



**UNIVERSITÉ
DE GENÈVE**

FACULTY OF SCIENCE

Physics Section

FASER Trigger Logic Board

**Hardware Commissioning and Data Acquisition Software
Development**



Eliott Philippe Johnson

Supervisor: Prof. A. Sfyrla

Tutor: Dr. C. Antel

Department of Physics
University of Geneva

This dissertation is submitted for the degree of
Master of Physics

Acknowledgements

I would like to acknowledge those who have made this study possible:

Prof. A. Sfyrla who has suggested this study, as well as Dr. C. Antel for their help, their encouragement and the many discussions that has made it possible to clarify a great number of ideas and concepts.

Dr. B. Petersen from CERN for his advice and suggestions during the TLB commissioning.

Dr. K. Schmieden for his help and indications in understanding the code.

Dr. S. Meehan for his explanations and help working in the scintillator lab.

My father and grandfather who battled through the typos and shared their expertise.

Abstract

The ForwArd Search ExpeRiment (FASER) searches for light and weakly interacting particles produced in the decays of light mesons in the far forward region of proton-proton collisions at the Large Hadron Collider. A trigger logic board (TLB) designed and built by the University of Geneva has been commissioned and proved to be suitable for use in the first run of the FASER experiment. Software used for configuring and reading out the TLB has been written and successfully integrated in the FASER digital acquisition system framework.

Table of contents

Nomenclature	ix
1 The Future of High Energy Physics	1
1.1 Introduction	1
1.2 Beyond the Standard Model	2
1.3 How does FASER fit in the context of ATLAS and CMS	3
2 The Faser Experiment	5
2.1 Experiment Location and Schedule	5
2.1.1 Civil Engineering Work	5
2.2 Magnet Design	7
2.3 Expected Signature	7
2.4 Detector Break Down	8
2.4.1 Trackers	9
2.4.2 Calorimeter	10
2.4.3 Scintillators	10
3 Data Acquisition Architecture	13
3.1 DAQ Dataflow	15
3.2 Run Control	16
3.2.1 Displaying Data using Grafana	17
4 Trigger and Data Acquisition	19
4.1 Unige USB3 GPIO	19
4.2 TLB Board	21
4.3 LUT	24
4.4 TLB Commissioning	25
4.4.1 Sampling Phase	25
4.4.2 Input Delay	27
4.4.3 Coincidence Logic Test	28
4.4.4 Prescales	29

4.4.5	Random Trigger	30
4.4.6	Software Trigger	32
4.4.7	Deadtime from the Rate Limiter	32
4.4.8	Simple Deadtime	32
4.4.9	Deadtime from the BCR veto	33
4.4.10	Output Delay	34
5	TLB Software	37
5.1	TRB/TLB communication program structure	37
5.2	Configuration json	38
5.3	Software contribution	38
5.3.1	GPIONDrivers	39
5.3.2	Setting values in the TLB's configuration	42
5.3.3	Reading out Data	44
5.3.4	TLBDecode	44
5.3.5	TriggerReceiver Module.	44
6	Conclusion and Outlook	47
	References	49
	Appendix A Instruments panels	51
	Appendix B TLB Data Configuration Format	53
B.1	TLB Configuration Variable Layout	53

Nomenclature

Roman Symbols

pp Proton-proton

p_T Transverse Momentum

Acronyms / Abbreviations

ALP Axion like particles

ATLAS A Toroidal LHC ApparatuS

BCID Bunch Counter Identifier

BCR Bunch Counter Reset

BST Beam Synchronous Timing system

CERN Conseil Européen pour la Recherche Nucléaire

CMS Compact Muon Solenoid

DAQ Data Acquisition

DCS Detector Control System

EB Event Builder

ECR Event Counter Reset

eV Electronvolt

FASER ForwArd Search ExpeRiment

HL High Luminosity

IP Interaction Point

L1A Level 1 Accept

LHC	Large Hadron Collider
LOS	Line Of Sight
LUT	Look Up Table
PBC	Physics Beyond Colliders
RO	Read-Out
ROR	Read-Out Receiver
SC	Slow Control
SCT	SemiConductor Tracker
SM	Standard Model
TAN	Target Absorber Neutral
TAS	Target Absorbers
TLB	Trigger Logic Board
TRB	Tracker Readout Board

Chapter 1

The Future of High Energy Physics

1.1 Introduction

The Large Hadron Collider (LHC) is one of the wonders of the modern world - the largest and highest-energy particle collider designed to explore the laws governing the forces among the fundamental particles and phenomena that can help us understand the structure of space and time and the correlation between quantum mechanics and general relativity. It has achieved to study in great detail the Standard Model (SM) as well as to discover the Higgs boson, the particle that gives mass to all other particles. Nevertheless, many fundamental questions remain open in physics such as the apparent violation of symmetry between matter and antimatter and the Hierarchy problem between the four fundamental forces, to name a few. After the discovery of the Higgs boson, no physics beyond the SM has been found at the LHC, which motivates the exploration of unprobed regions such as the search for light and weakly coupled long-lived particles.

ForwArD Search ExpeRiment (FASER) is a small and inexpensive detector installed in the far forward region of proton-proton (pp) collisions that will collect data during Run 3 and probe new regions of the parameter space for new particles with masses in the 10 MeV to GeV range, see Fig. 1.1. FASER requires no beam modifications and only requests luminosity information. Further into the

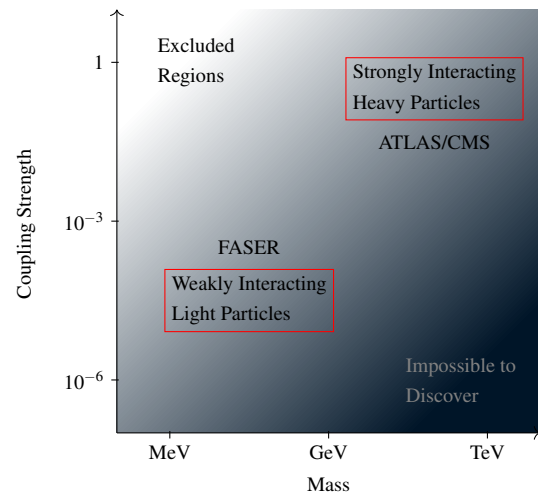


Fig. 1.1 ATLAS and CMS focus on large transverse momentum (p_T) signatures that emerge in the roughly isotropic decay of such particles. There is a complementary class of viable new particles that are much lighter, with mass in the MeV to GeV range, much more weakly coupled to the SM.

future, a larger FASER 2 will extend the sensitivity to even larger masses but will require significant civil engineering.

1.2 Beyond the Standard Model

The FASER collaboration hopes to discover new particles through this experiment. An example for such a new particles is the dark photon, Fig.1.2a. Dark photons are hypothetical hidden sector (a collection of yet unobserved quantum fields) particles. Interaction between hidden sector and SM are weak, indirect and would be mediated via gravity or new particles. Dark photons are proposed as a force carrier, similar to the photon in the SM with a new abelian U(1) gauge symmetry. This new spin-1 gauge boson could couple very weakly to electrically charged particles through kinetic mixing with the normal photon [1]. They could be produced in meson decays, see Fig. 1.4, where we see that the branching ratio (B) of the neutral pion to A' and a photon is the same as the branching ratio for $\pi^0 \rightarrow \gamma\gamma$ but modified by the mass of the A' :

$$B(\pi^0 \rightarrow A' \gamma) = 2\varepsilon^2 \left(1 - \frac{m_{A'}^2}{m_{\pi^0}^2}\right)^3 B(\pi^0 \rightarrow \gamma\gamma)$$

Another type of new physics could be the discovery of axion-like-particles (ALPs). ALPs are hypothetical particles supposed stable, neutral and of very low mass (1 meV- μ eV). They are a solution by Peccei-Quinn to the problem of CP violation in QCD. There exists a possibility of using the LHC as a beam-dump experiment. see Fig. 1.5. Very high energy photons produced in the IP could interact with material in the Neutral Beam Absorbers (TAN, see Fig. 2.1) and produce ALPs (TeV energy) via Primakoff effect, explained on Fig. 1.2b, that could decay in FASER.

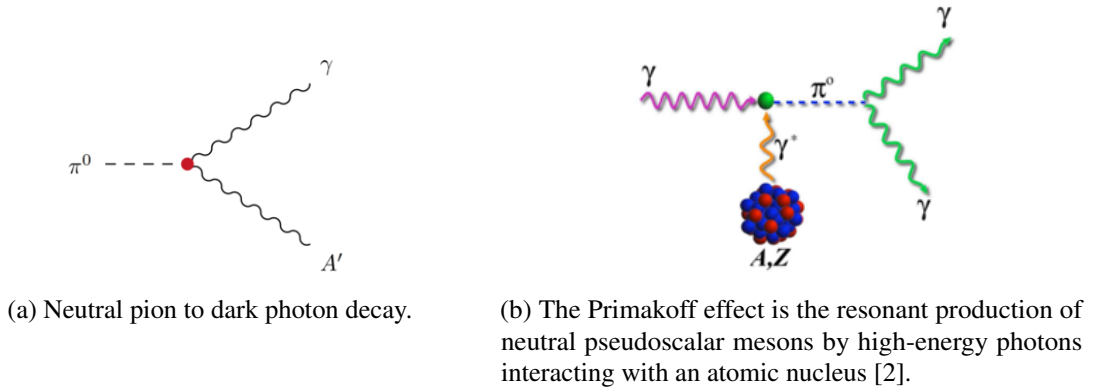


Fig. 1.2 Beyond the standard model particle production.

1.3 How does FASER fit in the context of ATLAS and CMS

The four main LHC experiments (ATLAS, CMS, ALICE, LHCb) that search for TeV-scale particles with high transverse momentum (p_T) may be misguided in the search for new physics. An area that isn't covered by these experiments, is the search for light particles with masses in the MeV to GeV range with low p_T . These new particles, that might be extremely weakly-coupled, may travel hundreds of meters without interacting with any material before decaying into known SM particles. During their travel, they would not be bent by magnets. Instead, they will continue along a straight line and their decay products have the potential to be spotted by the FASER experiment. These hypothetical new particles would be long lived (LLP) and very collimated with the beam (of the order of the milliradian) allowing a small detector far from the IP to be built. For example, new particles that are produced in pion decays are typically produced within angles

$$\theta \sim \frac{\Lambda_{QCD}}{E} \sim \frac{m_{pion}}{E}$$

of the beam collision axis, where E is the energy of the particle and Λ_{QCD} is the scale of the pion mass. This implies that around 500 meters after the interaction point the downstream particle spread is 10 to 100 cm [3].

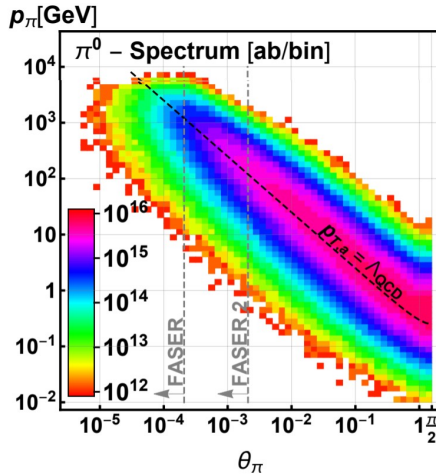


Fig. 1.3 Differential π^0 production rate in the (θ, p) plane, where θ and p are the meson's angle with respect to the beam axis and momentum, respectively. The angular acceptances for FASER and FASER 2 are indicated by the vertical gray dashed lines [4].

FASER will search for light and extremely weakly interacting particles in the far forward region of the LHC where a large number of mesons (10^{16} pions per year) is expected, as shown on Fig. 1.3. This large production rate, needed if these particles are weakly coupled, might allow to find dark photons (A') - a product of the decay of pions - that are otherwise swamped in background at higher θ , within the ATLAS detector. The mesons would decay in ATLAS ($\pi^0 \rightarrow A' \gamma$) and the A' would travel and decay in FASER, situated 480 m along the line-of-sight of the proton collisions in front of the ATLAS interaction point at the LHC. FASER also has the prospect of discovering axion-like particles - produced in LHC collisions via high energy photons interacting with material in the LHC neutral particle absorber (TAN).

FASER is part of the CERN's Physics Beyond Colliders study group which is an exploratory study aimed at exploiting the full scientific potential of CERN's accelerator complex and its scientific infrastructure through projects complementary to the existing LHC, HL-LHC and other possible future colliders. The

cost of the experiment is less than 1 MCHF and funding has been secured from the Simons and Heising-Simons foundations.

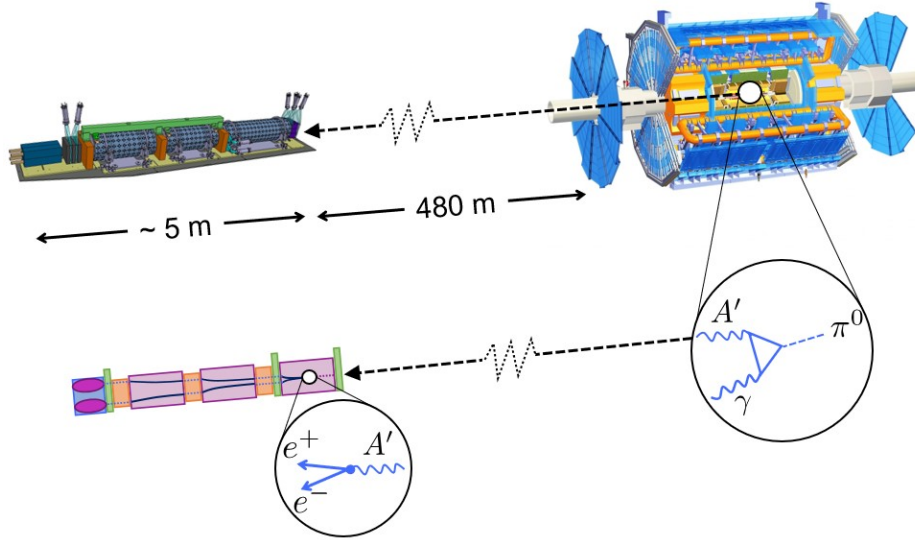


Fig. 1.4 Dark Photon (A') production: neutral pions decaying in ATLAS produce dark photons that would travel 480 m and decay in FASER.

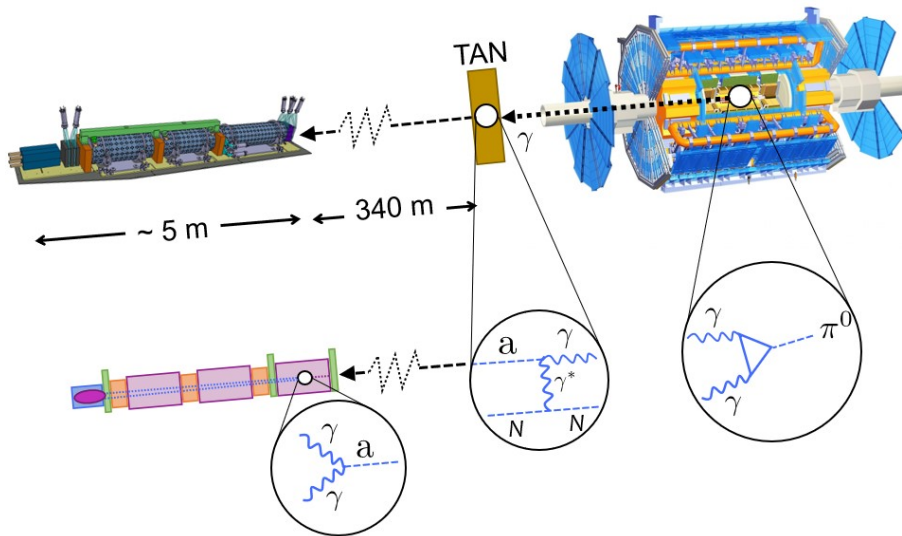


Fig. 1.5 Axion-like particle production: photons from ATLAS interact with the neutral particle absorber (TAN) and create axions through the Primakoff effect.

Chapter 2

The Faser Experiment

2.1 Experiment Location and Schedule

The FASER collaboration makes use of a disused tunnel, called TI12 (Fig. 2.1), in just the right location to intercept these new particles that could be escaping from collisions in the ATLAS detector. The idea is that these light and weakly-interacting particles produced at the ATLAS IP will travel along the beam collision axis, unaffected by the magnets that bend the beams of particles around the ring of the LHC, through matter (mostly rock and concrete) without interacting and then interact within the FASER detector in TI12 [5]. The schedule is tight. The LHC is warmed up for maintenance until end of 2020, and FASER needs to be built before Run 3. A third long shutdown is planned for 2024 where the collaboration could do some maintenance/upgrade on FASER or install FASER 2, a bigger version of FASER.

In Fig. 2.1 we see that charged particles are deflected by the LHC magnets and that neutral hadrons are absorbed by either the Target Absorber for secondary particles (TAS) or the Target Absorber Neutral (TAN) that are designed to protect the magnets. This means that only LLPs will travel from the IP to FASER through 10 meters of concrete and 90 meters of rock. In the SM only muons and neutrinos can reach FASER. However the LLPs is expected to easily pass through all of the material without interacting and then decay in FASER. We also see in the bottom right corner of Fig. 2.1 that FASER will be located roughly 5 meters from the LHC tunnel.

2.1.1 Civil Engineering Work

Some civil engineering work is needed to install the experiment in the tunnel. TI12 has a slight upwards slope which requires a trench to be cut out to align FASER with the beam collision axis. The digging depth will be around 50 cm and 5 m long. Figs. 2.2 and 2.3 show the civil engineering progress made in the tunnel.

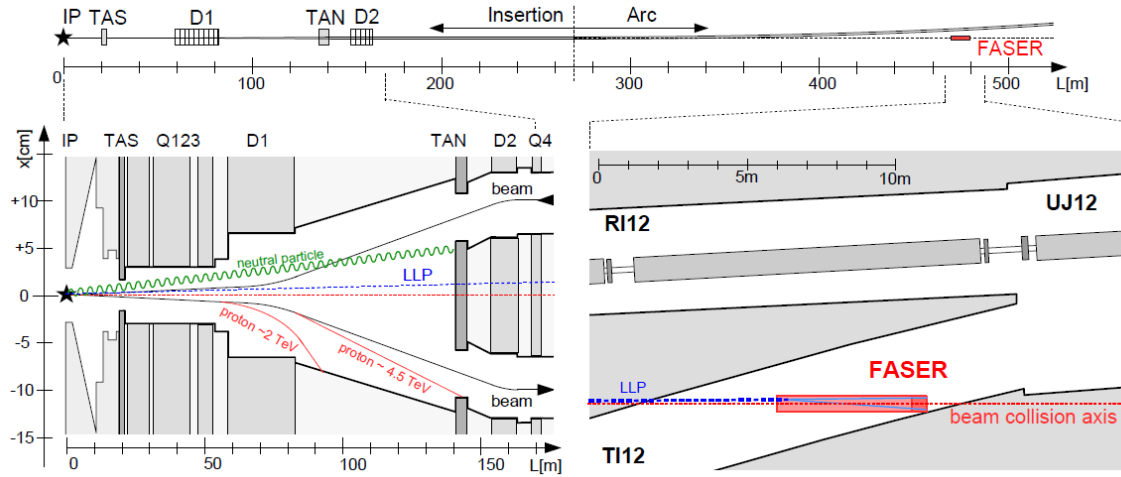


Fig. 2.1 TI12's infrastructure



Fig. 2.2 The disused TI12 tunnel before any accommodation for FASER.



Fig. 2.3 Ventilation has been removed and a trench has been dug in preparation for the installation of the experiment.

2.2 Magnet Design

FASER will use a 0.6 T permanent dipole magnet (divided in three parts) made of neodymium iron boron (NdFeB), which unlike electromagnets does not require high voltage power or cooling. It will be thin enough to allow the line of sight to pass through the magnet's centre with minimum digging in the floor of TI12. This is by far the most expensive piece of FASER. It will cost ~ 450 kCHF, making up half the price of the experiment.

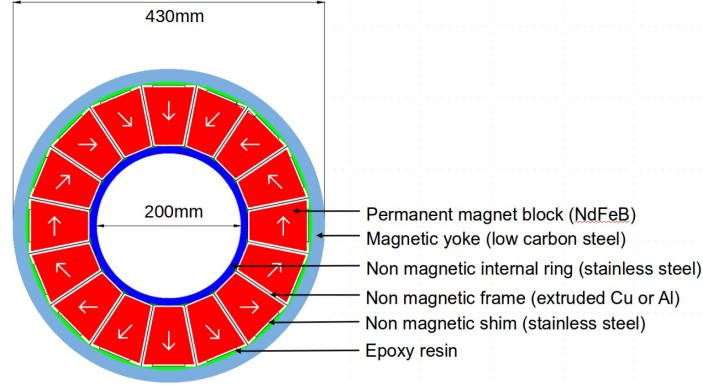


Fig. 2.4 The magnet uses a Hallbach array, a special arrangement of permanent magnets that in our case of a solenoid augments the interior magnetic field on one side of the array while cancelling the outer field to near zero on the other side. This is achieved by having a spatially rotating pattern of magnetisation.

2.3 Expected Signature

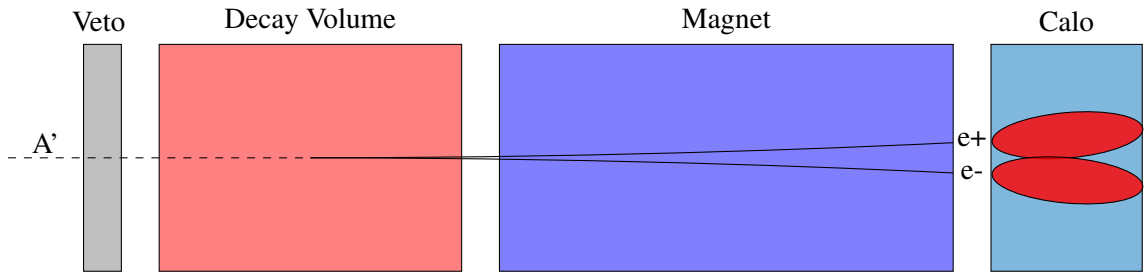


Fig. 2.5 Signal signature $A' \rightarrow e^+ e^-$.

The long-lived particles that will travel and enter the 20 cm width of FASER will be highly energetic of the order of the TeV [5]. If we use that $p_T \sim m$:

$$\theta \sim \frac{m}{E} \sim \frac{p_T}{E} \leq 1 \text{ mrad} \implies E \sim \text{TeV}$$

A characteristic signal event would be a long lived particle production after a pp collision followed by the LLP's decay into TeV SM particles after the travel to FASER:

$$pp \rightarrow \text{LLP} + X, \text{ LLP travels } \sim 480 \text{ m}, \text{ LLP} \rightarrow e^+e^-, \mu^+\mu^-, \pi^+\pi^-, \gamma\gamma, \dots$$

Signal signature $A' \rightarrow e^+e^-$

A typical signature, shown in Fig. 2.5, for the $A' \rightarrow e^+e^-$ would be no signal in the veto scintillator and two high energy oppositely charged tracks, consistent with originating from a common vertex in the decay volume, and with a combined momentum pointing back to the IP. A large electromagnetic energy would be deposited in the calorimeter. The magnets are extremely important to separate the decay products of such light and highly energetic particles. Without them the decay products would only be separated by a few hundred μm after traveling 1 m in the detector. For a LLP with mass $\sim 100 \text{ MeV}$ and energy $E \sim 1 \text{ TeV}$:

$$\theta \sim \frac{m}{E} \sim \frac{100 \text{ MeV}}{1 \text{ TeV}} \sim 100 \mu\text{rad}$$

This is why the FASER magnets are needed to separate the tracks of the products.

2.4 Detector Break Down

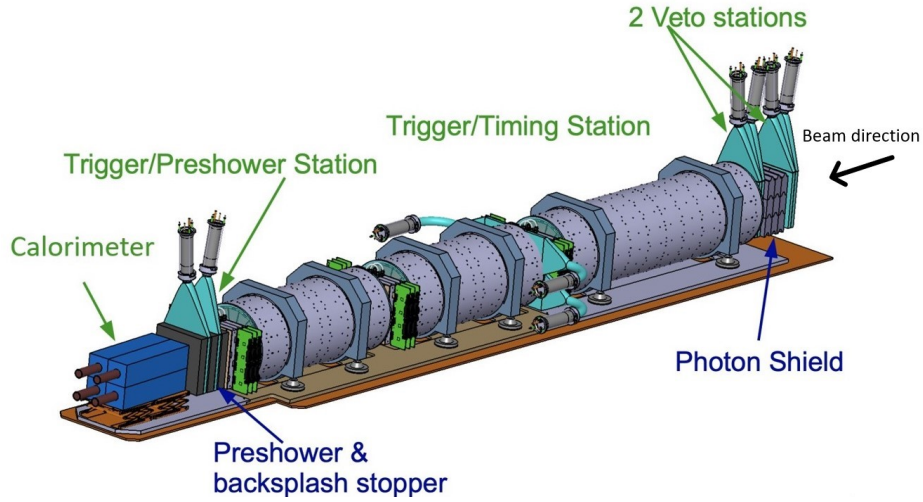


Fig. 2.6 FASER Schematic.

The FASER experiment is composed of multiple detectors, shown in Fig. 2.6, that must all work in harmony to discover BSM particles. The beam first enters the front of the detector (right of Fig. 2.6) composed of two scintillator stations that are used to veto charged particles, primarily high-energy muons, coming from the IP. Each of these station is made up of two layers of scintillators and has in between them a lead photon shield that converts photons into electromagnetic showers that can be

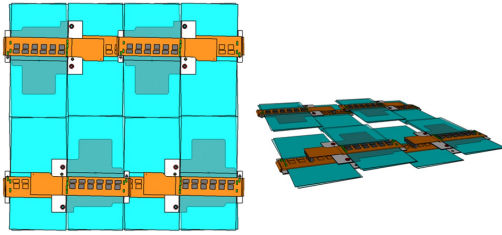


Fig. 2.8 A Tracking Layer. Each SCT module is $6\text{ cm} \times 12\text{ cm}$ and the tracking plane is made of 8 modules, giving a square of about $24\text{ cm} \times 24\text{ cm}$ covering the full active area of the detector

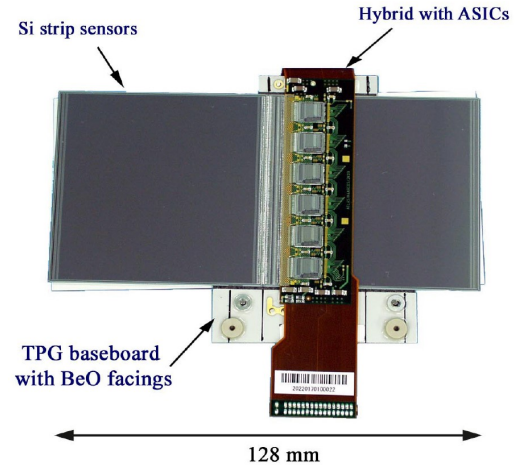


Fig. 2.9 An SCT Module. Visible in the middle are 6 of the 12 on-board ASIC chips that must be actively cooled with a water loop.

vetoed by the scintillators. After the veto stations, the product of the particles decaying in the decay volume will be separated by the 1.5 m long magnet. The decay volume is followed by a spectrometer consisting of two 1 m long magnet and three tracking stations to reconstruct the path of the pair of charged particles and measure their momenta. At the entrance and exit of the spectrometer are placed scintillator stations for triggering and precision time measurements. Finally, an electromagnetic calorimeter will identify high-energy electrons and photons and measure the total electromagnetic energy [6].

2.4.1 Trackers

The FASER tracker's purpose is to reconstruct the trajectories of energetic charged particles that will have decayed in the decay volume. The tracker will be made up of 3 tracking stations, shown on Fig. 2.7, located after each of the three magnet segments. Each of the tracking stations is composed of 3 layers, like the ones displayed in Fig. 2.8, of double sided silicon strip detectors that each contain 8 SCT modules shown on Fig. 2.9. All of the SCT modules used are spare modules from the SCT module production for the ATLAS experiment that allows the FASER collaboration to save resources [6].

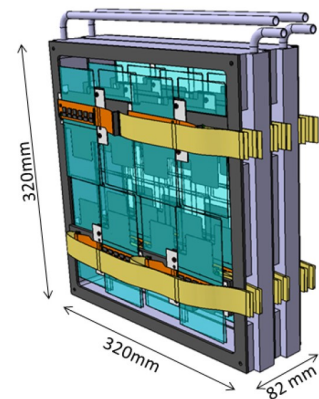


Fig. 2.7 One of the three Tracking Station.

2.4.2 Calorimeter

The calorimeter installed at the back of FASER will measure the electromagnetic energy of the event. It will be able to identify electrons and photons and will be used for triggering. As for the Tracker, the calorimeter will use 4 spare LHCb outer ECAL modules. It is composed of 66 layers of lead/scintillator that are readout by photomultipliers (PMTs) (no longitudinal shower information). Its dimensions are $12\text{cm} \times 12\text{cm} - 75\text{cm}$ long (including PMT) and provides $\sim 1\%$ energy resolution for 1 TeV electrons. However, the resolution will degrade at higher energy due to not containing full shower in calorimeter.

2.4.3 Scintillators

Scintillators will be used to identify charged particles entering the decay volume, and for triggering. They will use PMT for the readout. They are produced at CERN and are capable of extremely efficient charged particle veto.

Eight scintillators will be used in FASER. A double layer of scintillators, shown in Fig. 2.10, will be located at the entrance of the detector to veto incoming charged particles, primarily high-energy muons followed by a thick layer of lead converting photons into electromagnetic shower that can be efficiently vetoed by an additional double layer of scintillator.

After the first magnet and tracking layer is located the timing/trigger station composed of a double layer of wider scintillators, see Fig. 2.11. They are used to detect the first appearance of a charged particle pair from the decay of an LLP in the decay volume. It will be used as the primary trigger and to measure time elapsed since the proton-proton collision at the IP.

A final scintillators station is located in front of the calorimeter. It is used to provide an additional trigger that can be used in tandem with the first one and has a thin layer of material (either tungsten or lead) in front of it to create a simple preshower detector. This will allow the detection of the physics signal of two close-by energetic photons [6].

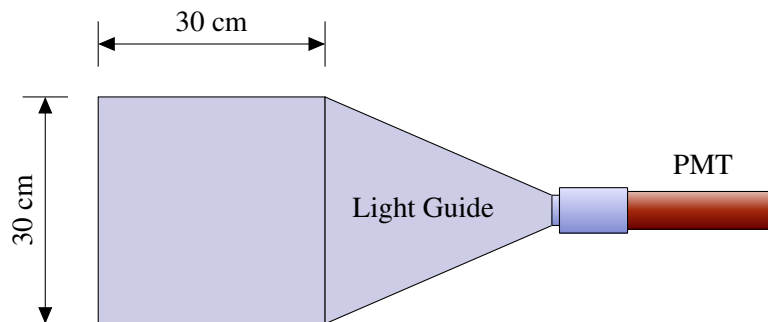


Fig. 2.10 Six of these scintillators will be used for the veto and preshower stations. The basic design is 2 cm-thick, $30\text{ cm} \times 30\text{ cm}$ plastic scintillator made of Bicron connected through a light guide to a Hamamatsu H3178-51 PMT.



Fig. 2.11 Scintillators used for the timing/trigger station. Their dimension are 1 cm-thick, 20×40 cm. Because they are wider, PMTs have been installed on both sides. Two scintillators will be offset along the line of sight to introduce an overlap and avoid inefficiency.

Chapter 3

Data Acquisition Architecture

My work focused on the understanding of the Trigger and Data Acquisition System (DAQ), as well as, implement the controls for the Trigger Logic Board (TLB) that is at the center of Fig.3.1 and a core component of the data taking. I will first, however, describe the general layout of the data acquisition so that the reader has a better understanding of the TLB's role in FASER.

The DAQ architecture digests information from three different kinds of detectors (in grey in Fig. 3.1), 72 SemiConductor Tracker modules (SCT), 10 scintillators and 4 calorimeters. The SCT modules are connected to tracker readout boards whereas the scintillators and calorimeter are connected to the digitizer. The signals from the scintillators and calorimeters are combined in pairs (AND, OR or one-only) to form 8 trigger outputs that go straight into the TLB. The TLB then maps the 8 trigger outputs into 4 final trigger lines via a Look-up Table described in section 4.3. Any signal in the 4 trigger lines will ultimately decide whether we want to save the signal as a physical event or not. The tracker modules do not contribute to forming a trigger decision, this is done solely using calorimeter and scintillator information. They are readout on a trigger accept signal only. On top of the four trigger lines coming from the scintillators and the calorimeters, two additional trigger accept signals can be created by a random trigger or software trigger generated internally by the TLB. In total, the TLB outputs six trigger lines, that can all be prescaled or vetoed, and form a Level 1 Accept (L1A). If the board decides to save the event for further analysis, it sends the L1A signal to both the digitizer and to the tracker readout boards which initiates the read-out of the subdetector's event data from the respective onboard buffers to the data acquisition server in a server room above ground.

Additionally, the TLB needs to receive time information through the LHC beam synchronous timing system (BST). The orbit can be passed directly via LEMO cable to the TLB. This information is then used to assign the Bunch Counter Reset (BCR) signal. The clock signal however needs to pass through a "clock cleaner" which is a commercial board that ensures stable and high quality clock signal.

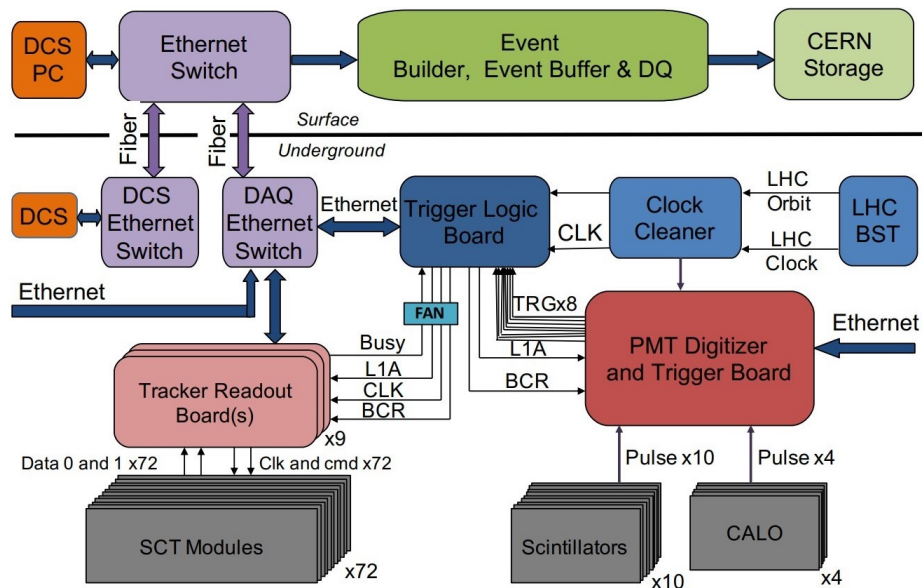


Fig. 3.1 DAQ Architecture - The TLB processes trigger signals from the scintillators and the calorimeter and returns an L1A to the tracker board and the digitizer.

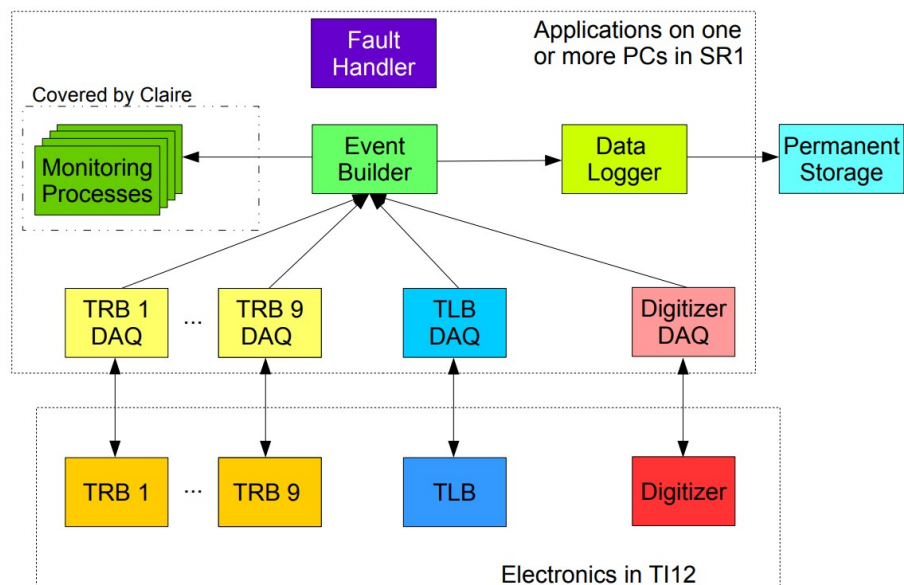


Fig. 3.2 DAQ Dataflow - The front-end electronics will each have their own receiver application which takes the initial raw data, adds a fragment header and sends it to the Event Builder.

3.1 DAQ Dataflow

Once an L1A has been sent out to all detectors, the event's data acquisition can start. It takes place in three different stages - combining more and more the different detectors data into a single event, see Fig.3.2.

Each front-end electronics board located in TI12 (TRB, TLB, Digitizer) will have its own "receiver" application which will take the initial raw data from the read-out boards (ROs) whenever it receives a trigger. These receiver applications are board-specific because they need to send and check board configuration before each run. They will also be able to send special commands to the electronics during runs such as the ECR (Event Counter Reset). Moreover, each read-out receiver application (RORs) will take the raw data payload and add a "fragment header" that contains important summary information of the event such as Event ID, BCID and size of the payload to form "fragments", see Table 3.1. The Event ID (number of triggers received since start) and BCID (bunch crossing ID) are embedded information in the raw payload that is extracted to construct the fragments.

After that, each receiver application sends out its data fragments to a single Event Builder (EB) application that will merge all fragments with the same event ID into a single event with an additional "event header", see Table 3.2. The event builder can check at this moment if no error is present by cross checking between each fragments if the BCID match.

Table 3.1 ROR event header assigned by the receiver applications

ROR event header	Size (bits)
Start of header marker	8
Fragment tag	8
Trigger bits	16
Format version number	16
Total header size	16
Total data size	16
Source identifier	32
Event ID	64
Bunch crossing ID	16
Data status	16
Time-stamp	64
Total size	288 (9 4-byte words)

A small note on why the ECR is needed: the electronic boards only use 24-bit counters which are filled in ~ 7 h at the maximum expected rate of 650 Hz that we need to reset as we want to run for longer periods. This is why the Event ID in the ROR and EB is 64 bits, ensuring each event has a unique 64-bit event ID even though the event counter is periodically reset. The first bits will be used to count the number of ECR.

Table 3.2 Full event header assigned by the EB

Full event header	Size (bits)
Start of header marker	8
Event tag	8
Trigger bits	16
Format version number	16
Total header size	16
Total data size	16
Number of fragments	8
Run number	24
Event ID	64
Event Counter	64
Bunch crossing ID	16
Data status	16
Time-stamp	64
Total size	352 (11 4-byte words)

3.2 Run Control

During FASER runs, everything will be launched, controlled and configured using the web-based run-control application, see Fig. 3.3. It will feature basic DAQ control commands such as initialise, start, ECR, stop and shutdown. Additionally, a configuration window will quickly let the user change the configuration json (details: 5.2) and a run information panel will help visually monitor the physics and monitoring rate. Finally, a status and settings panel will indicate whether each module (trigger, eventbuilder, datalogger, etc.) is running or stopped and let the user easily access the logs.

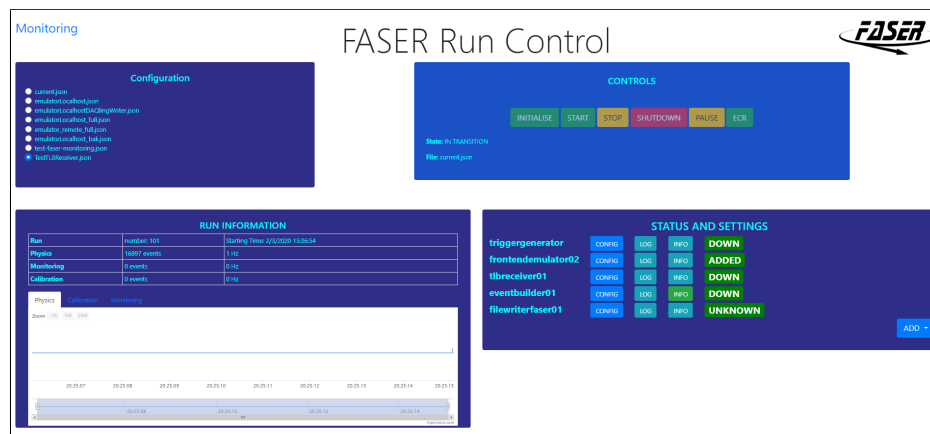


Fig. 3.3 DAQ Run Control - This run control allows to interface with the experiment during data taking. It is accessed on a browser and used to set configurations on the different detectors and start/stop the DAQ. It can also display live data from the experiment using metrics.

3.2.1 Displaying Data using Grafana

Monitoring will be done using Grafana [7], a multi-platform open source analytics and interactive visualization software. This will let the user do basic statistics such as average and rate of physics event. The implementation in the different modules, see section 5.3.5, is done by adding a metric to the selected variable chosen to be visualized in the code. For example, one would simply declare the metric

```
std::atomic<int> my_metric1;
```

and then increment it whenever a physics event has been detected.

```
my_metric1 +=1;
```

The metric is periodically registered in a monitoring database which Grafana interfaces with. The monitoring database used is Influxdb [8] - an open-source time series database.

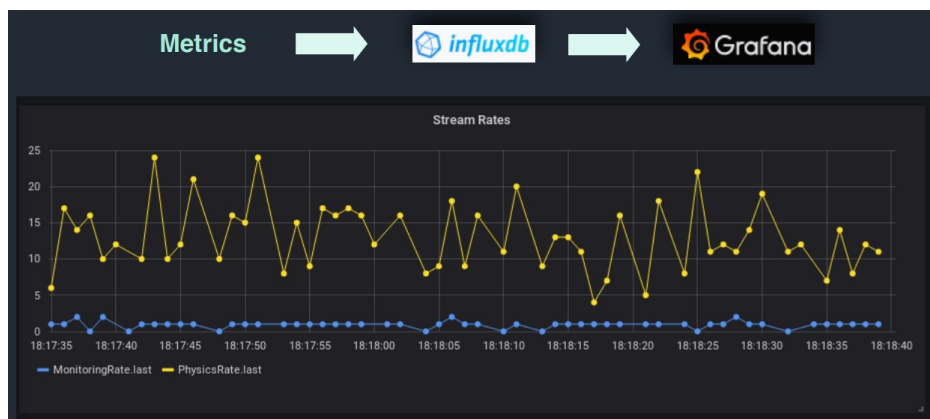


Fig. 3.4 The metric variable would be processed by Influxdb and then by Grafana to produces visualisations graphs.

Chapter 4

Trigger and Data Acquisition

4.1 Unige USB3 GPIO

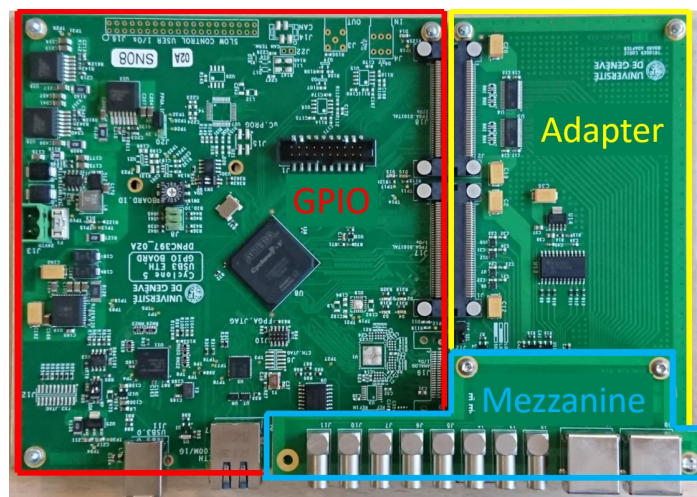


Fig. 4.1 The TLB is composed of three sub-parts and uses the same Unige USB3 GPIO board as the TRB for the "core", highlighted in red. An additional adapter board and mezzanine are used to interface with hardware signals.

The Trigger Logic Board (TLB) and the Tracker Readout Boards (TRB) are both designed to be the interface between the hardware signals and the readout and control. The TRB digest signals straight from the SCT modules whereas the TLB receives the scintillators and calorimeter signals through the Digitizer. The solution is to use a GPIO (General Purpose Input-Output) developed by the University of Geneva called the "Unige USB3 GPIO" for both the TLB and the TRB as core module ¹ with an additional dedicated interface board for each application, see Fig. 4.1. For the readout and the slow control, two interfaces can be used: USB3 and Ethernet. USB3 has initially been used but ultimately Ethernet will be the means of communication [6].

¹The core of the board is an Intel Cyclone V A7 FPGA

Both TLB and TRB boards have a standalone GUI to perform the initial testing. As an example, Fig. 4.2 shows the TLB's GUI and its three tabs. The "board" tab controls the DAQ start and stop as well as sending direct parameters through slow control such as enabling or disabling the trigger and sending software triggers. The second tab is used to toggle the eight inputs that come in through the LVDS. It also allows the prescales to be set and additional control such as enabling the LHC Clock or using the on-board one. The last tab is a 256 entries matrix that allows to set the LUT (more information in 4.3).



Fig. 4.2 The DAQ GUI has three tabs, one to start DAQ and send slow control commands, one to configure inputs and prescales and the last one to set the LUT.

Once the data acquisition is started, the TLB can output two types of data: trigger data or monitoring data depending on which one was enabled. The trigger data is sent out in chunks of 5 "words" of 32 bits, see Fig. 4.4.

First a header is used to mark the beginning of the incoming trigger data. The second word contains the Event Counter coded into 24 bits and the third word uses the whole 32 bits to code the orbit counter. The fourth word is the Bunch Counter (0 to 3563) that uses the required 12 bits. The fifth and last word code different information. The first bits code the Input bits (next clock) and Input Bits. Bits 16 to 21 code the Trigger Before Prescale (TBP) where each of the bits is a trigger signal. Bits 24 to 29 code the Trigger After Prescale (TAP).

The Monitoring Data, used for debugging, follows the same word order but has a total of 25 words to code more information. Because the Monitoring Data rate is low compared to the Trigger

Data², several triggers will arrive in between monitoring data as well as information on events that are vetoed and thus not recorded as Trigger Data. This is why TBP now uses whole words to count up the hundreds or thousands of trigger that will have arrived and are now labelled TBP0 to TBP5. Additional Trigger after veto are used to count the trigger that have been vetoed and finally, the last four word code the different sources of veto. The full TLB data format is showed on Fig. 4.3.

4.2 TLB Board

The TLB is the central triggering module of the FASER TDAQ system. It receives input signals at 125 MHz from the scintillators and the calorimeter that have been converted into computer-readable information via the digitizer³. The TLB processes these signals to create the trigger decision (L1A) which is sent to the detector readout boards. The TLB is the brain of FASER as it receives signals from every detector and then takes the decision to store the event or not. A flow chart presenting the operation of the TLB is shown on Fig.4.5. The eight signals from the digitizer come in the TLB in the bottom right corner of the figure 4.5. These eight trigger lines coming from the CAEN digitizer are synchronized to the 40 MHz clock which can be selected to either come from an on-board clock or - during operations - from the LHC clock. The LHC clock signal is processed by a clock cleaner board before coming in the TLB. Each of the eight channels can then be individually delayed to account for differences in cable length and then go into the coincidence logic detailed in subsection 4.3 LUT. These four outgoing trigger lines, as well as the random trigger (TBP5) and a software trigger (TBP6) both controlled via SC (Slow Control), can now be individually prescaled. More information on these modules can be found in subsection 4.4.4 - 4.4.6. The signal from these six signals is almost ready to become an L1A. However, FASER has different sources of deadtime, see Table 4.1, where it is unable to read data. In fact, the TLB must first verify that a previous L1A wasn't sent out the previous 5-10 μs as the detector need this time to readout the data. It must also check that the signal is not within a few clock cycles of a BCR.

Table 4.1 Sources of deadtime.

Simple Deadtime:	Number of 25ns clock cycles after a L1A where a new L1A is vetoed.
BCR:	A dead time is issued if a BCR was sent in the last 6 CLKs, this CLK or the next two CLK cycles.
Busy (tracker):	Designed not to allow a new L1A while the tracker is reading out an event.
Rate Limiter:	Introduces dead time after three L1As in 15 turns. Triggers are rejected and the output rate is kept below 2.25 kHz.

²Max trigger rate: 1 kHz, Monitoring rate: 0.001 Hz to 11.2 kHz

³The digitizer, a CAEN VX1730 VME board, samples at 500 MHz, but its coincidence logic and outputs run at 125 MHz.

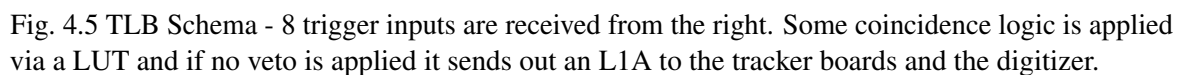
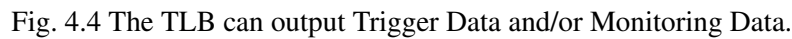
TLB Monitoring Data

Word 0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header																															
Word 1																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Event Counter																							
Word 2																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Orbit Counter																															
Word 3																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Bunch Counter												
Word 4 to 9																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	TBP0 to TBP5																							
Word 10 to 15																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	TAP0 to TAP5																							
Word 16 to 21																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	TAV0 to TAV5																							
Word 22																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Deadtime Veto Counter																							
Word 23																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Busy Veto Counter																							
Word 24																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Rate Limiter Veto Counter																							
Word 25																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	BCR Veto Counter																							

TLB Trigger Data

Word 0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header																															
Word 1																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Event Counter																							
Word 2																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Orbit Counter																															
Word 3																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Bunch Counter											
Word 4																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	TAP					0	0	TBP					Input Bits								Input Bits (next CLK)									

Fig. 4.3 TLB Data Format



4.3 LUT

The coincidence logic is done using a Lookup Table (LUT). The LUT is a 256 input table (8 bits) where each input receives a value from 0-16 (4 bits). The benefits of a LUT are that it saves precious computing time and is easily reprogrammable via slow control because it is stored in RAM. It works by receiving the eight input channels (TSig1-8) from the digitizer and outputting a 4 bit value accordingly. This 4 bit value are 4 user defined coincidence trigger lines before prescale (TBP1-4). This allows to setup complicated logic via software as opposed to via hardware.

Figure 4.6 helps visualize how the LUT converts the eight signals from the digitizer to 4 coincidence lines. One important thing to note is that every 256 values from the LUT must be configured every time the TLB is powered on because, as mentioned before, the LUT is stored in RAM. Table 4.2 represents how the user would fill a LUT that triggers on TSig1 - by assigning LutArray #1 to 1.

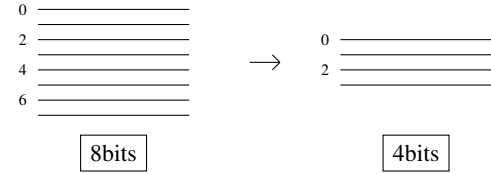


Fig. 4.6 The TLB's LUT receives 8 bits and converts it to 4 bits.

Table 4.2 An example of LUT setup in software.

LUTArray	#0	#1	#2	#3	#n	#255
Value	0	1	0	0	...	0

A few other examples are presented in Fig.4.7:

- The first example, in Fig.4.7a, shows a Trigger Signal (TSig1) from the digitizer in the first channel as visualized by the red line in the zeroth bit. In this example we want to attribute a certain coincidence logic to when we have TSig1 firing (and only the first channel firing, no other signal). This can be done by setting LUT#1 to 1 so that when you have a signal in TSig1 then you get 1 as output. It's important not to get confused by the indices, LUT#0 represent all 8 bits off, or in our example, all bits to black. Because the first line is red we need to assign a value to LUT#1.
- In our second example in Fig. 4.7b a signal is incoming in the second channel (TSig2) of the digitizer so we want to decide what LUT#2's logic will be. Here we decided we want to fire all the coincidence trigger lines before prescale so all the 4 bits are red, i.e. set to 16.
- In the last example we'll present a coincidence. Let's say the digitizer receives signals in TSig1 and TSig2 but you only want an L1A when they are both coming in together, then you can set LUT#1=0 and LUT#2=0 so that you don't trigger on single event but you set LUT#3 to 4 so that L1A's are only sent during a coincidence.

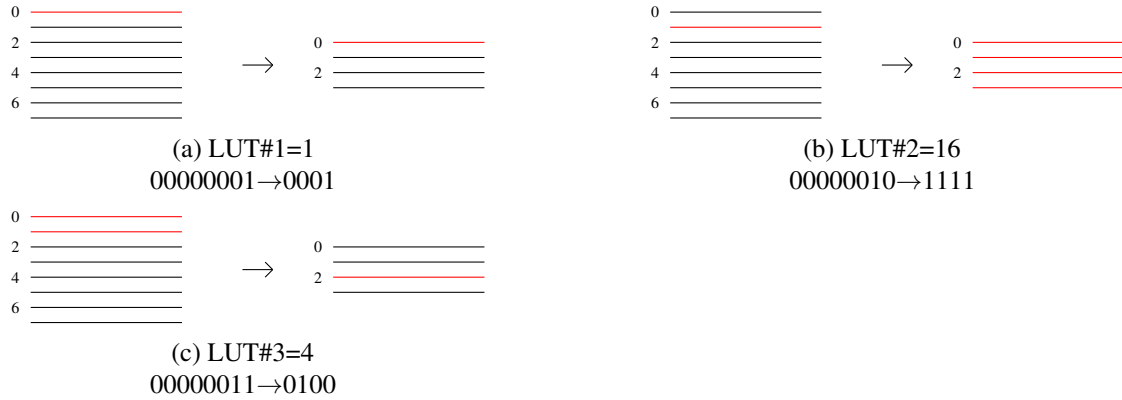


Fig. 4.7 LUT examples - cases are explained in main text.

4.4 TLB Commissioning

The first step in using the TLB at FASER is to make sure it respects all specifications. To do this, we install the TLB board in a scintillator laboratory at CERN. The setup to test the board consists of a pulse generator, an oscilloscope, the digitizer⁴ and the TLB, see Fig. 4.8.

4.4.1 Sampling Phase

The first element that has been tested is the sampling phase. We want the TLB input signal to be synchronized with the 40MHz LHC clock⁵ such that a signal is always generated a fixed number of clock cycles after a physics signal. However, because the digitizer outputs at 125 MHz we have an inherent 8 ns jitter⁶. This means that depending on the length of the cable the signal from the digitizer could, for example, fall close to the rising edge of the TLB clock - sometimes triggering, sometimes not. We could however trigger on the falling edge and that would trigger every time because of the length of the digitizer signal (32ns). This is the sampling phase option we need to test. In Fig. 4.9 the incoming physics signal is centered on the fourth rising edge of the clock. However because of the jitter, half of them will not trigger if we trigger on the falling edge but all of them will trigger on the rising edge.

To test this, we plug the BCR's⁷ output from the TLB to the Digitizer input1 (see Appendix A Instruments panels, page 51) as well as into the oscilloscope. We also plug the L1A output from the TLB to the oscilloscope. We then trigger on the L1A. We see that when we trigger on the falling edge, Fig.4.10a, we trigger on two distinct clock cycles whereas if we trigger on the rising edge, Fig. 4.10b, we only trigger on the same clock cycle. In conclusion, the sampling phase works and the actual setting for FASER will need to be calculated or measured once the FASER assembly is finished. Each

⁴The digitizer's brand and model are CAEN V1730.

⁵40 MHz or 25 ns between each rise of the clock signal and 12.5 ns between every phase change.

⁶This is a continuous jitter as opposed to discrete.

⁷11245 Hz so 88μs between each pulse

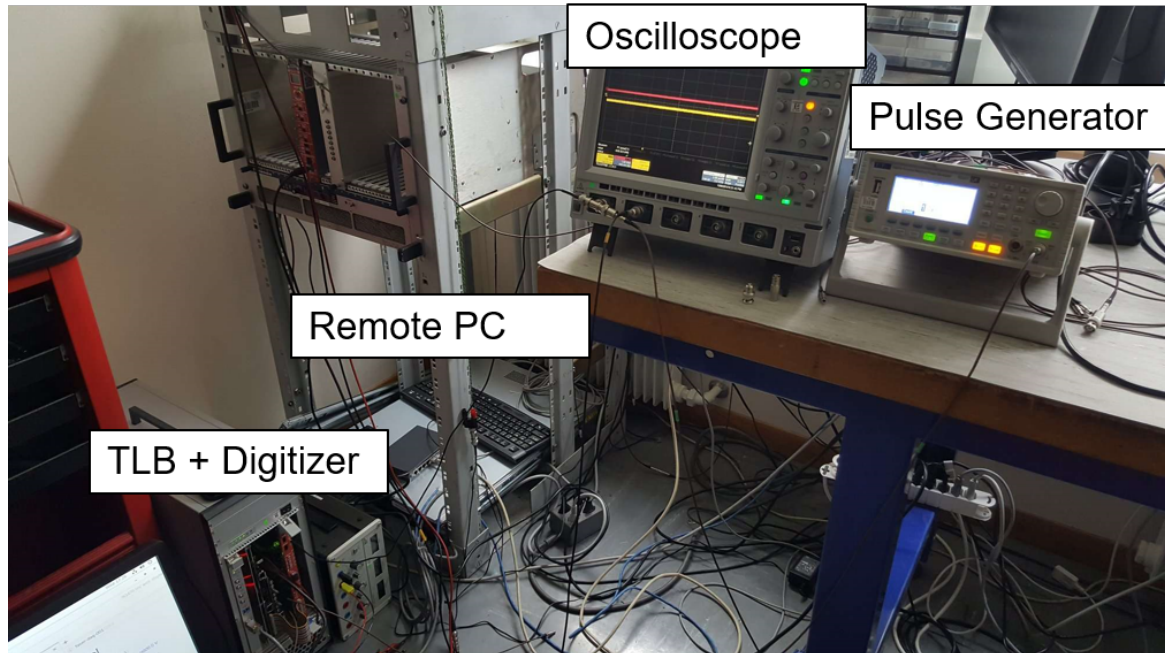


Fig. 4.8 TLB Test Setup in the scintillator lab.

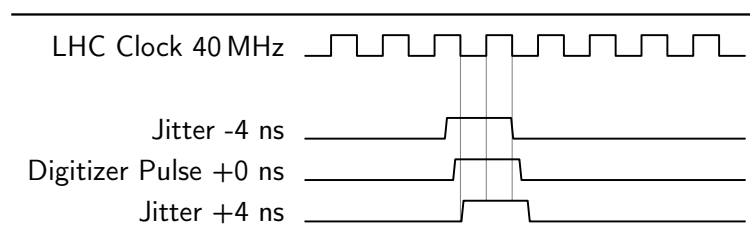


Fig. 4.9 Sampling on the falling edge trigger on two different clock cycle whereas sampling on the rising edge trigger only on one clock cycle.

input line might have a different setting depending on the length of the cables connecting the trackers to the TLB.



(a) Sampled on the falling edge

(b) Sampled on the rising edge

Fig. 4.10 Sampling on the falling edge we see two distinct signal which means we don't have a reliable time information as opposed to when we sample on the rising edge. The bottom part of each picture is a zoom of the top part. On each top right corner is the L1A the oscilloscope triggers on. The time division on the zoomed part is 11.8 ns/div which makes the signals 25 ns apart as predicted. The length of the digitizer signal is 32 ns.

4.4.2 Input Delay

The next important tuneable setting is the input delay. Because the different detectors are far apart, we must delay the signals that come from the front of FASER. This setting allows the user to delay a signal and to wait for the next incoming signals. This will be useful to coincide all tracking stations throughout the length of FASER. The Fig. 4.11 illustrates the concept of event arriving at different clock cycles depending on the travel length. It shows that from the ATLAS IP to FASER we will have a constant time delay of around 64 clock cycles. However, particles will interact with the three tracking stations at different clock cycles, roughly one clock cycle apart (to be determined with actual cable length). The TLB's input delay range is three clock cycles (75 ns) individually adjustable for each 8 channels. This is plenty for the approximately 16.5 ns length of FASER and the additional cabling.

The test setup, illustrated in Fig. 4.12, had the BCR0 signal from the TLB split with a T-connector into two cables - one 30 ns longer than the other - to the oscilloscope. Both signals then go to the digitizer channel0 and channel2 (the digitizer actually has 16 inputs but every odd channel is merged with the even one) with the 30 ns longer cable going to channel0. What we see on the oscilloscope on Fig. 4.13 is three signals. In yellow, the C1 delayed signal with respect to the pink C2 and the blue L1A. The test was successful because no L1A were detected if we asked a coincidence on channel0

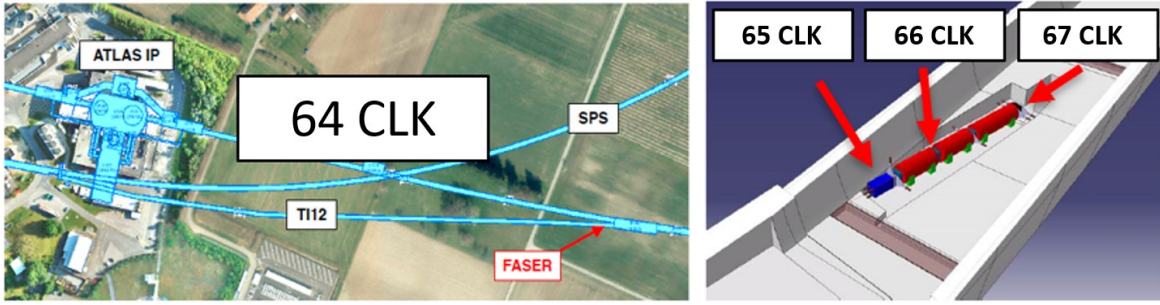


Fig. 4.11 Each physics event will have a constant clock cycle shift depending on where it is located in FASER. Example clock cycle values for illustration purposes.

and channel2 via SC. However, when we delayed the C2 signal by one or two clock cycle (25 - 50 ns) then we would get an L1A which is what happened in Fig. 4.13.

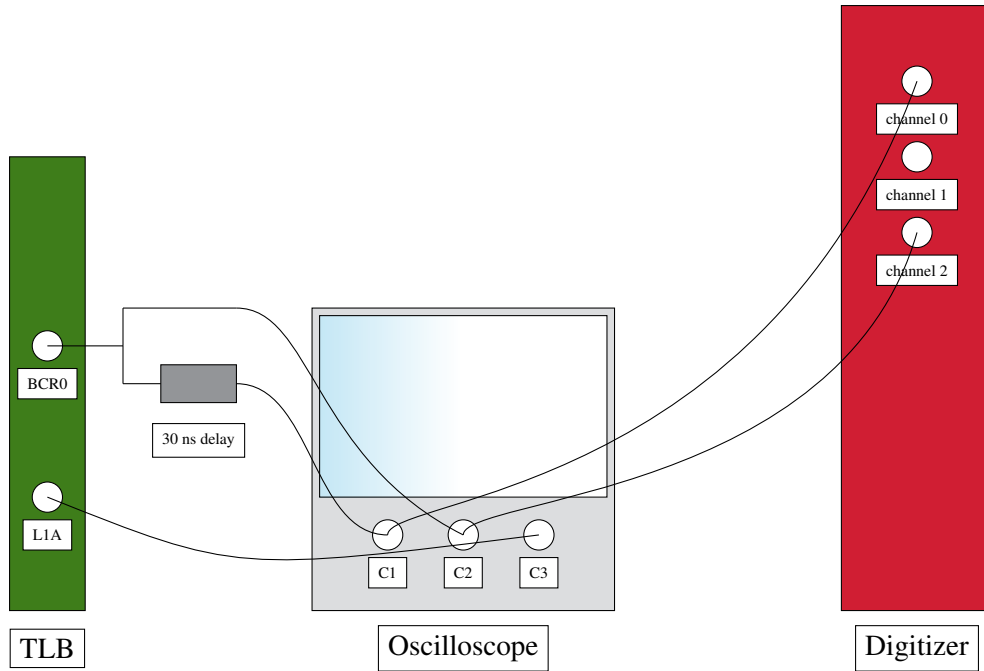


Fig. 4.12 Input Delay Schema. The BCR0 outputs a signal at 11245 Hz which is sent as an input in channel 0 and channel 2. We then see the L1A output. (Not shown on the figure is the LEMO cable connecting the digitizer to the TLB to communicate).

4.4.3 Coincidence Logic Test

The following test is the coincidence logic test. We took the same setup as in Fig. 4.12 but without the 30 ns delay and modified the look up table's value via SC with three different settings presented in Table 4.3. The first coincidence we tried was to set all the LUT values to 0 and LUT#1=1. That means we would only get an L1A if signals were received on channel0. In our setup we could unplug

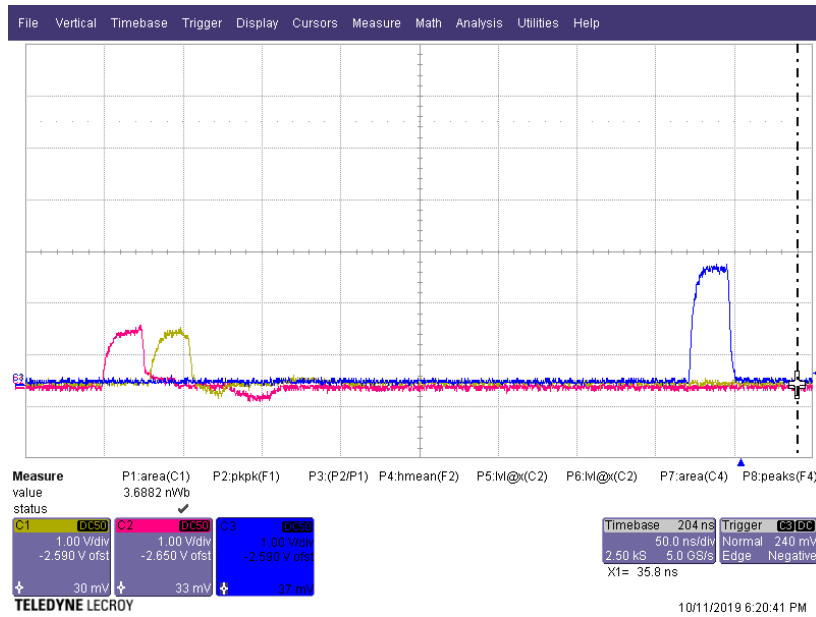


Fig. 4.13 Input delay. C1 (yellow) is delayed by 30 ns with respect to C2 (pink). The L1A (blue) is seen because of to the input delay of 1 clock cycle.

channel2 and still get L1As, however, if one were to unplug channel0 there wouldn't be any L1A as planned. Next is LUT#2=1, this means that the TLB would only send an L1A if the digitizer received a signal in channel2. This worked as expected and even with channel0 unplugged one got signals. The last test was to set LUT#3=1 and everything else to zero. This means requesting a coincidence from channel0 and channel2. As a result, L1As only got sent if channel0 and channel2 received a signal on the same clock cycle. Unplugging any one of the channels stopped the L1A generation.

Table 4.3 Look up table setting for the coincidence logic test

LUTArray	#0	#1	#2	#3	#n	#255
Triggers on ch0	0	1	0	0	...	0
Triggers on ch2	0	0	1	0	...	0
Coincidence on ch0 and ch2	0	0	0	1	...	0

4.4.4 Prescales

Prescales allows to sample the incoming data by selecting only a fraction of it. For example if you set prescale to 2 then only every 2nd trigger is passed, for a prescale of 3 only every 3rd signal is passed and so on. To test if the prescales where working, we sent fixed pulses with a signal generator - Fig.4.14 - at 1 kHz with a signal width of 40 ns and 1 Volt peak to peak. Table 4.4 contains data taken from the TLB's GUI. As expected the time to fill increase by two and seven-fold with the prescale setting. Additional tests, using the oscilloscope to visualize, were performed and can be seen on Fig.

4.15. The yellow signal on channel1, C1, represents the pulse generator whereas the pink signal C2 represents the L1As. In Fig.4.15a yellow and pink signals overlap because for every pulse sent, an L1A is emitted. However for Fig.4.15b one in two signals gets vetoed by the prescale setting and in Fig. 4.15c one in three.

Table 4.4 Prescale table for a pulse frequency of 1 kHz and a width of 40 ns. Time values are to fill a 1024 KB file.

Prescale setting	Time to fill [h:m:s]	T [s]	# events	L1A frq [Hz]
1	00:00:52	52	52432	1008.31
2	00:01:45	105	52432	499.35
7	00:06:07	367	52430	142.86

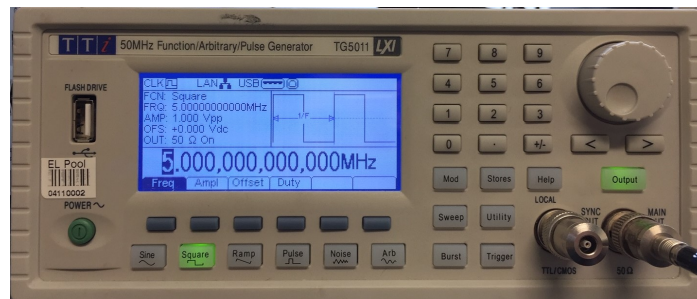


Fig. 4.14 Signal generator installed in the scintillator lab used to generator pulses that mimic physics signal.

4.4.5 Random Trigger

Random triggers are a pseudo-random generator generating 25 ns wide pulses (TBP5) at a user-settable (SC) average pulse rate of about 10-1000Hz. The random generator is based on a Linear-feedback shift register. A higher value uses more bits and hence a higher range of random numbers, potentially increasing the time between 2 triggers. Each time a trigger is generated, a new (different) seed is loaded into the shift register. It is a unique input line that has 8 different settings ranging from 0 to 7. The setting value determines the number of bits used in the shift register. In our test we tried 3 different settings shown in Table 4.5. We see that the frequencies range from 20-360Hz close to what was specified in the TLB specifications⁸. Another measurements made was the time between two outgoing L1As shown in Fig. 4.16. We see for these two settings that the distribution is indeed random as intended.

⁸A setting of 0 is actually the highest frequency one can set and should be even higher in frequency

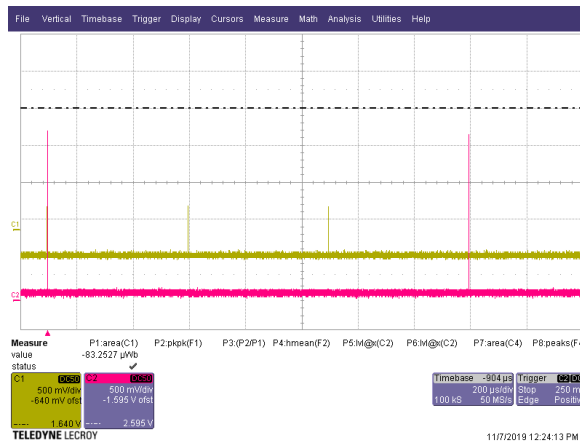
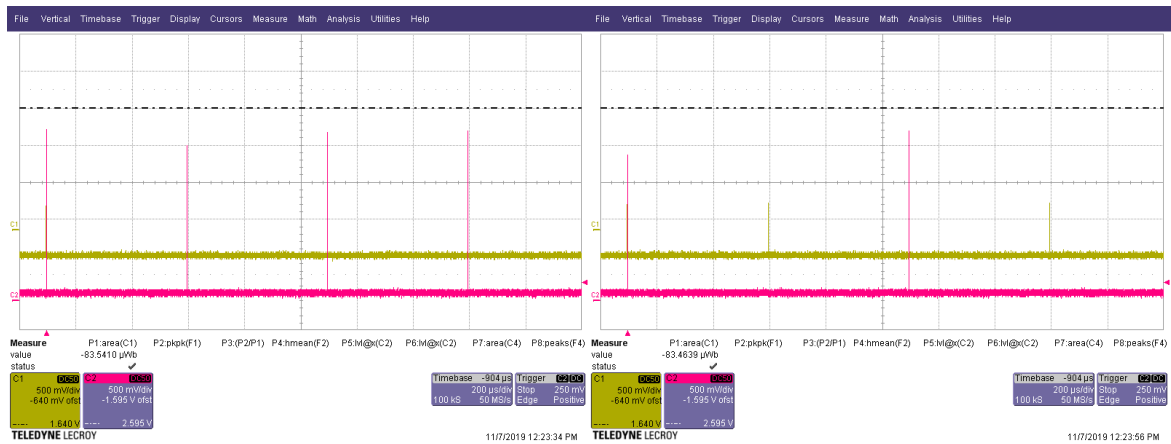


Fig. 4.15 Prescale results - the yellow signal is the pulse generator, the pink signal is the L1A.

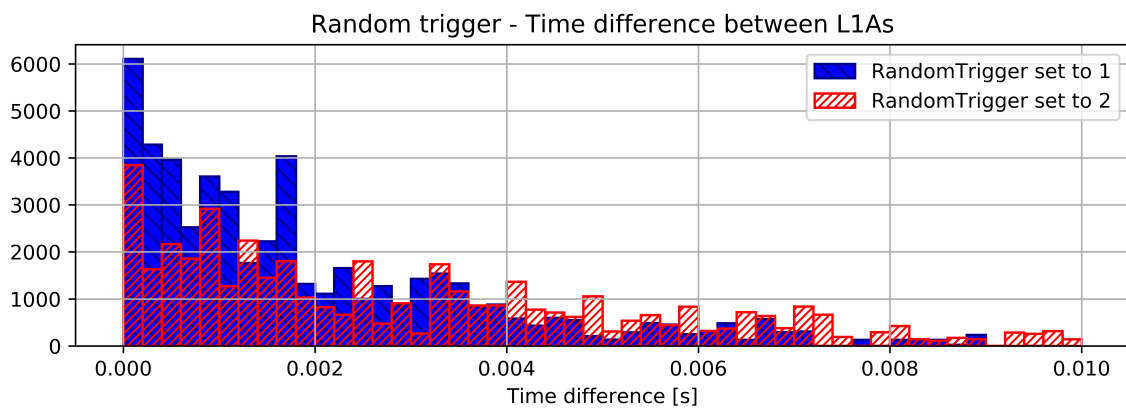


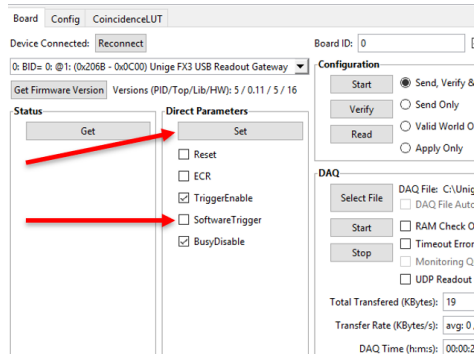
Fig. 4.16 The Random Generator has eight different settings from 0 to 7. 0 correspond to the fastest trigger rate and 7 to the lowest.

Table 4.5 Random trigger table. Time values are to fill a 1024KB file.

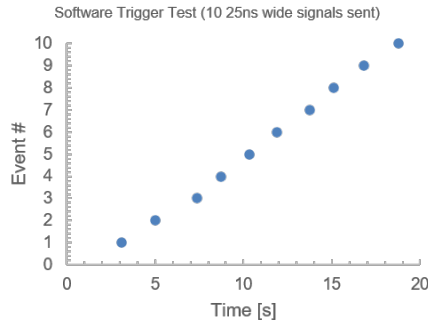
Random setting	Time to fill [h:m:s]	Avg time btwn triggers [s]	Avg frq [Hz]
1	1.71	$2.78 \cdot 10^{-3}$	359
2	4.37	$4.25 \cdot 10^{-3}$	235
7	12.8	$4.83 \cdot 10^{-2}$	20.7

4.4.6 Software Trigger

Software trigger is an additional trigger line that is user settable. For this module testing, we simply sent 10 software triggers via the GUI, see Fig. 4.17a, while taking data. This gives the plot on Fig. 4.17b which shows the ten software triggers. The x-axis represents the time it took to send the different trigger signals.



(a) GUI procedure.



(b) Software Trigger as a function of time.

Fig. 4.17 Software Triggers can be used as manual sources of trigger (and L1A) if no detector is set as an input.

4.4.7 Deadtime from the Rate Limiter

By design the TLB's rate is limited to three L1As per 15 orbits to protect from noisy triggers. This means that if you feed the BCR signal you'll only get an L1A every 5 orbits. This is what was tested on Fig. 4.18. The yellow signal on C1 is the BCR signal at 11.245 kHz and the pink signal are the L1As at 2.249 kHz⁹. As expected we see a pink L1A every 5 yellow BCR because of the vetoing.

4.4.8 Simple Deadtime

To test the simple deadtime, we generate with the pulse generator 10 Mhz pulses with 10 cycles, 40 ns width and 1 volt peak to peak. On Fig. 4.19a we see the three L1As in pink. There are only three because of the rate limiter is kicking in. They are at the same frequency as the driving signal with a

⁹Frequency: $\frac{3}{15 \cdot \frac{1}{11245}} = 2.249 \text{ kHz}$. Period: $T = 1/2.249 = 445 \mu\text{s}$

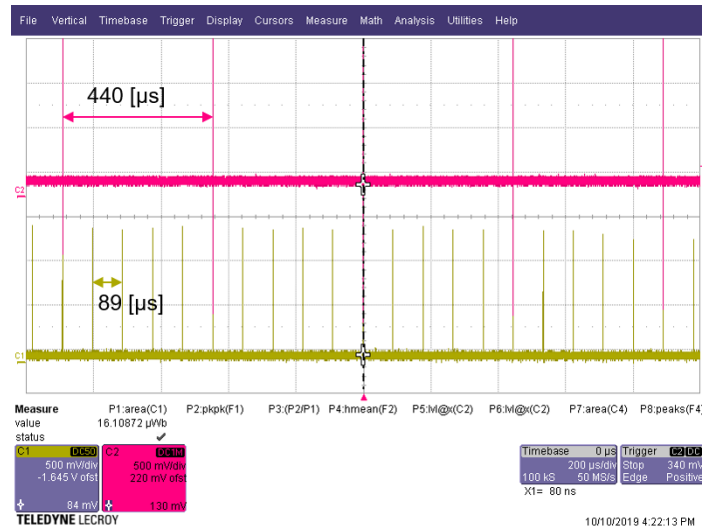


Fig. 4.18 Sampling Rate results - In yellow the TLB receives a BCR signal at 11.245 kHz. As the rate limiter kicks in, only 1 out of 5 trigger signal outputs an L1A.

phase because of the processing and cable length. Setting the simple deadtime parameter to 10 in Fig.4.19b results in the L1As being separated by 10 clock cycles or 250 ns as expected.

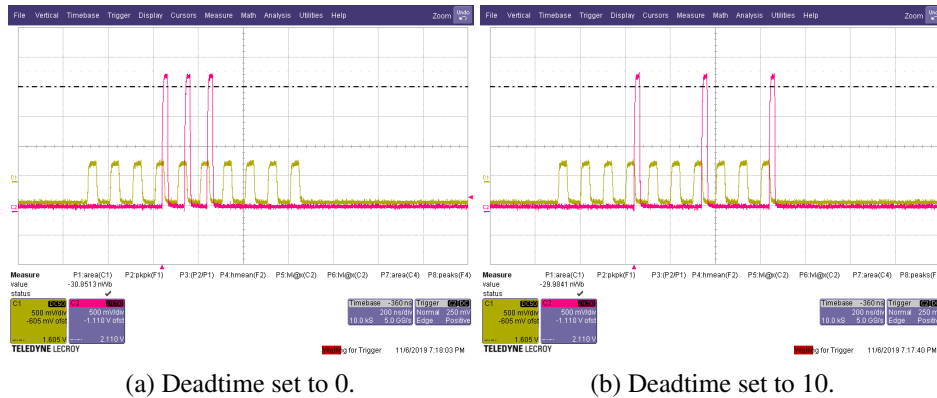
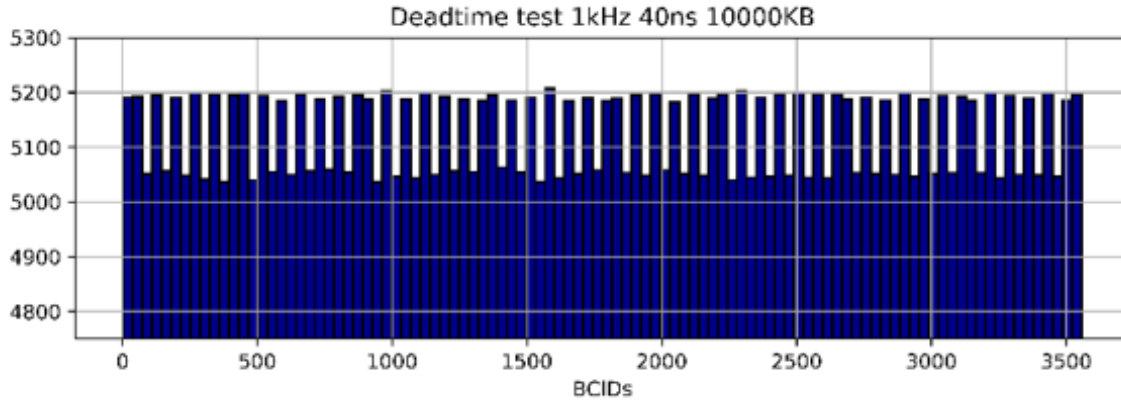


Fig. 4.19 Deadtime results - Changing the deadtime parameter forces the TLB to veto during the number of clock cycle selected.

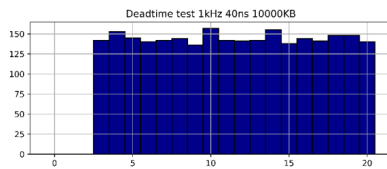
4.4.9 Deadtime from the BCR veto

The deadtime manager takes care of vetoing L1As via multiple criterium. One of them is that an L1A cannot be too close the a BCR. This behaviour is well described in the specifications. "A veto signal is generated if a BCR was sent in the last 6 CLKs, this CLK or the next two CLK cycles, i.e, if BCR-3 is set in the last 9 CLK cycles". The orbit delay generates the BCR-3 three CLK cycles earlier. In this test we are sending a pulse signal of 1[kHz], width 40ns and 1 Volt peak to peak and we fill a 10'000KB file with L1As. In Fig.4.20a we plot the BCID range from 0 to 3563. We find while

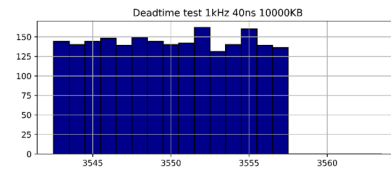
zooming in on the beginning in Fig.4.20b and on the end in Fig. 4.20c that it actually goes from 3 to 3557 and that we are missing 9 BCIDs as expected.



(a) Deadtime Manager whole range.



(b) Deadtime zoomed on beginning.



(c) Deadtime zoomed on the end.

Fig. 4.20 Deadtime results.

Another thing we are able to see in the output data are the sources of veto. For example in Fig. 4.21 we read the monitoring data for 11245 BCRs with the deadtime setting set to 10, the frequency of the signal generator set to 10 MHz, 100 pulses with a pulse period of 11 ns and a width of 40 ns. We are then able to see the simple deadtime, the BCR deadtime, the busy tracker (not working in our setup because no tracker connected) and the rate limiter.

4.4.10 Output Delay

The Output delay consists of delaying the outgoing L1A signal from the TLB. "It is a programmable delay for sending L1A signals at the right latency for each readout output. Each output delay is individually set by SC in CLK counts (7bits)". The option in the GUI is labelled DigitizerDelay and we tried three different options presented in Table 4.6. The results can be seen on Fig. 4.22. From left to right we see the first signal has the 16 clock cycle delay which corresponds to 400 ns, the second pulse is with the 4 clock cycle delay which corresponds to 100 ns and the third pulse is with no delay. We see the inherent delay from computing which is roughly 100 ns.

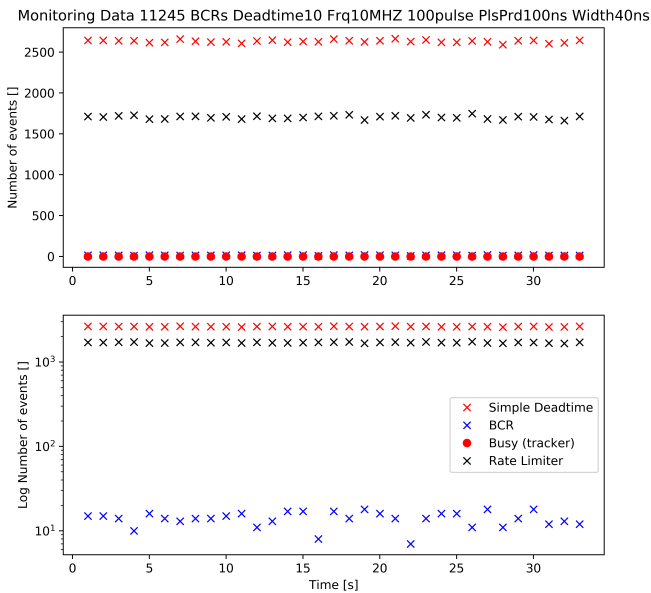


Fig. 4.21 Able to read 4 sources of veto. Pay attention to the vertical axis - bottom one is log scale. (The trackers weren't connected in this setup so the Tracker Busy is flat).

Table 4.6 Output delay.

Digitizer setting [CLK cycles]	Delay [ns]
0	0
4	100
16	400

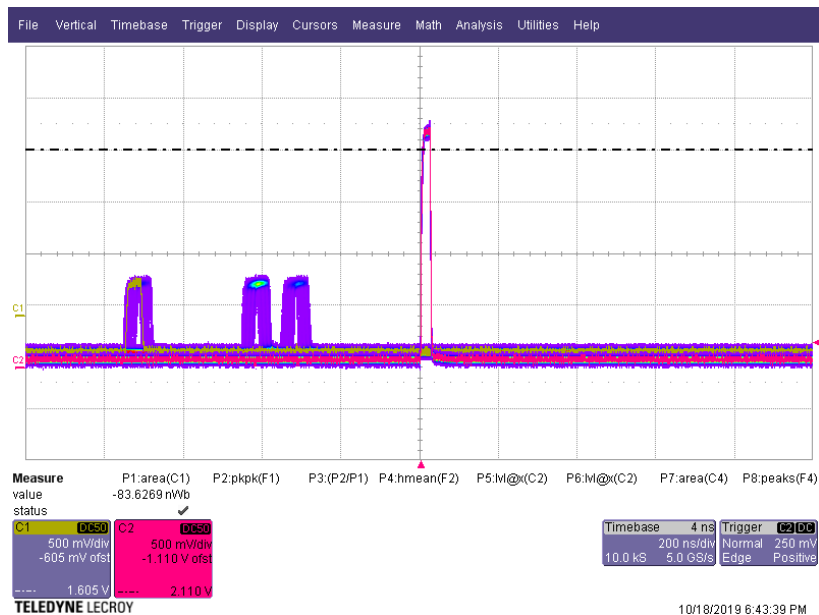


Fig. 4.22 Output delay with persistence on. Triggering on L1A allows to see the different delays.

Chapter 5

TLB Software

Both the TLB and the TRB use the same design of GPIO board and thus the low-level control software is shared. In what follows, the TRB software is introduced and its modifications into TLB software are explained.

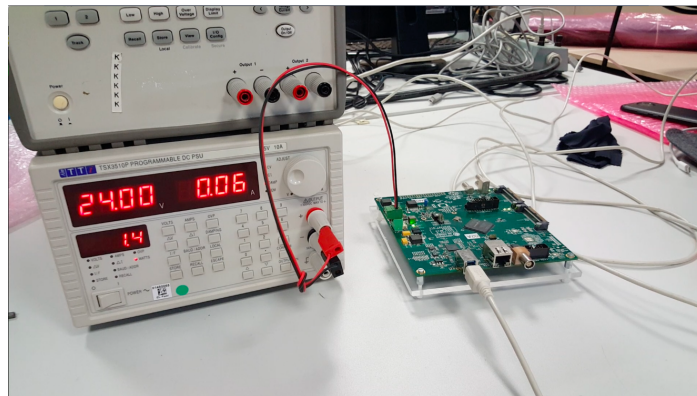


Fig. 5.1 TRB lab set-up in building 161, CERN, the TRB is powered via the generator with 24.0 V and 0.06 A and connected via USB 3.0 to a laptop for connection tests.

5.1 TRB/TLB communication program structure

To send instructions to the board, the GPIO base class is used. This contains the most important function called `SendAndRetrieve`. It allows the user to change bits at a certain address in memory. A typical program structure to access either the TLB or TRB would be.

```
#include "../TrackerReadout/GPIOBBaseClass.h"
using namespace FASER;
int main()
{
    GPIOBBaseClass * MyGPIOAccess = new GPIOBBaseClass(false);
    MyGPIOAccess->SendAndRetrieve(0x02, 0x1);
}
```

```
return 0;
}
```

Listing 5.1 In this example, SendAndRetrieve has two parameters: 0x02 and 0x1. 0x02 corresponds to the address in memory and 0x01 corresponds to the payload.

The SendAndRetrieve command expects the following arguments.

```
bool GPIOBaseClass::SendAndRetrieve(uint8_t CmdID, uint16_t CmdArg, uint16_t* AnswArg)
```

The first argument "CmdID" is the address in memory and tells the board where to write the second argument "CmdArg" which is the actual bit to change. The last argument "AnswArg" is what the program will write into when it receives something back from the board. In general it will send back the CmdID unless it is requesting specific data. A good check is to verify after each SendAndRetrieve that CmdID == AnswArg.

5.2 Configuration json

The full DAQ config will be stored in single json files and the setup can be modified either by directly changing it or via a GUI interaction, see Fig.5.2. Most of the configuration will be in this file. The trigger configuration data will be stored on a separate file which can be linked from the main configuration. During each run the configuration used will need to be archived for offline analysis either via common DAQling solution or via a simple text blob in database indexed by run-number.

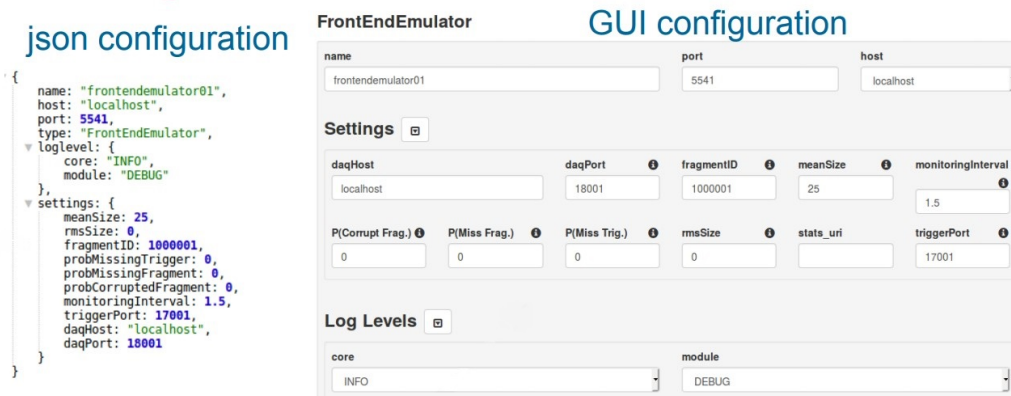


Fig. 5.2 Json and GUI configuration.

5.3 Software contribution

My contribution to the software has been on two fronts: implementation of both the TLB readout and TLB receiver DAQ software.

5.3.1 GPIODrivers

The GPIODrivers sub-folder located in the main FASER DAQ repository (gitlab.cern.ch/faser) is a package that contains the readout libraries as well as the standalone driver programs for the UniGe GPIO boards used within the FASER experiment's trigger and readout system, namely the Trigger Logic Board and the Tracker Readout Board. Each readout library depends on the GPIOBaseClass library, the base library to access a UniGe GPIO board¹, which is contained in the package.

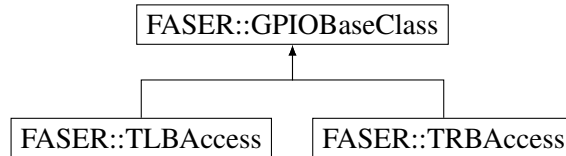


Fig. 5.3 GPIOBaseClass inheritance diagram.

The TLBAccess² is an API to communicate with the Trigger Logic Board included in the above package. It inherits from the GPIOBaseClass and mostly uses the SendAndRetrieve command described earlier. A typical use of the library would be to parse a json configuration file and store it as a variable.

```

{
  "SamplingPhase": [false, false, false, false, false, false, false, false],
  "InputDelay": [0, 0, 0, 0, 0, 0, 0, 0],
  "RandomTriggerRate": 3,
  "Prescale": [0, 0, 0, 0, 1, 0],
  "TrackerDelay": 0,
  "DigitizerDelay": 0,
  "LHC_CLK": false,
  "OrbitDelay": 0,
  "Deadtime": 0,
  "MonitoringRate": 1,
  "OutputDestination": 0,
  "Input": [false, false, false, false, false, false, false, false],
  "Reset": true,
  "ECR": false,
  "TriggerEnable": true,
  "SoftwareTrigger": false,
  "BusyDisable": true,
  "EnableTriggerData": true,
  "EnableMonitoringData": false,
  "ReadoutFIFOReset": true
}
  
```

Listing 5.2 Example json configuration format used to configure the TLB.

¹Faser internal information: [GPIO front-end interace specifications on sharepoint](#)

²TLBAccess's documentation.

The user would then pass the stored parsed json to the `ConfigureAndVerifyTLB(json)` function that "cleans" the board, i.e. sets all configuration memory bits to zero and sends the configuration. After that, it also verifies if the configuration has been applied correctly. If the configuration has failed it will try again to configure the board. `ConfigureAndVerifyTLB()` is a large function that uses different functions such as `CleanConfig()` and `ConfigureTLB()`. A diagram presenting its functionality is shown on Fig. 5.4.

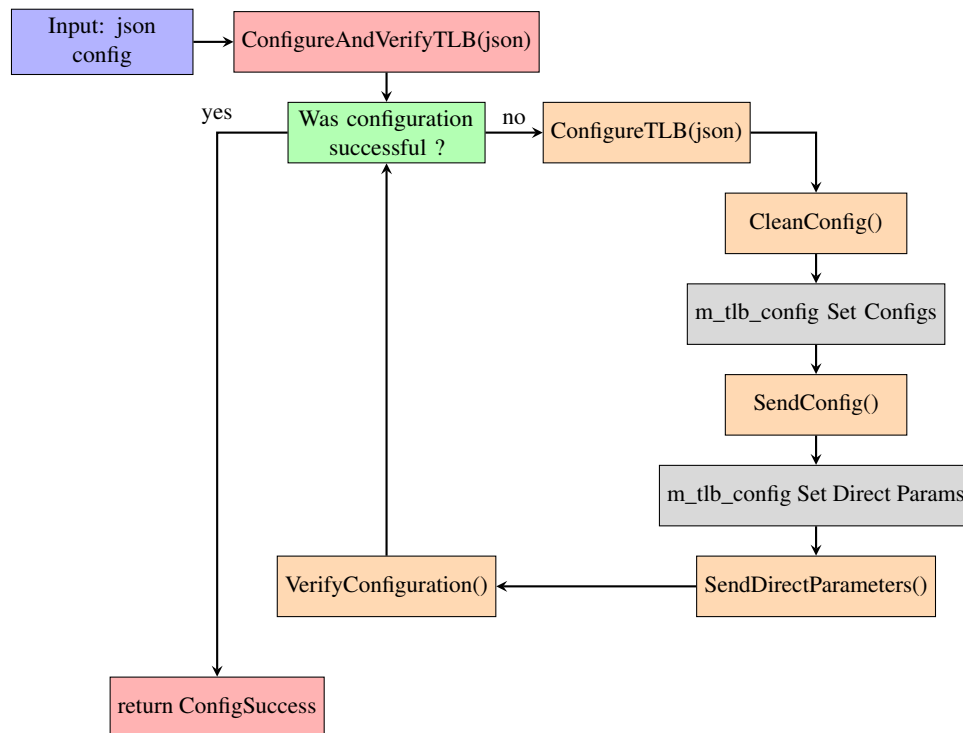


Fig. 5.4 The `ConfigureAndVerifyTLB()` function takes as an input a json configuration file and will try a pre-defined amount of time, in case of an error, to configure the board.

To clean all bits stored on the on-board memory the `CleanConfig` function sends 15 commands with two inputs. First the address `USER_SET_CONFIG` which is a unsigned integer of length 8 bits that stores `0x08`, the address of the configuration and second, the payload. Because we need to send 15 different words at the same address the first 4 bits of the 16 bits payload is reserved to specify which word number the payload is assigned to, see Fig.5.5.

```

void TLBAccess::CleanConfig(){
    SendAndRetrieve(TLBCmdID::USER_SET_CONFIG, 0x0000);
    SendAndRetrieve(TLBCmdID::USER_SET_CONFIG, 0x1000);
    SendAndRetrieve(TLBCmdID::USER_SET_CONFIG, 0x2000);
    ...
    SendAndRetrieve(TLBCmdID::USER_SET_CONFIG, 0xE000);}
  
```

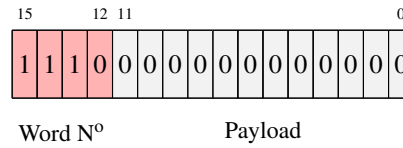


Fig. 5.5 Example of how 0xE000, the last word of the clean configuration is divided into two parts.

After the board has been cleaned, we use the `m_tlb_config` object, an instantiation of the class `ConfigReg` to set the configuration. Let's explain for example how we set the third word or word2 (0010) as we start counting from zero. To know what is coded into the payload of word 2, we must take a look at the variable layout in table 5.1. The zeroth word contains 8 sampling phases variable and two input delays because they are of size 1 and 2 bits respectively and that the payload size is 12 bits. The first word contains only the last `InputDelay` variables whereas the second word has three components. It starts with the random trigger rate that is coded on 3 bits, prescale 0 on 8 bit and a truncated prescale 1 on the last bit of the payload, see Fig. 5.6.

Table 5.1 Snippet of the variable layout TLB Configuration. Names highlighted in purple and blue are coded on the second word. The second prescale, in red, is coded on the second **and** the third word. The full layout is in the Appendix B.1.

Start Index	Next Index	Name	Bit Size	Min	Max
0	1	SamplingPhase	1	0	1
1	2	SamplingPhase	1	0	1
2	3	SamplingPhase	1	0	1
3	4	SamplingPhase	1	0	1
4	5	SamplingPhase	1	0	1
5	6	SamplingPhase	1	0	1
6	7	SamplingPhase	1	0	1
7	8	SamplingPhase	1	0	1
8	10	InputDelay	2	0	3
10	12	InputDelay	2	0	3
12	14	InputDelay	2	0	3
14	16	InputDelay	2	0	3
16	18	InputDelay	2	0	3
18	20	InputDelay	2	0	3
20	22	InputDelay	2	0	3
22	24	InputDelay	2	0	3
24	27	RandomTriggerRate	3	0	7
27	35	Prescale 0	8	0	255
35	43	Prescale 1	8	0	255
43	51	Prescale 2	8	0	255

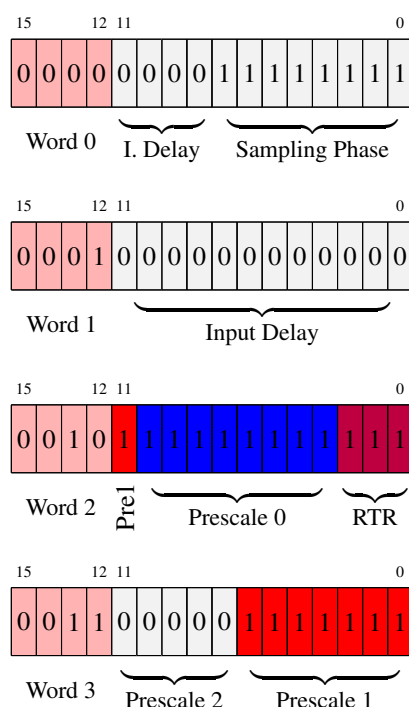


Fig. 5.6 Word 2 is divided into three parts, prescale 1, prescale 0 and RandomTriggerRate. Prescale 1 is split and coded on two different words. In this example, all variables are coded to their maximum value: prescales to 255 and RandomTriggerRate to 7.

5.3.2 Setting values in the TLB's configuration

The ConfigReg's header file has 15 ConfigWord variables that are sent by the TLBAccess to configure the TLB. They are initialized empty so one must fill them using the function built in ConfigReg. To fill the random trigger rate we simply call the SetRandomTriggerRate function that takes as an input the value from the configuration json. It performs a safety check to verify that the value respects the maximum value (of 7 for the RTR) and wipes the random trigger rate bits. The wiping must be performed if the user modifies more than once the same variable while the program is running. An example of wiping an old payload is shown on table 5.2. An example of setting a new value on a cleaned payload is shown on table 5.3.

Table 5.2 Wiping process using compound bitwise AND (&=). This cleans the first three bits and leave the others untouched.

Payload												
0	0	1	0	0	1	0	1	0	1	0	1	Old payload
1	1	1	1	1	1	1	1	1	1	0	0	Cleaning payload (0xFF8)
0	0	1	0	0	1	0	1	0	0	0	0	Ready to write new RTR

```
void ConfigReg::SetRandomTriggerRate(int RandomTriggerRateValue) {
```



```

if (RandomTriggerRateValue>7){RandomTriggerRateValue=7;std::cout<<"RandomTriggerRateValue
    is too big, setting it to max possible value"<<std::endl;}
ConfigWord2&=0xFFF8; //We clean all the random trigger rate bits so we can rewrite them.
ConfigWord2=(ConfigWord2|RandomTriggerRateValue); //We write the Random Trigger Rate bits
.}

```

Table 5.3 Setting the value using compound bitwise OR (|=). The three first bits have been cleaned and are now set to 101.

Payload												
0	0	1	0	0	1	0	1	0	0	0	0	Ready to write new RTR
0	0	0	0	0	0	0	0	0	1	0	1	RandomTriggerRateValue
<hr/>												
0	0	1	0	0	1	0	1	0	1	0	1	ConfigWord2

To set the first prescale, Prescale 0, there is a slight difference compared to the RandomTriggerRate as we must shift the value by 3 bits to the left to position the information at the correct spot. So just after cleaning the appropriate bits we shift and then we apply the value:

```
value=value<<3;
```

To set the second prescale, Prescale 1, we add a little complexity as it is on two different words. We first ensure that the value does not exceed the maximum value of 255 and then we clean both words at the appropriate bit position. The value is then copied onto two new variables and manipulated so as to split the information. Finally it is applied onto its respective ConfigWords.

```

//Set Prescale 1 (is on word2 and 3).
if (value>255){value=255;std::cout<<"Prescale 1 value is too big, setting it to 255."
<<std::endl;}
ConfigWord2&=0xF7FF; //We clean all the prescale 1 bits so we can rewrite them.
ConfigWord3&=0xFF80;
//now we need to split the value into two pieces
int value1=value;
int value2=value;
value1&=0x1; //meaning we only keep the first bit
value1=value1<<11;
value2&=0xFE; //meaning we erase the last bit 11111110
value2=value2>>1;
ConfigWord2=(ConfigWord2|value1);
ConfigWord3=(ConfigWord3|value2);

```

Sending the configuration words.

Once all of the 15 ConfigWords have been set we can simply use the TLBAccess's SendConfig() function to send them using SendAndRetrieve.

```
void TLBAccess::SendConfig(){
    SendAndRetrieve(TLBCmdID::USER_SET_CONFIG,m_tlb_config.ConfigWord0);
    SendAndRetrieve(TLBCmdID::USER_SET_CONFIG,m_tlb_config.ConfigWord1);
    ...
    SendAndRetrieve(TLBCmdID::USER_SET_CONFIG,m_tlb_config.ConfigWord14);}

```

The direct parameters use the same logic so won't be explained into details here and are simply separated because they are stored in another location in memory.

5.3.3 Reading out Data

Now that the board is configured³, the data can be read out. To start the data acquisition we call the StartReadout function that takes as an argument an integer that specifies if we want to enable Trigger/Monitoring data.

```
tlb->StartReadout(WhatToRead);

```

This function turns on the TLB's data outputting system and creates a secondary thread that runs the PollData function. This polling function stores completed words into the m_TLBEventData variable that is a vector of vectors. The main loop can then call the GetTLBEventData to copy and clear the m_TLBEventData and use it to parse/decode it or send it to the Trigger Receiver Module.

5.3.4 TLBDecode

Also implemented is a TLBDecode code to check if is a trigger or monitoring header during the PollData. This can also be used to decode standalone the TLB's data but ultimately the decoding will be done after the event builder.

5.3.5 TriggerReceiver Module.

For the daq modules, I worked on the TriggerReceiver located in Faser/daq/src/Modules/TriggerReceiver in the gitlab repository. This folder contains two files, the header and the main. The TriggerReceiver module works similarly to the test program in Fig. 5.7 in that it is also able to configure, start/stop and readout the TLB. The main difference is that the TLBAccess can run as a standalone software whereas the TriggerReceiver Module is used to integrate the TLB into a common framework

³The LUT's configuration would be set by now as well using functions written by O. Theiner.

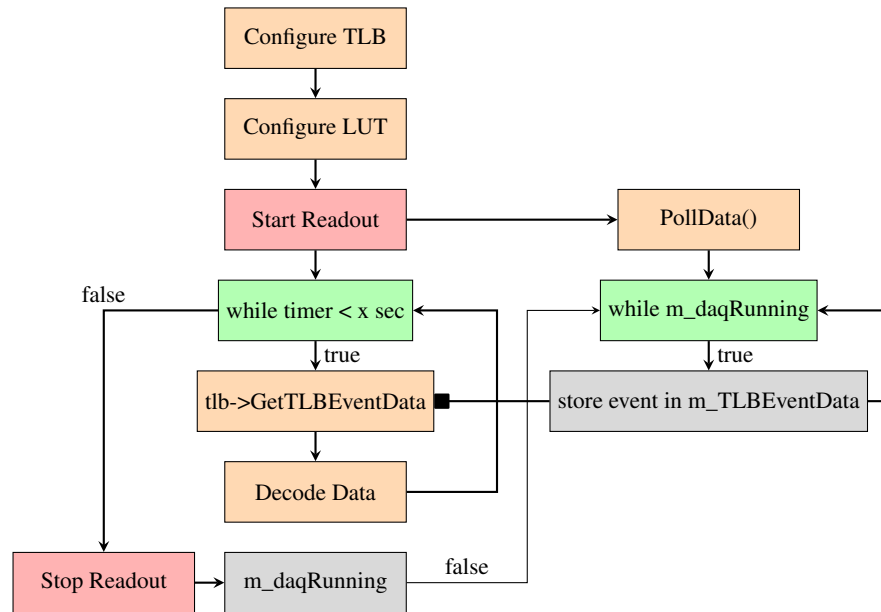


Fig. 5.7 TLBAccess example program diagram. [EliottTestProgram.cxx](#)

provided by DAQling that allows all individual hardware from the FASER experiment to be integrated together.

Chapter 6

Conclusion and Outlook

No presence of new physics in high p_T of pp collisions has led the search for light and weakly coupled particles with a characteristic long lifetime. These will be searched in the new ForwArd Search ExpeRiment (FASER), a small and inexpensive detector installed in the far forward region of pp collision. FASER will probe new regions of the parameter space that might confirm the existence of Beyond the Standard Model particles such as the axion or the dark photon. FASER has a small 5 m long footprint consisting of magnets, tracker planes, scintillators and a small calorimeter. A trigger and data acquisition system (TDAQ) provides the experiment with trigger and read-out capability using a Trigger Logic Board (TLB) as the main component.

Hardware commissioning and peer-reviewed software development of the TLB has been completed in the context of this master project. As a result, the TLB is now ready to be used for commissioning the FASER experiment. Integration tests with other components of the TDAQ system have taken place in a dedicated laboratory at CERN and further testing is in progress. An important milestone will be cosmic data taking with the TLB providing triggers to read out detector components upon the passage of cosmic rays; this will initially be done using scintillators and calorimeter modules and soon after, with tracker planes.

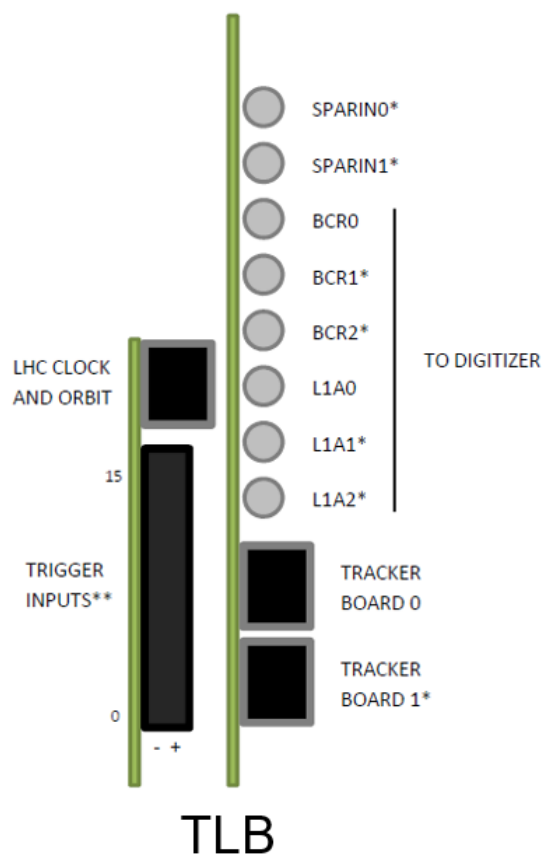
The TLB is an essential component for FASER's operation during Run 3. It can be modified to function in the future on the prospective bigger and more advanced FASER 2 experiment that will extend the sensitivity to include new particles produced in heavy meson decays.

References

- [1] “Dark photon,” Feb. 2019, page Version ID: 883393752. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Dark_photon&oldid=883393752
- [2] J. Kang and B. Nelson, “The Standard Model and Strings - Can They Be Connected?” *Physical Review D*, p. 47, Nov. 1978.
- [3] {FASER Collaboration}, A. Ariga, T. Ariga, J. Boyd, D. W. Casper, J. L. Feng, I. Galon, S.-C. Hsu, F. Kling, H. Otono, B. Petersen, O. Sato, A. M. Soffa, J. R. Swaney, and S. Trojanowski, “Letter of Intent for FASER: ForwArd Search ExpeRiment at the LHC,” *arXiv:1811.10243 [hep-ex, physics:hep-ph, physics:physics]*, Nov. 2018, arXiv: 1811.10243. [Online]. Available: <http://arxiv.org/abs/1811.10243>
- [4] {FASER Collaboration}, A. Ariga, T. Ariga, J. Boyd, F. Cadoux, D. W. Casper, Y. Favre, J. L. Feng, D. Ferrere, I. Galon, S. Gonzalez-Sevilla, S.-C. Hsu, G. Iacobucci, E. Kajomovitz, F. Kling, S. Kuehn, L. Levinson, H. Otono, B. Petersen, O. Sato, M. Schott, A. Sfyrla, J. Smolinsky, A. M. Soffa, Y. Takubo, E. Torrence, S. Trojanowski, and G. Zhang, “FASER’s Physics Reach for Long-Lived Particles,” *Physical Review D*, vol. 99, no. 9, p. 095011, May 2019, arXiv: 1811.12522. [Online]. Available: <http://arxiv.org/abs/1811.12522>
- [5] —, “FASER: ForwArd Search ExpeRiment at the LHC,” *arXiv:1901.04468 [hep-ex, physics:hep-ph, physics:physics]*, Jan. 2019, arXiv: 1901.04468. [Online]. Available: <http://arxiv.org/abs/1901.04468>
- [6] {FASER Collaboration}, A. Ariga, T. Ariga, J. Boyd, F. Cadoux, D. W. Casper, F. Cerutti, S. Danzeca, L. Dougherty, Y. Favre, J. L. Feng, D. Ferrere, J. Gall, I. Galon, S. Gonzalez-Sevilla, S.-C. Hsu, G. Iacobucci, E. Kajomovitz, F. Kling, S. Kuehn, M. Lamont, L. Levinson, H. Otono, J. Osborne, B. Petersen, O. Sato, M. Sabate-Gilarte, M. Schott, A. Sfyrla, J. Smolinsky, A. M. Soffa, Y. Takubo, P. Thonet, E. Torrence, S. Trojanowski, and G. Zhang, “Technical Proposal for FASER: ForwArd Search ExpeRiment at the LHC,” *arXiv:1812.09139 [hep-ex, physics:hep-ph, physics:physics]*, Dec. 2018, arXiv: 1812.09139. [Online]. Available: <http://arxiv.org/abs/1812.09139>
- [7] “Grafana documentation,” library Catalog: grafana.com. [Online]. Available: <https://grafana.com/docs/grafana/latest/>
- [8] “InfluxDB 1.8 documentation | InfluxData Documentation,” library Catalog: docs.influxdata.com. [Online]. Available: <https://docs.influxdata.com/>

Appendix A

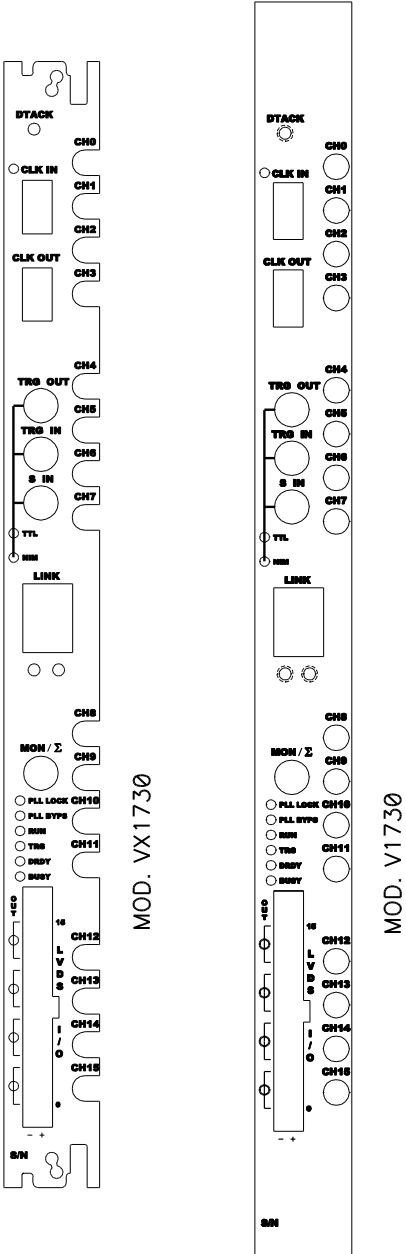
Instruments panels



* FOR FUTURE USE – NOT IMPLEMENTED IN FIRMWARE
** INPUT 8 TO 15 FOR FUTURE USE – NOT IMPLEMENTED IN FIRMWARE

Fig. A.1 TLB panel view.

Fig. A.2 Digitizer front panel view. VX1730, V1730.



Appendix B

TLB Data Configuration Format

B.1 TLB Configuration Variable Layout

Table B.1 Full Variable Layout TLB Configuration

Start Index	Next Index	Name	Bit Size	Min	Max
0	1	SamplingPhase	1	0	1
1	2	SamplingPhase	1	0	1
2	3	SamplingPhase	1	0	1
3	4	SamplingPhase	1	0	1
4	5	SamplingPhase	1	0	1
5	6	SamplingPhase	1	0	1
6	7	SamplingPhase	1	0	1
7	8	SamplingPhase	1	0	1
8	10	InputDelay	2	0	3
10	12	InputDelay	2	0	3
12	14	InputDelay	2	0	3
14	16	InputDelay	2	0	3
16	18	InputDelay	2	0	3
18	20	InputDelay	2	0	3
20	22	InputDelay	2	0	3
22	24	InputDelay	2	0	3
24	27	RandomTriggerRate	3	0	7
27	35	Prescale	8	0	255
35	43	Prescale	8	0	255
43	51	Prescale	8	0	255
51	59	Prescale	8	0	255
59	67	Prescale	8	0	255
67	75	Prescale	8	0	255
75	82	TrackerDelay	7	0	127
82	89	DigitizerDelay	7	0	127
89	90	LHC_Clock	1	0	1
90	102	OrbitDelay	12	0	4095
102	114	Deadtime	12	0	4095
114	134	MonitoringRate	20	0	1048575
134	166	OutputDestination	32	0	4294967295
166	167	Enable	1	0	1
167	168	Enable	1	0	1
168	169	Enable	1	0	1
169	170	Enable	1	0	1
170	171	Enable	1	0	1
171	172	Enable	1	0	1
172	173	Enable	1	0	1
173	174	Enable	1	0	1