ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών

«Φυσική και Τεχνολογικές Εφαρμογές»

Upgrade of the NSW and Basic Functions of the VMM3

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του Αθανάσιου Γιόκαρη

Επιβλέπων: Θεόδωρος Αλεξόπουλος

Αθήνα, Ιούλιος,2021

περίληψη

Η νέα αναβάθμιση μικρού τροχού του πειράματος ATLAS στο CERN, που σχεδιάζεται να πραγματοποιηθεί το 2020, απαιτεί μια νέα γενιά ηλεκτρονικών front-end που θα υποστηρίζει τις απαιτήσεις απόκτησης δεδομένων. Το Ολοκληρωμένο Κύκλωμα Ειδικής Εφαρμογής VMM βρίσκεται σε εξέλιξη τα τελευταία επτά χρόνια για να χρησιμεύσει ως το θεμέλιο του σχεδίου ανάγνωσης του New Small Wheel. Έχει περάσει από τρεις σημαντικές αναθεωρήσεις και μια μικρή, η τελευταία είναι η έκδοση παραγωγής. Για τη διευκόλυνση της δοκιμής και της ανάγνωσης του VMM, καθώς και για τη μελέτη της απόδοσης του ανιχνευτή του, έχει αναπτυχθεί ένα πλήρες σύστημα ανάγνωσης. Αποτελείται από ευέλικτη λογική Field-Programmable Gate Array με εκτεταμένη λειτουργικότητα και ένα αποτελεσματικό πλαίσιο λογισμικού που παρέχει το περιβάλλον εργασίας χρήστη. Αυτό το σύστημα, που αναφέρεται ως "Το σύστημα ανάγνωσης VMM", έχει χρησιμοποιηθεί σε εκστρατείες δοκιμών δέσμης στο CERN, καθώς και σε σενάρια βαθμονόμησης πάγκων και δοκιμών για τα τελευταία χρόνια, υποστηρίζοντας όλους τους τρόπους ανάγνωσης και τα χαρακτηριστικά του VMM .

A B S T R A C T

The New Small Wheel Upgrade of the ATLAS experiment at CERN, planned to take place at 2020, requires a new generation of front-end electronics that will support its data acquisition requirements. The VMM Application-Specific Integrated Circuit has been in development for the last seven years to serve as the foundation of the New Small Wheel's readout scheme. It has gone through three major revisions and a minor one, the latter being the production version. To facilitate the testing and readout of the VMM, as well as to study its detector performance, a complete readout system has been developed. It consists of flexible Field-Programmable Gate Array logic with extensive functionality, and an efficient software framework providing the user interface. This system, referred to as the "The VMM Readout System", has been used in test-beam campaigns at CERN, as well as in bench calibration and testing measurement scenarios for the past several years, supporting all readout modes and features of the VMM.

Contents

CHAPTER 1

1.1 LHC (large Hadron Collider)

The Large Hadron Collider (LHC) is by far the most powerful particle accelerator built to date. Following an upgrade, the LHC now operates at an energy that is 7 times higher than any previous machine! The LHC is based at the European particle physics laboratory CERN, near Geneva in Switzerland. CERN is the world's largest laboratory and is dedicated to the pursuit of fundamental science.

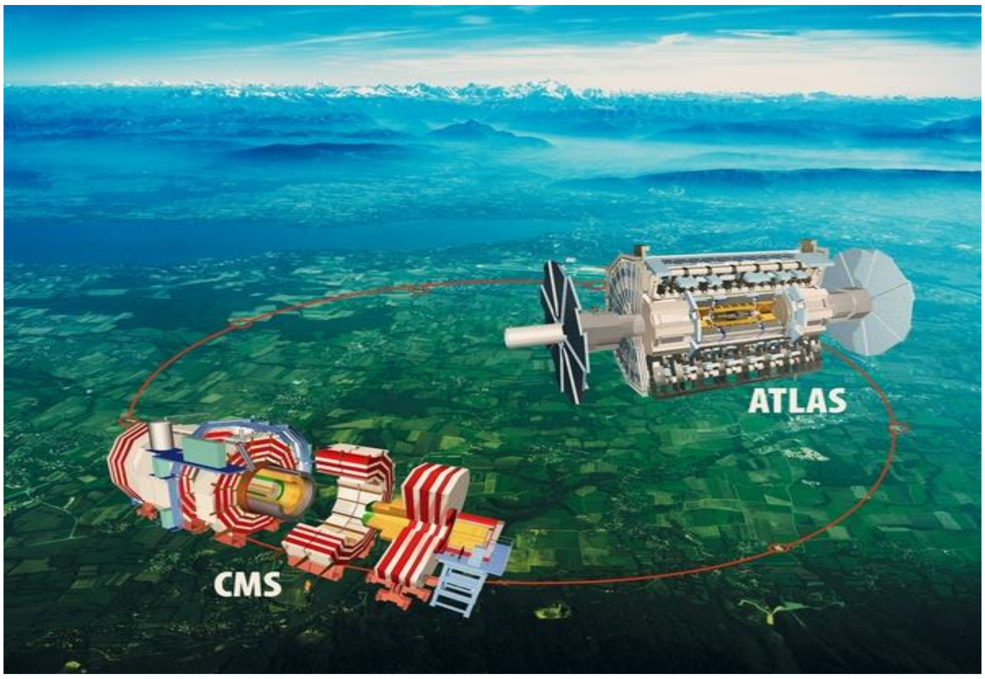


Figure 1.1: A bird's eye view of the LHC (Credit: CERN)

The LHC [5] allows scientists to reproduce the conditions that existed within a billionth of a second after the Big Bang by colliding beams of high-energy protons or ions at colossal speeds, close to the speed of light. This was the moment, around 13.7 billion years ago, when the Universe is believed to have started with an explosion of energy and matter. During these first moments all the particles and forces that shape our Universe came into existence, defining what we now see.

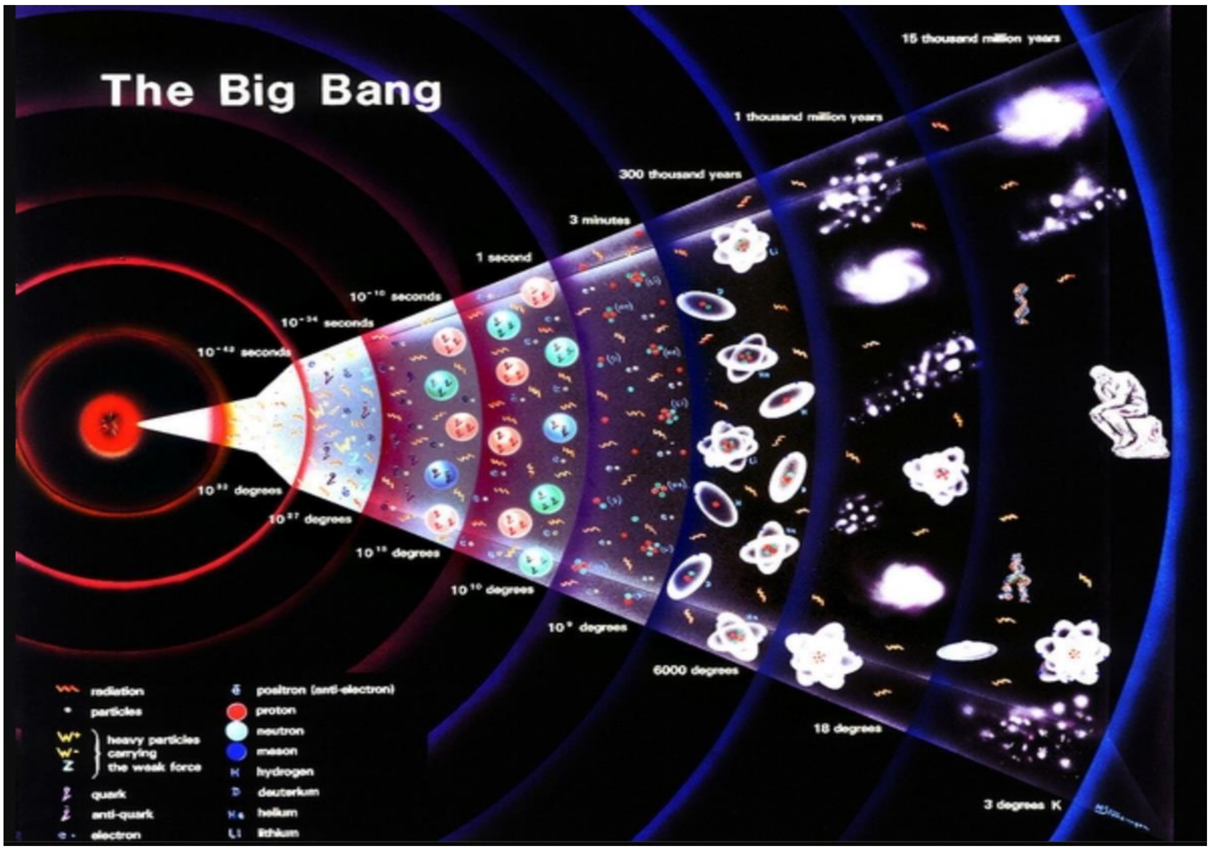


Figure 1.2 : The big Bang

The LHC is exactly what its name suggests - a large collider of hadrons (any particle made up of quarks). Strictly, LHC refers to the collider, a machine that deserves to be labelled ‘large’, it not only weighs more than 38,000 tonnes, but runs for 27km (16.5mi) in a circular tunnel 100 metres beneath the ground. Particles are propelled in two beams going around the LHC to speeds of 11,000 circuits per seconds, guided by massive superconducting magnets! These two beams are then made to cross paths and some of the particles smash head on into one another.

However, the collider is only one of three essential parts of the LHC project. The other two are:

1.2 The Detectors

Each of the four main detectors sit in huge chambers around the LHC ring to detect the outcomes of the particles colliding. ATLAS, ALICE, CMS and LHCb.

1.3 Worldwide LHC Computing Grid (WLCG)

A global network of computers and software, that is essential to processing the masses of data recorded by all of the LHC's detectors.
The LHC is truly global in scope because the LHC project is supported by an enormous international community of scientists and engineers. Working in multinational teams all over the world, they are building and testing equipment and software, participating in experiments and analysing data. Greece has a significant role in the project and has scientists and engineers working on all the main experiments.
The LHC project involved 111 nations in designing, building and testing equipment and software, and now continues with them participating in experiments and analysing data. The degree of involvement varies between countries, with some able to contribute more financial and human resource than others.

1.4 NSW (New small wheel)

The ATLAS detector is one of the largest [7] detectors built for particle experiments. The detector features a cylindrical shape allowing particle reconstruction in a 4π enclosure. It is possible to determine the momentum of a charged particle through reconstruction of its track in the detector. Parts of the muon trackers of the ATLAS detector, are consisting of Monitored Drift Tubes (MDTs) with a single tube spatial resolution of 80 µm, Cathode Strip Chambers (CSCs) and for the trigger signals Resistive-Plate Chambers (RPCs) and small-strip Thin Gap Chambers (sTGCs), realized in two small wheels and two big wheels [ATLAS Collaboration, 2008].

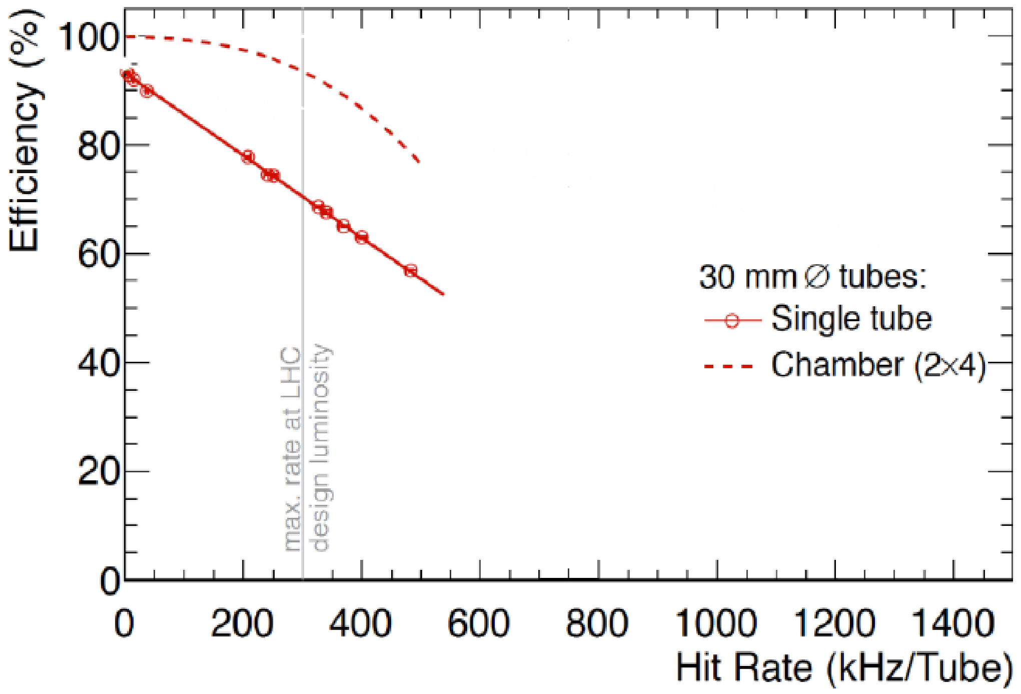


Figure 1.3: Efficiencies of the Monitored Drift Tubes as a function of the hit rate. The points are measured data from a single tube, while the dotted line represents extrapolated efficiencies for a chamber with two times four tube layers. Figure taken from ATLAS collaboration 2013

For the design luminosity (at a hit rate of 300 kHz/tube) of the LHC the MDT detector's efficiency is above 90% and drops significantly for even higher frequencies. The reasons for these inefficiencies are ions that are created next to the central wire of the tube. These ions have to drift to the wall of the MDT and thus influence the electric field close to the wire for the electron drift velocity and the avalanche process. This leads to a wrong signal reconstruction especially at high rates which are expected after the high luminosity LHC upgrade.
The upgrade for the ATLAS detector during the long shutdown from 2019 to 2020 is called the phase 1 upgrade. It includes the exchange of the small wheel detectors as well as the exchange of the readout chain and front-end electronics of the small wheel. To operate the New Small Wheel (NSW) efficiently, a high rate stable front-end chip was specifically designed for the new detectors to cope with the challenge of an increased collision frequency. The MDTs and CSCs are going to be replaced by sTGC and Micromegas [ATLAS collaboration, 2013], which are going to be used for the trigger system and for high precision track reconstruction. The Micromegas detectors for the NSW have trapezoidal shapes varying with the position in

the small wheel and are sandwiched by the sTGC detectors (see fig 1.4). These Specific Micromegas allow for readout in two dimensions: The η-coordinate of the ATLAS detector, which can be reconstructed with a precision of around 70 _m for perpendicularly incident particles and the φ-coordinate with a precision in the order of mm.

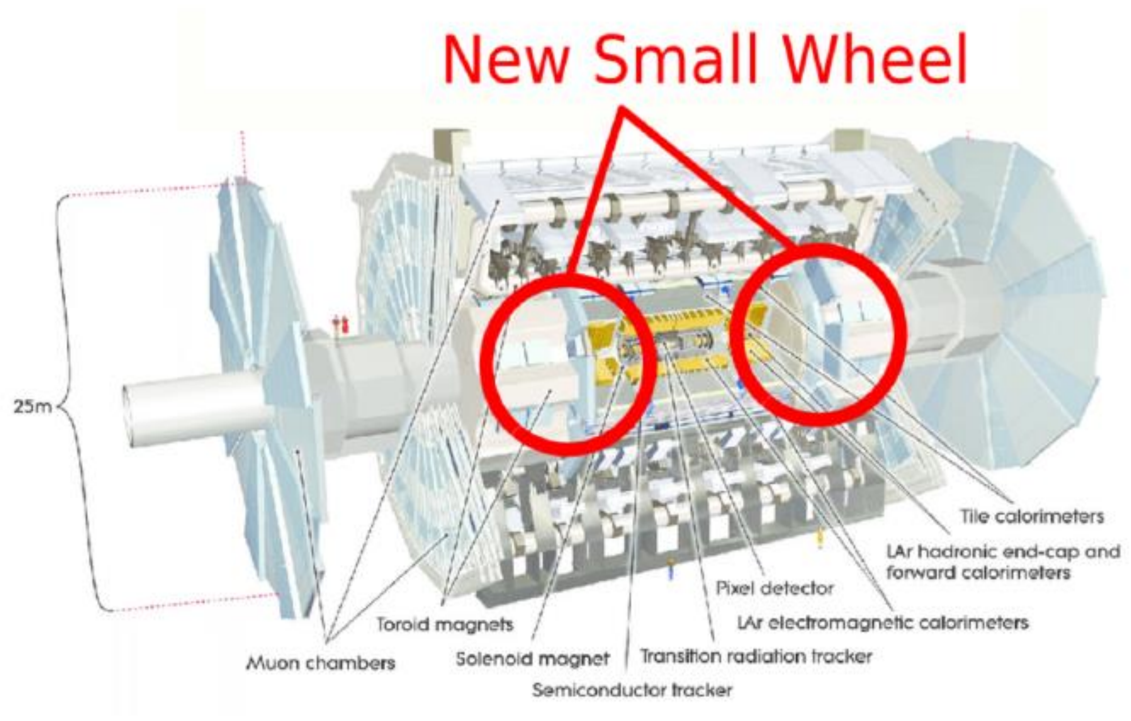


Figure 1.4: The ATLAS detector in a cross-sectional view. The detector itself consists of the inner barrel with pixel detectors, semiconductor trackers and transition radiation trackers located around the interaction point. The hadronic and electromagnetic calorimeters surrounding the inner barrel follow with scintillating material and absorbers. The TGCs and RPCs are used for the trigger system in the muon spectrometer, whereas the MDTs and CSCs are used for online track reconstruction. These detectors are going to be replaced by the NSW with Micromegas and sTGCs technology. Figure taken from [ATLAS Collaboration, 2008].

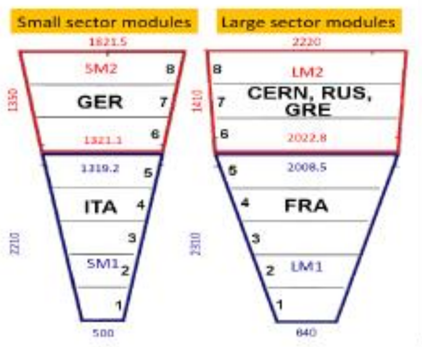


Figure 1.5 : The Micromegas small and large sectors are built out of two smaller trapezoidal detectors. The sizes are given in mm and the countries responsible for the construction are indicated.

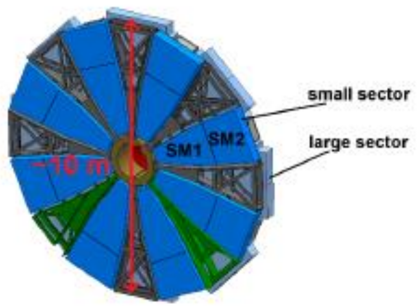


Figure 1.6 The NSW muon spectrometer consisting of small and large detectors sectors. These sectors consist of Micr-omegas layers sandwiched by sTGCs.

The motivation [4] for the luminosity upgrade of the Large Hadron Collider (LHC) is to precisely study the Higgs sector and to extend the sensitivity to new physics to the multi-TeV range. In order to achieve these goals the ATLAS experiment has to maintain the capability to trigger on moderate momentum leptons under background conditions much harder than those currently present at the LHC. For the Muon Spectrometer (MS) such requirements necessitate the replacement of the forward muon-tracking region (called the muon Small Wheel) with new detectors capable of precision tracking and triggering simultaneously. The New Small Wheel (NSW) upgrade [3] is designed to cope with the high background rate that is expected at L = 2-5×10^{34} $cm^{-2}s^{-1}$ during Run-3 and the high luminosity
LHC (HL-LHC). Figure 1 shows the z-y view of one quarter of the ATLAS detector, the three stations of the MS and the location of the Small Wheel.

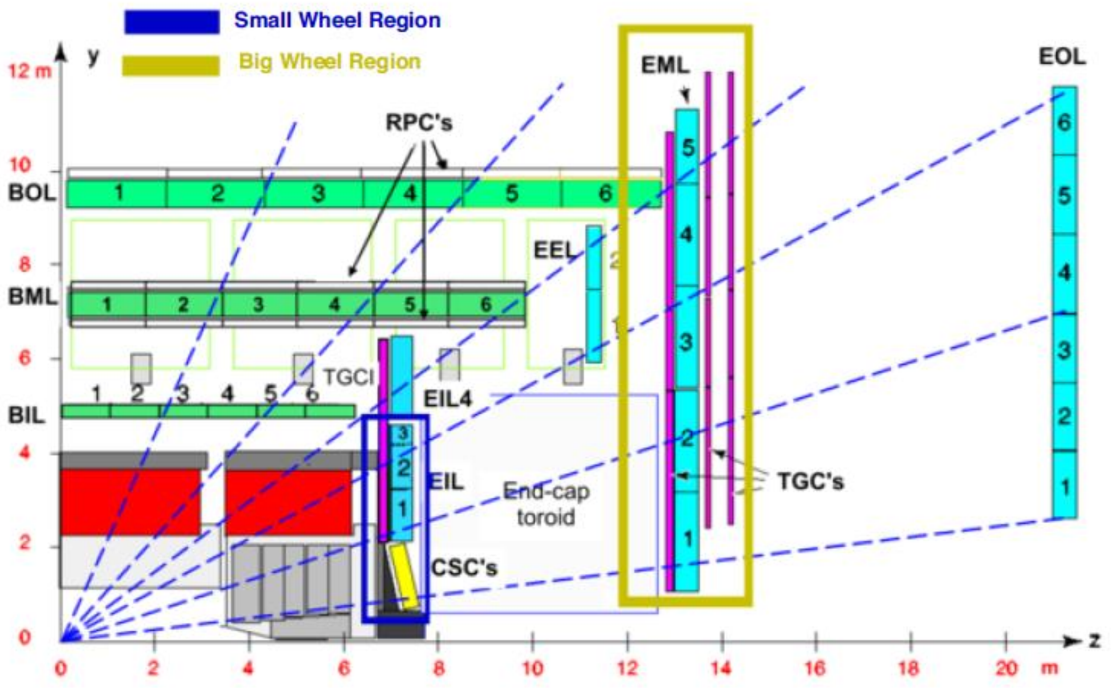


Figure 1.7 : A z-y view of 1/4 of the ATLAS detector. It shows the three stations of the Muon Spectrometer in the forward region: the End-cap Inner Large (EIL), End-cap Middle Large (EML) and End-cap Outer Large (EOL). The detector regions of the so called Small Wheel (blue rectangular) and the muon Big Wheel is shown (yellow rectangular).

The rate of the ATLAS muon trigger increases proportional to the instantaneous luminosity. Muon triggers have currently a very high fake rate (about 90%) due to low energy particles generated in the materials between the Small and the Big Wheel, entering the trigger chambers of the Big Wheel. Simply raising the muon trigger p_T thresholds from 20 GeV to 40 Gev would maintain the trigger rate at the current level but would result in a significant loss in physics, especially for signatures with a relatively soft single leptons relevant for Higgs boson analyses (e.g. $WH \rightarrow l\nu bb, H \rightarrow \tau_{lep}\tau_{had}$) as shown in Fig. 1.8 (left). For high p_T muons, the resolution is dominated by the muon spectrometer. Requiring segments in all three muon stations ensures the best momentum resolution, important for searches for new physics at the highest energies. However, the current SW chambers, based on monitoring drift tubes, can not cope with rates up to 15
kHz/cm^2. Losing the track segments in the SW means losing high quality muon candidates and results in reduced efficiency as shown in Fig. 1.8(right).

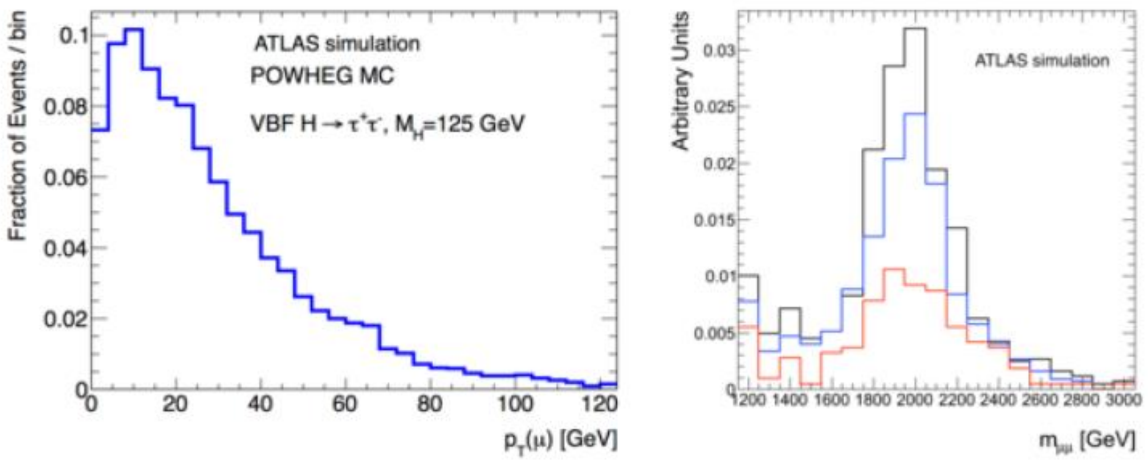


Figure 1.8: (Left) Distribution of transverse momenta of leptons from $H \rightarrow \tau\tau$ decays where one of the τ decays into a muon and two neutrinos. (Right) Reconstructed dimuon mass in simulated $Z\_ \rightarrow \mu\mu$ events with three different levels of background realized by a data overlay technique. The black, blue and red histograms correspond to luminosity of 0.3, 3 and 5×1034 $cm^{-2}s^{-1}$ respectively

The increased muon trigger rates and the degraded muon track reconstruction performance, a clear indication of the origin of the triggered muons is necessary. This will be provided by replacing the SW with the New Small Wheel (NSW). NSW is a set of precision tracking detectors that are fast, capable to perform bunch crossing identification at rates up to 15 kHz/cm^2 and have spacial resolution of less than 100 μm per detection plane. The NSW detectors can therefore provide the muon trigger system with reconstructed track segments of good angular resolution (< 1 mrad) that can clearly indicate whether the triggered muons originated from the collision point or not as shown in Figure 3. The existing ATLAS Big Wheel trigger accepts all three tracks shown. The fake tracks (B and C) can be rejected in the trigger by the addition of the New Small Wheel. At the same time, the singlemuon p_T threshold can be kept at the current low levels and be comfortably managed by the subsequent muon trigger stages.

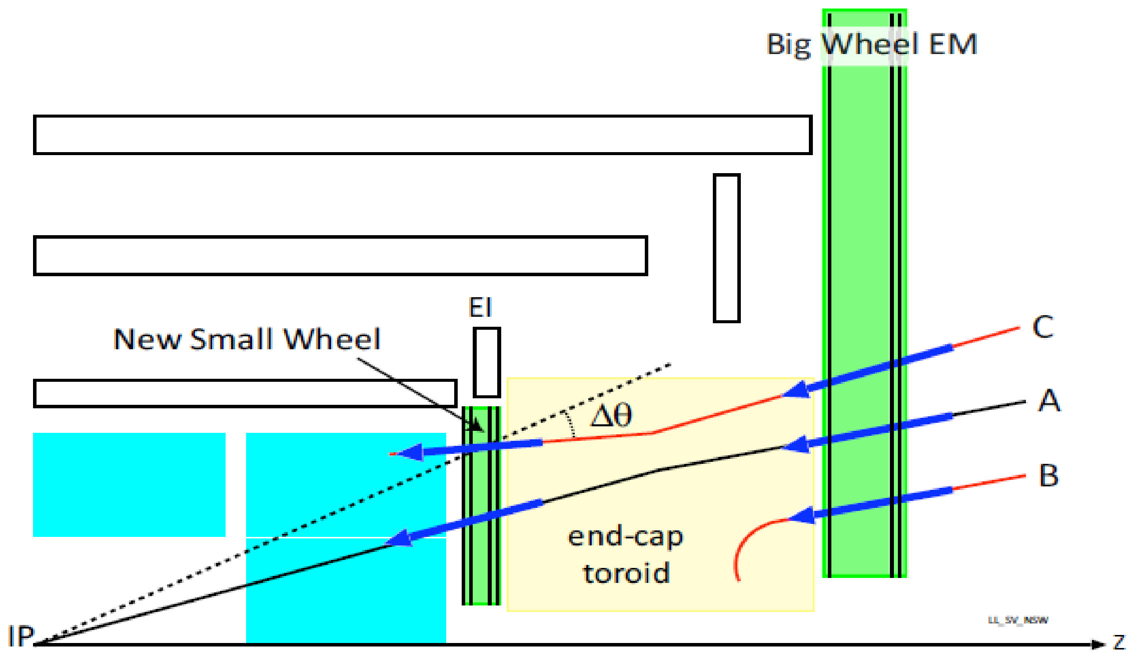


Figure 1.9: Schematic of the ATLAS muon trigger. The existing BigWheel trigger accepts all three tracks shown. The fake tracks (B and C) can be rejected in the trigger by the addition of the New Small

1.5 Detector Layout

The NSW will utilize two detectors, the small-strip Thin Gap Chambers (sTGC) as the primary trigger and Micromegas (MM) as the primary precision tracker. The NSW consists of 16 detector planes arranged in two multilayers.

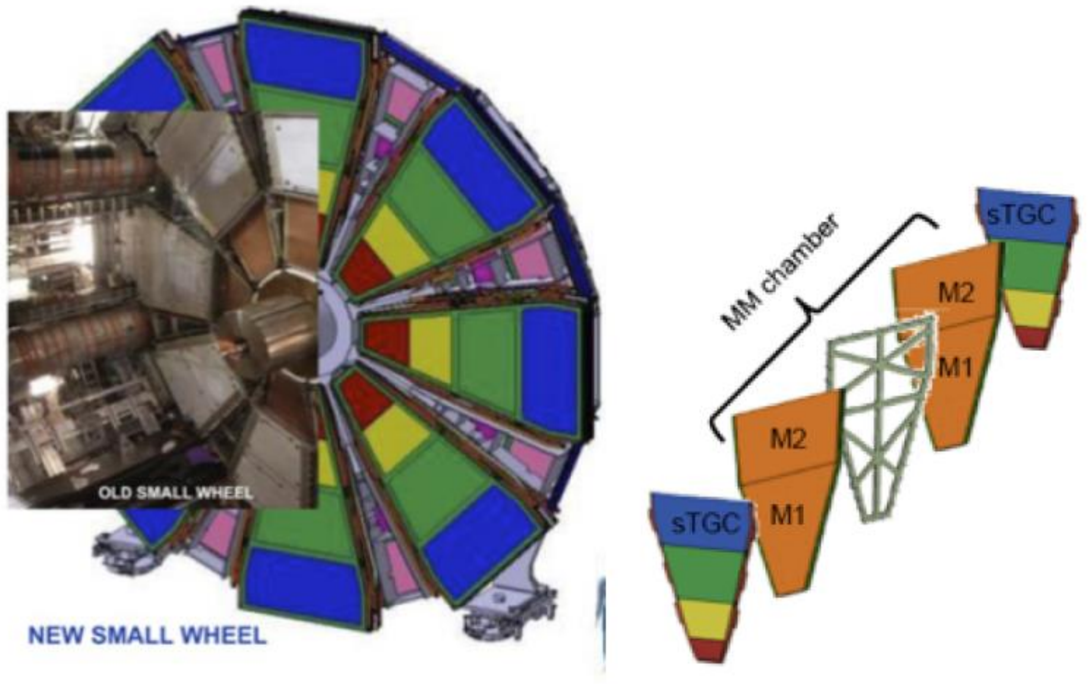


Figure 1.9 : Left: Schematic of the 16 sectors of the NSW together with a partial picture of the current SW and the central cylindrical brass plug. Right: Schematic of the arrangement of the four multilayers of sTGC and MM detectors including support structure.

Each multilayer comprises four sTGC and four MM detector planes. A sandwich arrangement of sTGC-MM-MM-sTGC is used to maximize the distance between the two sTGCs multilayers for improved track segment angular resolution at the trigger level as shown in Fig. 4 (right). The choice of eight planes per detector was dictated by the need to provide a robust, fully functional detector system over its whole lifetime. Following the present structure of the MS, each wheel is composed of 16 sectors (8 small and 8 large) as shown in Fig. (left).

1.6 MicroMegas Detectors

The "Micromegas [2] (Micro-MEsh Gaseous Structure) detector is a gaseous particle detector coming from the development of wire chamber. The Micromegas detectors are mainly used in experimental physics, in particular in particle physics, nuclear physics and astrophysics for the detection of ionising particles. The Micromegas are light detectors in order to minimize the perturbation on the impinging particle. From their small amplification gap, they have fast signals in the order of 100 nanoseconds They are a type of micropattern gaseous detector with a spatial resolution below one hundred micrometers Nowadays, the use of the Micromegas technology is growing over the different fields of experimental physics. It consists of two asymmetric regions, the amplification region (128 μm,40 - 45 kV/cm) and the Conversion/Drift region (5mm, 0.6 kV/cm).For the NSW micromegas detectors, the two regions are filled with a gas mixture of Ar+ 7%CO2 and are separated by a thin metallic mesh called micromesh. The readout strips of the detector are covered by an insulator and a layer of resistive strips to protect the detector from discharges. Charged particles ionize the gas in the drift area and the electron produced drift toward the mesh. The electrons that pass through the mesh, enter the amplification region where the avalanche effect happens. The electrons now have enough energy to produce new ion-electron pairs that will also ionize the gas

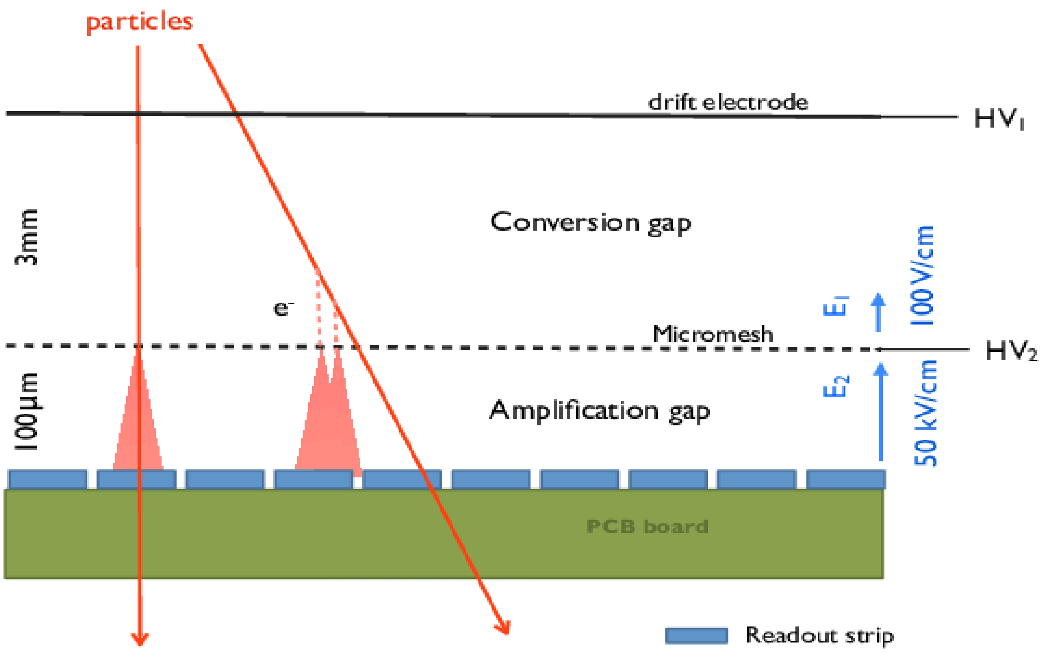
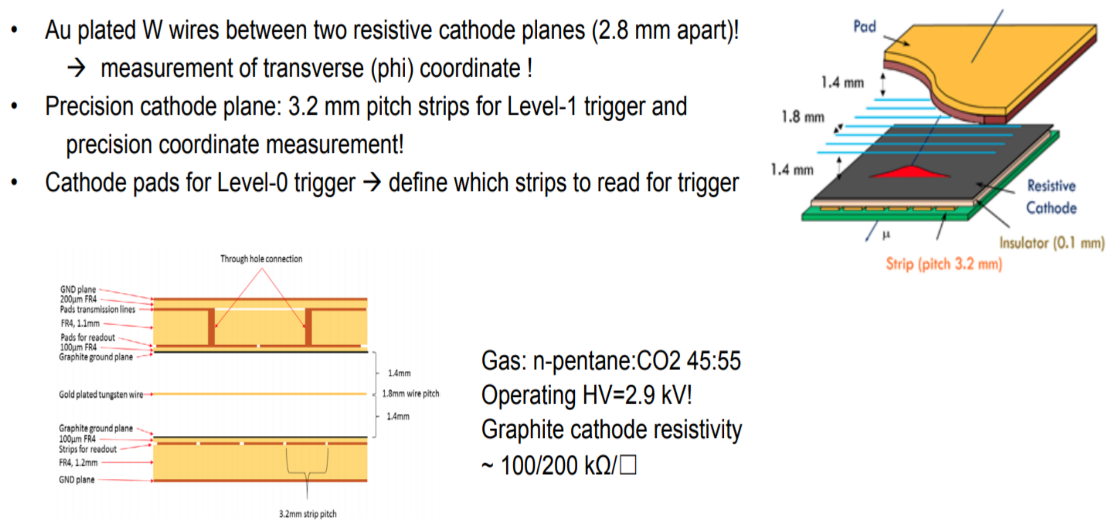


Figure 1.10 : Operation principle of the Micromegas detector

1.7 The sTGC detectors

They [1] are gaseous detectors and are made of and consist of gold plated tungsten wires and strips and pads both from copper. The wires have diameter of 50 µm and a potential of 2.6Kv is applied. They have a pitch of 1.8mm and are placed between two graphite epoxy cathode planes each at a distance of 1.4mm from the wires. The planes of the cathode are made of graphite-epoxy-mixture and have resistivity of 100-200 kΩ/sq. The precision strips on the outer side of the cathodes and the pads, acting as readout electrodes. The strips have 3.2 mm pitch and the pads have a much larger pitch 80 mm. They are used to identify muon tracks at the interaction point and to define a region of interest for which wires are read out. The chamber is filled with a gas mixture of 55% CO2 and 45% n-pentane which has an electron drift velocity of about 3 cm/mus under an electric field of 2.9 kV/cm.



- Au plated W wires between two resistive cathode planes (2.8 mm apart)!
  → measurement of transverse (phi) coordinate !
- Precision cathode plane: 3.2 mm pitch strips for Level-1 trigger and precision coordinate measurement!
- Cathode pads for Level-0 trigger → define which strips to read for trigger

Gas: n-pentane:CO2 45:55
Operating HV=2.9 kV!
Graphite cathode resistivity
~ 100/200 kΩ/□

Figure 1.11 : structure of STGC detectors

They [1] are three different types of active elements on a detector: strips, wires and pads. All the three are read out via the VMM. Strips provide the precision coordinate measurement for track reconstruction, wires for the second coordinate and pads are used for triggering purposes requiring a 3 out of 4 coincidence between the signals of pads in consecutive layers. The wire signals are negative while both the strip and pads are positive, so the VMM has to handle both polarities. Below we have a typical, simulated waveform of an sTGC detector.

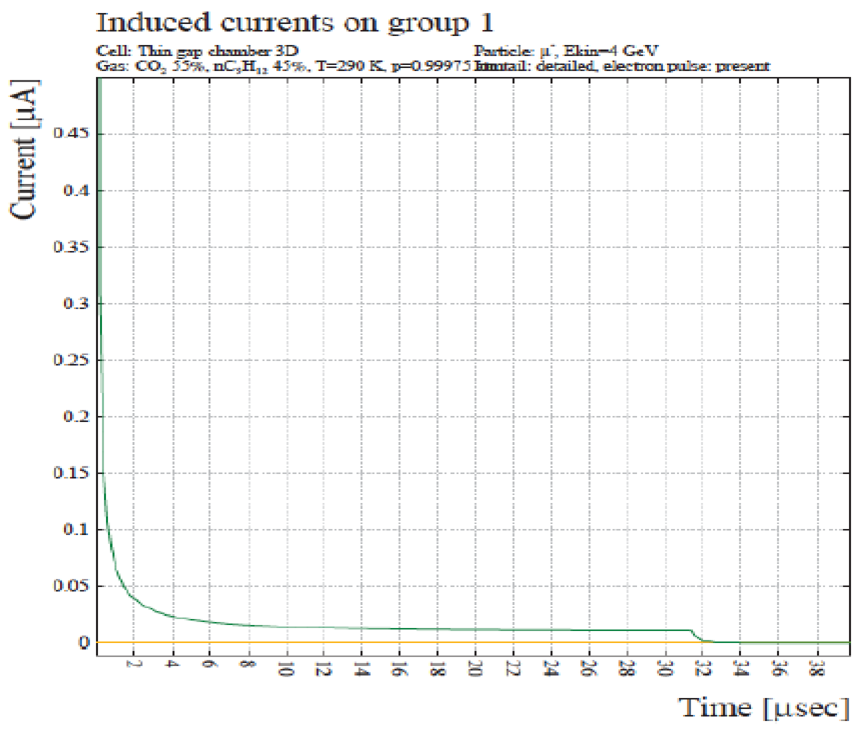


Figure 1.12 : Simulated current induced by the motion of electrons and positive ions in an sTGC detector.

Chapter 2

2.1 VMM (VENETIS MICRO MEGA)

The VMM [3] is a custom Application Specific Integrated Circuit (ASIC). It is used in the front end readout electronics of both the Micromegas and sTGC detectors of the New Small Wheels Phase I upgrade project. It is being developed at Brookhaven National Laboratory by Gianluigi de Geronimo and his microelectronics design group. It is fabricated in the 130 nm Global Foundries 8RF-DM process (former IBM 8RF-DM). The 64 channels with highly configurable parameters will meet the processing needs of signals from all sources of both detector types

1) Negative anode strip signals from the Micromegas detectors.
2) Negative wire-group signals from the sTGC detectors.
3) Positive cathode strip signals from the sTGC for precision spatial reconstruction.
4) Positive cathode pad signals used in the sTGC trigger.

The VMM has four independent data output paths:

1) Precision (10-bit) amplitude and (effective) 20-bit time stamp read out at Level 0 accept.

2) A serial out Address in Real Time (ART) synchronized to a 160MHz clock which is used for the Micromegas Trigger

3) Parallel prompt outputs from all 64 channels in a variety of selectable formats (including a 6-bit ADC) for the sTGC trigger.

4) Multiplexed analog amplitude and timing outputs will not be used in the NSW but can be valuable in development and debugging.

The third version of VMM3 has all the design features including Level-0 buffer logic and seu mitigation circuitry for the configuration registers, the state machines and the FIFO pointers. The device will be packaged in a Ball Grid Array with outline dimensions of consistent with the Micromegas strip pitch

The VMM has 64 linear front-end channels. In figure 1 a block diagram shows us how a channel looks like. Each channel is integrated with a low-noise charge amplifier (CA), adaptive feedback, test capacitor and an adjustable polarity so we can process positive or negative charge. We use an input mosfet as a p-channel with gate area and a drain current ID = 2 mA. This corresponds to an inversion coefficient IC=0.22 a transconductor g_m=50 mS and a gate capacitance C_g . The filter (shaper) is a third-order (one real pole and two complex conjugate poles) designed in delayed dissipative feedback (DDF).The DDF architecture offers higher analog dynamic range, making possible a relatively high resolution at input capacitance much smaller than 200 pF. Next to the shapers are sub-hysteresis discriminators with neighbour enabling logic and individual threshold trimming, the peak detector and the time detector.
The discriminator which has a global as well as a per channel adjustable threshold will forward a signal that is higher than a certain threshold. The neighbour logic then allows to read out the channels adjacent to the hit one even though it did not cross the channel specific threshold. This is also communicated across different chips via a bidirectional signal. The output is then sent to the ART (Address in Real Time) Data Driver Card which receives the hit VMM channels as well as their timing and sends the data to the Micromegas trigger processor. The peak finder detects the peak of a detector signal or test pulse (PDO) and stores it first into analog memory. The linear

voltage ramp of the time-to-amplitude converter (TAC) is then activated after crossing a or at the time of the peak. The TAC is stopped at the next bunch crossing clock, which counts in 25 ns bins see below figure

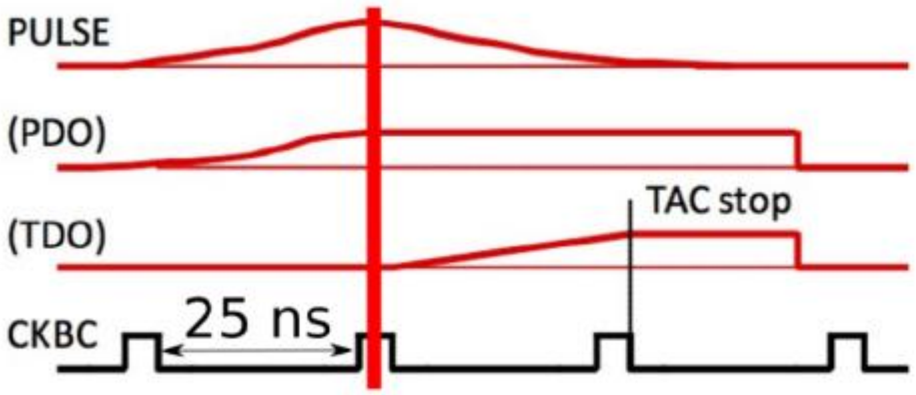


Figure 2.1: Sketch for the working principle of the peak detector output (PDO) and time detector output (TDO). The peak _nder of a VMM channel scans a test pulse or detector signal for its peak. As soon as the peak was found, the TDO is linearly ramped up. The ramp up is stopped at the next bunch crossing clock (CKBC) and saved as the TDO. The CKBC always counts in 25 ns steps.

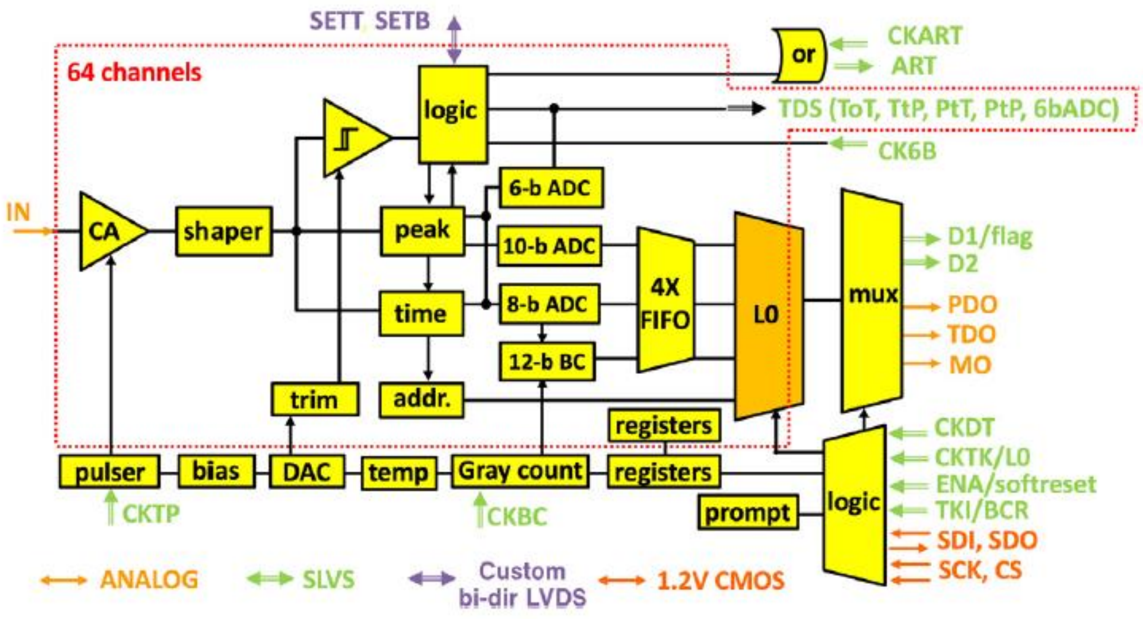


Figure 2.2: VMM3 Architecture

The ASIC can operate in either a two phase analog mode (not used in NSW) or a continuous, simultaneous read/write mode. In the two phase mode data are registered while the VMM is in acquisition mode and then read out, after the system is switched to the read out mode. Acquisition is re-enabled after the readout phase is completed. In continuous mode the simultaneous read/write of data assures dead-timeless operation that can handle rates up to the maximum 0f 4 MHz per channel. The ASIC has four independent output data paths:

1. Multiplexed analog amplitude and timing.

2. Digitized (10-bit amplitude, 20-bit vernier time stamp) in a 2-bit (DDR readout) digital multiplexed mode in either a short four-word buffer (existing already in VMM2) or with a deeper buffer sufficient for the expected Level-0 latency with the associated control logic

3. Address in Real Time (ART) used in the Micromegas trigger schema.

4. Direct SLVS-400 outputs of all 64 channels in parallel in one of five selectable formats, used in the sTGC trigger.

The overall connection scheme of these data paths with the rest of the NSW readout and trigger components are shown in the below figure

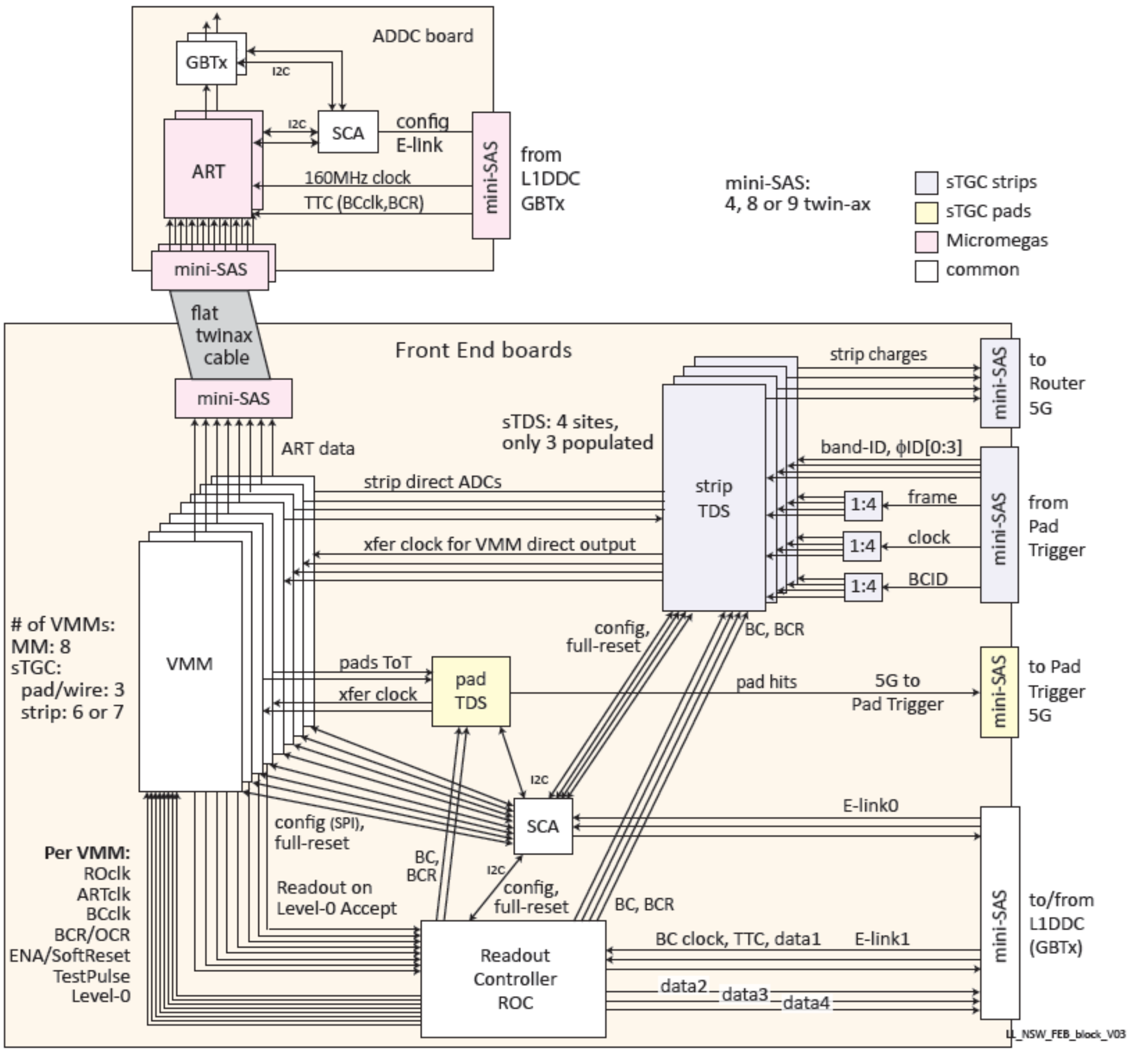


Figure 2.3 : Overall connection Diagram of the VMM

2.2 Readout modes

The VMM has three modes of operation: a two-phase analog mode, a continuous simultaneous read/write mode, and the Level-0 mode. Also, the chip provides trigger primitives via its fast outputs.

2.3 Two-phase analog mode

In the two-phase mode, data are registered while the VMM is in acquisition mode and then are read-out once the system is switched to the readout mode. Acquisition is re-enabled after the data extraction
phase is completed. The readout advances to the next channel by injecting a token to the relevant input. The token is sparse, passed only among those channels with valid events. Once the procedure is complete, the token is

routed to the output for reading-out the next chip, thus allowing for a daisy-chained readout with a single token input. In this mode, the internal ADCs are switched off and the analog signals are read-out through analog buffers with external ADCs.

2.4 Continuous mode

In continuous mode, the simultaneous read/write of data assures dead-timeless operation that can handle rates up to the maximum of 4MHz per channel.[2] Higher rates can be achieved by interrupting the 10-bit ADC once the 6-bit ADC has finished, since the 6-bit ADC conversion takes up ~40 ns including the channel reset. The peak and time detectors convert the voltages into currents that are routed into the 10 and 8-bit ADC, respectively. The channel is reset, once both conversions are completed and the digital values are latched in digital memories. The self-reset of each channel provides continuous and independent operation of all 64 channels. In this mode, each channel of the ASIC provides the 38-bit data stored in a four-event deep de-randomizing FIFO. The data are read-out via two serial data lines (D1 and D2 in Fig.

2.5 Level-0 mode

The Level-0 (L0) readout mode of the VMM was designed to be compatible with the readout scheme of the ATLAS experiment. It provides the ability to buffer the channel hit data in a deeper memory architecture and allows the VMM to select the data to be read-out after the reception of an external trigger input, namely the L0 signal. The L0 selection logic applies a configurable latency to the trigger and forwards out only hit data for which their timestamp is within a window of the trigger's time-of-arrival plus its latency offset. A block diagram of the logic is depicted in Fig. 2.4, where data from the VMM channel(s) are forwarded from left to right through various buffers. The first FIFO is a 64-deep latency buffer (LAT. FIFO in Fig. 2.4) where all hits can be buffered. Each channel has a selector circuit based on the arrival of the external trigger signal which finds events within the configurable window (L0 SEL in Fig. 2.4). Once there is a match, the data are copied to the channel FIFO (L0 CH FIFO in Fig. 2.4) and are ready to be read-out in a round-robin manner, starting with the information
of the BC timestamp. The data are forwarded out of the ASIC via two serial lines, synchronous to both edges of an externally-provided 160MHz data clock, resulting in a total bandwidth of 640 Mb⁄s. The data are encoded using the 8b/10b protocol [5], leading to an effective bandwidth of 512 Mb⁄s.

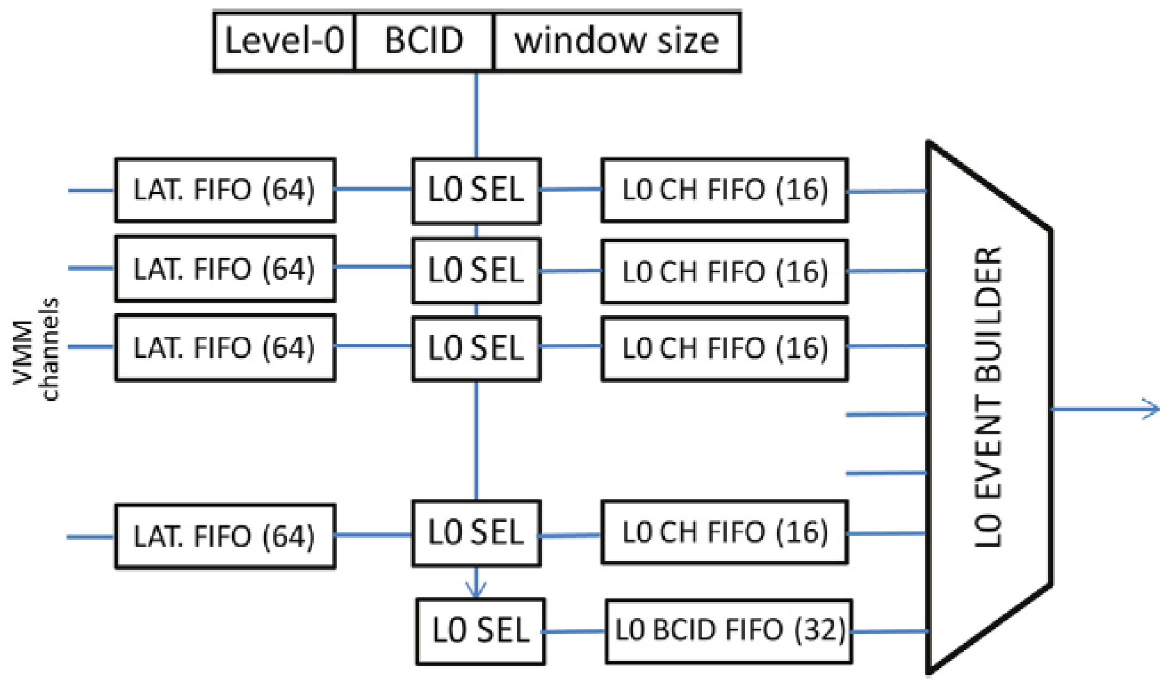


Figure 2.4 : L0 buffer and event selection logic block diagram. The trigger signal, or L_0,gets time stamped with a Bunch Crossing ID (BCID) value, then a configurable offset is applied to it. The hits stored in each 64-event-deep latency FIFO (LAT. FIFO) have a timestamp themselves as well. The L0 selection logic (L0 SEL) checks the hit timestamps against the ones of the trigger. Only hits within a window of configurable width enter the L0 CH FIFO before being sent out of the ASIC by the L0 Event Builder.

2.6 Fast outputs

The VMM has several fast serial outputs that can be used for different applications:

• The Address in Real Time (ART): This is the address (i.e. VMM channel number) of the first in time channel above-threshold. Depending on the chosen configuration parameters on the VMM, the total latency of the streamed ART address can be within ~15 ns, or ~20 ns plus the peaking time. This latency is the sum of several delays, some of which are the 5 ns delay between the peak and the peak-found signal, and the 5 ns digital latency from the comparator firing to the leading edge of the ART. The 6-bit address with an accompanying flag are transmitted serially, at 320 Mbps data rate, on every clock cycle of the BC clock.

• Direct Data Outputs (DDO): The VMM provides fast information via its 64 direct timing outputs, one for each of its channels. Each channel implements a fast digitization 6-bit ADC, that provides a coarse, but rapid amplitude measurement of the pulse. The conversion starts immediately when the peak-found signal is issued and the data are streamed

at each output, synchronized with an externally-provided 160MHz clock. The DDOs can also operate in four other modes: time-over-threshold (ToT), threshold-to-peak (TtP), peak-to-threshold (PtT), or a 10 ns pulse occurring at peak
(PtP).

2.7 The GPVMM board

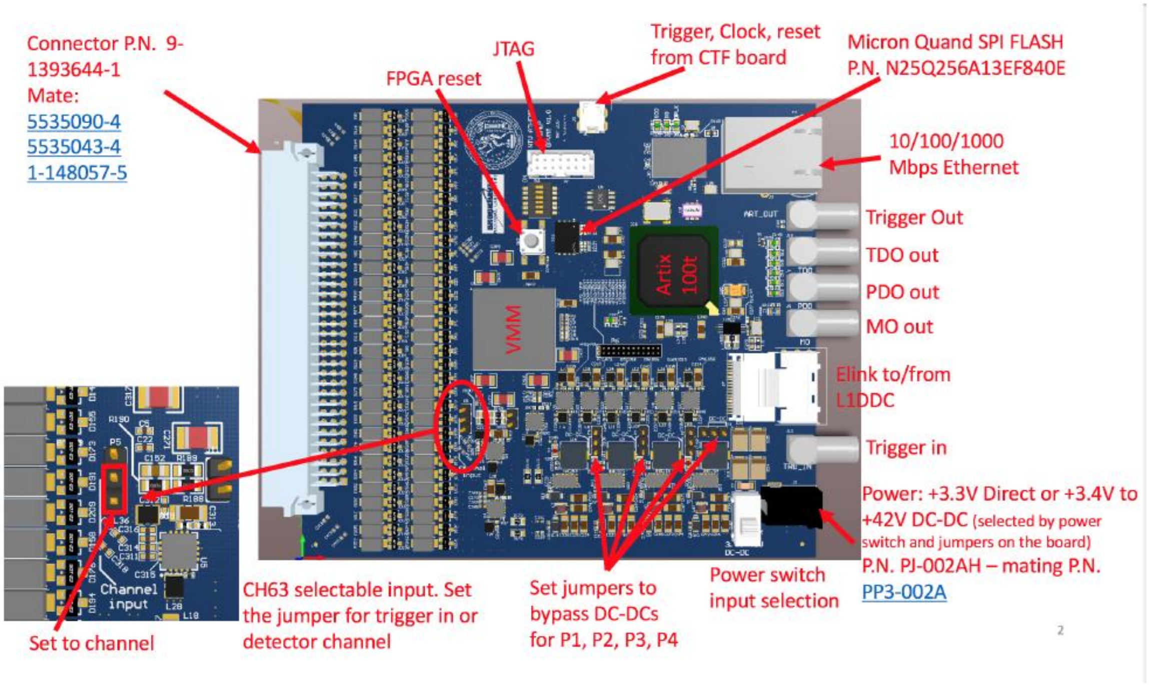

Connector P.N. 9-1393644-1
Mate:
5535090-4
5535043-4
1-148057-5
JTAG
FPGA reset
Trigger, Clock, reset from CTF board
Micron Quand SPI FLASH P.N. N25Q256A13EF840E
10/100/1000 Mbps Ethernet
Trigger Out
TDO out
PDO out
MO out
Elink to/from L1DDC
Trigger in
Power: +3.3V Direct or +3.4V to +42V DC-DC (selected by power switch and jumpers on the board)
P.N. PJ-002AH – mating P.N. PP3-002A
Power switch input selection
Set jumpers to bypass DC-DCs for P1, P2, P3, P4
CH63 selectable input. Set the jumper for trigger in or detector channel
Set to channel
Channel input
VMM
Artix 100t

Figure 2.5: The GPVMM board

The GPVMM board [6] houses a single VMM and an Artix®-7 FPGA by Xilinx®. It is shown in Fig 2. Communication with the software host is established using the FPGA and a commercial network Ethernet physical interface (PHY) at speeds of 10 or 100 or 1000 Mbps. The power

distribution circuit utilizes step-down regulators with an input voltage range from 3.4 up to 42V and an output ripple of less than 10mV peak-to-peak. The four power rails of the VMM (V_{ddp},V_{dd}, V_{ddad} and V_{ddd}) are independently powered by different Low Drop-Out (LDO) regulators to

mitigate any ripple effects. Three LEMO connectors provide access to the monitoring, peak-detector and time detector outputs (MO, PDO, TDO respectively) of the on-board VMM. The board has the ability to output the ART signal and receive an external trigger signal. The architecture of the GPVMM board is shown in Fig 2.5. The FPGA is accessible through a standard JTAG connector for programming and monitoring. The configuration file is stored in a flash memory which is accessible over the Serial Peripheral Interface (SPI) protocol. A second memory (EEPROM) is used to store the Media Access Control (MAC) and network addresses of the board, each of which are configurable. The EEPROM is accessed by the FPGA via the I2C protocol, thus allowing the on-board FPGA to dynamically reconfigure its network address, a crucial aspect for the scalability of the system. A �HDMI connector provides access to additional external signals such as a reference clock, trigger and reset. A miniSAS4 connector can be used to interface externally with other boards, such as the Level-1 Data Driver Card (L1DDC) [6], that has been developed for the NSW upgrade [1].
The input protection of the VMM is implemented using steering diodes in a rail-to-rail configuration along with series resistors. A number of front-end boards based on the same architecture as the one described here have been fabricated to interface with various types of detectors. The MMFE1 board, for example, was fabricated to validate the functionality of the VMM ASIC using 10 × 10 cm2 Micromegas [7] prototype chambers.

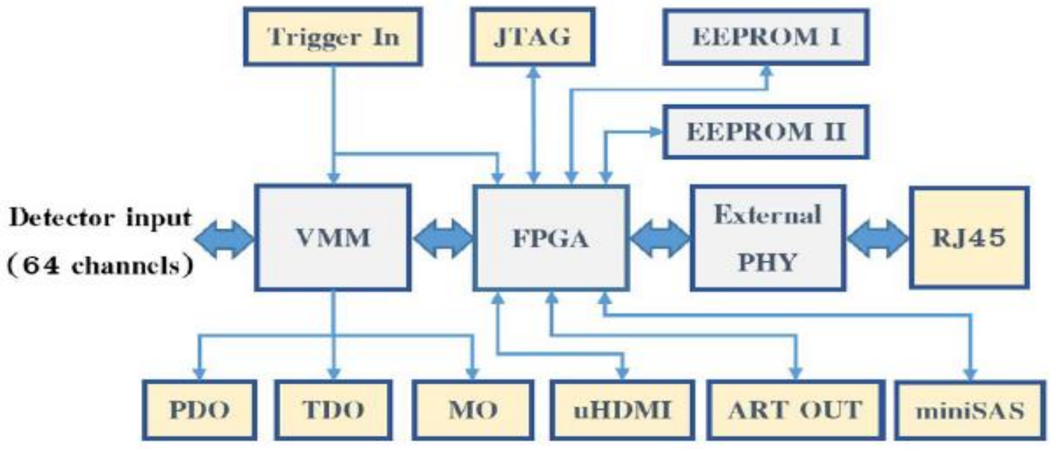


Figure 2.6 : Block diagram illustrating the architecture of the GPVMM front-end board. Blocks with yellow color indicate physical connectors while those in gray color indicate on-board components. The VMM is connected to the FPGA via differential lines, while its monitoring outputs drive three LEMO connectors. The trigger signal is propagated to the FPGA and to one of the VMM channels, so that the ASIC can perform precise timing

measurements to the trigger signal itself. The FPGA can be accessed via JTAG, while two flash memories store the FPGA's configuration and network access information. Ethernet connection is achieved via a standard RJ45 and External PHY pair that the FPGA can interface with. Finally, the FPGA can interface with off-board devices via a μHDMI and a miniSAS connector.

2.8 Firmware implementation

The on-board FPGA hosts the front-end VRS firmware, a design capable of reading-out, configuring and calibrating the ASIC. It interfaces to the on-board VMM(s) and the software. Written primarily in VHDL, the design can be implemented on several types of front-end boards that house one or multiple VMMs, such as the GPVMM. The firmware can also be easily ported to different FPGA packages and pinouts. As the scheme may be used either in standalone bench measurements, or in multiple-board applications, scalability and flexibility were two additional factors important in the development process. The main modules of the design are shown in Fig. 2.6 and are described below.

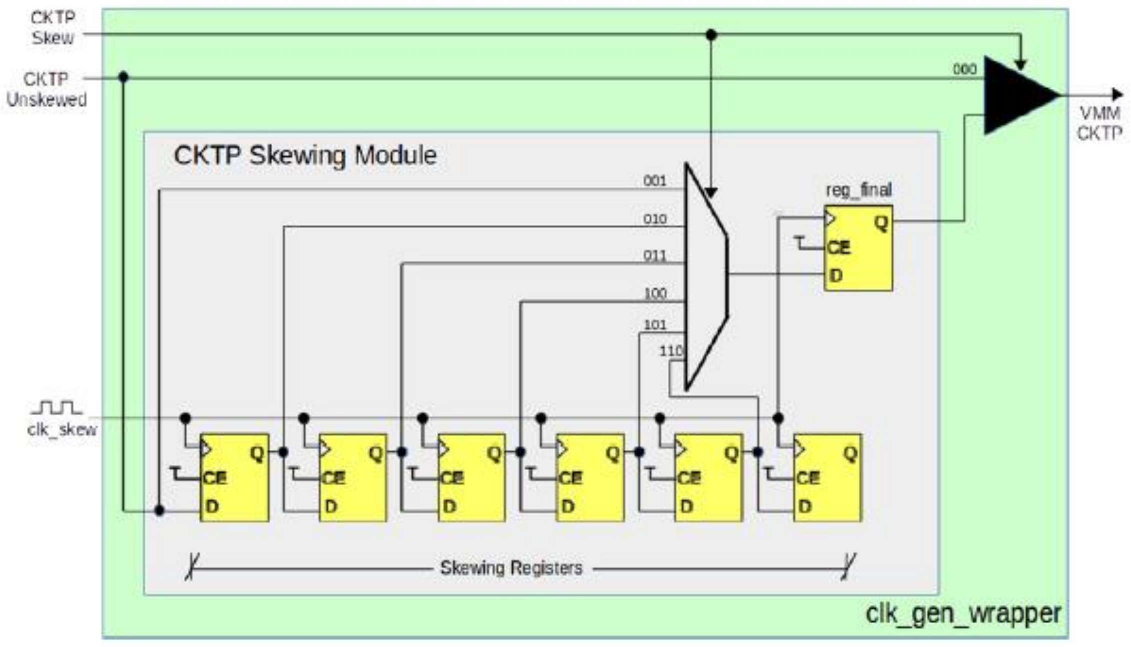


Figure 2.7: Behavioural representation of the CKTP skewing design. Only a limited number of the actual twenty-four registers and their associated multiplexer inputs is depicted here. Since the multiplexer is mainly implemented in FPGA LUTs that may be rearranged from implementation to implementation, an extra, physically constrained, register level before the global buffer that fans-out the CKTP ensures reliability and repeatability of the skewing scheme

2.8.1 The flow FSM

This component is the supervisory state machine of the FPGA's logic. It interfaces with all other building blocks of the design and defines the overall state in which the system is, depending on the user's directive as it is forwarded by the Configuration Block.

2.8.2 UDP/ICMP block

It is a sub-module provides interfacing capabilities with the software host. Modifications have been made to the original design to accommodate some Internet Control Message Protocol (ICMP) functionalities. The block also deploys the Xilinx® GTP Transceiver IP core that interfaces with the on-board PHY chip to provide 1 Gbps Ethernet-over-copper connection with the software host.

2.8.3 Configuration block

This module registers the incoming UDP payload data from the software and applies the values in internal configuration registers accordingly. It also forwards any commands for switching the system's state to the Flow FSM. Furthermore, it receives the VMM configuration data, and transmits it to the ASIC via the SPI protocol.

2.8.4 CKTP/CKBC generator

This module generates the BC clock (CKBC) and the test-pulse signal (CKTP). The CKBC is used as the VMM's reference clock, while the CKTP is driving the VMM internal test pulser. The test-pulse strobe is generated by the FPGA synchronously to the CKBC. The reference clock source may be an on-board oscillator or an external common clock source. The latter is used in multiple-board readout schemes to ensure synchronization between independent boards. This module is also responsible for the timing calibration of the ASIC.

2.8.5 Readout wrapper

This part of the logic deploys different sub-blocks (readout cores) that extract data from the VMM. Supporting all possible readout modes of the ASIC, it receives a trigger signal that initiates a readout cycle from the Packet Formation module and forwards this trigger to the VMM accordingly. Depending on the board implementation, the wrapper can instantiate more than one readout cores that can extract data from multiple on-board VMMs.

2.8.6 Packet formation

This module lies one hierarchical level above the Readout Wrapper. It forwards trigger signals to it, as received by the Trigger Module. It implements an internal trigger counter that the software uses to group events from multiple boards, when applicable. The module also aggregates the VMM data from the readout core(s). After accumulating all VMM data, it builds the event packet and finally forwards the data to the software via a buffer that interfaces with the UDP/ICMP Block.

2.8.7 Trigger module

This is the module that receives a trigger signal from either an external source or from the CKBC/CKTP Generator.

2.8.8 XADC module

This module implements the Xilinx® IP core of the same name. It essentially consists of a wrapper of the integrated 12-bit ADC that is driven by the VMM analog outputs. The only part of the logic written in Verilog HDL, the XADC is used when calibrating the threshold and pulser DACs of the VMM, or to sample the voltage variation of the VMM’s channel inputs, a measurement indicative of the noise each channel is subjected to. Finally, the module is responsible of digitizing the PDO and TDO voltage levels when acquiring data from the VMM in the Two-Phase Analog mode. Whichever the case may be, the samples are forwarded upstream to the software which handles them as typical data.

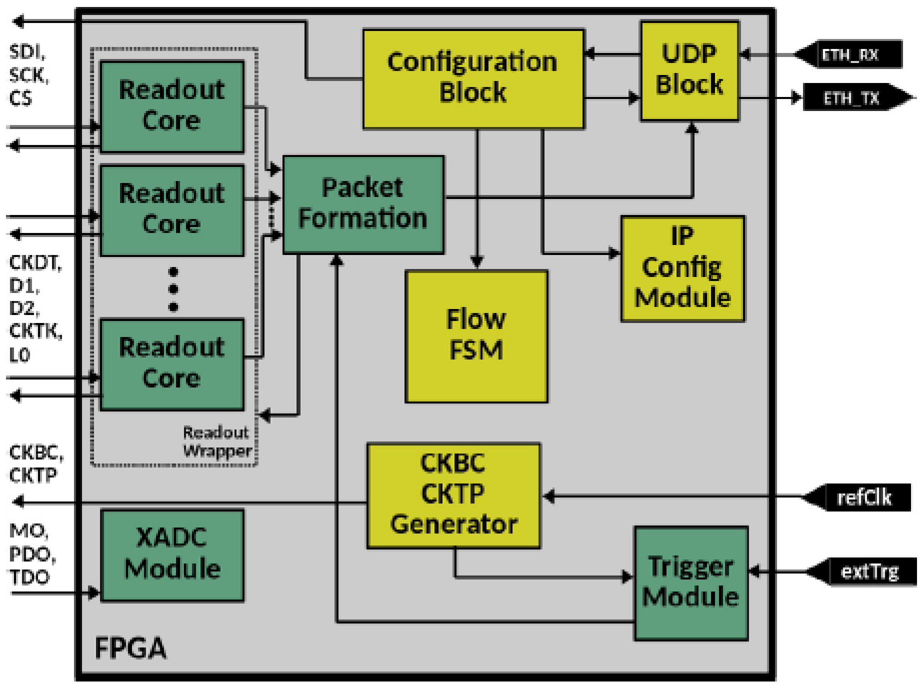


Figure 2.8 : Architecture of the VMM readout firmware

2.8.9 IP configuration modules

When using multiple readout boards within the same network domain, the ability to easily modify each FPGA's host IP and MAC address becomes a necessity. The desired IP address, which is configured by the software, is registered via the SPI or the I2C protocol. Upon a power start, the same module extracts the previously written address from the memory and stores the local IPv4 and MAC addresses.

Chapter 3

Software

In this chapter we will describe the instructions for installing and using the VMM Ethernet Readout Software (Verso). The central DAQ Software is used for configuring a user-set number of VMM-based front end boards in continuous or L0-trigger mode. This software performs UDP based readout and event building.. Verso is a graphical interface that monitors and calibrates the baseline NSW electronics NTUA/BNL firmware

3.1 Installation

The VERSO Software for the GPVMM board, and all VMM-bearing front-ends is hosted on NSWElectronics GitLab at :
https://gitlab.cern.ch/NSWelectronics/vmm_readout_software.

It is recommended to follow exactly the instructions on the Gitlab software if you have never installed it your host PC. Below are the instructions for installing VERSO. The recommended release is v4.4.0 which is for VMM2 and VMM3 readout. To obtain the software go to:

```
git clone -b v4.4.0 https://gitlab.cern.ch/NSWelectronics/vmm_readout_software.git
```

The requirements for the software are:

• Qt 5.7
• ROOT 5.34
• Boost 1.60
• C++11 (gcc>=4.7)

There are a few steps that need to be taken to obtain, install and run the DAQ software. You can follow the instructions given at the Gitlab url mentioned above. After the installation it is important for the VERSO to set the environmental parameters of Qt, Boost and ROOT packages in your bashrc file of your terminal.

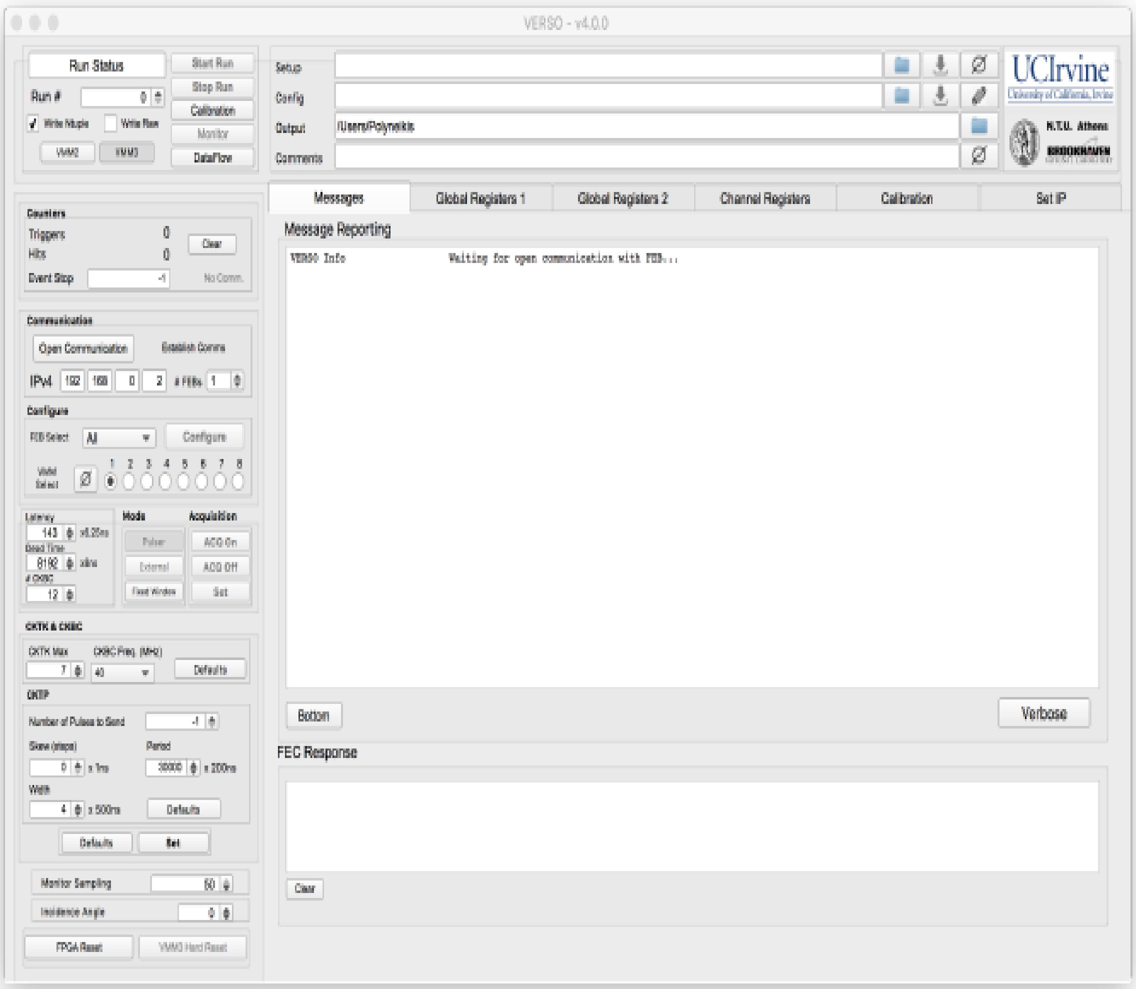
```
alias qmake=<path-to-Qt-directory>/Qt/5.7/clang_64/bin/qmake
export DYLD_LIBRARY_PATH=<path-to-boostdirectory>/boost/lib:$DYLD_LIBRARY_PATH
export ROOTSYS=<path-to-ROOT-directory>
export PATH=$PATH:$ROOTSYS/bin
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
export DYLD_LIBRARY_PATH=$ROOTSYS/lib:$DYLD_LIBRARY_PATH
```

3.2 Overview

After installing the VERSO software, the user is ready to start the graphic user interface GUI by running the executable verso file at the specific installation path of VERSO. On Linux machines this is executable by default, and will be located in the vmm_readout_software/build/ file. On a typical MAC machines this executable will be located in the vmm_readout_software/build/vmmdcs.app/Contents/MacOS/ . Whatever the case may be, the DAQ can be opened up by double-clicking on it from a graphical file-browser or by executing it from within a terminal:

```
./<path-to-vmmdcs-executable>/verso
```

The verso interaface will startup an the figure 3.1 is how the VERSO interface looks like.



Run # 0
✓ Write Ntuple   Write Raw
Output: /Users/Polyneikis

```
VERSO Info             Waiting for open communication with FEB...
```

Triggers 0
Hits 0
Event Stop -1
IPv4 192 168 0 2   # FEBs 1
FEB Select All
VMM Select 1 2 3 4 5 6 7 8
Latency 143 x6.25ns
Dead Time 8192 x6ns
CKBC 12
CKTK Max 7   CKBC Freq. (MHz) 40
Number of Pulses to Send -1
Skew (steps) 0 x 1ns   Period 30000 x 200ns
Width 4 x 500ns
Monitor Sampling 60
Incidence Angle 0

Figure 3.1: Startup of VERSO interface

3.3 Run Control

This tab displays the number and status of the data acquisition run which is in progress and gives the ability to the user to dump event-indexed ROOT n-tuple and/or raw data file. The user can also start & stop the readout/run and set VVM2, VMM3, L0 readout and config. By pressing calibration, the user can set the run type to calibration. Once the *ACQ On* button is hit, while both the Calibration and *Start Run* buttons are pressed, the calibration loop will commence, as it has been configured by the user in the following tab. By pressing DataFlow, the user opens up VERSO dataflow monitor and by pressing Monitor, user enable sending event samples to VMM monitor (https://gitlab.cern.ch/aikoulou/vmm-mon).

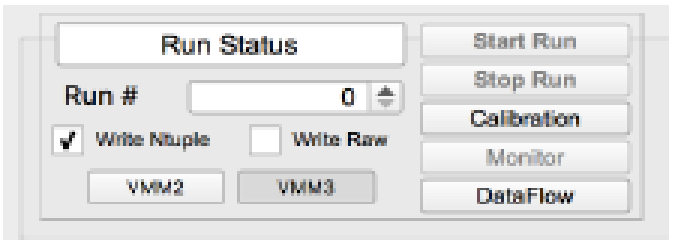


Figure 3.2

3.4 Monitor

The vmm-mon application is a single window app that monitors the data that VERSO records. The main panel shows the plots, depending on the settings in the left panel.

- The list of boards and VMMs is shown, and the user can select any item in the tree to show the corresponding plots in the main panel.

- Settings: refresh interval, type of plots to show (PDO, TDO etc),

• Clear selection in the tree, in order to make the main panel show only VMM plots.

• Reset histograms (resets all the histograms)
• Pause/Resume

• .png, makes a png snapshot of the main panels and saves to the directory of the vmm-mon.

Figure 3.3: Plots of the VMM board.

3.5 Top-level Configuration Parameters

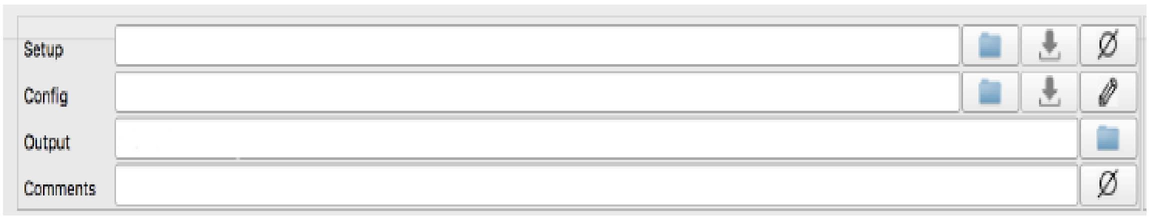


Figure 3.4

In the Config field, the user can load, set and write a configuration xml file with the VMM settings that have been set in the GUI. The procedure to write a new config file is to set the interface parameters, press the ”pen” button and specify the directory that user wants to save the config file in. This way, the user has the ability to load the VMM configuration by pressing the folder icon, load the config file and press the ”download” button to set the configuration parameters to the VERSO interface. In the Output field, the user can specify the directory location to dump the output files and in the Comment field, the user is able to comment/text or store as a field inside of ROOT n-tuple. Also, VERSO provides the optional Setup field, where the user can load the DAQ setup configuration such as detector-to-elx channel mapping and detector mapping of detectors and electronics that he/she can find under vmm_readout_software/readout_configuration directory.

3.6 Counters

Figure 3.5

Displays the total number of triggers (events) recorded and the total number of the unique VMM channel hits recorded. In Event Stop box the user can set the number of events to record in the run by setting -1 there is no limit of the events. Be advised that the number of triggers/hits is not exact. One has to open the ntuple after stopping the run to see exactly how many events were recorded by the DAQ system.

3.7 Communication and Configuration

Figure 3.6

By pressing Open Communication, VERSO opens the UDP sockets. In the IPv4 field, so the user can specify the IP of the VMM(FEB) board and sets the address of the first FEB and also the number of FEBs to communicate with. In the Configuration Frame, the user can select the FEB number that he wants to send the VMM/FPGA configuration parameters to. In VMM Select, the user specifies the VMM ID to configure on the selected FEB. By pressing the Configure button, we send the VMM configuration to the selected FEBs and VMMs. For a successful data acquistion of multiple boards, FEBs ip addresses must be in ascending order, e.g 1st FEB 192.168.0.2 2st FEB 192.168.0.3 etc.

3.8 Trigger and Acquisition Mode

Figure 3.7

The user is able to change the trigger latency and can set the Acquistion/Trigger Mode by enabling the Pulser (internal) or External VMM trigger mode. There is a third option called Fixed Window, where the FPGA can send a specific number of CKBCs upon reception of a trigger. The Latency step-size is measured in steps of 25 ns in L0 readout mode, and in units of 6.25 ns in continuous/Fixed Window mode. By pressing Set,
the user sends the settings to the FPGA. Via the AQN Off and AQN On, the user can turn off/on VMM acquisition mode.

3.9 FPGA clocks Configuration

Figure 3.8

The user can set the maximum number of CKTKs (if in continuous readout mode) and also the frequency of CKBC (40,20 and 10 MHz CKBC frequencies are possible). The CKTP frame is for the configuration of the CKTP (test pulse), where the user sets a fixed number of CKTP pulses to send (-1 for no limit) and also the CKTP skew w.r.t CKBC (1ns steps for 40 ns CKBC clock, 6.25 ms steps otherwise). By pressing Set,the user sends CKTK, CKBC and CKTP configuration to the FEBs.

3.10 Monitor Sampling and Angle

Figure 3.9

User selects the event sampling rate parameters for sending to
vmm-mon application and also the incident angle of detector
to be stored in output ROOT n-tuple.

3.11 FPGA & VMM Hard Reset

Figure 3.10

Resets tha FPGA or hard resets the VMM

Global Register I & II & Channel Registers

The VERSO software gives the user the ability to set the Global Configuration Registers of the VMM3. The bit names of the registers in the Global Registers Tabs are described in the official VMM3 documentation Under the Global Register I & II, tabs, the user can set most common registers such as Test Pulse amplitude in DAC Counts, Threshold in DAC Counts, Channel Gain, Peak Time, TAC Slope, Channel Polarity, Channel Monitoring etc. In Channel Registers, figure 3.12, the user can modify the settings of each of the 64 VMM channels, like masking (SM), or enabling of the test pulse circuitry (ST).

Figure 3.11 : Global registers I & II Tabs

Figure 3.12 : Channel Registers Tab

3.12 FEB IP Configuration Panel

Figure 3.13

Under the Set IP tab, you can change the default ip of the FEB board (default: 192.168.0.2) to a new board ip. By pressing the Set button, the user sends the IP configuration to the FPGA. After the change of ip, it is mandatory for the user to reset the IPv4 field with the new FEB's ip under the Communication Frame of VERSO software.

3.13 Calibration

Figure 3.14 : Calibration tab

At the calibration tab you can perform calibration routines using the FPGA's xADC or in non-xADC mode. In xADC based sampling calibration type, you can perform calibration routines to sample:

• Channel-by-channel threshold variations, stepping over the VMM threshold trimmers values

• Analog DAC levels (threshold and pulser)

• Channel baseline & noise levels

It is important to acknowledge that the calibration loop may ''freeze'', in the baseline xADC readout mode. This happens if the voltage, sampled by the xADC is above a certain limit. For the baseline runs, this happens if a channel is dead. The loop will then stop and the user has to press the Manual xADC configuration button in the GUI to have VERSO ignore the channel and move on. You can realize if the loop has frozen by looking at the Ch. Monitor field of the Global Registers 1 tab. The number of monitored channel for the baseline check should always be incrementing.
In non-xADC based sampling calibration type, the user can perform calibration routines based on the loop parameters at the bottom of the window (Gain, Peak Time, TAC Slope, Test Pulse DAC etc.)

Chapter 4 Running Procedure

Now we will describe successfully data acquisition with internal trigger (pulser) and trigger (scintillator).

4.1 Communication

First we want to configure the right configuration with the board by setting the static IPv4 of the desktops network card with the network settings:

- IP Address : 192.168.0.16
- Subnet Mask : 255.255.255.0

After setting the network settings, the user plugs the VMM board in the PC through an Ethernet cable and supplies the board 3.3V-3.4V to 42 V DC-DC (the selection of power is based on the power switch and jumpers of the board). In the newer version of the firmware , a led blinking pattern will occur at the on-board USER_ LEDs,if the reference clock is being received correctly by the FPGA. The user can check the communication with the board by pinging the board's IP address (default IP: 192.168.0.2) :

```
ping 192.168.0.2
```

Or the user can find the ip the broadcast ping:

```
ping -b 192.168.0.255
```

If the ping is successful and the Ethernet LEDs are blinking, the user can continue with the experiment setup. If not, the user should connect GPVMM board to CTF board via uHDMI to get the reference clock, once the connection is succesful the D17 LED ("PLL/MMCM Lock" signal) will turn on and Ethernet LEDs will start to blink. In the newer version of the firmware the LED pattern will also be visible for the first few seconds.
Also, it is important to mention that there are two different versions of the GPVMM board's firmware, with reference clock via uHDMI/CTF and another without the external reference clock. The default firmware of GPVMM boards needs CTF reference clock. User should follow Chapter 3 to program GPVMM with the right firmware.

4.2 Setup

In the below figure is the experiment setup for data acquisition of the VMM board. It consists of the users detector, PC for communicating with the board through an ethernet cable, oscilloscope for the monitor output of the VMM channels, power supply, scintillators for the trigger mode(TTL pulses) and also a fan is appropriate for the cooling of VMM and FPGA chip. Also, the use of a CTF board for reference clock is optional and it depends on the GPVMM's firmware. The user in order to use the reference clock from CTF board must not connect GPVMM board directly to CTF via uHDMI, but one must use a compatible HDMI adapter which has been designed by the NTUA

Figure 4.1 : Experimental setup

4.3 Internal Trigger (Pulser)

The steps below describe the procedure the user must follow to be able to record data with the Internal Trigger Mode (Pulser):

1. Start VERSO software
2. Set board's IP in IPv4 field
3. Press Open Communication button
4. Set Global Register's 1 & 2 parameters
5. Set Internal Pulse[ST] or Mask[SM] channels through Channel Registers Tabs
6. Select Pulser Mode In Mode Frame
7. Press VMM3 Hard Reset button
8. Press Set button in CKTP Frame
9. Press Set button in Mode Frame
10. Press Configure button to sent the configuration parametes to VMM

4.4 External Trigger

The steps below describe the procedure the user must follow to be able to record data with the External Trigger Mode (e.g scintillator):

1. Connect Scintillator Coincidence output to TRG_IN port1.
2. Start VERSO software
3. Set board's IP in IPv4 field
4. Press Open Communication button
5. Set Global Register's 1 & 2 parameters
6. Mask[SM] channels through Channel Registers Tabs (optional)
7. Select External Mode In Mode Frame
8. Set trigger parameters (latency, lat.extra) in Trigger Frame
9. Press VMM3 Hard Reset button
10. Press Set button in CKTP Frame
11. Press Set button in Mode Frame
12. Press Configure button to sent the configuration parametes to VMM

4.5 Acquisition and Monitor Guidelines

After following the steps above, the user is ready to start the acquisition of data by pressing ACQ On in Acquistion Frame as well as record them in a ROOT n-tuple by pressing Start Run. If the procedure is successful, the counter of Triggers and Hits will start to count the number of triggers and hits that user is recording. If the counter is not incrementing, then it can either be that the VMM/FPGA is not receiving triggers at all, or that the VMM is receiving triggers/l0 signals, but no events are being recorded. At any given time, the user can stop the recording procedure by pressing ACQ_ Off and Stop Run. The ROOT n-tuple file is placed under the directory that the user had specified in the Output field. For every change in the VERSO parameters and registers, it is mandatory to repeat steps 9-12 for the successful sending of registers to the VMM and FPGA. It is recommended to use an oscilloscope connected to the GPVMM3 board through a LEMO cable in MO ouput, to monitor the VMM channel output pulse, an example of internal pulser shown in figure , which have selected to monitor by the Ch. Monitor [sm5-sm0] selection in Global Registers 1 tab in VERSO software. Otherwise, a useful tool to monitor the transactions between the GPVMM board and the PC that hosts the software is Wireshark. The packets that are exchanged between the software and firmware can be seen at this applications GUI, thus giving an overview of the state of the setup (e.g., if the FPGA sends packets of only a few bytes while in ACQ ON, then the VMM does not send out any data).

4.6 xADC Calibration Run

This subsection gives information on how to run a baseline and threshold DAC calibration loop.

Figure 4.2 : Monitor Output of VMM channel through Internal Trigger Mode (Pulser)

4.7 Baselines

1. Start VERSO software
2. Set board's IP in IPv4 field
3. Press Open Communication button
4. Set Global Register's 1 & 2 parameters
5. Go to Calibration Tab
6. Select xADC, select which VMMs to sample from (one for GPVMM, eight for MMFE8)
7. Select number of samples for each calibration step and the channel range
8. Select Baselines in the list on the left

9. Select Calibration at the top left of the VERSO GUI, and press Start Run which is in the same area
10. Once the run finishes (the Stop Run should not be clickable anymore, and the Channel Monitor should have
stopped rolling at channel 63, if the final channel selected at the calibration tab was 63), one can open the
calibration file.
11. If the calibration loop hangs at a channel, one should press the Manual xADC Configuration button at the
Calibration Tab once, and VERSO will sample over the next channel.
12. If the FPGA is sending calibration packets to VERSO and the loop proceeds normally, but VERSO writes
an empty .ROOT file, then the user should close and restart VERSO without any config .xml loaded.
The command in the .ROOT prompt to see the baseline voltage profile across an entire eight-VMM board is (e.g.
MMFE8):
calib->Draw("samples/4.096:channel+chip*64>>h1(512,0,511, 4096, 0, 4095)", "", "colz")
And for a 1-VMM board (e.g. GPVMM):
calib->Draw("samples/4.096:channel>>h1(64,0,63, 4096, 0, 4095)", "", "colz")

4.8 Threshold DAC

1. Start VERSO software
2. Set board’s IP in IPv4 field
3. Press Open Communication button
4. Set Global Register’s 1 & 2 parameters
5. Go to Calibration Tab
6. Select xADC, select which VMMs to sample from (one for GPVMM, eight for MMFE8)
7. Select number of samples for each calibration step and the channel range
8. Select Threshold DAC in the list on the left
9. Select Threshold DAC scan range, and step size
10. Select Calibration at the top left of the VERSO GUI, and press Start Run which is in the same area
11. Once the run finishes (the Stop Run should not be clickable anymore, and the Threshold DAC should have
stopped rolling at the maximum value, as set by the user two steps before.

11. If the calibration loop hangs, one should press the Manual xADC Configuration button at the Calibration Tab.

12. If the FPGA is sending calibration packets and the loop proceeds normally, but VERSO writes an empty
.ROOT file, then the user should close and restart VERSO without any config .xml loaded.
The command in the .ROOT prompt to see the threshold DAC voltage for chip 0 of a given board is:
calib->Draw("samples/4.096:dacCounts>>h1(1024,0,1023, 4096, 0, 4095)", "chip==0", "colz")

Obviously one can change the chip number to see the DAC->Voltage relation for all chips if a multi-chip board is used. For the GPVMM board, the above command is sufficient.

4.9 VMM3 Registers

The table below includes the global registers of the VMM:

Figure 4.3.1 : Global Configuration Registers of the VMM

Figure 4.3.2 : Global Configuration Registers of the VMM

Figure 4.3.3 : Channel Configuration Registers of the VMM

5.1 Program Scope

Program scope in an extension of verso so we don't have to use the large oscilloscope, it is more convenient. With this program we are able to read signals from our VMM board. In the picture below is the oscilloscope that we used to capture our signals.

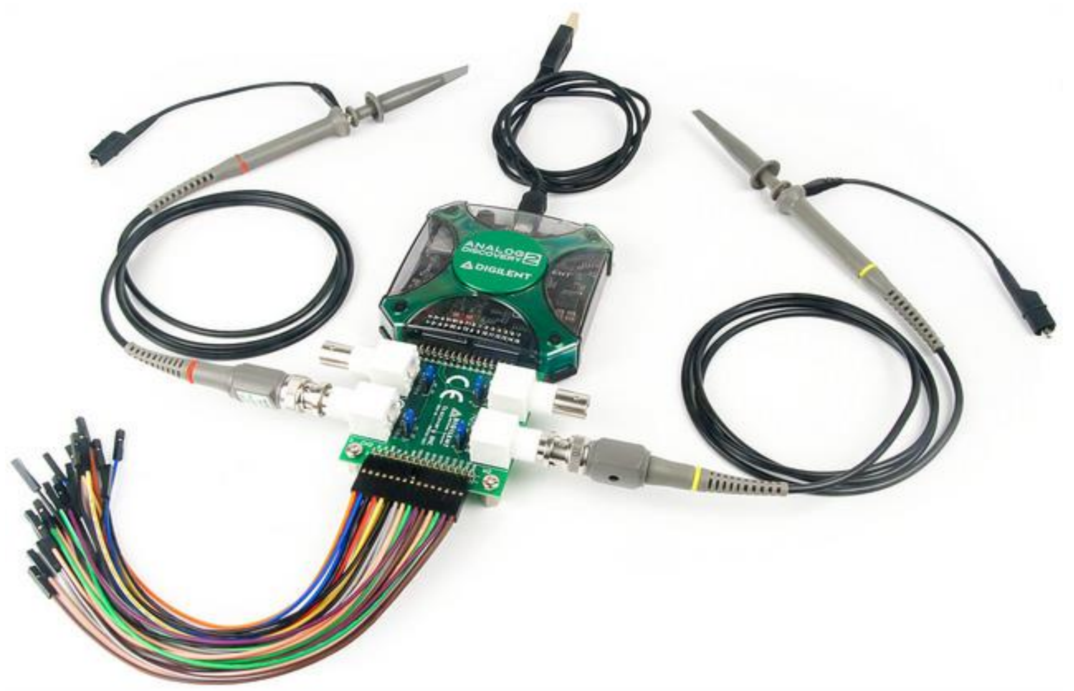
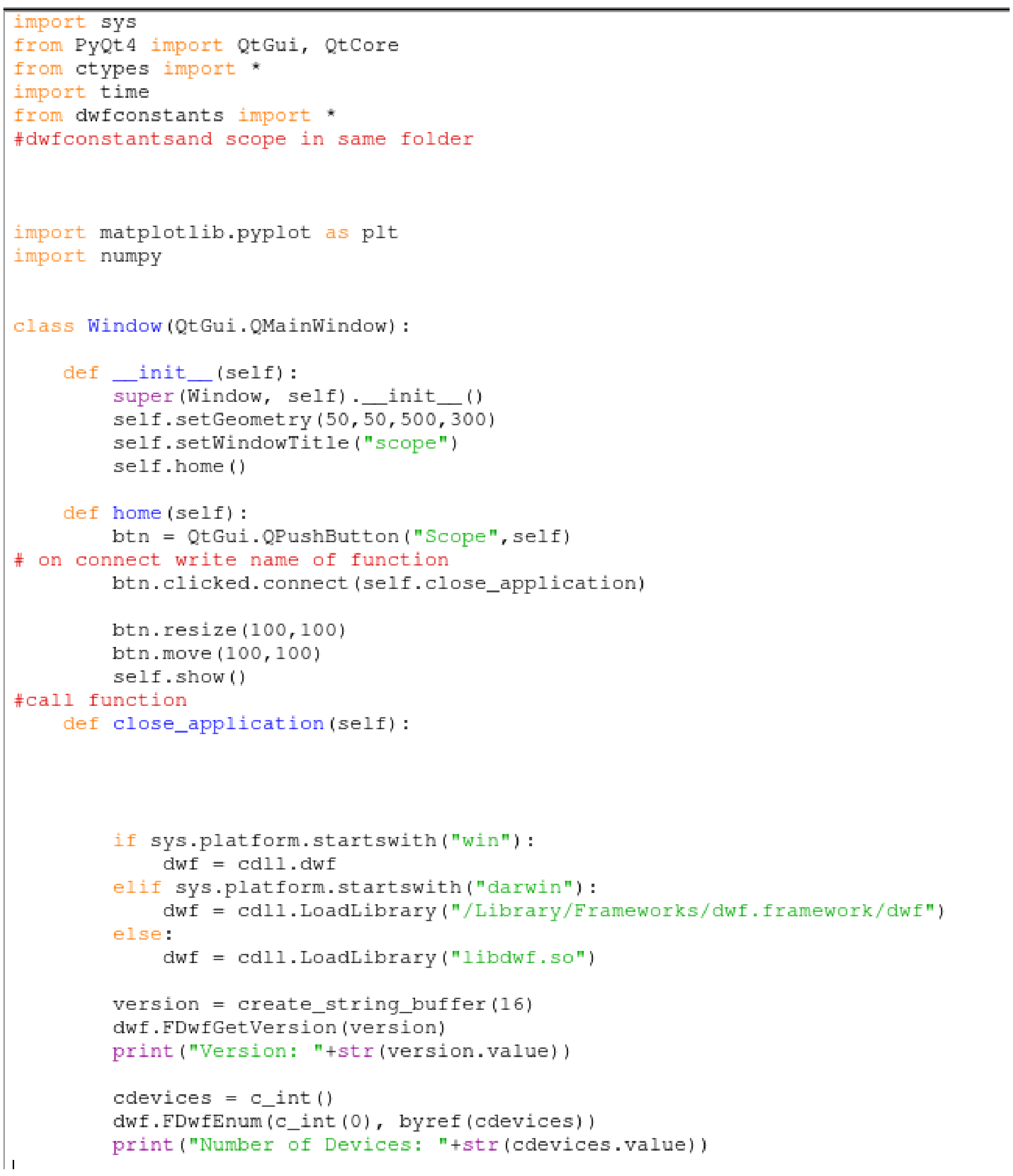


Figure 5.1: Digilents analog discovery oscilloscope

To be able to communicate with this device we will use the scope program. It is a graphic interface made by pyQt4. After importing the libraries of

Digilents analog discovery 2, we are able to call specific functions that will allow us to communicate, acquire data and then visualize our signals.

To run this program first we have to install python on our computer. Then you have to install PyQt4. With these tools we our capable now to make our program which is shown below.



```
import sys
from PyQt4 import QtGui, QtCore
from ctypes import *
import time
from dwfconstants import *
#dwfconstantsand scope in same folder



import matplotlib.pyplot as plt
import numpy


class Window(QtGui.QMainWindow):

    def __init__(self):
        super(Window, self).__init__()
        self.setGeometry(50,50,500,300)
        self.setWindowTitle("scope")
        self.home()

    def home(self):
        btn = QtGui.QPushButton("Scope",self)
# on connect write name of function
        btn.clicked.connect(self.close_application)

        btn.resize(100,100)
        btn.move(100,100)
        self.show()
#call function
    def close_application(self):




        if sys.platform.startswith("win"):
            dwf = cdll.dwf
        elif sys.platform.startswith("darwin"):
            dwf = cdll.LoadLibrary("/Library/Frameworks/dwf.framework/dwf")
        else:
            dwf = cdll.LoadLibrary("libdwf.so")

        version = create_string_buffer(16)
        dwf.FDwfGetVersion(version)
        print("Version: "+str(version.value))

        cdevices = c_int()
        dwf.FDwfEnum(c_int(0), byref(cdevices))
        print("Number of Devices: "+str(cdevices.value))
```

```
if cdevices.value == 0:
    print("no device detected")
    quit()

print("Opening first device")
hdwf = c_int()
dwf.FDwfDeviceOpen(c_int(0), byref(hdwf))

if hdwf.value == hdwfNone.value:
    print("failed to open device")
    quit()


print("Configure and start first analog out channel")


print("Configure analog in")
# change frequency (smaller number to the left bigger to the right smaller time<might be 2us
dwf.FDwfAnalogInFrequencySet(hdwf, c_double(500000000))
print("Set range for all channels")
#have to see how dividee works properly
#dwf.FDwfDigitalInDividerSet(hdwf,c_int(0))
dwf.FDwfAnalogInChannelRangeSet(hdwf, c_int(-1), c_double(4))
dwf.FDwfAnalogInBufferSizeSet(hdwf, c_int(1000))
#sample rate





print("Wait after first device opening the analog in offset to stabilize")
time.sleep(2)

print("Starting acquisition...")
#set up trigger
#dwf.FDwfDigitalInDividerSet(hdwf,c_int(0))
#dwf.FDwfDigitalInDividerGet(hdwf,c_int(0))
dwf.FDwfAnalogInTriggerAutoTimeoutSet(hdwf, c_double(0)) #disable auto trigger
dwf.FDwfAnalogInTriggerSourceSet(hdwf, trigsrcDetectorAnalogIn) #one of the analog in channels
#dwf.FDwfAnalogInTriggerTypeSet(hdwf, trigtypeEdge)
dwf.FDwfAnalogInTriggerChannelSet(hdwf, c_int(0)) # first channel
dwf.FDwfAnalogInTriggerLevelSet(hdwf, c_double(0.4)) # set trigger volts
dwf.FDwfAnalogInConfigure(hdwf, c_int(1), c_int(1))
```

```
        sts = c_int()
        while True:
            dwf.FDwfAnalogInStatus(hdwf, c_int(1), byref(sts))
            #set up trigger
            #dwf.FDwfAnalogInTriggerAutoTimeoutSet(hdwf, c_double(0)) #disable auto trigger
            #dwf.FDwfAnalogInTriggerSourceSet(hdwf, trigsrcDetectorAnalogIn) #one of the analog in channels
            #dwf.FDwfAnalogInTriggerTypeSet(hdwf, trigtypeEdge)
            #dwf.FDwfAnalogInTriggerChannelSet(hdwf, c_int(0)) # first channel
            #dwf.FDwfAnalogInTriggerLevelSet(hdwf, c_double(2)) # 0.5V,0.6V

            #dwf.FDwfAnalogInTriggerConditionSet(hdwf, trigcondRisingPositive)
            #dwf.FDwfAnalogOutEnableSet(hdwf, c_int(0), c_int(1)) # 1 = Sine wave")
            #dwf.FDwfAnalogOutFunctionSet(hdwf, c_int(0), c_int(1))
            #dwf.FDwfAnalogOutFrequencySet(hdwf, c_int(0), c_double(3000))
            #dwf.FDwfAnalogOutConfigure(hdwf, c_int(0), c_int(1))
            if sts.value == DwfStateDone.value :
                break
            time.sleep(0.1)
        print("    done")

        rg = (c_double*1000)()
        dwf.FDwfAnalogInStatusData(hdwf, c_int(0), rg, len(rg)) # get channel 1 data
        #dwf.FDwfAnalogInStatusData(hdwf, c_int(1), rg, len(rg)) # get channel 2 data

        dwf.FDwfAnalogOutReset(hdwf, c_int(0))
        dwf.FDwfDeviceCloseAll()

        dc = sum(rg)/len(rg)
        print("DC: "+str(dc)+"V")

        plt.plot(numpy.fromiter(rg, dtype = numpy.float))
        plt.show()


def run():

    app = QtGui.QApplication(sys.argv)
    GUI = Window()
    sys.exit(app.exec_())

run()
```

Figure 5.2 : Scope program

On the first line a codes we are making a button, that when clicked will call our scope function and our program will run. In our def close_application function, we see our Scope program.

5.2 functions of Digilents analog discovery 2

It is important to understand what each function from our Digilent library does, so we can understand better how this program works. These functions can be understood better from the [9] SDK reference manual of Digilents analog discovery 2.

1.)**FDwfGetVersion**(char szVersion[32])
Description: Retrieves the version string. The version string is composed of major, minor, and build numbers (i.e., "2.0.19").

Parameters:

-szVersion – Pointer to buffer to receive version string

2.)**FDwfEnum**(ENUMFILTER enumfilter, int *pnDevice)
Description: Builds an internal list of detected devices filtered by the *enumfilter* parameter. It must be called before using other *FDwfEnum* functions because they obtain information about enumerated devices from this list identified by the device index.

Parameters:

- enumfilter – Filter value to be used for device enumeration. Use the *enumfilterAll* constant to discover all compatible devices.

- pnDevice – Integer pointer to return count of found devices by reference

3.) **FDwfDeviceOpen**(int idxDevice, HDWF *phdwf)
Description: Opens a device identified by the enumeration index and retrieves a handle. To automatically enumerate all connected devices and open the first discovered device, use index -1.

Parameters:

- idxDevice – Zero based index of the enumerated device.

- phdwf – Pointer to *HDWF* variable to receive opened interface handle by reference.

4.)FDwfAnalogOutEnableSet(HDWF hdwf, int idxChannel, BOOL fEnable)
Description: Enables or disables the channel specified by idxChannel. With channel index -1, each Analog Out channel enable will be configured to use the same, new option.

Parameters:

- hdwf – Open interface handle on a device.

- idxChannel – Channel index.

- fEnable – TRUE to enable, FALSE to disable.

5.)**FDwfAnalogOutFunctionSet**(HDWF hdwf, int idxChannel, FUNC func)
Description: Sets the generator output function for the specified instrument channel. With channel index -1, each enabled Analog Out channel function will be configured to use the same, new option.

Parameters:

- hdwf – Open interface handle on a device.

- idxChannel – Channel index.

- func – Generator function option to set.

6.) **FDwfAnalogOutFrequencySet**(HDWF hdwf, int idxChannel, double hzFrequency)
Description: Sets the frequency. With channel index -1, each enabled Analog Out channel frequency will be configured to use the same, new option.

Parameters:

- hdwf – Open interface handle on a device.

- idxChannel – Channel index.

– hzFrequency – Frequency value to set expressed in Hz.

7.)**FDwfAnalogOutConfigure**(HDWF hdwf, int idxChannel, BOOL fStart)
Description: Starts or stops the instrument. Value 3 will apply the configuration dynamically without changing the state of the instrument. With channel index -1, each enabled Analog Out channel will be configured.

Parameters:

- hdwf – Interface handle.

- idxChannel – Channel index.

- fStart – Start the instrument: 0 stop, 1 start, 3 apply

8.) **FDwfAnalogInFrequencySet**(HDWF hdwf, double hzFrequency)
Description: Sets the sample frequency for the instrument.

Parameters:

- hdwf – Interface handle.

– hzFrequency – Acquisition frequency to set.

9.) **FDwfAnalogInChannelRangeSet**(HDWF hdwf, int idxChannel, double voltsRange)
Description: Configures the range for each channel. With channel index -1, each enabled Analog In channel range will be configured to the same, new value.

Parameters:

- hdwf – Interface handle.

- idxChannel – Channel index.

-voltsRange – Voltage range to set.

10.)**FDwfAnalogInBufferSizeSet**(HDWF hdwf, int nSize)
Description: Adjusts the AnalogIn instrument buffer size.

Parameters:

- hdwf – Interface handle.

-nSize – Buffer size to set.

11.)**FDwfAnalogInConfigure**(HDWF hdwf, BOOL fReconfigure, BOOL fStart)
Description: Configures the instrument and start or stop the acquisition. To reset the Auto trigger timeout, set fReconfigure to TRUE.

Parameters:

- hdwf – Interface handle.

- fReconfigure – Configure the device.

- fStart – Start the acquisition

12.)**FDwfAnalogInStatus(HDWF hdwf, BOOL fReadData,** DwfState *psts)
Description: Checks the state of the acquisition. To read the data from the device, set fReadData to TRUE. For single acquisition mode, the data will be read only when the acquisition is finished.

Parameters:

- hdwf – Interface handle.

- fReadData – TRUE if data should be read.

- psts – Variable to receive the acquisition state.

13.)**FDwfAnalogInStatusData**(HDWF hdwf, int idxChannel, double *rgdVoltData, int cdData)
Description: Retrieves the acquired data samples from the specified idxChannel on the AnalogIn instrument. It copies the data samples to the provided buffer.

Parameters:

- hdwf – Interface handle.

- idxChannel – Channel index.

- rgdVoltData – Pointer to allocated buffer to copy the acquisition data.

- cdData – Number of samples to copy

14.)**FDwfAnalogOutReset**(HDWF hdwf, int idxChannel)
Description: Resets and configures (by default, having auto configure enabled) all AnalogOut instrument parameters to default values for the specified channel. To reset instrument parameters across all channels, set idxChannel to -1.

Parameters:

- hdwf – Interface handle.

-idxChannel – Channel index

15.)**FDwfAnalogInTriggerAutoTimeoutSet**(HDWF hdwf, double secTimeout)
Description: Configures the auto trigger timeout value in seconds.
Parameters:
- hdwf – Interface handle.
- secTimeout – Timeout to set.

16.)**FDwfAnalogInTriggerSourceSet(HDWF hdwf,** TRIGSRC trigsrc)

Description: Configures the AnalogIn acquisition trigger source.

Parameters:

- hdwf – Interface handle.

- trigsrc – Trigger source to set.

17.)**FDwfAnalogInTriggerChannelSet**(HDWF hdwf, int idxChannel)

Description: Sets the trigger channel.

Parameters:

- hdwf – Interface handle.

- idxChannel – Trigger channel index to set.

18.)**FDwfAnalogInTriggerLevelSet**(HDWF hdwf, double voltsLevel)

Description: Sets the trigger voltage level in V.

Parameters:

- hdwf – Interface handle.

- voltsLevel – Trigger voltage level to set.

19.)**FDwfDeviceCloseAll**()

Description: Closes all opened devices by the calling process. It does not close all devices across all processes.

Parameters: None.

After correctly calling this functions we can create our oscillator program that allows us to see signals.

5.3 Testing

We shall now connect our oscilloscope to our computer and open python version 2.7.5, to see if our program is working properly. This is our basic setup.

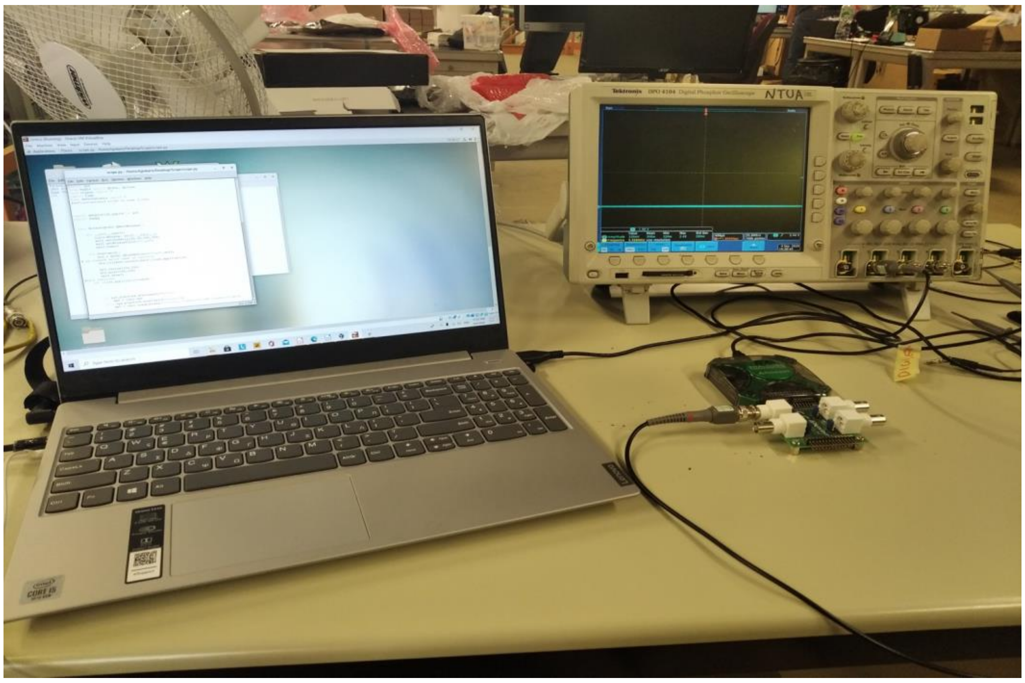


Figure 5.3 : Connection of oscilloscope with computer

In the picture above you can see that the laptop is connected to the small oscilloscope and the small oscilloscope is connected to another oscilloscope. We have done this to see that the signals that will be shown on on program scope will be the same on the oscilloscope we connected to.

First we shall send a test pulse 2.5 V from from Tektronix oscilloscope to the analog discovery 2 oscilloscope.

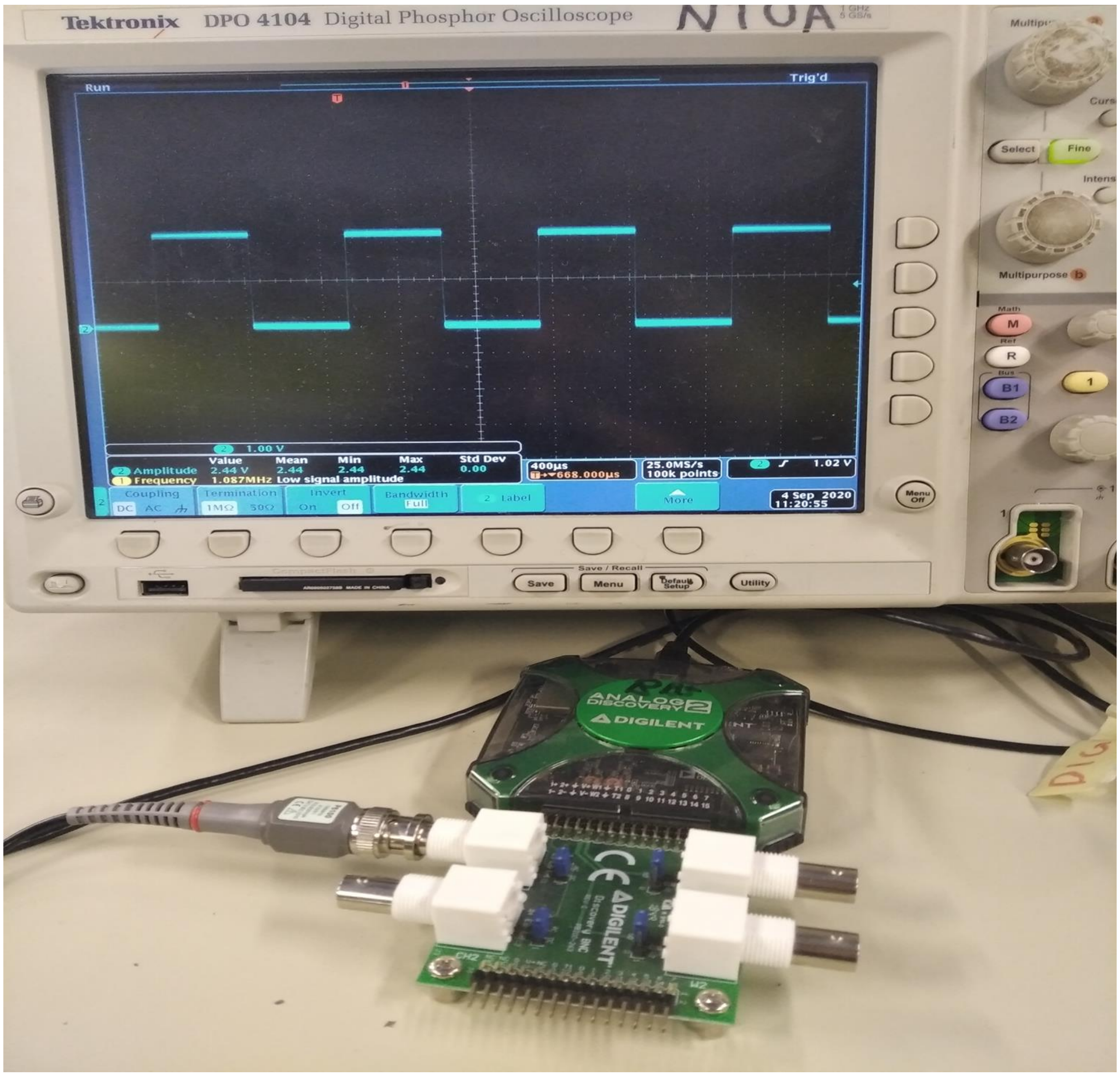
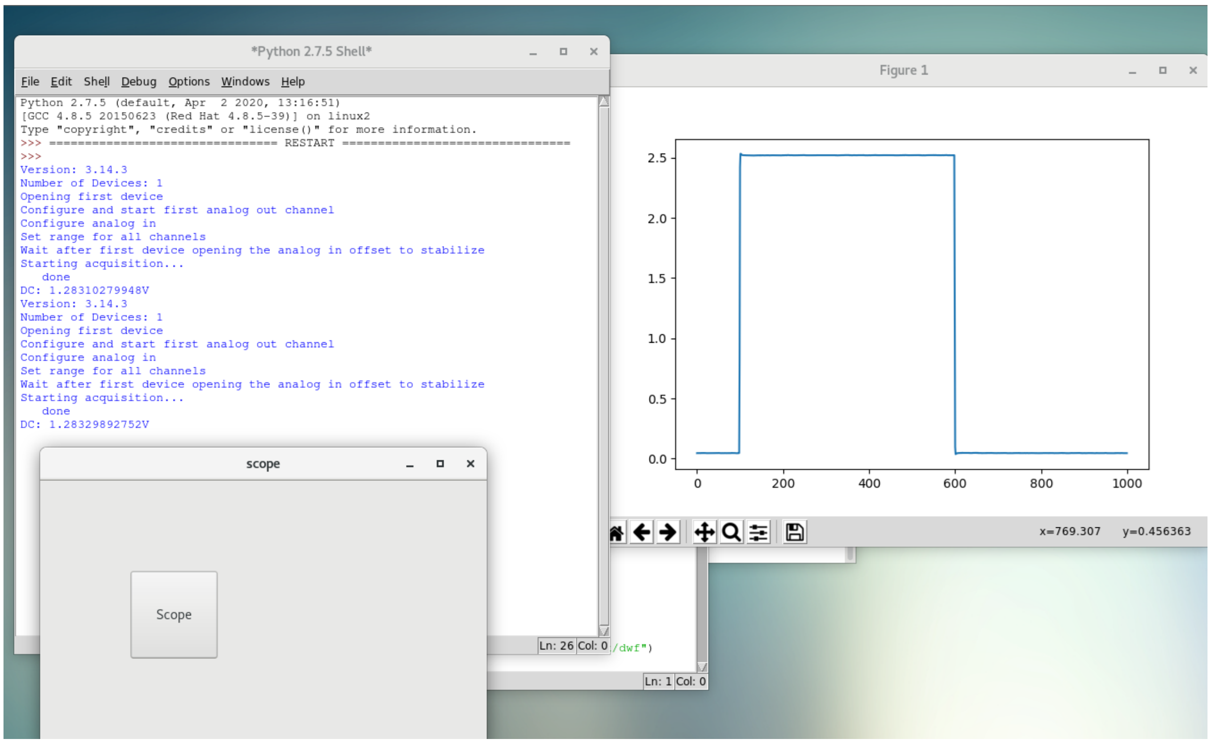


Figure 5.4: Sending 2.5 V test pulse from Tektronix oscilloscope to analog discovery oscilloscope

Hopefully after running the program we are supposed to see the same signal. At the below image this is what we see.



```
Python 2.7.5 (default, Apr  2 2020, 13:16:51)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Version: 3.14.3
Number of Devices: 1
Opening first device
Configure and start first analog out channel
Configure analog in
Set range for all channels
Wait after first device opening the analog in offset to stabilize
Starting acquisition...
   done
DC: 1.28310279948V
Version: 3.14.3
Number of Devices: 1
Opening first device
Configure and start first analog out channel
Configure analog in
Set range for all channels
Wait after first device opening the analog in offset to stabilize
Starting acquisition...
   done
DC: 1.28329892752V
```

Figure 5.5: Running the scope program

After reading fig 5.4 and 5.5 we can see that our program is reading the same 2.5 test pulse that is sent from the tektronix oscilloscope.

5.4 Reading signal from Gpvmm board

After reading our test pulse, we are now ready to communicate with our Gpvmm board. First we connect our laptop with the board.

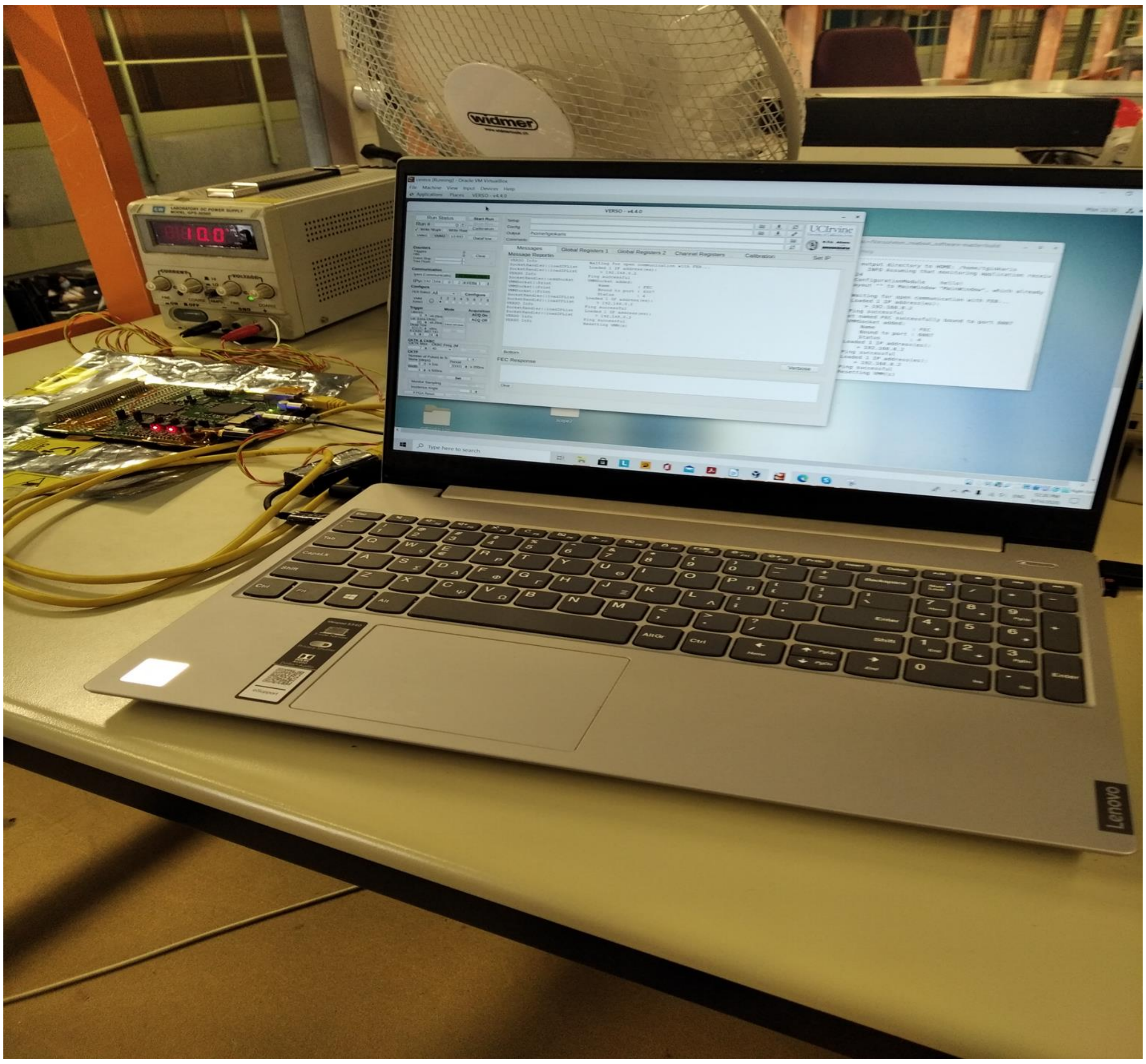
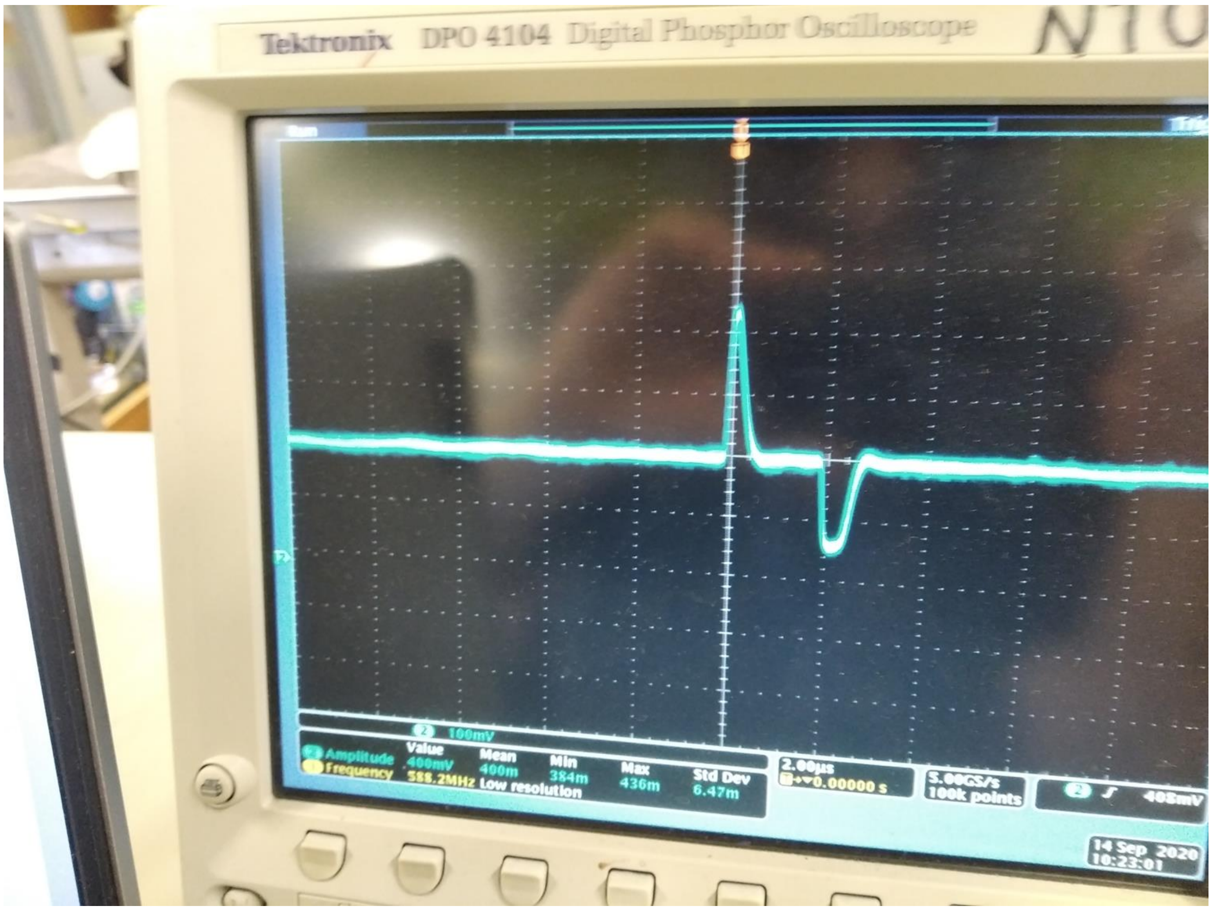


Figure 5.6: Connecting laptop to board

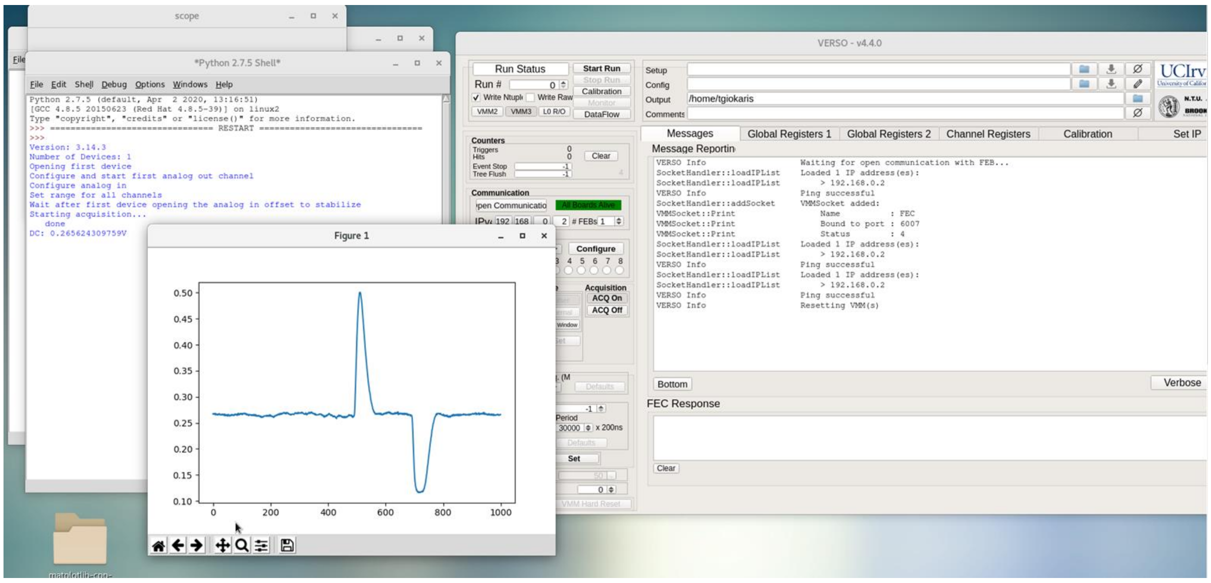
And here is the signal that we are supposed to see.



	Value	Mean	Min	Max	Std Dev
Amplitude	400mV	400m	384m	436m	6.47m
Frequency	588.2MHz	Low resolution			

Figure 5.7: Signal from Tektronix oscilloscope

We can see that our signal has an Amplitude about of 0.4 V. Now let run our program scope to see if we have a similar result.



```
Python 2.7.5 (default, Apr  2 2020, 13:16:51)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Version: 3.14.3
Number of Devices: 1
Opening first device
Configure and start first analog out channel
Configure analog in
Set range for all channels
Wait after first device opening the analog in offset to stabilize
Starting acquisition...
   done
DC: 0.265624309759V
```

```
VERSO Info                  Waiting for open communication with FEB...
SocketHandler::loadIPList   Loaded 1 IP address(es):
SocketHandler::loadIPList       > 192.168.0.2
VERSO Info                  Ping successful
SocketHandler::addSocket    VMMSocket added:
VMMSocket::Print                Name          : FEC
VMMSocket::Print                Bound to port : 6007
VMMSocket::Print                Status        : 4
SocketHandler::loadIPList   Loaded 1 IP address(es):
SocketHandler::loadIPList       > 192.168.0.2
VERSO Info                  Ping successful
SocketHandler::loadIPList   Loaded 1 IP address(es):
SocketHandler::loadIPList       > 192.168.0.2
VERSO Info                  Ping successful
VERSO Info                  Resetting VMM(s)
```

Figure 5.7: Signal from board after using program scope.
We have an amplitude of about 0.4 V too, which means that our oscilloscope is working. When using the oscilloscope it is important to calibrate the trigger correctly and it's frequency or else we might not capture our signal.

Conclusions

We see that we can properly retrieve signal with this program. It should be noted that we have to properly modify our trigger so we can capture our specific signal. That can be done changing the trigger function in our code. The basic reason for this program was to put this program in verso, so we can easily read our signals, without the need to use the bigger oscilloscope.

Bibliography

[1] Design and development of the Level1 Data Driver Card (L1DDC) for the New Small Wheel upgrade of the ATLAS experiment at CERN. Panagiotis K. Gkountoumis
[2] Wikipedia
[3] ATLAS NSW Electronics Specifications version v3.3
[4]The New Small Wheel Upgrade Project of the ATLAS Experiment Bernd Stelzer for the ATLAS Muon Collaborationa (available www.sciencedirect.com)

[5] https://stfc.ukri.org/research/particle-physics-and-particle-astrophysics/large-hadron-collider/
[6] Contents lists available at ScienceDirect, journal homepage: www.elsevier.com/locate/nima
[7] Investigations of the VMM readout chip for Micromegas Detectors of the ATLAS New Small Wheel Upgrade. Maximilian Paul Rinnagel
[8] User manual GPVMM3 board version 2.4 October 26 2018 Brookhaven national laboratory
[9] https://reference.digilentinc.com/reference/software/waveforms/waveforms-sdk/reference-manual