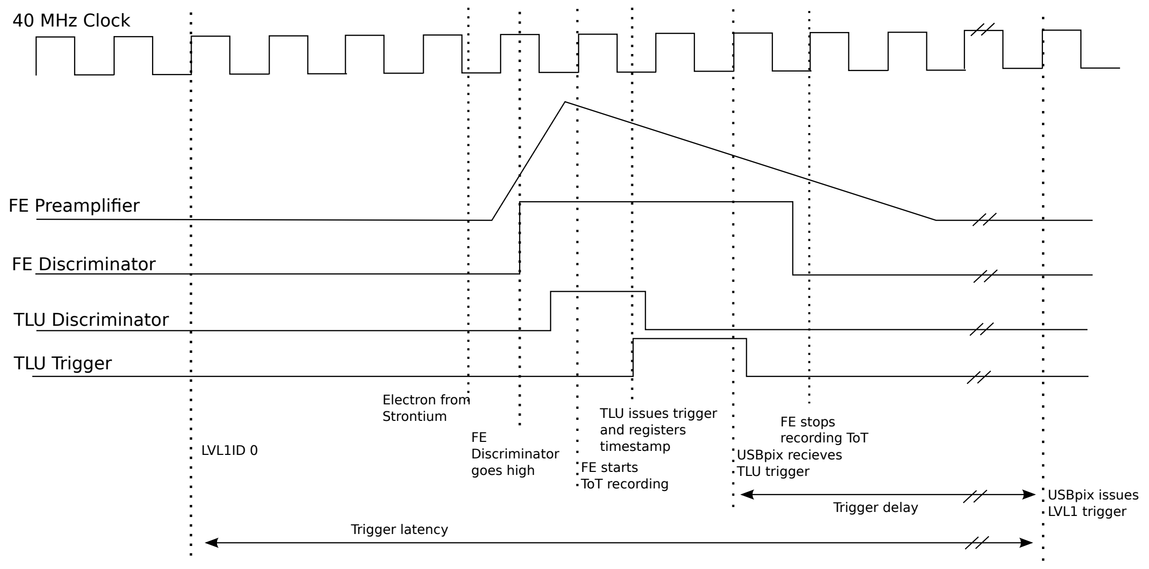


Figure 6.1.: Schematic of the timing of the DAQ components' responses for a single hit

Front-End

When an electron emitted by the Strontium source travels through the sensor, attached to the FE, it creates a certain amount of free charges, depending on its energy. These charges immediately induce a signal at the read-out electrode, connected to one of the FE's analogue pixels. The output of the analogue pixel's preamplifier starts rising and reaches its maximum after a fixed amount of time, independent of the energy deposited. The amplitude of the signal reflects the amount of charges collected at the sensor's read-out electrode and thus depends on the energy deposited, leading to a variation of the rise time of the signal depending on the energy of the electron. The signal is passed to the analogue pixel's discriminator, registering a hit once the signal reaches the discriminator's threshold. This means, the time passed between the hit occurring in the sensor and the FE registering the hit depends on the amount of energy being deposited by the electron. The delay is the *timewalk*.

For every hit the FE determines the time the discriminator was over threshold (ToT), proportional to the amount of charges collected. The ToT is stored in the 4-PDR attached to the hit pixel along with an internal timestamp.

USBpix

When the USBpix board receives the TLU Trigger the corresponding hit was already registered by the FE, and most likely more hits occurred since then. The already delayed trigger is purposely delayed even further by the USBpix board, for a timespan called *trigger delay*. It is adjustable via the `lvl1_delay` option of the USBpix Producer. After the trigger delay the USBpix board sends a *LVL1 trigger* to the FE, which contains a timestamp generated based on the time the TLU trigger was received by subtracting an amount of time called *trigger latency*, accounting for the delay in between the hit occurring and the LVL1 trigger being received by the FE. The FE then checks for hits saved in the 4-PDRs with a timestamp matching the LVL1 timestamp. If a hit was found in the clock cycle matching the LVL1 timestamp or one of the following 15 cycles a LVL1ID corresponding to the clock cycle is attached to the hit, the hit data read out, and passed to the Run Control.

6.1.2. Statistics

Since the time within a clock cycle at which a hit occurs and the energy of the incident electron are randomly distributed a statistical process was used to determine the timewalk. The values needed were the timestamp provided by the TLU, and the LVL1ID and ToT provided by USBpix or rather the FE.

First the position of a hit within the clock cycle it occurred in is determined by evaluating the timestamp t of the hit modulo eight, since the timestamp is incremented eight times per clock cycle[2]. To determine the time passed between the hit and the rising flank of the first LVL1 clock cycle the LVL1ID times eight is added to the timestamp modulo eight, resulting in a value in units of 3.125 ns[3] referred to as *adjusted timestamp* t_{adj} with

$$t_{\text{adj}} = t \bmod 8 + \text{LVL1ID} \cdot 8\,.$$

Evaluating the 16 LVL1IDs times eight timestamps per clock cycle results in 128 possible adjusted timestamps, ranging from 0 to 127. This number of possible adjusted timestamps exists for each of the 13 possible ToTs a hit can have associated, ranging from 1 to 13, resulting in a total of 1664 possibilities to bin a hit. For each of the 128 bins per ToT the number of hits occupying it is determined and plotted versus the number of the bin, representing the adjusted timestamps. The resulting 13 plots are shown and discussed in Subsection 6.2.1.

[2]Increased once per clock cycle by eight when using a TLU v0.2a, allowing to use the same analysis
[3]25 ns when using a TLU v0.2a

From these plots the adjusted timestamps at half maximum were calculated and plotted dependent on the ToT plot they were extracted from. The resulting plot (Figure 6.4) represents the timewalk in the FE-I4 read-out chip.

6.2. Results

6.2.1. Box Plots

The Figures 6.2–6.3 should show box plots, each box approximately one ToT wide. Since a TLU v0.2a had to be used the timestamps, providing the temporal resolution of the measurement, were spaced 25 ns apart. This led to a coarse measurement. Instead of a steeply rising flank, a plateau, and a steeply falling flank forming a box only a sharp peak was observed in each plot, making it impossible to fit a box function to the data. A Gaussian function

$$\mathrm{Occurrence}(t_{\mathrm{adj}}) = \mathrm{Constant} \cdot \exp\left(-\frac{1}{2}\left(\frac{t_{\mathrm{adj}} - \mathrm{Mean}}{\mathrm{Sigma}}\right)^2\right)$$

was determined as the best possible description of the data and fitted using the χ^2-method. The values of the fit parameters are displayed in each plot.

6.2.2. Timewalk

The values at half maximum of the box plots (Figures 6.2–6.3) were calculated and plotted dependent on the ToT represented by the box plot, resulting in a final plot (Figure 6.4) representing the timewalk in the FE-I4. The value of the adjusted timestamp at half maximum (HM) of the rising flank of each plot was calculated via

$$t_{\mathrm{HM}} = \mathrm{Mean} - \sqrt{\ln 4} \cdot \mathrm{Sigma}\,,$$

the error σ_t via Gaussian error propagation

$$\sigma_t = \sqrt{\left(\sigma_{\mathrm{Mean}} \frac{\partial t_{\mathrm{HM}}}{\partial \mathrm{Mean}}\right)^2 + \left(\sigma_{\mathrm{Sigma}} \frac{\partial t_{\mathrm{HM}}}{\partial \mathrm{Sigma}}\right)^2} = \sqrt{\sigma_{\mathrm{Mean}}^2 + \ln 4 \cdot \sigma_{\mathrm{Sigma}}^2}\,.$$

The plot shows the expected inverse correlation between the deposited charge and the delay before the hit is registered by the FE. The large delays are caused by the LVL1ID of the hits, irrelevant for the following analysis. The important difference Δt between the largest and smallest delay is

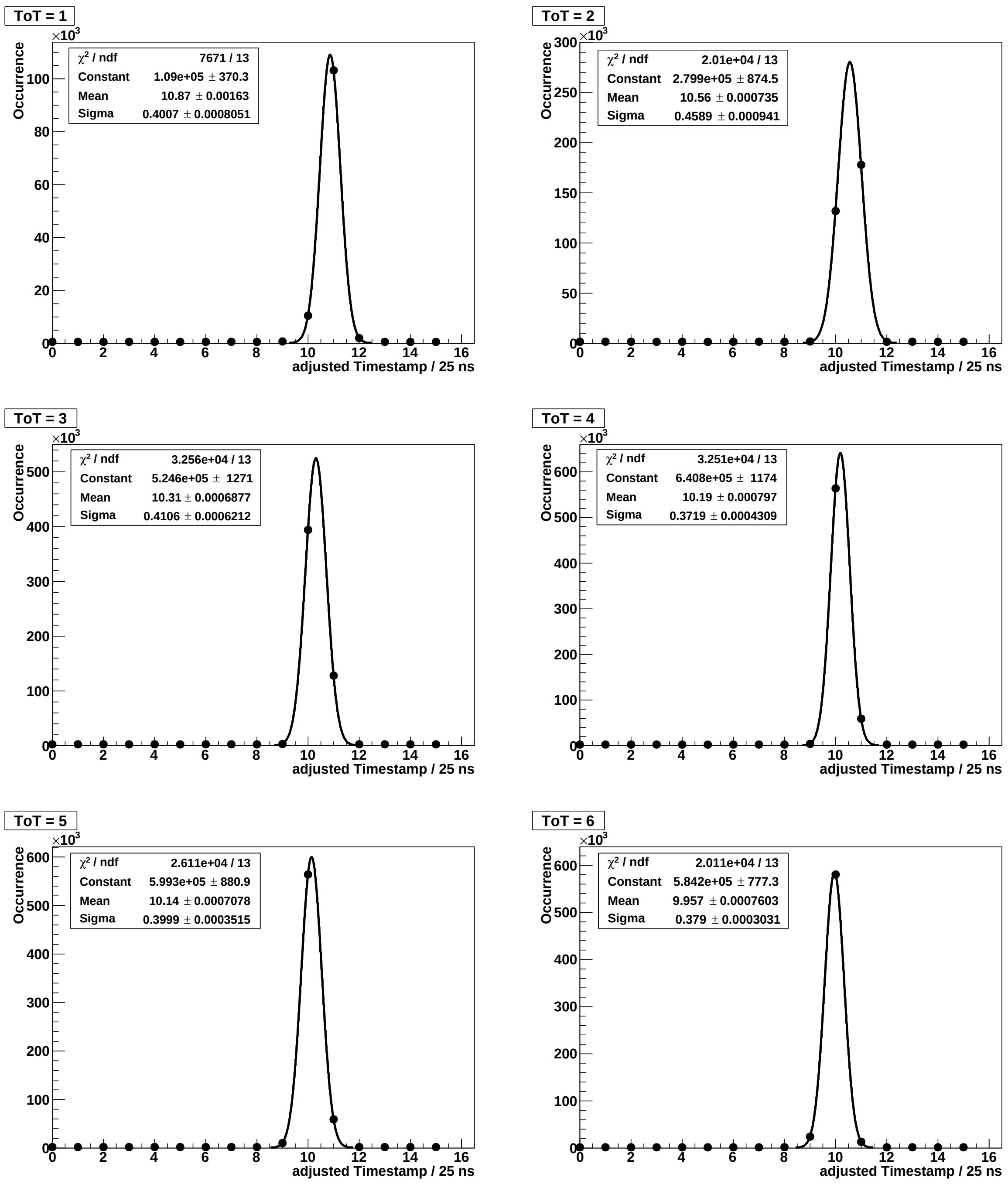


Figure 6.2.: The plots show the number of occurrences of each adjusted timestamp for the ToT values 1 to 6. A Gaussian function was fitted to the values, visibly shifting towards lower adjusted timestamps for higher ToTs

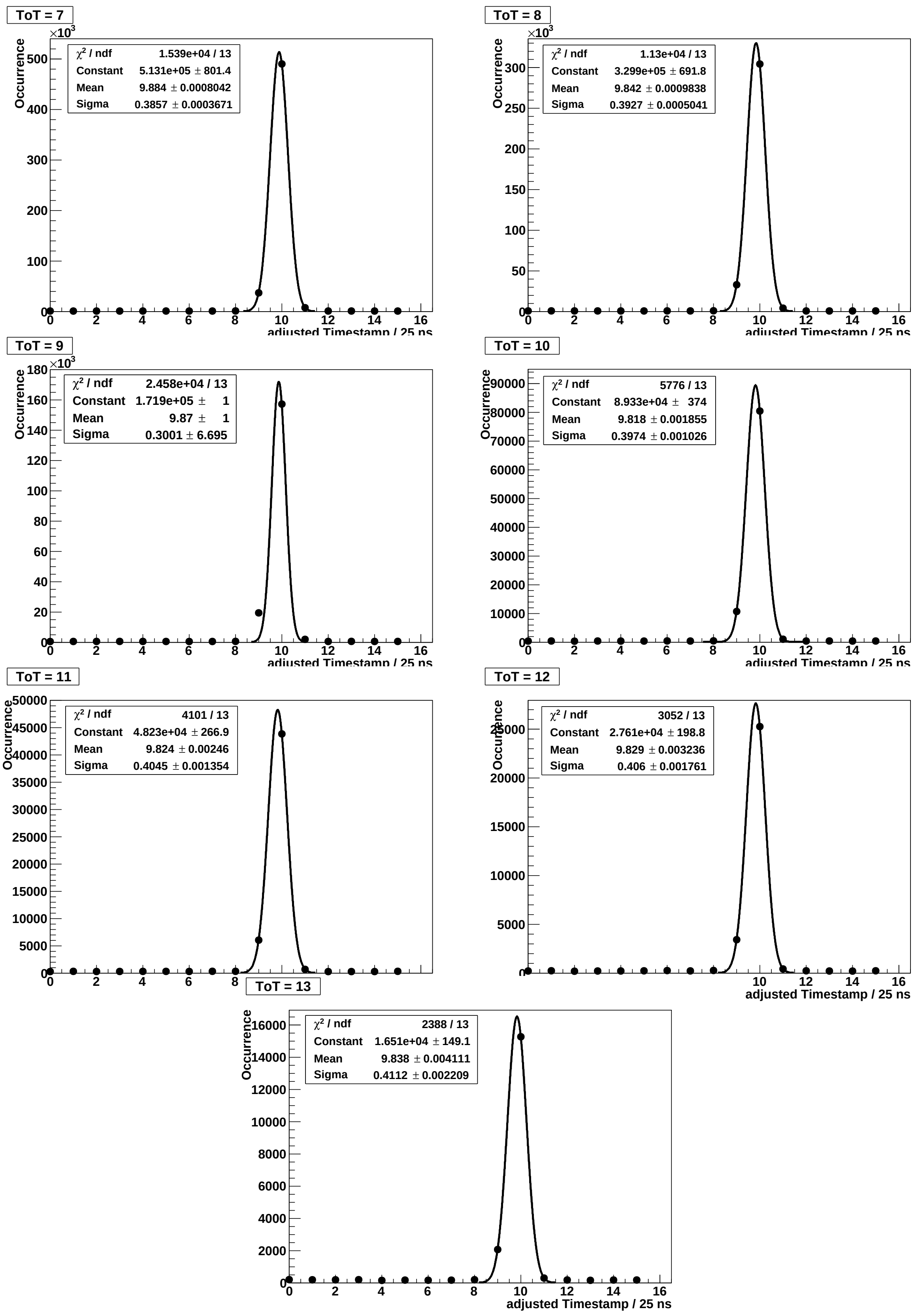


Figure 6.3.: The plots show the number of occurrences of each adjusted timestamp for the ToT values 7 to 13. A Gaussian function was fitted to the values

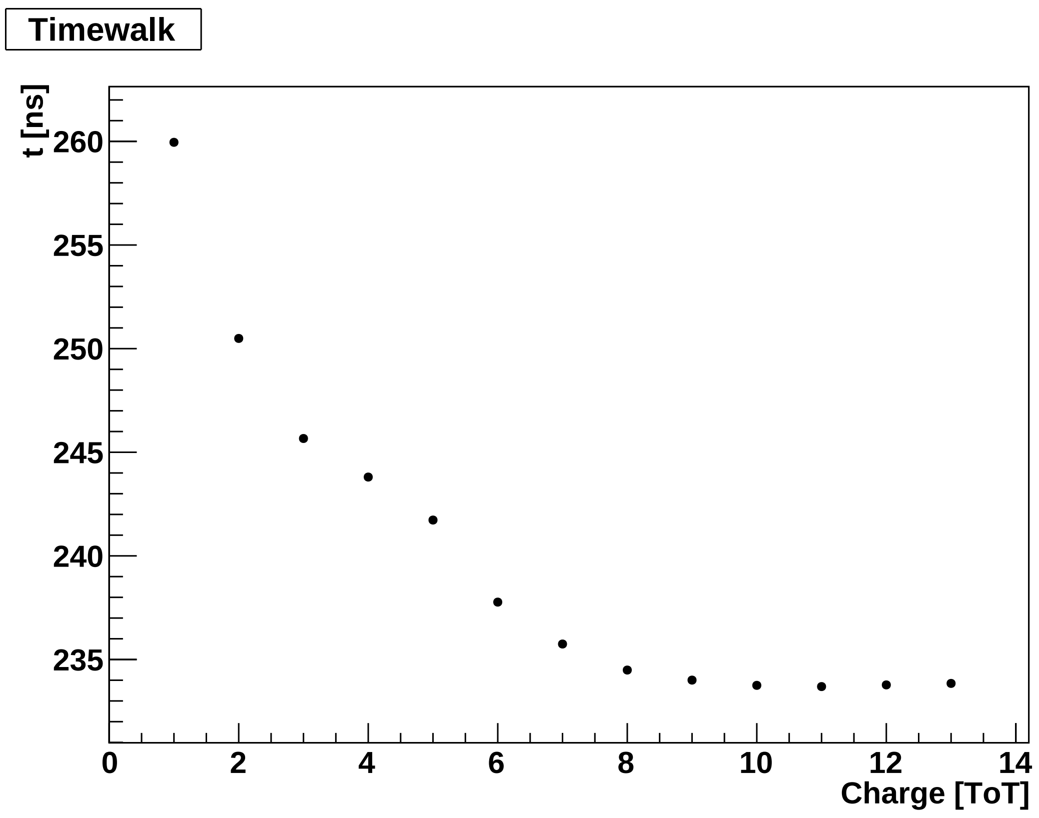


Figure 6.4.: The timewalk in the FE-I4 read-out chip, calculated using timestamps with a 25 ns resolution, provided by a TLU v0.2a

$$\begin{aligned}\Delta t &= t(\mathrm{ToT}=1) - t(\mathrm{ToT}=11)\\ &= (259.96 \pm 0.05)\,\mathrm{ns} - (233.70 \pm 0.07)\,\mathrm{ns}\\ &= (26.26 \pm 0.09)\,\mathrm{ns}\,.\end{aligned}$$

Clearly visible is the asymptotic behaviour seen for ToT > 9, resulting in a difference $\Delta t_{\mathrm{asym}} = t(\mathrm{ToT} = 9) - t(\mathrm{ToT} = 13)$ of only (0.16 ± 0.13) ns. The errors σ_t calculated using only the errors of the fit parameters are very small ($\sigma_{t,\mathrm{rel}} = \sigma_t/t < 0.05\%$) and are therefore not visible in the plot. They cannot represent the total errors of the delays, since the large reduced χ^2 of the fits and the alternative function used for fitting have to be accounted for, leading to a much larger error, probably in the order of 10 ns.

7. Conclusion and Outlook

7.1. The Set-up

During the building and testing phases as well as during physics operation the set-up was found to be prone to errors of various types. On the hardware side the trigger mechanism of the TLU v0.2c was found to be defunct, requiring a replacement unit. This was a prototype TLU v0.2a, missing the *precise timestamp* feature implemented in the final version. Additionally it showed errors regarding operation using different firmware versions and clock IO. The scintillation detector required some fine tuning of the control voltage before functioning flawlessly, while the USBpix board, FE and sensor performed well from the start.

Setting up the software proved to be difficult, due to issues during both compilation and operation. Whenever the attempt was made to configure STControl more than once via the Run Control an error occurred, requiring both applications to be restarted. The modifications implemented in the TLU Producer could not be properly tested, since the TLU did not function using the default firmware. No documentation was provided for the custom firmware distributed with the EUDAQ software, making it impossible to adapt the modifications to this version.

Once the system was fully set up and well configured the data acquisition worked flawlessly, recording several million events in a matter of hours.

7.2. Timewalk Measurement

Due to the issues described above the measurements could not be performed as intended. Inconsistencies were likely introduced in the recorded data, caused by the TLU and USBpix clocks not being synchronized. Using the TLU's internal clock rather than the externally provided USBpix clock introduced a random element to the timestamps, supposed to provide a static reference point. Due to the low temporal resolution of the TLU v0.2a's timestamp of 25 ns the impact of this effect was likely minor. The low resolution itself was a far greater problem, resulting in only 12.5% of the data points aimed for being actually recorded. As a consequence the supposed to be box shaped plots were found to show sharp peaks, consisting of only one or two data points. As expected fitting a box function to these plots was impossible, and the Gaussian functions used instead resulted in reduced χ^2s in the order of 10^2 to 10^3. The small errors of the fit parameters, resulting in small errors of the final values, are therefore misleading and expected to be much larger than the error propagation suggests.

The timewalk seen in Figure 6.4 shows the expected behaviour of decreasing delays for increasing charges. The maximal delay was determined to be (26.26 ± 0.09) ns, observed between the ToTs 1 and 11. A near constant delay was found for ToTs greater than 9. The errors of the data points are four orders of magnitude smaller than the values, suggesting a high precision of the measurement. This contradicts the actual measurement and is due to the high χ^2s not accounted for in the final data.

7.3. Outlook

Even though quite probable results were obtained the timewalk measurement needs to be repeated. A working TLU v0.2c has to be acquired, using which the modifications to the TLU Producer have to tested. After successful synchronization of the USBpix and TLU clocks data sets containing the precise timestamp need to be recorded. The adapted File Writer and developed analysis tools can then be used to evaluate the data and calculate the timewalk. Optionally the precision of the result can be improved further by extending the File Writer to extract the coordinates of the hit pixels as well, so an extended version of the analysis tools can use these coordinates in combination with calibration data specific to the FE for an accurate ToT to charge conversion.

The developed File Writer and analysis software handled the data processing well, but limitations were suspected, especially when evaluating larger amounts of data. To increase usability and performance of the `Data` class several improvements need to be implemented. The arrays currently used should be replaced by `vector` objects for easier creation and access. The memory footprint could be reduced by implementing structs designed for specific data sets. Versatility could be greatly improved by using templates, while user-friendliness would benefit from moving the software to Root, requiring less user input when going from data analysis to plotting.

A. Source Code

A.1. EUDAQ

Listing A.1: Public member function added to the TLUController class, called by the TLU Producer

```
void TLUController::SetClockSourceSelect(int bit_val){
  if (m_addr) {
    WriteRegister(m_addr->TLU_CLOCK_SOURCE_SELECT_ADDRESS,bit_val);
    std::cout << ``Set CLOCK_SOURCE_SELECT_ADDRESS to '' << bit_val /
       << std::sendl;
  }
}
```

Listing A.2: Configuration of the DAQ system, applied via the EUDAQ RunControl

```
[RunControl]
RunSizeLimit = 100000000
NoTrigWarnTime = 10

[DataCollector]

[LogCollector]
SaveLevel = EXTRA
PrintLevel = INFO

[Producer.TLU]
AndMask = 1
OrMask = 0
VetoMask = 0
DutMask = 2
TriggerInterval = 0
TrigRollover = 0
EnableDUTVeto = 3
```

```

[Producer.USBpixI4]
SkipConfiguration = no
config_file = F:\Users\bklaas\66-W13-IBL1_3000_ToT.cfg.root
fpga_firmware = C:\USBPix\releaseI4-4.3-beta\config\usbpixi4.bit
SRAM_READOUT_AT=30
# lvl1_delay=26
```

A.2. Analysis Tools

Listing A.3: FileWriterTwAnalyses.cc,
written to extract Timestamps, LVL1_ID and ToT from EUDAQ binary raw files

```
#include "eudaq/FileWriter.hh"
#include "eudaq/FileNamer.hh"
#include "eudaq/PluginManager.hh"
#include <iostream>
#include <fstream>

namespace eudaq {

  class FileWriterTwAnalysis : public FileWriter {
    public:
      FileWriterTwAnalysis(const std::string &);
      virtual void StartRun(unsigned);
      virtual void WriteEvent(const DetectorEvent &);
      virtual unsigned long long FileBytes() const;
      virtual ~FileWriterTwAnalysis();
    private:
      std::ofstream m_file;
  };

  namespace {
    static RegisterFileWriter<FileWriterTwAnalysis> reg("TwAnalysis")/
       ;
  }

  FileWriterTwAnalysis::FileWriterTwAnalysis(const std::string & /
     param)
      : m_file("dummy.txt") {
```

```
    std::cout << "EUDAQ_DEBUG: This is FileWriterTwAnalysis::/
       FileWriterTwAnalysis("<< param <<")"<< std::endl;
  }

  void FileWriterTwAnalysis::StartRun(unsigned runnumber) {
    std::cout << "EUDAQ_DEBUG: FileWriterTwAnalysis::StartRun("<< /
       runnumber << ")" << std::endl;
    // close an open file
    if (m_file.is_open()) {
      m_file.close();
    }

    // open a new file
    std::string fname(FileNamer(m_filepattern).Set('X', ".txt").Set('/
       R', runnumber));
    m_file.open(fname.c_str());
    if (!m_file.is_open()) EUDAQ_THROW("Error opening file: " + fname/
       );
  }

  void FileWriterTwAnalysis::WriteEvent(const DetectorEvent & devent)/
      {
      if(devent.GetEventNumber() % 10000 == 0)
          std::cout << "EUDAQ_DEBUG: FileWriterTwAnalysis::WriteEvent/
              () processing event "
              <<  devent.GetRunNumber() <<"." << devent./
                  GetEventNumber() << std::endl;

      if (devent.IsBORE()) {
          PluginManager::Initialize(devent);
          return;
      }

      //disentangle the detector event
      StandardEvent sevent(PluginManager::ConvertToStandard(devent));
      sevent.PrintTsLvl1Tot(m_file);
  }

  FileWriterTwAnalysis::~FileWriterTwAnalysis() {
//      if (m_file.is_open()) {
//         m_file.close();
//      }
  }
```

```
    // Just to avoid errors
    unsigned long long FileWriterTwAnalysis::FileBytes() const { /
       return 1337; }

}
```

Listing A.4: Public member function added to the StandardEvent class, called by FileWriterTwAnalysis

```
void StandardEvent::PrintTsLvl1Tot(std::ofstream & os) const {
      // If it is an event plane...
    if(m_planes.size() > 0)
      // cycle through all 16 LVL1_IDs...
      for(unsigned lvl1 = 0; lvl1 < m_planes[0].NumFrames(); lvl1++)
      // and for each LVL1 cycle through all recorded hits.
        for(unsigned HitNo = 0; HitNo < m_planes[0].HitPixels(lvl1); /
           HitNo++) {
          os << Event::GetTimestamp() << " ";
          os << lvl1 << " ";
      // Print ToT
          os << m_planes[0].GetPixel(HitNo, lvl1) << "    ";
      // Optionally print TLU_ID and EventNumber for consistency check
          os << m_planes[0].TLUEvent() << " ";
          os << Event::GetEventNumber() << std::endl;
        }
}
```

Listing A.5: Header file of the Data class, designed to easily read, edit, and store the acquired data

```
#ifndef Data_H
#define Data_H

class Data{
  private:
    int m_numberOfColumns;
    int m_numberOfRows;
    unsigned long long** m_data;    // FIXME: waste of memory to save/
        lvl1 and tot as typ
  public:
    Data(){};
    Data(char* dataFile, int numberOfColumns);
    Data(int numberOfColumns, int numberOfRows, unsigned long long /
       initial = 0);

    void readData(char* input_file, int numberOfColumns);
    void writeData(char* newFile);

    int getColumns(){return m_numberOfColumns;};
    int getRows(){return m_numberOfRows;};

    void setColRow(int column, int row, unsigned long long value);
    unsigned long long getColRow(int column, int row);
};
#endif      // Data_H
```

Listing A.6: Source of the Data class, designed to easily read, edit, and store the acquired data

```
#include <iostream>
#include <fstream>

#include "Data.h"

using namespace std;

Data::Data(char* dataFile, int numberOfColumns){
  readData(dataFile, numberOfColumns);
}

Data::Data(int numberOfColumns, int numberOfRows, unsigned long long /
    initial){
  m_numberOfColumns = numberOfColumns;
  m_numberOfRows = numberOfRows;

  // needs type const as argument
  const int x = m_numberOfColumns;
  const int y = m_numberOfRows;
  m_data = new unsigned long long*[x];
  for(int i = 0; i < m_numberOfColumns; i++)
    m_data[i] = new unsigned long long[y];

  for(int i = 0; i < m_numberOfColumns; i++)
    for(int j = 0; j < m_numberOfRows; j++)
      m_data[i][j] = initial;
}

void Data::readData(char* input_file, int numberOfColumns){
  double dummyElement;              // Hilfsvariable zum Datenzaehlen
  int elementCounter=0;             // Nimmt Anzahl der Datenpunkte /
     auf
  ifstream dataFile(input_file);    // Gegebenen Datensatz oeffnen
  cout << "\nReading File '" << input_file << "'..." << endl;

  // Anzahl der gegeben Datenpunkte ermitteln
  while(true){
    dataFile >> dummyElement;       // Durchlaeuft zeilenweise die /
       Inputdatei
    if(dataFile.eof())
      break;                        // Am Ende der Datei die Schleife/
          verlassen ,
    elementCounter++;               // sonst Datenpunktzaehler /
       erhoehen
  }

  m_numberOfRows = elementCounter/numberOfColumns;
  m_numberOfColumns = numberOfColumns;
  cout << "Number of elements: " << elementCounter << endl;
  cout << "Number of rows: " << m_numberOfRows << endl;
  cout << "Reading data... ";

  // needs type const as argument
  const int x = m_numberOfColumns;
  const int y = m_numberOfRows;
  m_data = new unsigned long long*[x];
  for(int i = 0; i < m_numberOfColumns; i++)
    m_data[i] = new unsigned long long[y];

  dataFile.clear();
  dataFile.seekg(0);                // Dateizeiger zuruecksetzen

  // Daten aus der Datei in die Arrays schreiben
  for(int i = 0; i < m_numberOfRows; i++)
    for(int j = 0; j < m_numberOfColumns; j++)
      dataFile >> m_data[j][i];

  dataFile.close();
  cout << "done!\n" << endl;
}
```

```
void Data::writeData(char* newFile){
  ofstream dataFile(newFile);
  cout << "Writing to " << newFile << "... ";

  for(int i = 0; i < m_numberOfRows; i++)
    for(int j = 0; j < m_numberOfColumns; j++){
      dataFile << m_data[j][i];
      if(j == m_numberOfColumns-1)
        dataFile << endl;
      else
        dataFile << " ";
    }

  cout << "done!\n" << endl;
}

void Data::setColRow(int column, int row, unsigned long long value){
  m_data[column][row] = value;
}

unsigned long long Data::getColRow(int column, int row){
  return m_data[column][row];
}
```

Listing A.7: Source of the TwAnalysis application, designed to calculate the timewalk using the Data class

```
#include <iostream>
#include <cstdlib>

#include "Data.h"

#define INC_PER_CLOCK 8
#define TIMESTAMP 0
#define LVL1 1
#define TOT 2

using namespace std;

int main(int argc, char* argv[]){
  // parse number of file given, do computing for all files
  // argv[1] is number of columns
```

```
const int numberOfFiles = argc - 2;

// get timestamp, lvl1 and tot from files
Data* tsLvl1Tot = new Data[numberOfFiles];
for(int i = 0; i < numberOfFiles; i++)
    tsLvl1Tot[i].readData(argv[i+2], atoi(argv[1]));

cout << "Calculating adjusted timestamps... " << flush;
// all timestamps modulo 8
for(int i = 0; i < numberOfFiles; i++)
  for(int j = 0; j < tsLvl1Tot[i].getRows(); j++)
    tsLvl1Tot[i].setColRow(TIMESTAMP, j, tsLvl1Tot[i].getColRow(/
        TIMESTAMP, j) % INC_PER_CLOCK);

// correct mod. timestamps by lvl1-offset
for(int i = 0; i < numberOfFiles; i++)
  for(int j = 0; j < tsLvl1Tot[i].getRows(); j++)
    tsLvl1Tot[i].setColRow(TIMESTAMP, j, tsLvl1Tot[i].getColRow(/
        TIMESTAMP, j)
         + tsLvl1Tot[i].getColRow(LVL1, j) * INC_PER_CLOCK);
cout << "done!\n" << endl;

cout << "Counting adjusted timestamps... " << flush;
// count the occurences of an adjusted timestamp in a ToT bin
  // create an array for each tot with an entry for each possible /
      timewalk (128)
  // which holds the number of occurences of that ad. timestamp
Data totAdTsOc(15, 128);    // automatically initilized to 0

// count the number of occurences
// FIXME: incrementing should be more elegant
for(int i = 0; i < numberOfFiles; i++)
  for(int j = 0; j < tsLvl1Tot[i].getRows(); j++)
    totAdTsOc.setColRow( tsLvl1Tot[i].getColRow(TOT-1, j), /
        tsLvl1Tot[i].getColRow(TIMESTAMP, j),
         totAdTsOc.getColRow( tsLvl1Tot[i].getColRow(TOT-1, j), /
           tsLvl1Tot[i].getColRow(TIMESTAMP, j))+1);
cout << "done!\n" << endl;

// save ad. timestamp (second index in totAdTsOc) and
// counter (entry in totAdTsOc) to file
char* outFileName = "binnedTimestamps.dat";
totAdTsOc.writeData(outFileName);
```

```
    return 0;
}
```

Bibliography

[1] D. Griffiths, *Introduction to Elementary Particles*, Wiley-VCH, second edition (2004)

[2] J. Rieger, *Comparison of Thin n- and p-type Bulk Silicon Pixel Sensors*, Master's thesis, Georg-August-Universität Göttingen, Germany (2012), iI.Physik-UniGö-MSc-2012/08

[3] J. Beringer et al. (Particle Data Group), *Review of Particle Physics*, Phys. Rev. D **86**, 010001 (2012), URL `http://link.aps.org/doi/10.1103/PhysRevD.86.010001`

[4] G. Aad et al. (ATLAS), *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Physics Letters B **716(1)**, 1 (2012), URL `http://www.sciencedirect.com/science/article/pii/S037026931200857X`

[5] S. Brüning et al., *LHC Design Report*, CERN, Geneva (2004)

[6] *Measurements of Higgs boson production and couplings in diboson final states with the ATLAS detector at the LHC*, Technical Report CERN-PH-EP-2013-103, CERN, Geneva (2013)

[7] ATLAS Collaboration (ATLAS), *ATLAS Letter of Intent for a General-Purpose pp Experiment at the Large Hadron Collider at CERN*, Technical Report CERN/LHCC/92-4, LHCC/I 2, CERN (1992)

[8] ATLAS Collaboration (ATLAS), *ATLAS DETECTOR AND PHYSICS PERFORMANCE Technical Design Report Volume I*, Technical Report ATLAS TDR 14, CERN/LHCC 99-14, CERN (1999)

[9] J. Große-Knetter (ATLAS), *Overview of the ATLAS Insertable B-Layer (IBL) Project*, Technical Report ATL-INDET-PROC-2011-022 (2011)

[10] IBL Community (ATLAS IBL), *ATLAS Insertable B-Layer Technical Design Report*, Technical Report ATLAS TDR 19, CERN/LHCC 2010-013, CERN (2010)

[11] IBL collaboration (ATLAS IBL), *Prototype ATLAS IBL modules using the FE-I4A front-end readout chip*, Journal of Instrumentation **7(11)**, P11010 (2012), URL `http://stacks.iop.org/1748-0221/7/i=11/a=P11010`

[12] M. Barbero et al. (ATLAS IBL), *The FE-I4 Pixel Readout Chip and the IBL Module*, Technical Report PoS(Vertex 2011)038, CERN (2011)

[13] W. R. Leo, *Techniques for Nuclear and Particle Physics Experiments*, Springer Verlag, Berlin Heidelberg, second edition (1994)

[14] Z. He, *Review of the Shockley–Ramo theorem and its application in semiconductor gamma-ray detectors*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **463(1–2)**, 250 (2001), URL `http://www.sciencedirect.com/science/article/pii/S0168900201002236`

[15] M. Backhaus et al., *Development of a versatile and modular test system for ATLAS hybrid pixel detectors*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **650(1)**, 37 (2011), URL `http://www.sciencedirect.com/science/article/pii/S0168900210028676`

[16] J. Schneider, H. Krüger, *S3 Multi IO System – S3 Multi IO USB Card Version V1.03*, Technical report, Universität Bonn (2010)

[17] E. Corrin, *EUDAQ Software User Manual*, EUDET-Memo-2010-01 (2010)

[18] D. Cussans, *Description of the JRA1 Trigger Logic Unit (TLU), v0.2c*, EUDET-Memo-2009-4 (2009)

[19] D. Cussans, *Description of the JRA1 Trigger Logic Unit (TLU), v0.2*, EUDET-Memo-2008-50 (2008)

Acknowledgements

Zum Schluss bedanke ich mich allen, die zur Entstehung und Vollendung dieser Arbeit beigetragen haben.

Allen voran danke ich Jörn Große-Knetter und Jens Weingarten, die beide viele Stunden damit verbracht haben meine Fragen zu beantworten und viele Probleme zu lösen. Jens danke ich zudem für das Korrekturlesen dieser Arbeit, und Jörn für die Betreuung und das Erstellen des Erstgutachtens.

Weitere große Unterstützung habe ich von Julia und Johannes erhalten, denen ich dafür ebenfalls herzlich danke. Egal ob es um Fragen, den Einbau der Quelle oder das Löten von Platinen ging, ihr wart immer zur Stelle.

Ein weiterer Dank geht an Herrn Quadt, für die Möglichkeit diese Arbeit im II. Institut zu schreiben und einen Beitrag zum Atlas zu leisten. Ebenso bedanke ich mich für das Erstellen des Zweitgutachtens.

Zuletzt möchte ich noch Lars, Matthias, und allen weiteren Personen im II. Institut für die freundliche Aufnahme und die angenehme Umgebung danken.

Erklärung nach §13(8) der Prüfungsordnung für den Bachelor-Studiengang Physik und den Master-Studiengang Physik an der Universität Göttingen:

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.
Darüberhinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, im Rahmen einer nichtbestandenen Prüfung an dieser oder einer anderen Hochschule eingereicht wurde.

Göttingen, den March 18, 2016

(Björn Klaas)