# Data Acquisition System for Quality Tests of the ATLAS Muon Endcap Trigger Chambers.

Jacek Wasilewski

July 2002

# Data Acquisition System for Quality Tests of the ATLAS Muon Endcap Trigger Chambers.

MSc Dissertation of Jacek Wasilewski, Technical University of Łódź

## Abstract of the dissertation

The ATLAS The Collaboration is building a general-purpose  pp detector which is design to exploit the full discovery potential of the high energy proton-proton interaction  Large Hadron Collider (LHC) at CERN.

The LHC offers a large range of physics opportunities, among which the origin of mass at the electroweak scale is a major focus of interest for ATLAS. The detector optimization is therefore guided by physics issues such a sensitivity to the largest possible Higgs mass range.

The Thin Gap Chambers (TGCs) are detectors designed to detect the high transverse momentum muons in the endcaps of the ATLAS detector. The short response time of the TGCs makes it an ideal trigger system for selecting interesting events in the highly packed environment of the LHC accelerator.

The TGCs are designed and built in Weizmann Institute and are tested at the Tel-Aviv University and at the Technion.

The subject of this dissertation is the design and operation of the data acqusition system, which serves to automatize the procedure of testing the performance of the TGC detector, before are to be installed in the ATLAS experiment.

## Dissertation Committee

## Chair

Professor Andrzej Napieralski, Department of Microelectronics and Computer Science at Technical University of Łódź, Poland.

## Supervisor

Mgr Zbigniew Kulesza, Department of Microelectronics and Computer Science,Technical University of Łódź, Poland.
Dr Erez Etzion, School of Physics and Astronomy, Tel-Aviv University, Israel.

## Acknowledgements

I would like to thank Prof. Andrzej Napieralski and mgr Zbigniew Kulesza for giving me this unique opportunity to join the research program of the Tel Aviv group during the preparation of my dissertation. This allowed me to have a hands-on experience with the VME standard and its application to data acquisition systems. I am grateful to Dr Erez Etzion for hosting me at Tel Aviv University, introducing me into the activities of his laboratory and for providing professional help. I am also grateful to Prof. Halina Abramowicz for her support in this undertaking.

I am deeply indebted to my friends and colleagues from the lab., Meni Ben-Moshe and Jonatan Ginzburg for their daily professional and personal support. Without their company and help I would have been lost. I am also very appreciative of the support given to me by Dr Juliana Cohen. I am greatly indepted to Dr Yan Benhamu for showing me all kinds of new tricks to make my work more efficient and I will always remember the time we worked together, however short it was.

I also wish to express my thanks to Yoram Granitzky and Eduardo Warszawski from the Technion for their help with the hardware and software design. Thanks also go to Marcus Joos from CERN and to Galia Gilitz from the Weizmann Institute for their support with the VME software.

I cannot possibly name all the people who made my work easier and my stay in Israel more pleasant. But I thank them all from my heart.

I dedicate this dissertation to my parents. Without them I would have never achieved what I did.

Table Of Contents.

1 Introduction ........ 8
1.1 The Large Hadron Collider ........ 8
1.2 A Toroidal LHC Apparatus ........ 9
1.2.1 Components of the ATLAS detector ........ 10
1.2.2 Moun chamber layout. ........ 11
1.2.3 Trigger and data-acquisition system. ........ 12
1.3 Thin Gap Chamber. ........ 12
1.3.1 Principle of operation of TGCs. ........ 14
1.3.2 TGC test bench ........ 14
2 Readout system. ........ 15
2.1 The position measurement system ........ 15
2.1.1 The GassiPlex. ........ 16
2.1.2 The V551B CAEN C-RAMS SEQUENCER. ........ 18
2.1.3 The two channels C-RAMS ........ 21
2.2 TGC readout scheme ........ 23
2.2.1 The 128 channels Multihit Time to Digital Conversion ........ 23
2.2.2 The Amplifier Shaper Discriminator ........ 26
2.2.3 The ASD READOUT ........ 28
2.2.4 The ASD TO TDC. ........ 29
2.2.5 The NIM to ECL translator. ........ 30
2.3 Trigger system. ........ 30
2.4 Electronics general scheme. ........ 33
3 VME Interface and functionality ........ 35
3.1 SEQUENCER ........ 35
3.1.1 The SEQUENCER registers ........ 36
3.2 C-RAMS ........ 39
3.2.1 C-RAMS registers ........ 39
3.3 TDC ........ 42
3.3.1 The TDC registers. ........ 42
3.3.2 TDC programming procedure ........ 46
4 VME Software ........ 49
4.1 Drivers ........ 49
4.1.1 The memory mapping ........ 49
4.1.2 The VMEbus error detections ........ 50
4.2 Online program structure ........ 51
4.2.1 The SEQUENCER setting ........ 52
4.2.2 The C-RAMS setting ........ 54
4.2.3 The TDC setting ........ 55
4.3 Pedestals and Thresholds ........ 56
4.4 The Data Acquisition program ........ 58
4.5 Data acqusition system monitoring ........ 61
5 Results ........ 63
5.1 The timing of the Scintillators ........ 63
5.2 Example - The results of unit U08F2I-521.0 ........ 64
5.3 Another example - The result of unit U08F2I-548.9. ........ 67
6 Summary ........ 68

7 Bibliography ........................................................................69
Appendix I: VMEBUS Specification ...............................................70
Appendix II: VMEbus Application Program Interface ............................73
Appendix III: A structure of binary output file ................................83

List of figures.

*Figure 1.1 Overall layout of the ATLAS Detector*......9
*Figure 1.2 Components of a Detector Atlas*......10
*Figure 1.3 Three-dimensional view of the muon spectrometer instrumentation indicating the areas covered by the four chamber technologies.*......11
*Figure 1.4 Block diagram of the Trigger/DAQ system.*......12
*Figure 1.5 Basic configuration of TGC.*......13
*Figure 1.6 Electric field lines and potentials in a TGC.*......13
*Figure 1.7 TGC structure showig anode wires, graphite cathodes, G-10 layers, and a read-out strip orthogonal to the wires.*......13
*Figure 1.8 Schematic diagram of cosmic ray telescope.*......14
*Figure 2.1 Functional block of a GassiChip*......16
*Figure 2.2 The readout sequence of GassiPlex.*......17
*Figure 2.3 A typical distribution of the pedestal for one channel of the GassiChip.*......18
*Figure 2.4 Model v551B. The front panel.*......19
*Figure 2.5 Standard operation sequence.*......21
*Figure 2.6 Model v550. The front panel.*......22
*Figure 2.7 Model v767. The front panel and Input connector pin assignment.*......24
*Figure 2.8 Common Stop Emulation*......25
*Figure 2.9 Common Stop Sequence.*......26
*Figure 2.10 The ASD board*......27
*Figure 2.11 Block diagram of the ASD chip.*......27
*Figure 2.12 The signal propagations of ASD chip.*......27
*Figure 2.13 The ASDR top and back panel and the input connector pin assignment.*......28
*Figure 2.14 The TGC output signal from the ASDR*......29
*Figure 2.15 The ASDTDC top panel and input connectors pin assignment.*......29
*Figure 2.16 The ASD TO TDC connections system layout.*......30
*Figure 2.17 Schematic diagram of the trigger logical connection.*......31
*Figure 2.18 Scheme of the trigger design.*......31
*Figure 2.19 The TDC trigger (channel 1), signal from the TGC (channel 2) oscilogram.*......32
*Figure 2.20 The leyout of the readout system.*......33
*Figure 2.21 The C-RAMS-SEQUENCER connections layout.*......34
*Figure 2.22 The ASDR-ASDTDC-TDC connections  layout.*......35
*Figure 3.1 SEQUENCER Status Register*......36
*Figure 3.2 SEQUENCER Number of Channels Register*......37
*Figure 3.3 SEQUENCER T1 Register*......37
*Figure 3.4 SEQUENCER T2 Register*......37
*Figure 3.5 SEQUENCER T3 Register*......38
*Figure 3.6 SEQUENCER T4 Register*......38
*Figure 3.7 SEQUENCER T5 Register*......38
*Figure3.8 C-RAMS Word Counter Register*......39
*Figure 3.9 Mod. v550 FIFO structure*......40
*Figure 3.10 C-RAMS Number of channels*......40
*Figure 3.11 C-RAMS Status Register*......40
*Figure 3.12 C-RAMS Pedestal/Threshold memories*......41
*Figure 3.13 TDC Output Buffer Data Structure*......43
*Figure 3.14 TDC Status Register 1*......43
*Figure 3.15 TDC Status Register 2*......44
*Figure 3.16 TDC Control Register 2*......45

*Figure 3.17 TDC Opcode Handshake Register....................45*
*Figure 3.18 TDC Opcode Register....................46*
*Figure 3.19 Mod. V767 R/W operations: software logic blocks. ....................47*
*Figure 3.20 Mod. V767 OPCODE Word ....................47*
*Figure 4.1 Example of the relationship between the virtual address spaces of two processes and physical memory....................50*
*Figure 4.2 Mapping of the Virtual Memory ....................50*
*Figure4.3 The SEQUENCER timing oscilogram ....................53*
*Figure 4.4 The oscilogram of the signals from the Precision Chambers....................53*
*Figure 4.5 The signal from the PC when the CLOck pulse is not passing all GassiChips. .....54*
*Figure 4.6 Readout sequence inside GassiChips ....................54*
*Figure 4.7 The back window width of the TDC....................55*
*Figure 4.8 Typical distribution of pedestals of top Presicion Chamber ....................56*
*Figure 4.9 Typical distribution of pedestals of the bottom Presicion Chamber ....................57*
*Figure 4.10 The sigma for all channels of both Precision Chambers....................58*
*Figure 4.11 The example of the hit of the Precision Chamber....................58*
*Figure 4.12 The structure of data acquisition routine. ....................58*
*Figure 4.13 Algoritm of the online code....................60*
*Figure 4.14 The monitoring....................61*
*Figure 4.15 The monitoring....................62*
*Figure 5.1 The timing of the Scintillators....................63*
*Figure 5.2 The efficiency map of unit U08F2I-521.0....................65*
*Figure 5.3 The timing of the unit U08F2I-521.0 ....................65*
*Figure 5.4 The efficiency map of unit U08F2I-548.9....................67*
*Figure 5.5 The timing of the unit U08F2I-548.9....................68*

## List Of Tables.

*Table 1.1. The LHC general parameters....................8*
*Table 1.2 Principal TGC parameters....................12*
*Table 2.1 The inputs and outputs of the SEQUENCER....................20*
*Table 2.2 The inputs and outputs of the C-RAMS ....................23*
*Table 2.3 The inputs and outputs of the TDC....................25*
*Table 3.1 SEQUENCER Address Map....................36*
*Table 3.2 C-Rams Address Map....................39*
*Table 3.3 Address Map for the TDC....................42*
*Table 3.4 Mod. V767 OPCODE Words....................49*
*Table 5.1 The efficiency of the unit U08F2I-521.0....................65*
*Table 5.2 The efficiency of the unit U08F2I-548.9....................68*
*Table AIII.1: The structure of the binary online output file. ....................84*

# 1 Introduction

## 1.1 The Large Hadron Collider

The Large Hadron Collider (LHC) is a proton-proton (pp) collider designed to reach a centre of mass energy of 14 TeV and a luminosity of $10^{34}cm^{-2}s^{-1}$. The design parameters of LHC are listed in Table 1.1. Typical of such a high energy accelerator is a beam crossing time of 25 ns with about 23 pp interactions per crossing.

The LHC is an unprecedented project, not only in terms of centre-of-mass energy and luminosity of the colloding beams, but also in terms of cost, complexity and size of the experiments for which it is designed. It is supported by strong physics motivation, which is to understand the origin elementary particle masses, and to search for physics beyond the Standard Model of fundamental interactions.

| **LHC General Parameters** | | |
|---|---|---|
| Energy at collision | 7 | TeV |
| Energy at injection | 450 | GeV |
| Dipole field at 7 TeV | 8.33 | T |
| Coil inner diameter | 56 | mm |
| Distance between aperture axes (1.9 K) | 194 | mm |
| Luminosity | 1 E34 | $cm^{-2}s^{-1}$ |
| Beam beam parameter | 3.6 | $10^{-3}$ |
| DC beam current | 0.56 | A |
| Bunch spacing | 7.48 | M |
| Bunch separation | 24.95 | Ns |
| Number of particles per bunch | 1.1 | $10^{11}$ |
| Normalized transverse emittance (r.m.s.) | 3.75 | µm |
| Total crossing angle | 300 | µrad |
| Luminosity lifetime | 10 | H |
| Energy loss per turn | 7 | KeV |
| Critical photon energy | 44.1 | eV |
| Total radiated power per beam | 3.8 | kW |
| Stored energy per beam | 350 | MJ |
| Filling time per ring | 4.3 | min |

**Table 1.1. The LHC general parameters.**

## 1.2 *A Toroidal LHC Apparatus*

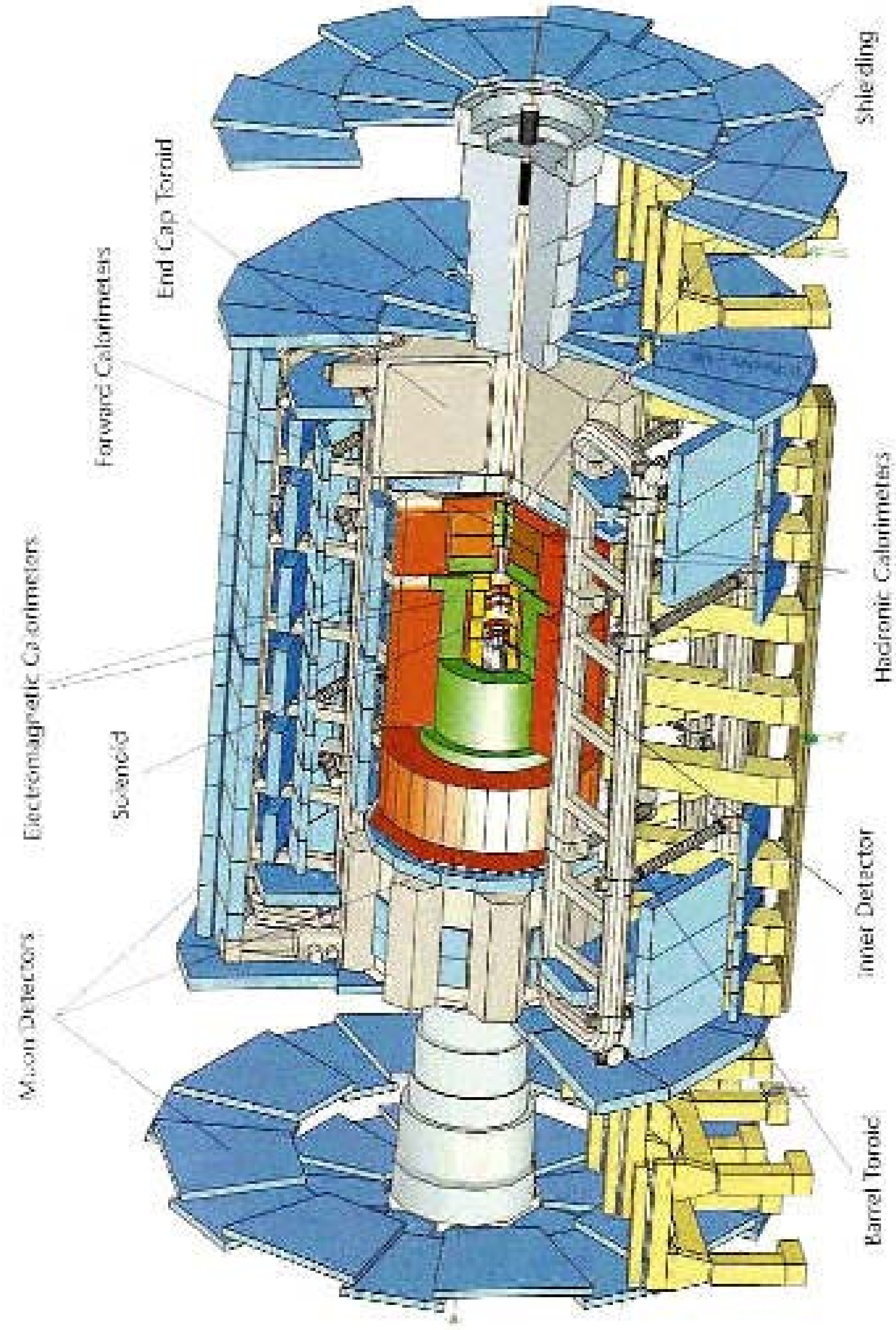


**Figure 1.1 Overall layout of the ATLAS Detector**

Two experiments have been proposed and approved to study pp interactions. One of them is the so called ATLAS expermient (ATLAS stands for **A** **T**oroidal **L**HC **A**paratu**S**) and one of its components, the forword muon trigger, is related to the work presented in this thesis. The overall ALTAS detector layout is shown in Figure 1.1. The basic criteria of the detector design [1] include the following:

- very good electromagnetic calorimetry for electron and photon indentification and measurements;
- high-precision muon momentum measurements;
- large acceptance in pseudorapidity, η, where $\eta = -\ln ctg\frac{\Theta}{2}$ and Θ is the polar angle of produced particles with respect to the beam line.
- triggering and measurement of particles with a low transverse-momentum, $p_T$, threshold;

## 1.2.1 Components of the ATLAS detector

The reason that high energy detectors are divided into many components is that each component tests for a special set of particle properties. These components are stacked so that all particles go through the different layers sequentially. A particle will not be detected unless it either interacts with a given detector component in a measurable fashion, or decays into detectable particles.

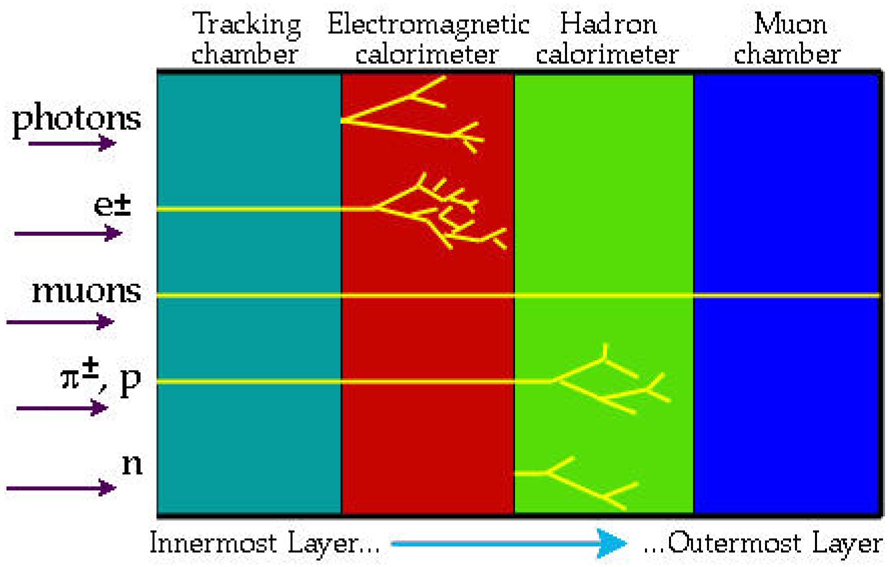


**Figure 1.2 Components of a Detector Atlas**

An example of how various particles are detected is shown in Figure 1.2
As it happens, each particle type has its own "signature" in the detector.

- The trajectory of charged particles are detected in the tracking chambers.
- Photons and electorns are absorbed in the electromagnetic sections of the calorimeter.
- Strongly interacting particles like neutrons, protons and pions are absorbed in the hadron calorimeter.
- Characteristic of muons is that they interact very weekly with matter and their trajectories can be observed in tracking chambers (muon chambers) placed as the outermost layer.

### 1.2.2 Moun chamber layout.

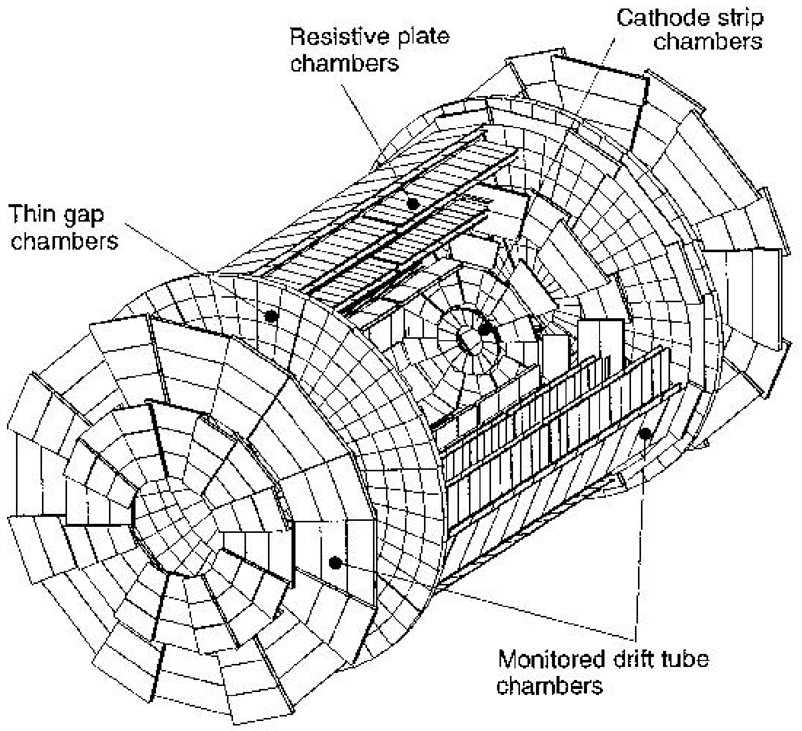


**Figure 1.3 Three-dimensional view of the muon spectrometer instrumentation indicating the areas covered by the four chamber technologies.**

The overall layout of the muon chambers in the ATLAS detector is shown in Figure 1.3. For reasons of costs and detection efficiency, four different chamber technologies are used as indicated in the figure. The chambers are arranged such that particles from the interaction point traverse three stations of chambers. The barrel chambers form three cylinders concentric with the beam axis, at radii of about 5, 7.5 and 10 m. They cover the pseudorapidity range $|\eta|<1$. The end-cap chambers cover the range $1<|\eta|<2.7$ and are arranged in four disks at distances of 7, 10, 14 and 21 – 23 m from the interaction point, concentric with the beam axis. The trigger function in the end-caps is provided by first three discs, the three stations of Thin Gap Chambers (TGC). The three stations consist of 192 TGC units and 440,000 readout channels, and cover an area of 2900 $m^2$.

### 1.2.3 Trigger and data-acquisition system.

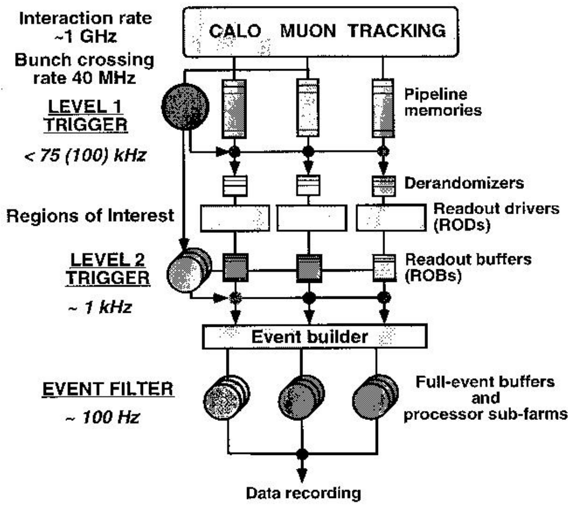


**Figure 1.4 Block diagram of the Trigger/DAQ system.**

The ATLAS trigger and the data-acquisition (DAQ) system is based on three levels of online event selection [3]. Each trigger level refines the decisions made at the previous level and, where necessary, applies additional selections criteria. Starting form an initial bunch-crossing rate of 40 MHz (interaction rate of ~$10^9$ Hz at luminosity of $10^{34}$ $cm^{-2}s^{-1}$), the rate of selected events must be reduced to ~100 Hz for permanent storage. While this requires an overall rejection factor of $10^7$ against the so called 'minimum-bias' events, excellent efficiency must be retained for the rare new physics processes, such as Higgs boson production and decay, which will be searched for in ATLAS. Figure 1.4 shows a simplified functional view of the Trigger/DAQ system.

An essential requirement on the level 1 trigger is that it should uniquely identify the bunchcrossing of interest within the short (25 ns) bunch-crossing intetrval.

The TGCs were selected for the level one muon trigger becouse of their adequately short response time. The timing resolution of the system is such that there is a 99% efficiency for signals arriving within the 25ns gate.

## *1.3 Thin Gap Chamber.*

The parameters characterising a single TGC unit are given in Table 1.2.

| | |
|---|---|
| Gas gap | 2.8 mm |
| Anode wire pitch | 1.8 mm |
| Wire diameter | 50 μm |
| Wire potential | 3100 V |
| Gas mixture | $CO_2$/n-$C_5H_{12}$(55%/45%) |
| Gas amplification | $10^6$ |
| Anode r/o pitch | 7.2-39.0 mm |
| Time resolution | >99% efficienty for 25 ns gate |
| Rate capability | Tested at 30 kHz/$cm^2$ |
| Read-out strip width | 14.6-49.1 mm |
| Total number of anode r/o channels | 280 000 |
| Total number strip r/o channels | 95 000 |

**Table 1.2 Principal TGC parameters.**

The TGC unit consists of a plane of equally spaced anode wires centered between two cathode planes as shown in Figure 1.5 [2]. The volume between two cathodes is field with a quenching gas.

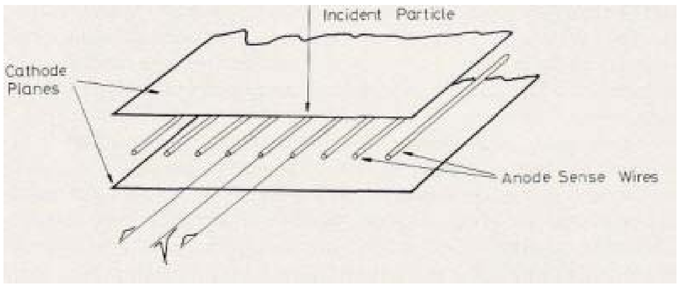


**Figure 1.5 Basic configuration of TGC.**

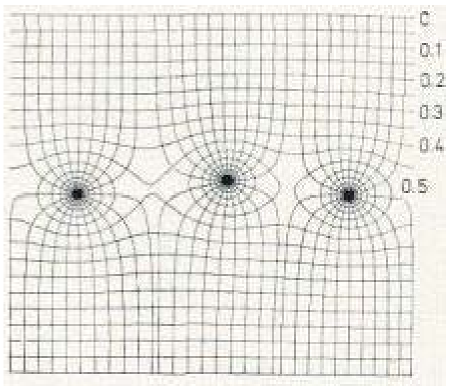


**Figure 1.6 Electric field lines and potentials in a TGC.**

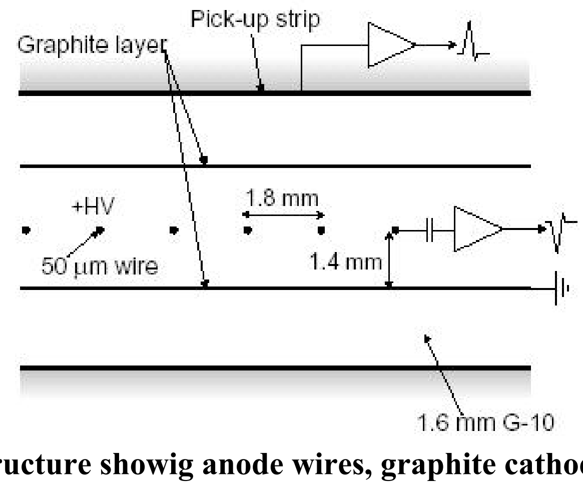


**Figure 1.7 TGC structure showig anode wires, graphite cathodes, G-10 layers, and a read-out strip orthogonal to the wires.**

Each wire sandwiched between two cathode planes acts as an independent proportional counter. The operating high voltage foreseen is 3.1 kV. The electric field configuration (see Figure 1.6) and the small wire distance provide for a short drift time and thus a good time resolution [2]. Except for the region very close to the anode wires, the field lines are essentially parallel and almost constant.

If electrons and ions are now liberated in the constant field region they will drift along the field lines toward the nearest anode wire and opposing cathode. Upon reaching the high field region, the electrons will be quickly accelerated to produce an avalanche. The positive ions liberated in the multiplication process then induce a negative signal on the anode wire.

The neighboring wires are also affected; however, the signals induced here are positive and of small amplitude as illustrated in Figure 1.7. In a similar manner, a positive signal is induced on the cathode. There is thus no ambiguity as to which wire is closest to the ionizing event.

### 1.3.1 Principle of operation of TGCs.

The TGCs are constructed in doublets and triplets. A TGC doublet is a unit which has two chambers. A TGC triplet has two chambers and one plane of wire between them. A chamber is a single layer of active detector element. For TGCs, this is a wire plane and its strips.

To obtain high granularity, good time resolution, high rate capabilities and ageing characteristics, a highly quenching gas that avoids streamers, while permitting a saturated operation is needed. This is achieved with a mixture of CO2–*n*-pentane (55%–45%), where the *n*-pentane plays the role of the quencher. No other gas that is less flammable has been found that provides the above performance.

### 1.3.2 TGC test bench

The TGC units are produced in the Weizmann institute in Israel and also in KEK in Japan and in China. There are three test benches to check the performance of TGC modules, two in Israel, in Technion and Tel-Aviv, and one in Kobe University in Japan. The purpose of this tests is too-fold. For each detector a detailed map of detection efficiency for the wires and strips is determined as well as their respective time resolution.

At the test site the TGC units pass the following steps of processing

- Preliminary checks
- Efficiency test
- Validation (Acceptance/Rejection)

A visual inspection of the incoming TGC units is performed to check that no damage was done to them during transport.

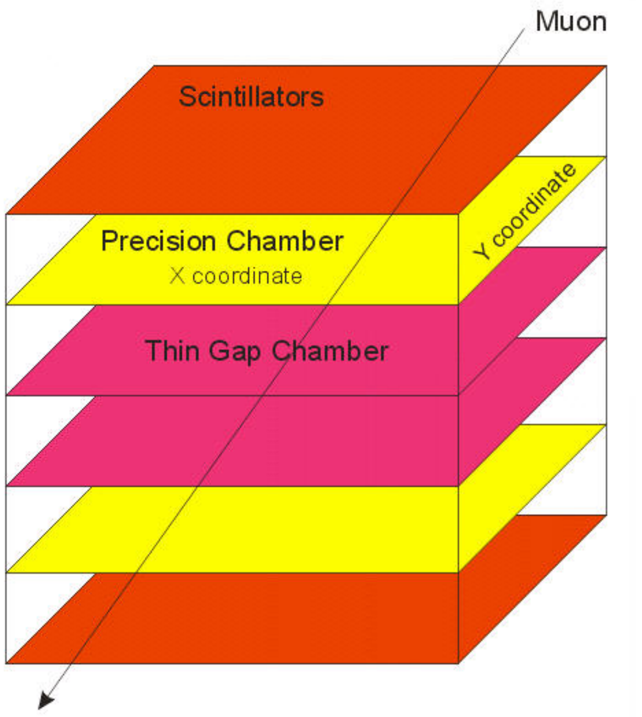


**Figure 1.8 Schematic diagram of cosmic ray telescope.**

The cosmic ray telescope is 2,2 m high, 1,6 m wide and 2,5 m long. There are 11 slots with a distance of 20 cm between them. The top and the bottom layers consist of scintillators planes. A signal in coincidence from the two planes signals the passage of a cosmic muon. The next layers from the bottom and from the top consist of the so called Precision Chambers (PC). The latter are made of TGC detectors which differ from the tested TGC chambers by their readout granularity. Their role is to accurately measure the positions where the muon crossed the chambers.

Accumulating events for a period of one week will permit a full mapping of the efficiency of each detector in the stack. The criteria for a good chamber is having 95% of its active area efficient at level of higher then 95% within a 25 nsec gate. If the unit failed the test it is removed from the test bench and returned to the Weizmann institute. After successfully passing this test, the detectors are flushed with $CO_2$, sealed and prepared for the transportation to CERN.

In the following, we present the test bench for the TGC chambers as designed and built at Tel Aviv University. In the chapter two I am describing the hardware readout sytem, in the chapter three I am presenting the VME interface and functionality, the chapter four is describing the online software. The results of the testing are shown in the chapter five.

# 2 Readout system.

## 2.1 The position measurement system

A Precision Chamber has similar construction as the TGC. The Precision Chamber has two orthogonal layers of strips which are built with the high granularity to provide a precise crossing moun position.

The readout system of the Precision Chamber is based on the C-Mos VLSI frontend chip GassiPlex, composed of four GassiPlex chips of 16 channels per one GassiBoard. A number, N, of chips can be "daisy chained", allowing the analog multiplexed outputs of N * 16 channels to be sent on a single line to a VME-read out module. Unit such as CAEN Readout for Analog Multiplexed Signals (CRAMS) performing the digitalization and zero suppresion of that analog signal sequence. The multiplexing and digitalization are two synchroneous operations operated by the same clock.

In case of a large number of channels, such a multiplexed scheme allows to minimize the number of expensive redout modules. The multiplexing frequency of 5 MHz allows a maximum data taking rate of a few hundred kHz for 16 channels.

### 2.1.1 The GassiPlex.

The single GassiChip consists of 16 channels as shown in Figure 2.1.

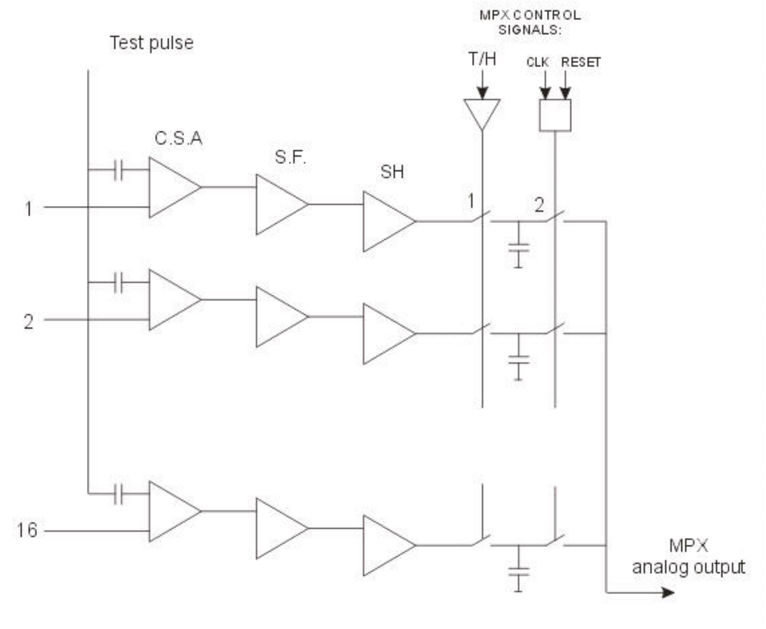


**Figure 2.1 Functional block of a GassiChip**

Each channel is composed of a charge sensitive amplifier (CSA), a switchable filter (SF), a shaping amplifier (SH) and Track/Hold stage (T/H).

The GassiChip is an ungated, then asynchronous, device, meaning that its inputs/outputs are always in a "live stage". Therefore, each output, or baseline, has to be kept at a constant DC-level in the absence of an input signal. That level is called "pedestal", evaluated in mV. Measuring the fluctuation of the pedestal level gives the noise level of that channel (in practice, as the sigma of a pedestal value distribution).

The shaping stage symmetries the pulse shape and allows a slight adjustment of the peaking time at 700 ns ± 100 ns via external bias resistor. That time is taken as a natural built-in delay used to wait for the external trigger decision.

In the operation mode, all the switches driven by the T/H stage are closed and those driven by the Multiplexed (MPX) stage open in the whole system. At the occurrence of an event, the capacitors associated with the hit channels are charged up by the corresponding detector currents. An external trigger signal, synchronous to that event, is used to generate a "HOLD" signal, sent to the whole system, in order to open all switches at a time corresponding to the peaking time of the shaping amplifier. Therefore, a charge particle crossing the detector leaves a signal in the strips close to the crossing point a signal which is larger then their pedestal level.

As long as the HOLD signal is kept active, the switches are open and the charges are freezed in the capacitors. As soon as the HOLD signal is released, the switches are closed and the charges is lost.

The decision for reading out an event (charges stored in the capacitors) is independent of the T/H operation. A train of MPX clock pulses is generated by an external device such as a SEQUENCER. The SEQUENCER allows adjusting the number and frequency of the clock pulses.

As soon as the clock train is over a RESET signal is sent to the GassiPlex, repositioning all the switches to their initial positions. That signal can be sent after the HOLD signal is released.

After the HOLD signal is generated, the trigger logic is vetoed by a BUSY signal to protect the whole system from taking new events. That protection is released by the SEQUENCER when all the tasks are terminated.

An example of the analog multiplexing train is shown in Figure 2.2. The clock train is sent also to the readout module where the digitization is performed. The digital conversion (and the suppression) of each analog step is done between two clock pulses, therefore:

- The ADC convert train must be synchroneous to the MPX clock train (same source). The SEQUENCER should give the possibility to manually adjust a delay between the 2 trains in order to enable the ADC conversion to be done in a stable part of each analog step.
- Although the two clock trains are issued from the same source, the MPX clock train is propagated trought all the GassiPlex chips to operate the MPX function. Therefore, the train is accumulating a delay (2 ns per channel, 32 ns per chip) with respect to the ADC convert train. If the time between two clock pulses is too short, the synchronization requirement is lost after a certain number of clock pulses.

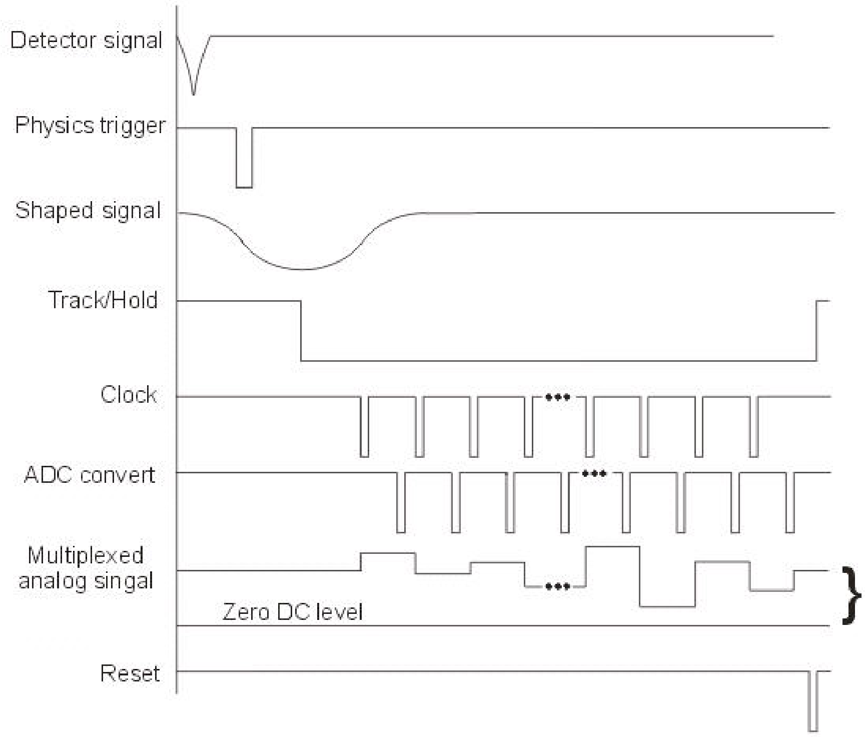


**Figure 2.2 The readout sequence of GassiPlex.**

In order to define a threshold value for every channel, one has to measure all pedestals level, PED(i), their RMS (noise figure), and sigma SIG(i). The usual way is to calculate the threshold table as TH(i) = PED(i) + N*SIG(i) where N is a selectable constant usually larger then three. In our case 2000 events are taken to calculate the threshold value. To minimize the noise level we choose to work with N equal to three. The threshold value for each channel is written to the pedestal and threshold memory of the C-RAMS [4]. The typical distribution of one channel of the PC is shown in Figure 2.3.

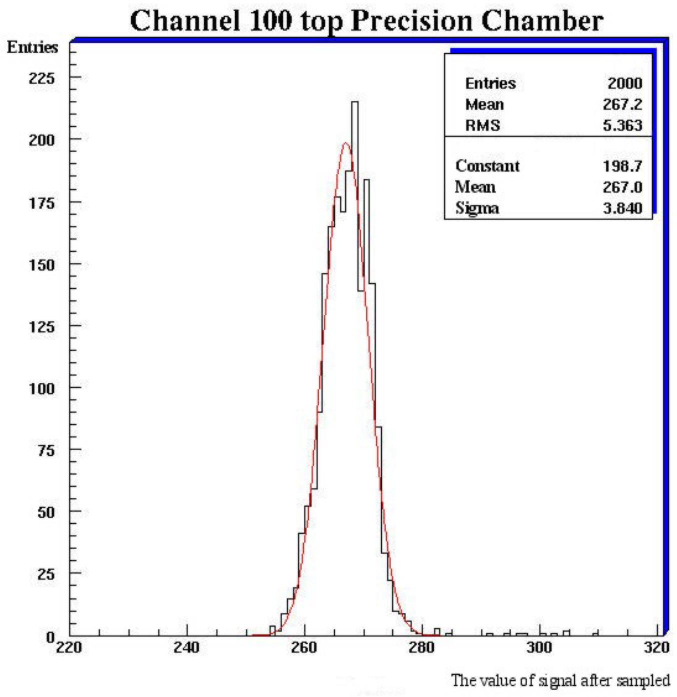


**Figure 2.3 A typical distribution of the pedestal for one channel of the GassiChip.**

### 2.1.2 The V551B CAEN C-RAMS SEQUENCER.

The V551B CAEN C-RAMS SEQUENCER (SEQUENCER) is a one-unit wide VME module that handles the Data Acquisition from multiplexing front-end chips, its front panel is shown in Figure 2. and its inputs outputs are described in the Table 2.1 [5]. The SEQUENCER has been developed to control the signals from/to the C-RAMS boards Mod. V550. A single SEQUENCER can control up to 19 C-RAMS modules in a complete VME crate, thus enabling the readout of 19*(2*2016) = 76608 multiplexed detector channels. The Mod. V551B is controlled via VME bus. The number N of detector channels to be read out by the C-RAMS can be programmed via VME up to 2047 (though the V550 C-RAMS can accept only 2016 detector channels). The multiplexing frequency can be set via VME from 100 kHz to 5 MHz, with programmable Duty Cycle. The delay between the multiplexing Clock signal and the Convert signal of the acquisition cards can be adjusted to wait for the settlement of the analog signal coming from the multiplexers. The delay between the TRIGGER and the HOLD signal and the delay between the HOLD and the Conversion Cycles (CONVERT) are also programmable.

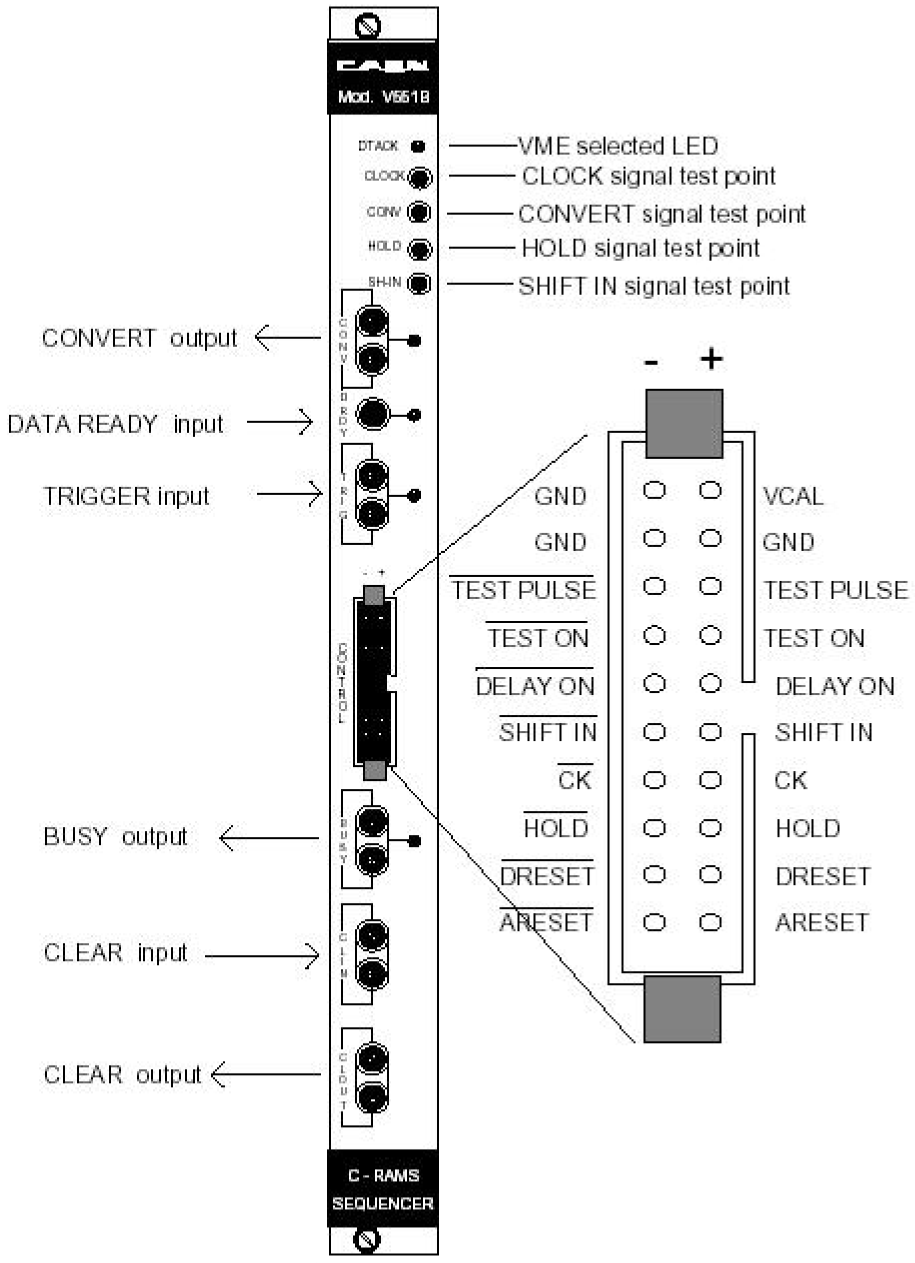


**Figure 2.4 Model v551B. The front panel.**

| **Inputs** | | | |
|---|---|---|---|
| **Signal** | **Standard** | **Active** | **Connector** |
| DATA READY | TTL | High | LEMO |
| TRIG | NIM | High | LEMO |
| CLEAR IN | NIM | High | LEMO |
| **Outputs** | | | |
| CONVERT | NIM | High | LEMO |
| BUSY | TTL | High | LEMO |
| CLEAR OUT | NIM | High | LEMO |
| VCAL | Analog Voltage | Positive or negative | CONTROL |
| TEST PULSE | TTL differential level | High | CONTROL |
| TEST ON | TTL differential level | High | CONTROL |
| DELAY ON | TTL differential level | High | CONTROL |
| SHIFT IN | TTL differential level | Low | CONTROL |

|  |  |  |  |
|---|---|---|---|
| CLOCK | TTL differential level | Low | CONTROL |
| HOLD | TTL differential level | Low | CONTROL |
| DRESET | TTL differential level | High | CONTROL |
| ARESET | TTL differential level | Low | CONTROL |

**Table 2.1 The inputs and outputs of the SEQUENCER.**

The readout sequence starts with an external or a VME TRIGGER. The BUSY becomes active, indicating that the module cannot accept another TRIGGER. The leading edge of the TRIGGER starts a monostable multivibrator circuit. After a t1 time programmable via VME the circuite activates the HOLD and the SHIFT IN signals. The HOLD signal is used to sample the signal at the output of the shapers at peaking time. Once the HOLD is asserted, a CLOCK cycles begins after a programmable t2 time. In the following readout sequence, N CLOCK and CONVERT pulses are generated. The number N of detector channels (between 1 and 2,047) can be programmed via VME. A CONVERT pulse follows each CLOCK pulse with a delay of t5 ns, programmable via VME.

The purpose of the delay is to wait for the relaxation of the analog signal coming from the multiplexers. The width of the active phase of the CLOCK and CONVERT pulses is t3 ns, programmable via VME. The V550 C-RAM accepts only CONVERT pulses with active phase $\geq$ 100 ns. The repetition period t4 of the CLOCK and CONVERT pulses (and consequently the multiplexing frequency) is programmable via VME. The CLOCK, CONVERT, HOLD and SHIFT-IN signals are available on four test points placed on the front panel. The active level of the test points is always high, disregarding of the normal level of the relevant signals on the front panel connectors.

With the last CONVERT pulse, the DRESET line (also the ARESET if enabled) becomes active for resetting the front-end circuitry. After the generation of the last CONVERT pulse, if none of the C-RAMS acqusition cards has asserted the DRDY signal the BUSY signal becomes inactive. Otherwise, when the DRDY raised (at least one channel has data ready), the BUSY signal remains high until all the C-RAMS FIFOs are read out, i.e. until the DRDY becomes inactive. An example of the sequence is shown in Figure 2..

Software controlled VETO is available as well. The VETO forces the V551B in a BUSY condition (no TRIGGER accepted).

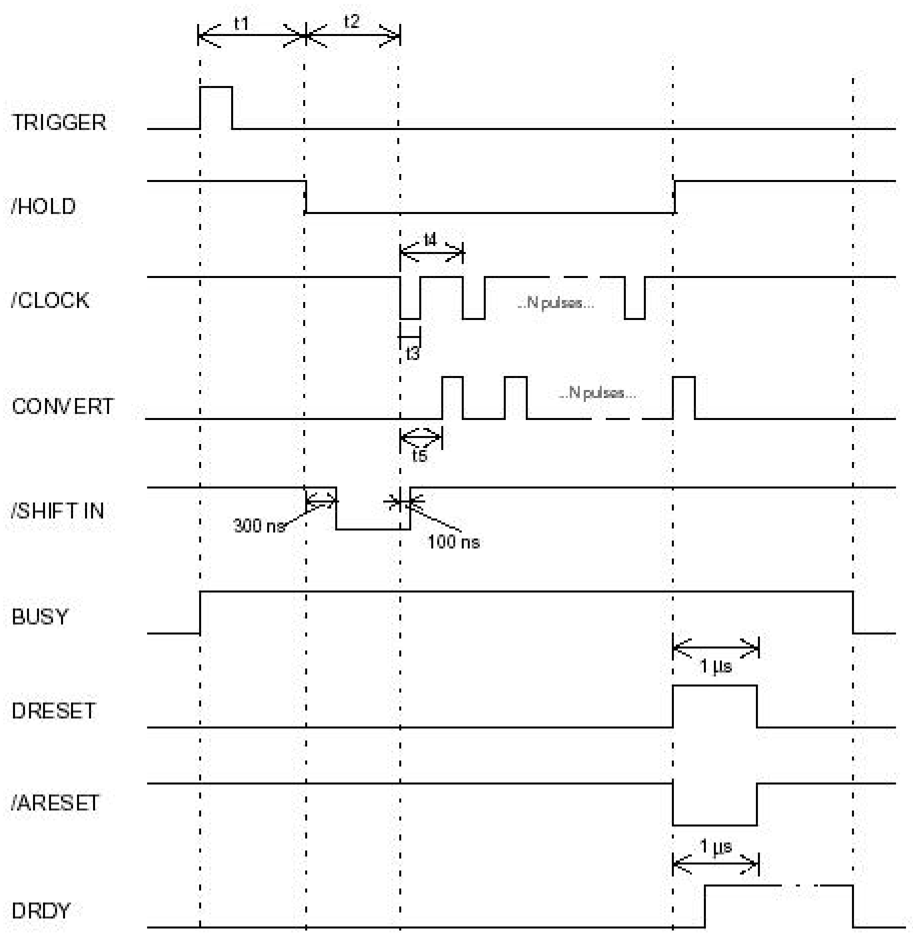


**Figure 2.5 Standard operation sequence.**

## 2.1.3 The two channels C-RAMS

The two channels C-RAMS is a one-unit wide VME module housing two independent Analog to Digital Conversion blocks to be used for the readout of analog multiplexed signals coming from several front-end chips [6]. The front panel is shown in Figure 2. and its inputs, outputs are shown in Table 2.2.

Each block of the module accepts positive, negative or differential input signals, the signals are amplified and fed to an ADC. The module has the following features:

- Conversion rate up to 5 MHz.
- Differential input with selectable amplification.
- 10 bit linear conversion.
- Zero suppression and pedestal subtraction.
- Diagnostics and self-test capabilities.

With the occurrence of an external CONVERT signal, the input signal is sampled by the ADC and its digital value is compared to a threshold value. If the signal is over threshold, the pedestal is subtracted and the result is stored in an Output Buffer arranged in FIFO logic 2K x 32 bit. For this purpose each block of the module houses two memories for the storage of the thresholds and the pedestals of each detector channel. The pedestal and threshold values are independent for each channel and the pedestal/threshold memory, which is arranged in 2K x 24 bit, can be filled (or read) via the VME.

The number of N detector channels to be read between 32 and 2016 in steps of 32out is controlled via VME. At the end of a conversion cycle (N CONVERT pulses), with the last word stored in the FIFO, if there is data in the FIFO, the module channel goes into the Data Ready state signaling that the data must be read via VME. A positive open collector signal ("DRDY") is available for each channel on the front panel and is provided with two bridged connectors for daisy chaining. A fast CLEAR signal is also available for cycle abort.

It is possible to operate the module in a TEST mode (VME selectable) by simulating some input patterns, as if they were coming from the ADC.

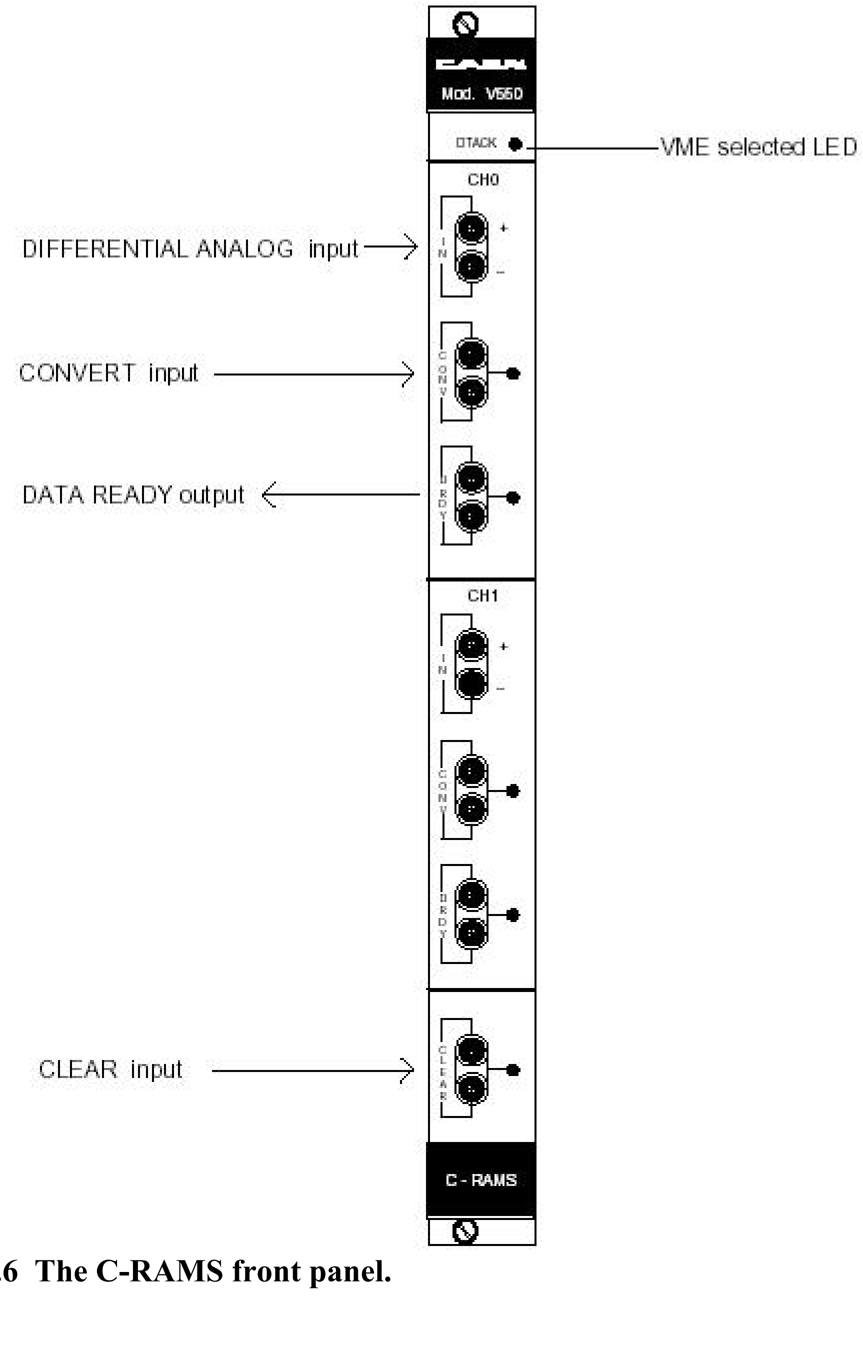


**Figure 2.6 The C-RAMS front panel.**

| **Inputs** | | | | |
|---|---|---|---|---|
| **Signal** | **Standard** | **Active** | **Connector** | **Minimum width** |
| INPUT CHANNELS | Positive or differential | | LEMO | |
| CONVERT | NIM | High | LEMO | 100 ns |
| CLEAR | NIM | High | LEMO | 50 ns |
| **Outputs** | | | | |
| DATA READY | TTL | High | LEMOI | |

**Table 2.2 The inputs and outputs of the C-RAMS**

The module can be used to calculate the values of the thresholds and the pedestals. For this purpose the threshold/pedestal memory must be zeroed via the VME. In this way the zero suppression and pedestal subtraction functions are disabled and the words written in the FIFO are the true converted values. The readout is performed in the way already described above. The VME CPU can compute the mean values and the variances which is used to determine the thresholds and pedestals to be stored back in the threshold/pedestal memory. The Memory Owner bit (MO) of the status register is used to switch the pedestal & threshold memory access between the VME bus (MO=0) and the channels Control Logic. At power on, the memorized pedestal and threshold values are not specified, therefore the pedestal and threshold memory must be initialized by the user.

## 2.2 TGC readout scheme

### 2.2.1 The 128 channels Multihit Time to Digital Conversion

The 128 channels Multihit Time to Digital Conversin (TDC) is a one-unit wide VME module that houses 128 independent Time to Digital Conversion channels. The unit houses four such TDC chips [7].

The TDC is a General Purpose time-to-digital converter, with 32 channels per chip. The integrated circuit is developed as a full custom device in CMOS 0.7 μm technology, allowing *Common Start* operations with a typical bin size of 0.8 ns. All channels can be enabled for the detection of rising and/or falling edges and for each channel there is a digital adjustment for the reset of any offsets and pedestals.

The data acquisition can be programmed in "EVENTS" (*Trigger Matching* with a programmable time window or *Start Gating* modes) or in "*Continuous Storage* " as well as the management of overlapping triggers. The *Common Stop* operation, though not existing in the chip itself, can be easily implemented on this board by assigning one of the 128 channels to a STOP signal and by an adequate programming of the trigger window.

The board houses a 32 kwords local memory FIFO buffer that can be read via the VME (as single data, Block Transfer and Chained Block Transfer) in parallel to the acquisition itself.

The programming is performed via a microcontroller that implements a high-level interface with the user in order to mask the board and the TDCs' hardware.

The unit accepts the following CONTROL signals (ECL differential) in common to all channels:

- START: a common START input;
- TRIGGER: a common TRIGGER input;
- RST: the RESET signal which allows to clear the buffers and reset the TDCs; upon programming, it can also reset other registers of the unit;
- CLOCK: providing an external Clock to the board.

The characteristics of the signals is shown in the Table 2.3.

The status of unit is shown by the LEDs on the front panel:

- DTACK lights up each time the module generates the VME signal DTACK;
- DATA READY lights up when the Data Ready condition occurs;
- BUSY lights up when no more data can be written;
- TERM ON lights up when all the lines of the CONTROL bus are terminated;
- TERM OFF lights up when no CONTROL bus line is terminated.

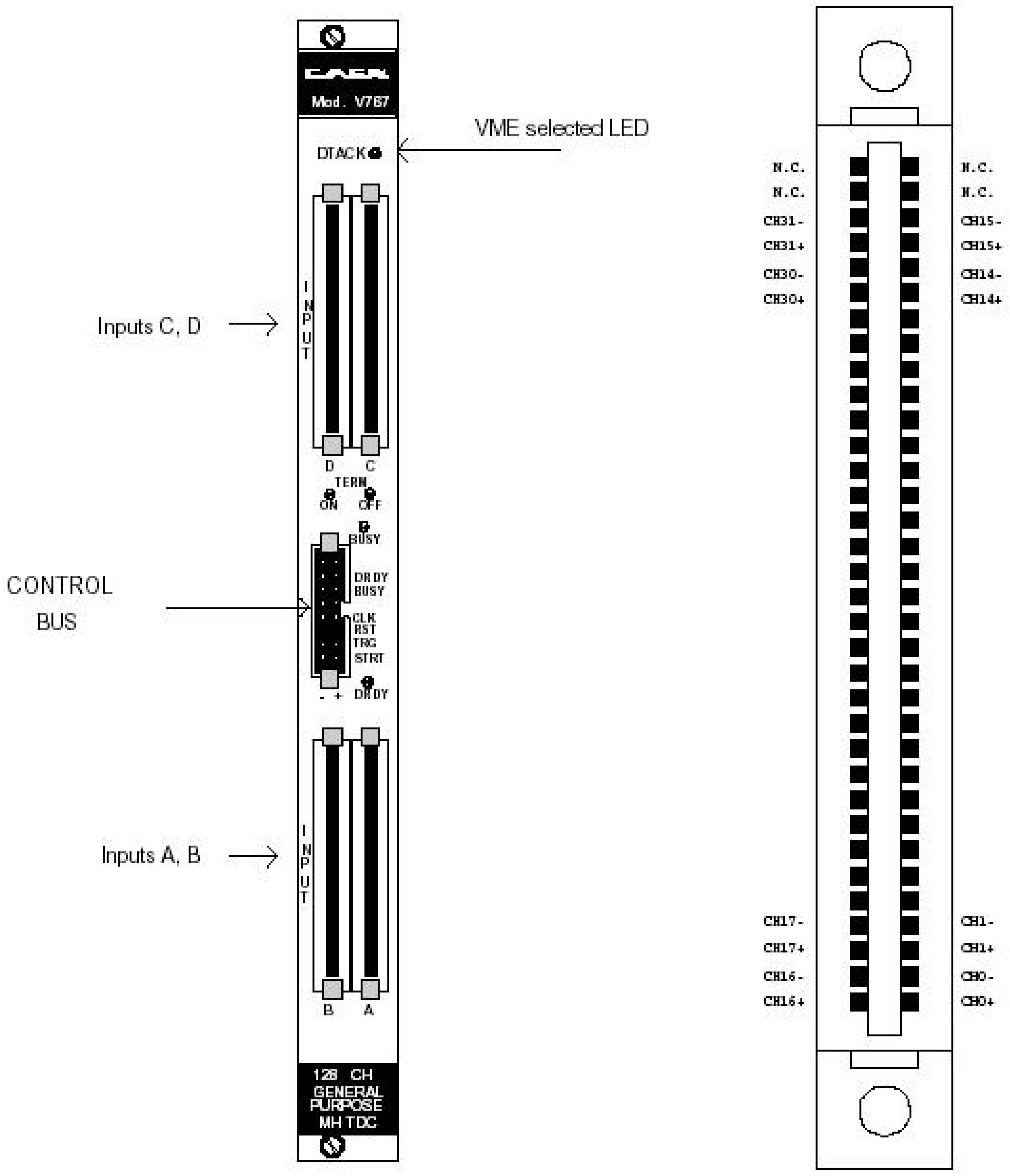


**Figure 2.7 The TDC front panel and Input connector pin assignment.**

| Inputs | | | |
|---|---|---|---|
| **Signal** | **Standard** | **Connector** | **Minimum width** |
| INPUT CHANNELS | ECL | Robinson Nugent | 10 ns |
| CLOCK | ECL, (rising egde active) | Header 3M | 25 ns |
| RESET | ECL, (active low) | Header 3M | 25 ns |
| TRIGGER | ECL, (rising egde active) | Header 3M | 25 ns |
| START | ECL, (rising egde active) | Header 3M | 10 ns |
| **Outputs** | | | |
| DRDY | ECL, active high | Header 3M | |
| BUSY | ECL, active high | Header 3M | |
| | | | |

**Table 2.3 The inputs and outputs of the TDC**

The input connectors refers to following channels:

- Connector A refers to Channels 0 to 31.
- Connector B refers to Channels 32 to 63.
- Connector C refers to Channels 64 to 95.
- Connector D refers to Channels 96 to 127.

The TDC operation is based on the functionality of the CERN/ECP-MIC TDC chip. Five different module setups are selectable via software for different acquisition scenarios, namely:

- *Stop Trigger Matching;*
- *Start Trigger Matching;*
- *Start Gating;*
- *Continuous Storage;*
- *Common Stop Emulation;*

It is possible to switch from one operation mode to the other by simply resetting and reprogramming the module (with this operation the data in memory will be lost). Since the trigger is delayed and the input signals from the TGCs come before the trigger, common stop emulation is used. The TDC chips do not foresee the common stop operation. It is implemented with the scheme shown in Figure 2.8 :

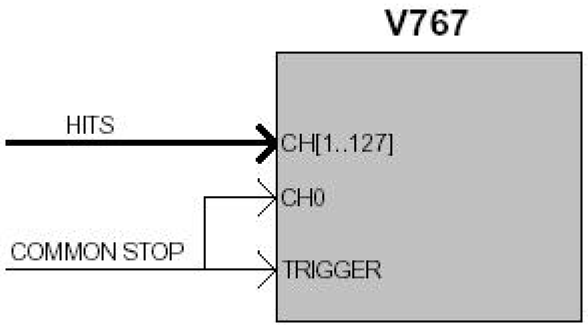


**Figure 2.8 Common Stop Emulation**

The *Common Stop Emulation* is done by programming the board to a *Stop Trigger Matching,* with the trigger window ending and the occurrence of the trigger/stop signal (see Figure 2.9).

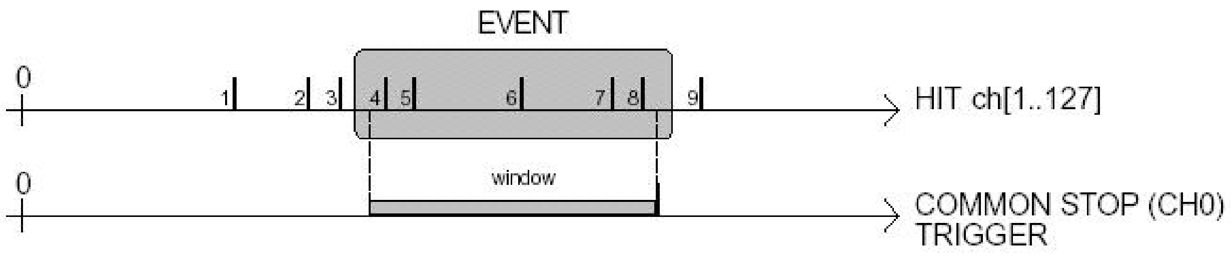


**Figure 2.9 Common Stop Sequence.**

The stored data consist of: absolute time measurement (the zero time reference is represented by the RESET) ; the timing information of the STOP (T stop – T hit) which can be obtained by the software. The event is stored in the local buffer of the board with the following structure:

HEADER event number
DATUM n.1 HIT time(4)
DATUM n.2 HIT time(5)
DATUM n.3 HIT time(6)
DATUM n.4 HIT time(7)
DATUM n.5 HIT time(8)
DATUM n.6 STOP time
EOB num. of data read = 6

### 2.2.2 The Amplifier Shaper Discriminator

TGC signals from wire-groups and strips pass through an Amplifier Shaper Discriminator IC (ASD-IC) placed on ASD Boards which are housed inside the shielding boxes attached to each TGC unit. An ASD Board is shown in Figure 2.11. The ASD-IC uses SONY Analog Master Slice technology (bipolar). Each ASD-IC contains four channels. The scheme of the one single channel is shown in Figure 2..

The ADC measures the signal charge in a given time window following the initial threshold crossing. The signal charge is encoded into pulse width (this is demonstrated in Figure 2.).This charge information enable to increase the accuracy of the timing measurement by performing time slewing correction. Two modes of operation are provided. In one mode the ASD output gives the time over threshold information, i.e. signal leading and trailing edge timing. The other mode measures the leading edge time and charge and is considered as the default-operating mode [8].

Its outputs signals are conform to the Low Voltage Differential Signaling standard, LVDS, to assure drivability and immunity against noise and minimizing power.

The 16-ch ASD board (ASDB) was designed to receive wire and strip signals from the TGCs. Each board contains four ASD ICs with protection circuits and a test pulse circuit that receives a test pulse and distributes a charge impulse to each channel. The board design is common for all TGC chambers. LVDS logic signals from the ASDB are transmitted through a 20 twisted-pair cable and a preamplifier output through a LEMO type connector. DC power (± 3V), ground, threshold voltage (common over a channel) and test pulse are supplied back to the ASDB via the same twisted-pair cable.

The circuit can successfully accept signals of 5 MHz or even higher frequencies.

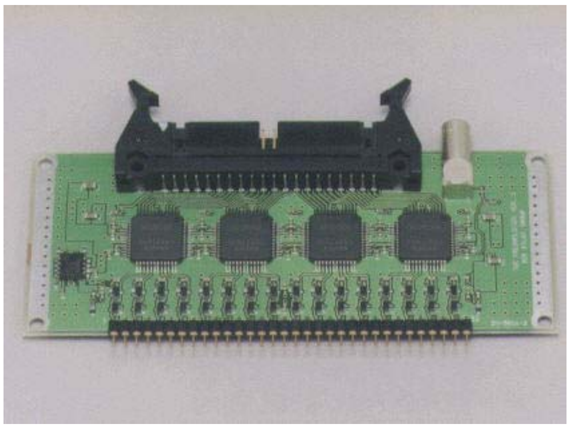


**Figure 2.110 The ASD board**

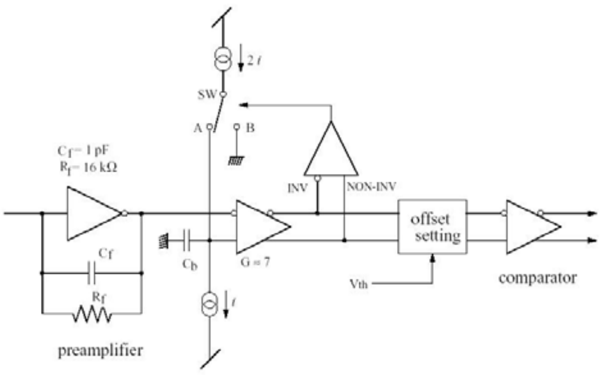


**Figure 2.11 Block diagram of the ASD chip.**

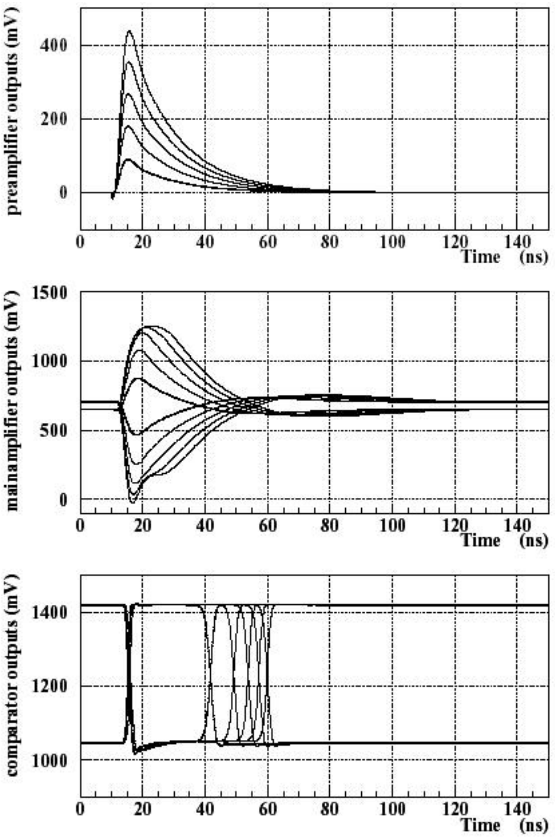


**Figure 2.12 The signal propagations of ASD chip.**

The Figure 2. shows the shape of the signal as measured in three different place in the ASD chip. First plot show the signal shape in the preamplifier, main-amplifier output shape is shown in the second plot where the comparator output is drawen in the last figure.

### 2.2.3 The ASD READOUT

The signals from ASDB pass through an ASD READOUT (ASDR) NIM card. ASDR consists of two boards of two inputs each. Each board accepts signals from one TGC layer via 20 twisted-pair cable, however it can accept either wires signals or strips signals. Since the TDC accepts only the ECL standard signals, ASDR converts the LVDS input signals to the ECL format. Signals from one input are summed and that sum is the one single output of this board. The input-output connector, Header 3M type, 8+8 pins, is placed on the top panel. The structure of the pins is shown in Figure 2.. These output signals are transmitted through an 8-pair flat-pair cable to an ASD to TDC NIM card. The trigger and the test pulse are supplied back from the ASD to TDC via the same 8-pair flat-pair cable. The layout of ASDR is shown in Figure 2..

The ASDR provides the DC power, ground, test pulse and threshold voltage to the ASDB via the same 20-pair twisted-pair cable. User can set the threshold voltage on the front panel. Each input has its own potentiometer. It is possible to check the threshold value on the control points on the front panel. In our case the threshold of wires is set to -50 mV, whereas the strip threshold is set to 60mv (see Figure 2. A). The output signal is depend of the thresholds value, in the case of wrong values the signal would have noise and the offline code will not able to calculate the maps of efficiency and the timing correclty. It is illustrated in Figure 2. B.

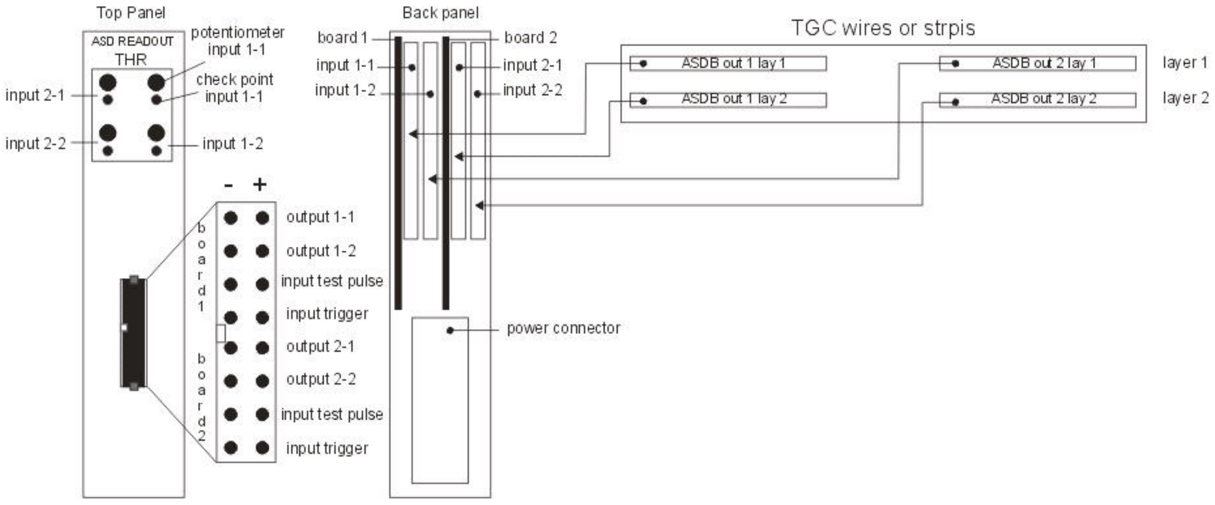
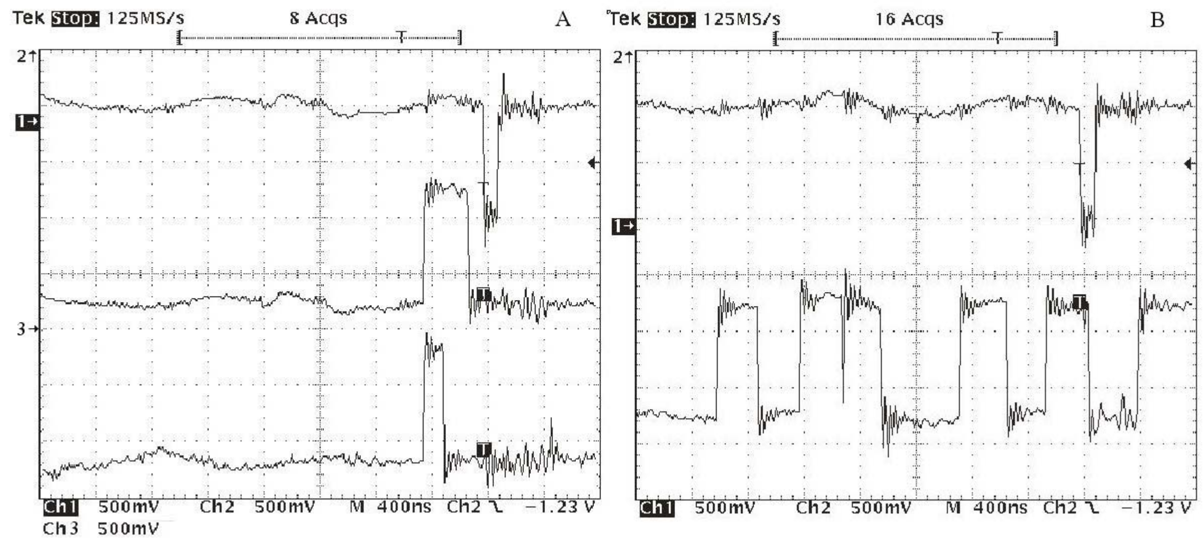


**Figure 2.13 The ASDR top and back panel and the input connector pin assignment.**



Ch 1: Trigger Ch 2: Wires signal Ch 3: Strips signal Ch 1: Trigger Ch 2: Wires signal

**Figure 2.14 The TGC output signal from the ASDR**

### 2.2.4 The ASD TO TDC.

The ASDR output are transmitted to the ASD TO TDC (ASDTDC) NIM card through an eight pair flat-pair cable. The trigger and the test pulse are provided to the ASDR via the same flat cable. The bottom eight ASDTDC inputs refer to the first output and top eight ASDTDC inputs refer to the secend output (see Figure 2.15 ).

Since the TDC input connectors are Robinson Nugent type (see Figure 2.7) thus ASDTDC changes only the type of connectors, from input connectors, Header 3M type, 8 + 8 pins, to the connector fitted the TDC. The input signals are transmitted to the output without any changes.

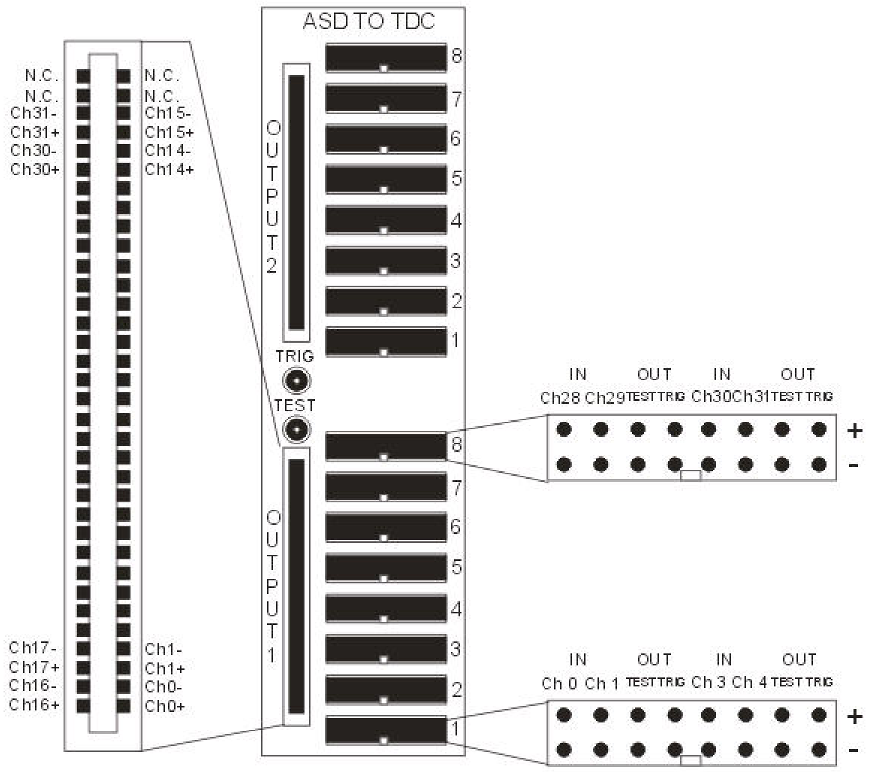


**Figure 2.15 The ASDTDC top panel and input connectors pin assignment.**

The ASDTDC card provides also the trigger to the TDC. The TDC trigger input is sent via the control bus of the TDC front panel (see Figure 2.7). That trigger has to be a standard ECL signal. The standard NIM trigger is connected to the LEMO connector on the ASDTDC front panel. The ASDTDC changes this trigger signal from NIM standard to the ECL standard. The ECL trigger is accessible from the trigger output in each input of the ASDTDC front panel. That trigger is connected to the TDC trigger input throught a special cable.
Since the TDC is working in the *Common Stop Emulation* mode, the trigger is to be provided to channel zero of the selected TDC chip. It is done by the trigger connection from the first ASDTDC input to  channel zero of the same fisrst input. The top and the bottom trigger signals are connected to the second and the third channel in the ASD TDC first input. This configuration provides the top and bottom triggers to the TDC, allowing the offline program to calculate the muon hit time in the scintillators (see Figure 2.16 ).

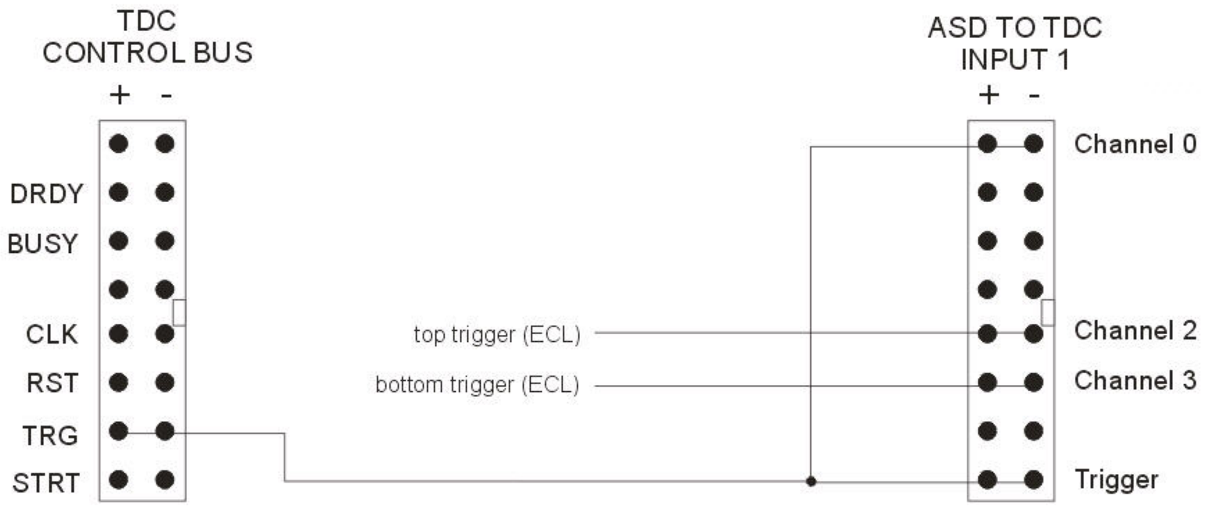


**Figure 2.16  The ASD TO TDC connections system layout.**

### 2.2.5 The NIM to ECL translator.

The top trigger and the bottom trigger are connected to a NIM to ECL card. These output signals are connected to the  second and the third channels of ASDTDC input one (see Figure 2.).

## *2.3 Trigger system.*

A charged particle (muon) which crosses the scintillator produces photon in the scintilating material. These photons are collected and amplified by Photomultiplier Tubes, which are attached at the ends of each scintillator. In this way a single analog signal is generated. All analog signals go to discriminators, where they are converted to a digital signal.

In our case eight scintillators are used, four at the bottom and four at the top of  the cosmic ray telescope (see Figure 2.) .

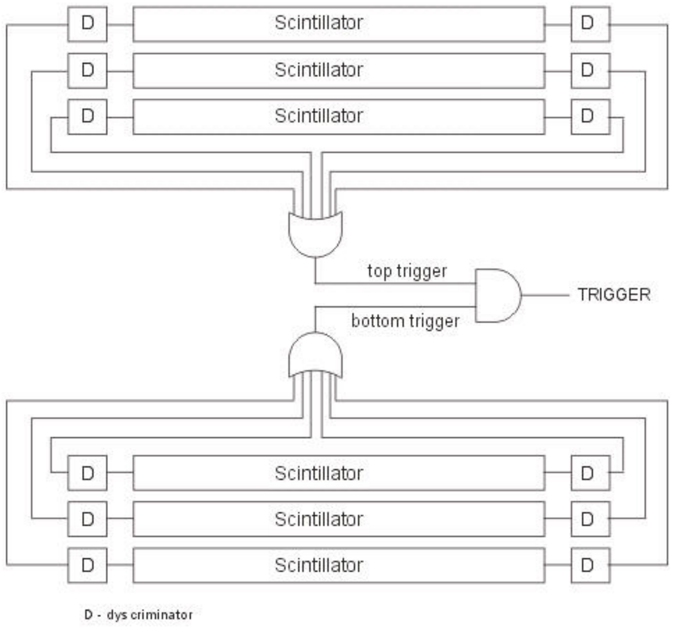


**Figure 2.17  Schematic diagram of the trigger logical connection.**

All signals from one couple of scintillators go to an OR logical gate. This signal represent the hit of that layer. The common trigger is made by an AND logical gate which takes a coincidence between the top and bottom layer signals.

To prevent a trigger signal when the system is not ready an AND coincidence between the BUSY and COMMON TRIGGER is made. It means that triggers are not allowed when the system is reading data from chambers. Since the TDC is working in a *Common Stop Emulation* mode the trigger has to be delayed. Otherwise the signals from the TGC will not be in within the proper time window. We choose a delay 100 ns. The scheme of the trigger design is presented in Figure 2..

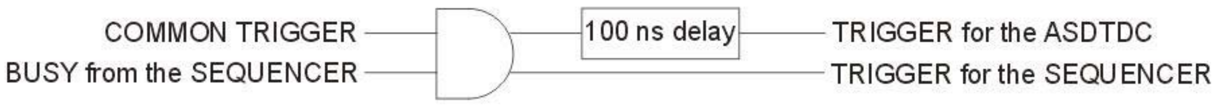


**Figure 2.18  Scheme of the trigger design.**

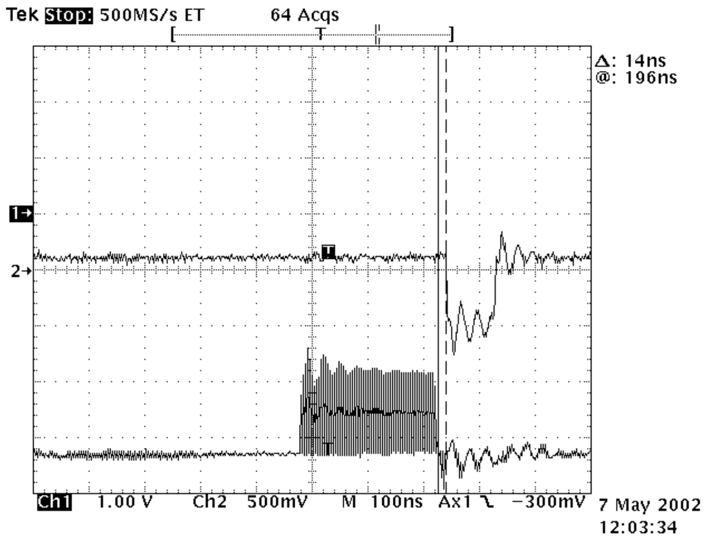


**Figure 2.19 The TDC trigger (channel 1), signal from the TGC (channel 2) oscilogram.**

The link between the TDC trigger and the signals from the TGC is shown in Figure 2.19 . Without the 100ns delay the TDC trigger will be in the middle of the TGC signal and the TDC will report a TDC Chip Error (see § 3.3.1).

## 2.4 *Electronics general scheme.*

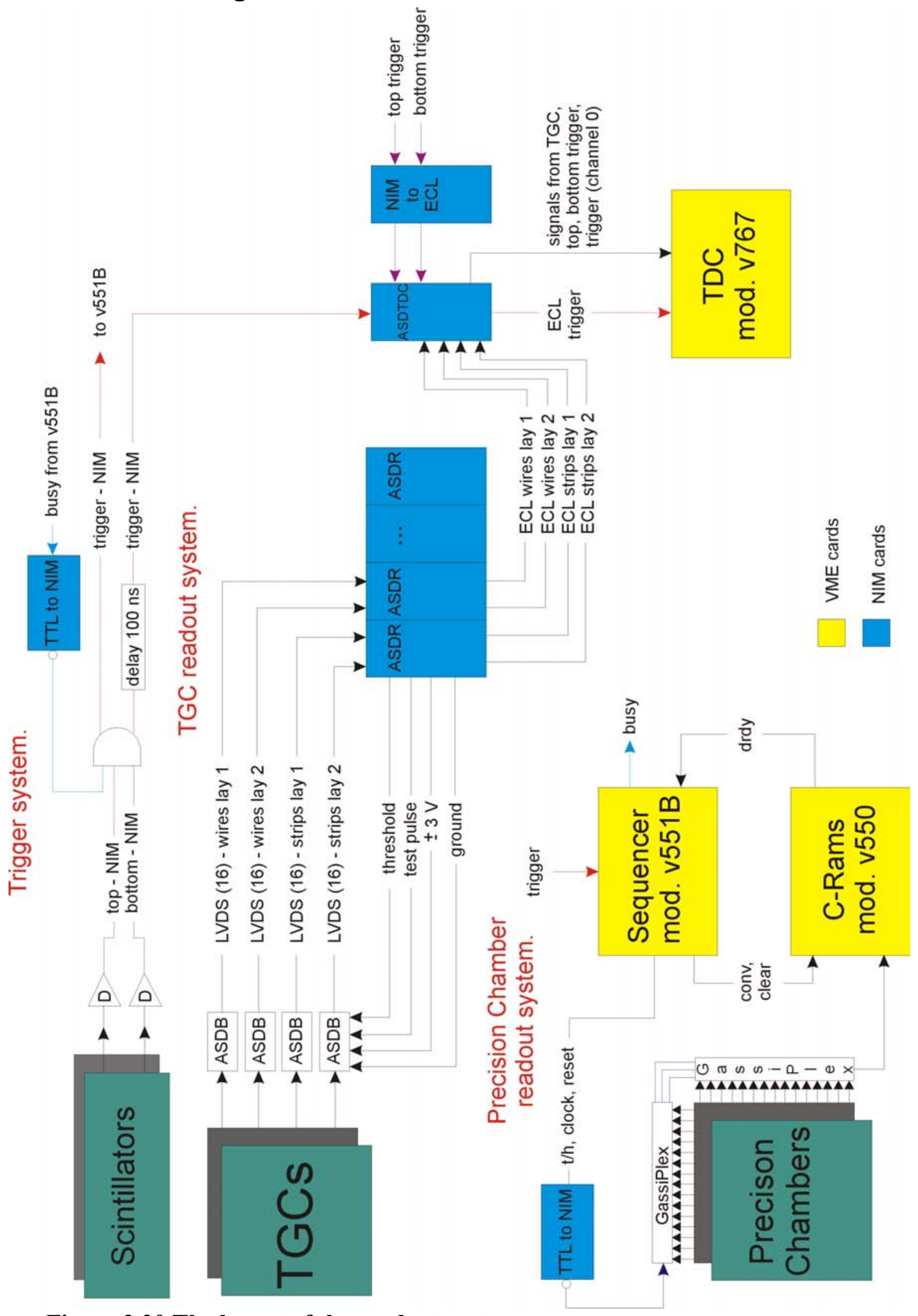


**Figure 2.20 The leyout of the readout system.**

To achieve synchronization between the C-RAMS and the SEQUENCER, the communication between these cards is done in the following way:
The TRIGGER is connected to the trigger input of SEQUENCER. With the occurrence of a TRIGGER signal the SEQUENCER starts generating the CLOCK, CONVERT, HOLD and BUSY signals.

To start taking data, a CONVERT signal is provided to the C-RAMS. The convert input of the C-RAMS is joined with the convert output of the SEQUENCER. The CLOCK and the CONVERT are synchronized at formation. After the train of CONVERT pulses the DRDY changes to high state indicating the presence of data in the output buffer. When the C-RAMS is in DRDY mode the generation of another CONVERT train pulses is prevented. To meet this condition, the DRDY is connected to the drdy input of the SEQUENCER. The BUSY of the SEQUENCER does not go low when the DRDY is high and the SEQUENCER does not accept any more triggers. The data, in the output buffer of C-RAMS, has to be read by a VME, otherwise the system will stop.

The output signals from the GassiPlexes cards are connected to the input channel of the C-RAMS. The C-RAMS has two input signals per one ADC channel, since the C-RAMS accepts a differential input signal. Since we use a positive signal, only one input channel is used. The second input has is terminated by a 50Ω resistor. The C-RAMS sequence system layout is shown in Figure 2..

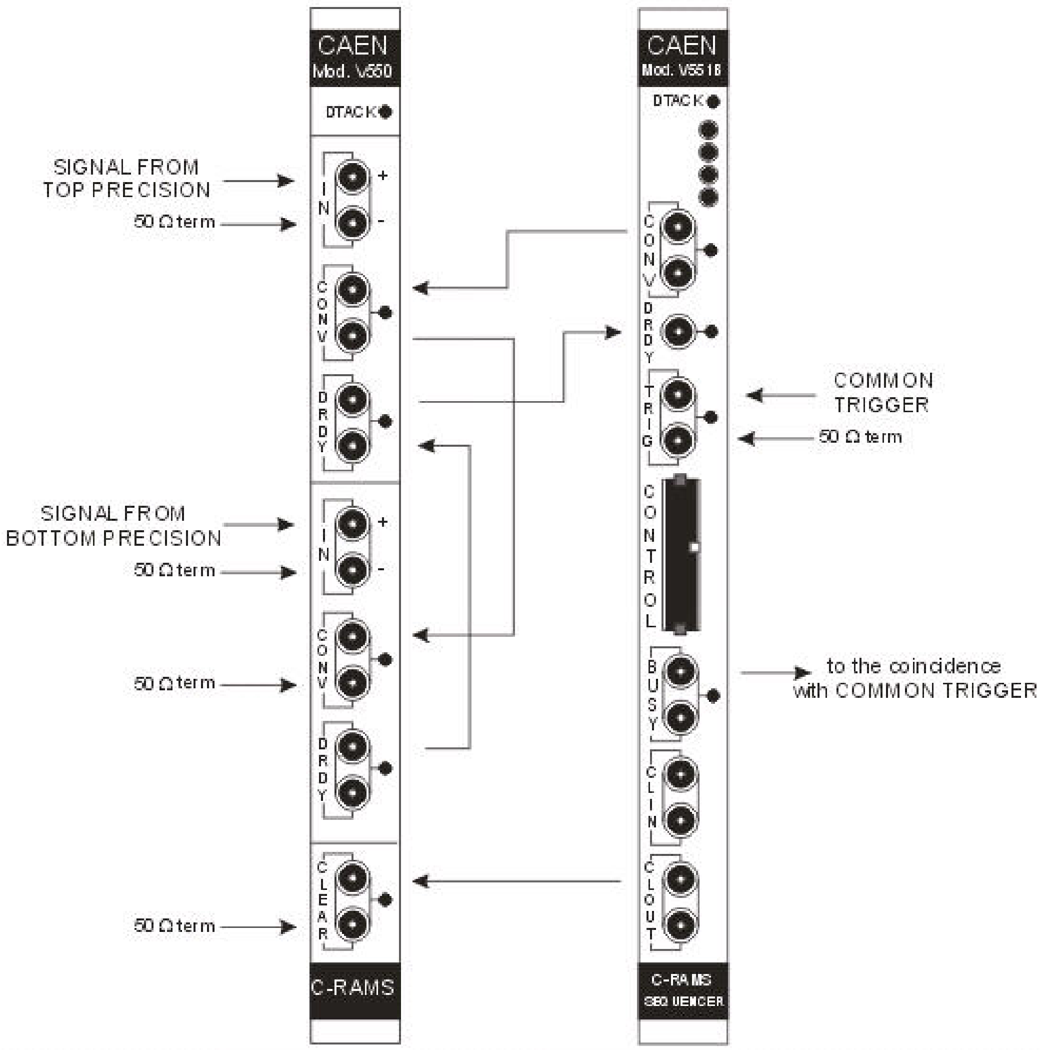


**Figure 2.21 The C-RAMS-SEQUENCER connections layout.**

To provide the signals from the TGCs chambers to the TDC the connections between the TGC, ASDR, ASTTDC and TDC is done in the following way:

The signals from the TGC are connected to the ASDR. The output signals go to the ASDTDC, the first input of ASDTDC is used to provide the triger, top and bottom trigger to the TDC. This input is connected to the TDC control bus. The trigger is connected to the LEMO input of this card as well. The output signals from ASDTDC go to the TDC. The layout of the connections is shown in Figure 2..

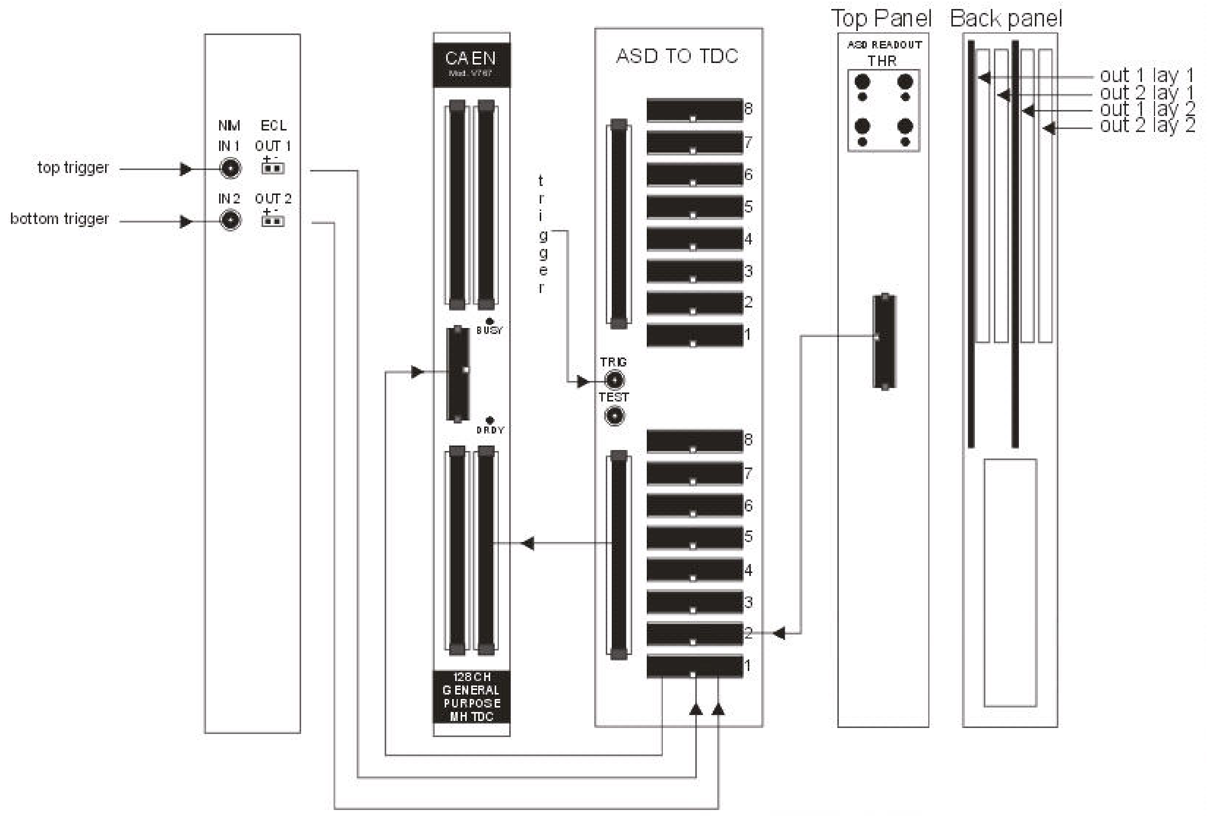


**Figure 2.22 The ASDR-ASDTDC-TDC connections layout.**

The layout of the electronics general scheme is shown in Figure 2..

# 3 VME Interface and functionality

## 3.1 SEQUENCER

The C-RAMS has CR/CRS space of 254 bytes length. This space consists of the registers. User can read and write the data from them, the map of the registers is shown in Table 3.1 [5]. The Base Address can be changed by four internal rotary switches houses on two piggyback boards plugged into the main printed circuit.

The Base Address can be selected within the range:

| | |
|---|---|
| 0x00 0000 ÷ 0xFF 0000 | A24 mode |
| 0x0000 0000 ÷ 0xFFFF 0000 | A32 mode |

| Address | Register/Content | Type |
|---|---|---|
| Base + 0xFE | Version & Series | read only |
| Base + 0xFC | Manufacturer & module type | read only |
| Base + 0xFA | Fixed code | read only |
| Base + 0x18 | Internal DAC | write only |
| Base + 0x16 | T5 Register | read/write |
| Base + 0x14 | T4 Register | read/write |
| Base + 0x12 | T3 Register | read/write |
| Base + 0x10 | T2 Register | read/write |
| Base + 0x0E | T1 Register | read/write |
| Base + 0x0C | Number of channels | read/write |
| Base + 0x0A | Test Register | read/write |
| Base + 0x08 | Status Register | read/write |
| Base + 0x06 | Software Trigger | read/write |
| Base + 0x04 | Software Clear | read/write |
| Base + 0x02 | Interrupt Level | write only |
| Base + 0x00 | Interrupt Vector | write only |

**Table 3.1 SEQUENCER Address Map**

### 3.1.1 The SEQUENCER registers

The following paragraphs are the registers description which are used by the online code.

**1. Clear Register**
(Base address + 0x04, read/write)
A VME access to this location causes the following:

1. a pulse (500 ns) is generated on the CLEAR output;
2. a pulse (500 ns) is generated on the DRESET line of the CONTROL BUS;
3. a pulse (500 ns) is generated on the ARESET line of the CONTROL BUS;
4. if the conversion sequence is in progress, it is aborted and the BUSY output becomes not active

**2. Status Register**
(Base address + 0x08, read/write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | AS | B | DY | AT | V | ID |

**Figure 3.1 SEQUENCER Status Register**
Figure 3.1 represents the bits structure of the Status register of the SEQUENCER
Where:
ID – Internal Delay
= 0 no Internal Delay;
= 1 Internal Delay selected.
V – Veto
= 0 no Veto state;
= 1 Veto state.
AT – Auto Trigger
= 0 Normal mode;

= 1 Auto Trigger mode.
DY – Data Ready (read only)
= 0 no Data Ready;
= 1 Data Ready.
B – Busy (read only)
= 0 not Busy;
= 1 Busy.
AS – Active Sequence
= 0 no Active Sequence;
= 1 Active Sequence.

### 3. Number of channels register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▨ | ▨ | ▨ | ▨ | ▨ | Number of channels | | | | | | | | | | |

**Figure 3.2 SEQUENCER Number of Channels Register**

The structure of the Number of Channels Register is shown in Figure 3.2. The number of detector channels (0 – 2047) to be read out by the C-RAMS is programmed via this register (though the V550 C-RAMS is limited to 2016 detector channels)

### 4. T1 Register

(Base address + 0x0E, read/write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | T1 | | | | | | | |

**Figure 3.3 SEQUENCER T1 Register**

The T1 register is used to set the 8 bits T1 parameters (see Figure 3.3). It sets the delay between the Leading Edge of the TRIGGER and the HOLD assertion. The actual delay t1 (in nanosecond) is calculated as follows:
$t1 = 500 + T1 * 10$ ns, where $0 \le T1 \le 255$.

### 5. T2 Register

(Base address + 0x10, read/write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | T2 | | | | | | | | |

**Figure 3.4 SEQUENCER T2 Register**

In the T2 register one sets 9 bits T2 parameters (see Figure 3.4). It provides the t2 delay between the HOLD assertion and the start of the CLOC/CONVERT sequence. The actual delay t2 (in nanosecond) is calculated as follows:
$t2 = 130 + T1 * 20$ ns, where $10 \le T1 \le 511$.

### 6. T3 Register

(Base address + 0x12, read/write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | T3 | | | | | | | |

**Figure 3.5 SEQUENCER T3 Register**

The T3 register sets the 8 bits T3 parameters it is shown in Figure 3.5. It adjust the duration t3 of the active phase of the CLOCK and the CONVERT. The actual duration t3 (in nanosecond) is calculated as follows:
$t3 = T3 * 20$ ns, where $1 \leq T3 \leq T4 \leq 255$. This constraint ($T3 \leq T4$) follows automatically from the fact that the active phase of the CLOCK and CONVERT must be less their own period.

**7. T4 Register**
(Base address + 0x14, read/write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | T4 | | | | | | | | |

**Figure 3.6 SEQUENCER T4 Register**

The T4 register allows to set the 9 bits T4 parameters (see Figure 3.6). It defines the period t4 of both the CLOCK and the CONVERT sequence.The actual duration t4 (in nanosecond) is calculated as follows:
$t4 = 20 + T4 * 20$ ns, where $1 \leq T4 \leq 511$.

**8. T5 Register**
(Base address + 0x16, read/write)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | T5 | | | | | | | | |

**Figure 3.7 SEQUENCER T5 Register**

The T5 register defines the 9 bits T5 parameters (see Figure 3.7). In Normal Mode (see Figure 3.1), it provides the delay t5 between the CLOCK and the relevant CONVERT. The actual delay t5 (in nanosecond) is calculated as follows:
$t5 = 40 + T5 * 20$ ns, where $2 \leq T5 \leq 511$.

## *3.2 C-RAMS*

The CR/SCR space of this card has 24572 bytes length. The 4 international rotary switches houses on two pigyback boards plugged into the main printed circuit board fix the module's Base Address. The module can work in the A24 and A32 mode [6].
The Base Address can be selected in the range:

| | |
|---|---|
| 0x00 0000 ÷ 0xFF 0000 | A24 mode |
| 0x0000 0000 ÷ 0xFFFF 0000 | A32 mode |

The map of the registers is shown in Table 3.2.

| **Address** | **Register/Content** | **Type** |
|---|---|---|
| Base + 0x5FFC<br>●<br>●<br>●<br>Base + 0x4000 | Pedestal/Threshold memory channel 1<br><br><br><br>Pedestal/Threshold memory channel 1 | read/write |
| Base + 0x3FFC<br>●<br>●<br>●<br>Base + 0x2000 | Pedestal/Threshold memory channel 0<br><br><br><br>Pedestal/Threshold memory channel 0 | read/write |
| Base + 0xFE | Version & Series | read only |
| Base + 0xFC | Manufactured & module type | read only |
| Base + 0xFA | Fixed code | read only |
| Base + 0x16 | Test pattern channel 1 | write only |
| Base + 0x14 | Test pattern channel 0 | write only |
| Base + 0x12 | Word counter channel 1 | read only |
| Base + 0x10 | Word counter channel 0 | read only |
| Base + 0x0C | FIFO channel 1 | read only |
| Base + 0x08 | FIFO channel 0 | read only |
| Base + 0x06 | Module clear | write only |
| Base + 0x04 | Number of channels | read/write |
| Base + 0x02 | Status Register | read/write |
| Base + 0x00 | Interrupt Register | write only |

**Table 3.2 C-Rams Address Map**

### 3.2.1 C-RAMS registers

1. **Word Counter Register**

(Base address + 0x12, read only channel 1)
(Base address + 0x10, read only channel 0)
The Word counter registers have the following structure:

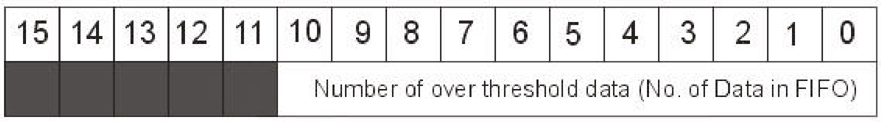


**Figure 3.8 C-RAMS Word Counter Register**

2. **FIFO Channel 0 and 1**

(Base address + 0x0C, read only channel 1)
(Base address + 0x08, read only channel 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| O | V | | | | | | | | Channel number | | | | | | | | | | | Channel data | | | | | | | | | | | |

**Figure 3.9 C-RAMS FIFO structure**

We can see the bits structure of the C-RAMS FIFO in Figure 3..
Where:
V – Validity bit :
= 0 converted value is under pedestal;
= 1 converted value is over pedestal.
O – Overange bit:
= 0 no FADC overange;
= 1 FADC overange.

3. **Clear Module**

(Base address + 0x06, write only )
A VME access to this location causes the following
1. Aborts the conversion process (if active);
2. clears the FIFOs;
3. clears the word counters.

4. **Number of channels**

(Base address + 0x04, read/write )
This register allows programming the number of detector channels to be read out in 32 steps. This number ranges from 32 (DCN=1) to 2016 (DCN=63), DCN=0 means 1 detector channel only. Its structure is shown in Figure 3.11.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▨ | ▨ | ▨ | ▨ | DCN Channel 1 | | | | | | DCN Channel 0 | | | | | |

**Figure 3.110 C-RAMS Number of channels**

5. **Status Register**

(Base address + %02, read/write )
The Status register contains information about the status and allows a few settings on the module.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | /F1 | /F0 | /H1 | /H0 | /E1 | /E0 | /D1 | /D0 | MO | T |

**Figure 3.131 C-RAMS Status Register**

Figure 3.13 represents the bits structure, where:

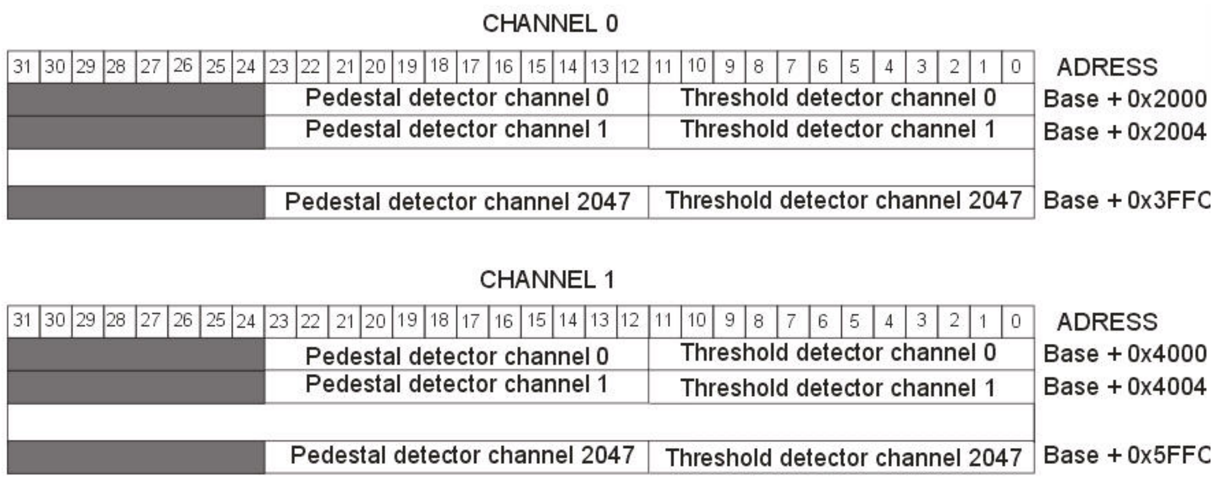
T - Test mode.
= 0 no Test mode;
= 1 Test mode;

MO - Pedestal Threshold memory owner.
= 0 memory owned by VME;
= 1 memory owned by the Conversion logic;
/D0 - Channel 0 Data Ready bit, read only.
= 0 data ready;
/D1 - Channel 1 Data Ready bit, read only.
= 0 data ready;
/E0 - FIFO channel 0 empty bit, read only.
= 0 empty;
/E1 - FIFO channel 1 empty bit, read only.
= 0 empty;
/H0 - FIFO CHANNEL 0 half full bit, read only.
= 0 half full;
/H1 - FIFO CHANNEL 1 half full bit, read only.
= 0 half full;
/F0 - FIFO CHANNEL 0 full bit, read only.
= 0 full;
/F1 - FIFO CHANNEL 1 full bit, read only.
= 0 full;

## 6. Pedestal and Threshold memories

(Base address + 0x2000…5FFE, read/write )

The pedestal and threshold memories are accessible via VME only if Memory owner bit is set to 0 (default mode).
They have the following structure which is shown in Figure 3.15:



CHANNEL 0

| 31–24 | 23–12 | 11–0 | ADRESS |
|---|---|---|---|
| ▨ | Pedestal detector channel 0 | Threshold detector channel 0 | Base + 0x2000 |
| ▨ | Pedestal detector channel 1 | Threshold detector channel 1 | Base + 0x2004 |
| | | | |
| ▨ | Pedestal detector channel 2047 | Threshold detector channel 2047 | Base + 0x3FFC |

CHANNEL 1

| 31–24 | 23–12 | 11–0 | ADRESS |
|---|---|---|---|
| ▨ | Pedestal detector channel 0 | Threshold detector channel 0 | Base + 0x4000 |
| ▨ | Pedestal detector channel 1 | Threshold detector channel 1 | Base + 0x4004 |
| | | | |
| ▨ | Pedestal detector channel 2047 | Threshold detector channel 2047 | Base + 0x5FFC |

**Figure 3.152 C-RAMS Pedestal/Threshold memories**

The memories size is 2048 words but only 2016 are actually used by the conversion logic (2016 is the maximum number of channels).

## 3.3 TDC

The TDC has CR/SCR space of 93 bytes length. Its base address can be changed by four internal rotary switches housed on two piggyback boards plugged into the main printed circuit board. It is also possible to reassign the Base Address via the VME. The new address is lost at Power-Off and the rotary switch setting will be restored at Power-On or after a Reset [7].

The Base Address can be selected in the range:

|  |  |
|---|---|
| 0x00 0000 ÷ 0xFF 0000 | A24 mode |
| 0x0000 0000 ÷ 0xFFFF 0000 | A32 mode |

The address of each register is the sum between the base address and the register offset . The map of the registers is shown in Table 3.3.

| Address | Register/Content | Type |
|---|---|---|
| Base + 0x00 | Output Buffer | read only |
| Base + 0x04 | Geographical Register | read only |
| Base + 0x06 | Bit Set | read/write |
| Base + 0x08 | Bit Clear | read/write |
| Base + 0x0A | Interrupt Level | read/write |
| Base + 0x0C | Interrupt Vector | read/write |
| Base + 0x0E | Status Register 1 | read only |
| Base + 0x10 | Control Register 1 | read/write |
| Base + 0x12 | ADER 32 | read/write |
| Base + 0x14 | ADER 24 | read/write |
| Base + 0x16 | MCST Address | read/write |
| Base + 0x18 | Single Shot Reset | read/write |
| Base + 0x20 | MCST Control | read/write |
| Base + 0x48 | Status Register 2 | read only |
| Base + 0x4A | Control Register 2 | read/write |
| Base + 0x4C | Event Counter | read only |
| Base + 0x4E | Clear Event Counter | write only |
| Base + 0x50 | Opcode Handshake | read only |
| Base + 0x52 | Opcode Register | read/write |
| Base + 0x54 | Clear | write only |
| Base + 0x56 | Testword H | write only |
| Base + 0x58 | Testword L | write only |
| Base + 0x5A | Software Trigger | write only |

**Table 3.3 Address Map for the TDC**

### 3.3.1 The TDC registers.

1. **Output Buffer**

(Base address + 0x0000 read only)

The output buffer contains the output data organized in 32-bit words. The data in the buffer is organized as follows:

*A*) In the case of *Start Trigger Matching*, *Stop Trigger Matching* (*Common Stop Emulation* included) and *Start Gating* mode the data in the buffer is organized in events.

Each event consists of (see Figure 3.173):

1. A **header** that contains the event number value;
2. The **data words** containing the 20 bit converted time values, the channel number and an edge bit;

3. An **End Of Block** (EOB) word containing the number of data words (i.e. the HEADER and EOB are not included) and a Status that is one if the TDC Chips have had an error, zero otherwise.

*B*) In the case of *Continuous Storage* mode there are neither the header nor the EOB word, and only the data words are written.
In both cases, if a read access to the buffer is performed when it is empty, the readout will provide a Not Valid Datum arranged as shown in Figure 3.173.

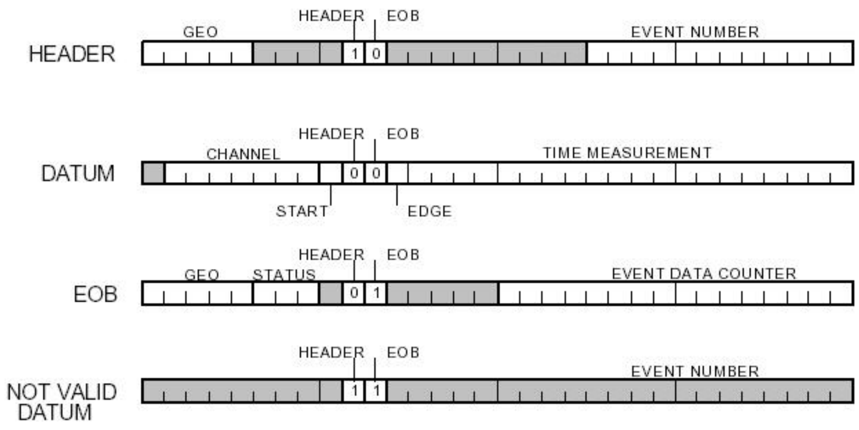


**Figure 3.173 TDC Output Buffer Data Structure**

4. **Status Register 1**

(Base address + 0x000E, read only)

This register contains information on the status of the module. Its bits structure is shown in Figure 3..

NO TERM and TERM OK refer to the termination of the lines in the front panel CONTROL bus: the last module is chain controlled via the front panel CONTROL connector and must have its termination ON, while all the others must have it OFF. The insertion or removal of the termination is performed via internal DIP switches.

The BUSY and DATA READY signals are available both for readout of a signals module as well as for readout of multi units system connected via the CONTROL BUS.

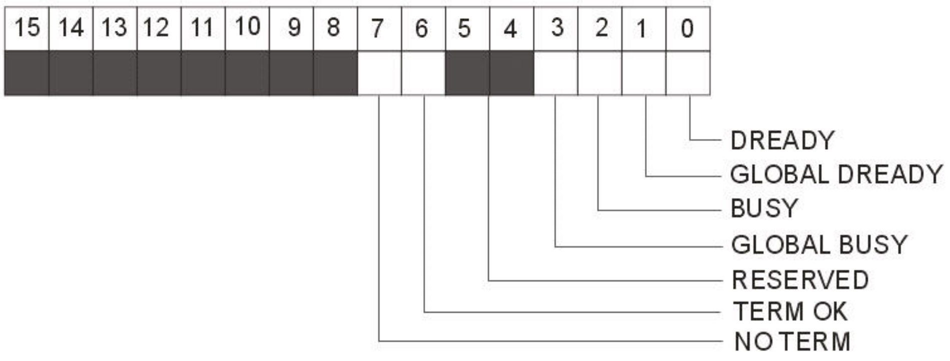


**Figure 3.14 TDC Status Register 1**

DREADY: Indicates that the Output Buffer has data.

= 0 No Data Ready.

= 1 Data Ready.

GLOBAL DREADY Indicates that at least a module in a chain has data.

= 0 No Module has Data Ready.
= 1 At least a Module has Data Ready.

BUSY Indicates either that a conversion is in progress, or the Output Buffer is full, or that the board is in TEST mode; goes low when the Output Buffer is not full and the module is in Normal Mode.
= 0 Module not Busy.
= 1 Module Busy.

GLOBAL BUSY Indicates that at least a module in a chain is BUSY.
= 0 No Module is Busy.
= 1 At least a Module is Busy.

TERM OK Termination ON bit.
= 0 Control Bus Termination is ON.

NO TERM Termination OFF bit.
= 0 Control Bus Termination is OFF.

### 5. Single Shot Reset Register

(Base address + 0x0018, read/write)

A dummy access to this register allows to generate a shot RESET single of the module. Once issued, the module Front End is reset.

### 6. Status Register 2

(Base address + 0x0048, read only)

This register contains further information about the status of the module buffer and TDC errors.

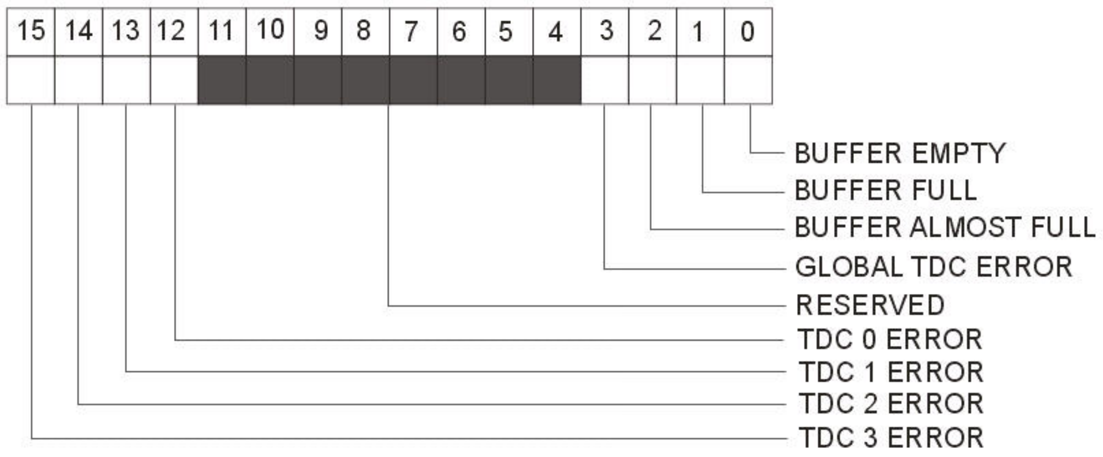


**Figure 3.15 TDC Status Register 2**

The bits structure of the Status Register 2 is shown in Figure 3., where:

BUFFER EMPTY: Indicates if the Output Buffer is empty.
= 0 Buffer Not Empty.
= 1 Buffer Empty.

BUFFER FULL: Indicates if the Output Buffer is full.
= 0 Buffer Not Full.
= 1 Buffer Full.

BUFF. ALMOST FULL: Indicates if the Output Buffer is almost full. The selection of the Almost Full status is programmable via OPCODE
= 0 Buffer Not Almost Full.
= 1 Buffer Almost Full.

GLOBAL TDC ERROR: Indicates that at least a TDC chip has an error.
= 0 No TDC chip has an error.
= 1 At least a TDC chip has an error.
TDC 0..3 ERROR: Indicates that a selected (0..3) TDC chip has an error.
= 0 The selected TDC chip has no errors.
= 1 The selected TDC chip has an error.

### 7. Control Register 2

(Base address + 0x004A, read only, bit 4 read/write)

This register allows performing a reading of some settings of the module. We can see its structure in Figure 3..

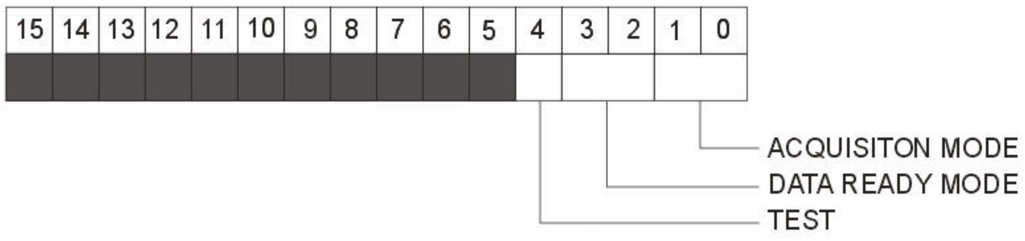


**Figure 3.16 TDC Control Register 2**

ACQUISITION MODE: Reads the status of the acquisition mode set via OPCODE without using the OPCODE reading. The two ACQ_MODE bits of this register are related to the Acquisition Mode according to the following:
ACQ_MODE = (0, 0) = Stop Trigger Matching Mode;
ACQ_MODE = (0, 1) = Start Trigger Matching Mode;
ACQ_MODE = (1, 0) = Start Gating Mode;
ACQ_MODE = (1, 1) = Continuous Storage Mode.
DATA READY MODE: Reads the status of the data in the buffer. The two DR_MODE bits of this register are related to the Data Ready Mode according to the following:
DR_MODE = (0, 0) = Buffer contains a complete event;
DR_MODE = (0, 1) = Buffer almost full;
DR_MODE = (1, 0) = Buffer not empty;
DR_MODE = (1, 1) = Meaningless.
TEST: Test bit.
= 0 Normal mode;
= 1 Test mode.

### 8. Opcode Handshake Register

(Base address + 0x0050, read only)

The Opcode Handshake Register is used for the Handshake Protocol between the VME and the micro-controller. It uses only 2 bits: READ OK and WRITE OK (see Figure 3.). All read and (write) operations of the Opcode Register can be performed, when the bit RO (WO) is set.

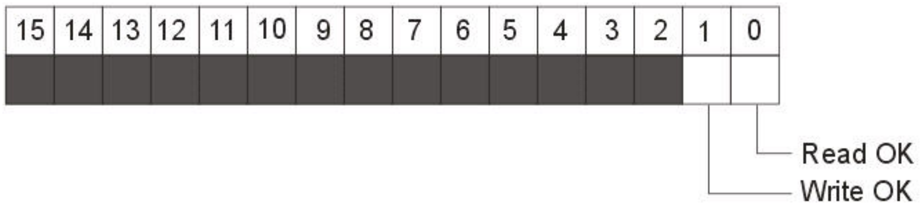


**Figure 3.17 TDC Opcode Handshake Register**

**9. Opcode Register**

(Base address + 0x0052, read/write)

The Opcode Register is used to send instructions to the microcontroller via 16-bit OPCODE setup words. The Figure 3.23 represents the structure of the Opcode Register.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMMAND | | | | | | | | OBJECT | | | | | | | |

**Figure 3.23 TDC Opcode Register**

**10. Clear Register**

(Base address + 0x0054 write only)

A VME access (read or write) to this location causes the following:

1. The TDCs are cleared;
2. The output buffer is cleared;
3. The readout controller is reset;
4. The Event counter is set to 0.

The same actions are performed at Power-ON and if the VME signal SYSRES is active.

## 3.3.2 TDC programming procedure

The TDC module programming is performed via the on-board micro-controller. The user sends and receives instructions and data to/from the microcontroller via 16-bit OPCODE setup words. The handshake is the following:

Write Operation:

- The VME (master) tests the WRITE_OK bit in the Opcode Handshake Register (see § 8); if the WO bit is set to one, the VME can write a datum;
- The WO bit is automatically reset after the datum is written and is set back to one when the datum has been acquired by the server;
- When the WO bit is set back to one, the VME can write another datum.

Read Operation;

- A valid datum can be read via VME only if the READ_OK (RO) bit in the Opcode Handshake Register (see § 8) is set to one;
- The RO bit is automatically reset after the datum is read out and is set back to one when a new datum is ready to be read out;
- When the RO bit is set back to one, the VME can read another datum.

Since the TDC module's internal delays it is necessary to insert a 10-ms delay in the software after the check of the RO/WO bit, i.e. before performing the next R/W operation on the Opcode Register.

In general, three different types of operation can be performed:

1. Write an opcode,
2. Write an operand,
3. Read an operand.

By default, at power-on or after a VME reset, the WO bit of the Handshake Register is set to one and the RO to zero: actually, the first operation to be performed is always the write operation of an opcode (operation of type one).

Conversely, the operation of type two must be always preceded by the write operation of an opcode which requires to write an operand. The operation of type three must be always preceded by a write operation of an opcode which issues a read access: actually, this makes the RO bit of the Handshake Register to be set to one in order to allow the following read operation. For example, the write operation of the opcode ‘READ_WIN_WIDTH’ makes the RO bit of the Handshake Register be set to one and in this way allows the following read operation. Before any read/write operation, it is necessary to run a while loop in order to wait until the module is ready. As a consequence, any R/W operation is in fact constituted by the following blocks (see Figure 3.25):

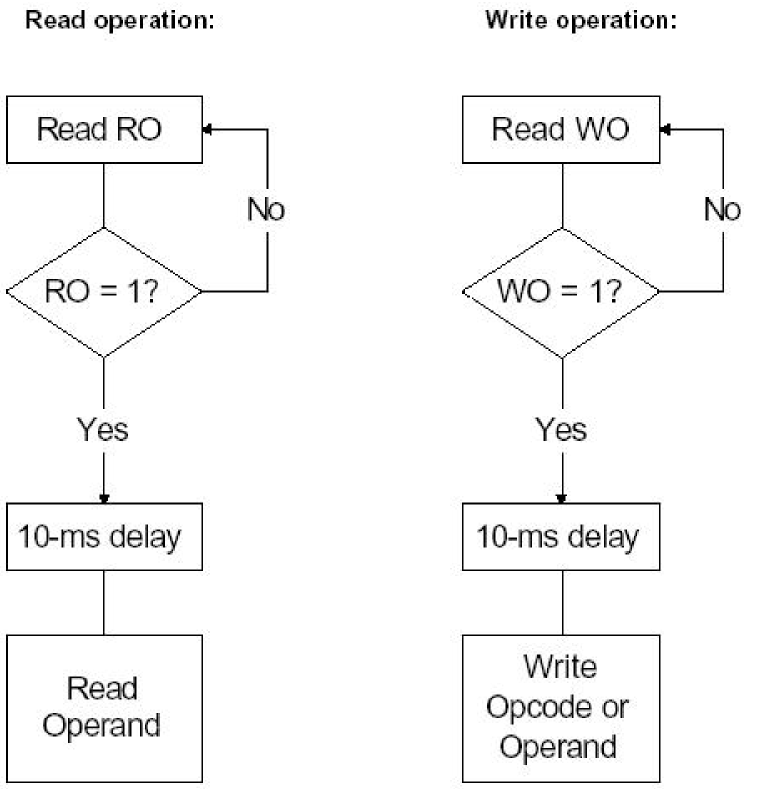


**Figure 3.259 TDC R/W operations: software logic blocks.**

The OPCODE setup words have the following format:

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMMAND | | | | | | | | OBJECT | | | | | | | |

**Figure 3.20 TDC OPCODE Word**

The COMMAND field specifies the operation to perform, while the OBJECT field (when required) specifies the object on which the operation must be performed on (e.g. the channel number). If the object refers to the channel number (OBJ = nn in Table 3.4), it can vary from 0 to 7F; if it refers to the TDC number (OBJ = c in Table 3.4), it can vary from 0 to 3.
When the operation does not foresee an object, the OBJECT field is meaningless.
The communication with the microcontroller begins always by sending an OPCODE; if no operands are foreseen (nR = 0 and nW = 0) the cycle ends, otherwise the microcontroller remains in a wait status until the user hasn't read or written all the foreseen operands.
The following Table 3.4 contains, for each OPCODE, the symbolic name, the performed operation, the number of written and read operands and the number of significant bits. The table contains only Opcodes that have been used by us in the online code.

| COM | OBJ | CODE | SYMBOLIC | OPERATION | nW | nR | nbit |
|---|---|---|---|---|---|---|---|
| **TEST** | | | | | | | |
| 01 | ------ | 01xx | EN_MEM_TEST | Enable memory test mode | --- | --- | --- |
| 02 | ------ | 02xx | DIS_MEM_TEST | Disable memory test mode | --- | --- | --- |
| 03 | ------ | 03xx | READ_MEM_TEST | Read memory test (on/off) | --- | 1 | 1 |
| ACQUISITION MODE | | | | | | | |
| 10 | ------ | 10xx | STOP_MATCH | Set stop trigger matching | --- | --- | --- |
| 11 | ------ | 11xx | START_MATCH | Set start trigger matching | --- | --- | --- |
| 12 | ------ | 12xx | START_GATE | Set start gating | --- | --- | --- |
| 13 | ------ | 13xx | CONT_STO | Set continuous storage | --- | --- | --- |
| 14 | ------ | 14xx | READ_ACQ_MOD | Read acquisition mode | --- | 1 | 2 |
| 15 | ------ | 15xx | LOAD_DEF_CONF | Load default configuration | --- | --- | --- |
| 16 | ------ | 16xx | SAVE_CONFIG | Save User configuration | --- | --- | --- |
| 17 | ------ | 17xx | LOAD_CONFIG | Load User configuration | --- | --- | --- |
| 18 | ------ | 18xx | EN_AUTO_LOAD | Enable auto load | --- | --- | --- |
| 19 | ------ | 19xx | DIS_AUTO_LOAD | Disable auto load | --- | --- | --- |
| 1A | ------ | 1Axx | READ_AUTO_LOAD | Read auto load | --- | 1 | 1 |
| **CHANNEL ENABLE** | | | | | | | |
| 20 | nn | 20nn | EN_CHANNEL | Enable channel nn | --- | --- | --- |
| 21 | nn | 21nn | DIS_CHANNEL | Disable channel nn | --- | --- | --- |
| 22 | nn | 22nn | READ_STAT_CH | Read status channel nn | --- | 1 | 1 |
| 23 | ------ | 23xx | EN_ALL_CH | Enable all channels | --- | --- | --- |
| 24 | ------ | 24xx | DIS_ALL_CH | Disable all channels | --- | --- | --- |
| 25 | ------ | 25xx | WRITE_EN_PATTERN | Write enable pattern for channels | 8 | --- | 16 |
| 26 | ------ | 26xx | READ_EN_PATTERN | Read enable pattern for channels | --- | 8 | 16 |
| **TRIGGER** | | | | | | | |
| 30 | ------ | 30xx | SET_WIN_WIDTH | Set window width | 1 | --- | 16 |
| 31 | ------ | 31xx | READ_WIN_WIDTH | Read window width | --- | 1 | 16 |
| 32 | ------ | 32xx | SET_WIN_OFFS | Set window offset | 1 | --- | 16 |
| 33 | ------ | 33xx | READ_WIN_OFFS | Read window offset | --- | 1 | 16 |
| 34 | ------ | 34xx | SET_TRG_LAT | Set trigger latency | 1 | --- | 16 |
| 35 | ------ | 35xx | READ_TRG_LAT | Rear trigger latency | --- | 1 | 16 |
| 36 | ------ | 36xx | EN_SUB_TRG | Enable subtraction of trigger time | --- | --- | --- |
| 37 | ------ | 37xx | DIS_SUB_TRG | Disable subtraction of trigger time | --- | --- | --- |
| 38 | ------ | 38xx | EN_OVL_TRG | Enable overlapping triggers | --- | --- | --- |
| 39 | ------ | 39xx | DIS_OVL_TRG | Disable overlapping triggers | --- | --- | --- |
| 3A | ------ | 3Axx | READ_TRG_CONF | Read trigger configuration | --- | 1 | 2 |
| **EDGE DETECTION** | | | | | | | |
| 60 | ------ | 60xx | RISE_ALL | Rising edge only on all channels | --- | --- | --- |
| 61 | ------ | 61xx | FALL_ALL | Falling edge only on all channels | --- | --- | --- |
| 62 | ------ | 62xx | ODDR_EVENF | Rising edge on odd ch. falling edge on even | --- | --- | --- |
| 63 | ------ | 63xx | ODDF_EVENR | Falling edge on odd ch. rising edge on even | --- | --- | --- |
| 64 | ------ | 64xx | RISE_START | Rising edge only start | --- | --- | --- |
| 65 | ------ | 65xx | FALL_START | Falling edge only start | --- | --- | --- |
| 66 | ------ | 66xx | BOTH_ALL | Both edges on all channels | --- | --- | --- |
| 67 | ------ | 67xx | READ_DETECTION | | --- | 3 | 2 |

| DATA READY | | | | | | | |
|---|---|---|---|---|---|---|---|
| 70 | ------ | 70xx | DR_EV_READY | Set D.R. = event ready | --- | --- | --- |
| 71 | ------ | 71xx | DR_ALMOST_FULL | Set D.R. = buffer almost full | --- | --- | --- |
| 72 | ------ | 72xx | DR_NOT_EMPTY | Set D.R. = buffer not empty | --- | --- | --- |
| 73 | ------ | 73xx | READ_DR_MODE | Read data ready mode | --- | 1 | 2 |
| 74 | ------ | 74xx | SET_ALM_FULL | Set almost full level | 1 | --- | 15 |
| 75 | ------ | 75xx | READ_ALM_FULL | Read almost full level | --- | 1 | 15 |

**Table 3.4 Mod. V767 OPCODE Words**

# 4 VME Software

The following section describes the way we used to access information in the VME cards. It details the Drivers used as well as describing the structure of the program which useses those drivers to access the electronics cards.

## 4.1 Drivers

### 4.1.1 The memory mapping

The VME module have Configuration ROM (CR) and Control and Status Register (CSR) memory. The CR area is “read only” and consists of information such as the module type. The CRS area is “read/write” and consist of the registers of that VME card.

The CR and the CRS space are part of the computer physical memory. The Base Address of the CR and CRS space can be changed by the switches which are on the each VME board.

The address of the individual register of the CR/CRS space is created by adding the offset address to the Base Address of CR/CRS space.

In a virtual memory system (VM) system, the program code deals with virtual address (logical address). While running, the virtual address is translated by the Memory Management Unit (MMU) to obtain a physical address that is used to access physical memory.
Each process typically has its own separate virtual address space with its own mappings and protections. It is illustrated in Figure 4.1.

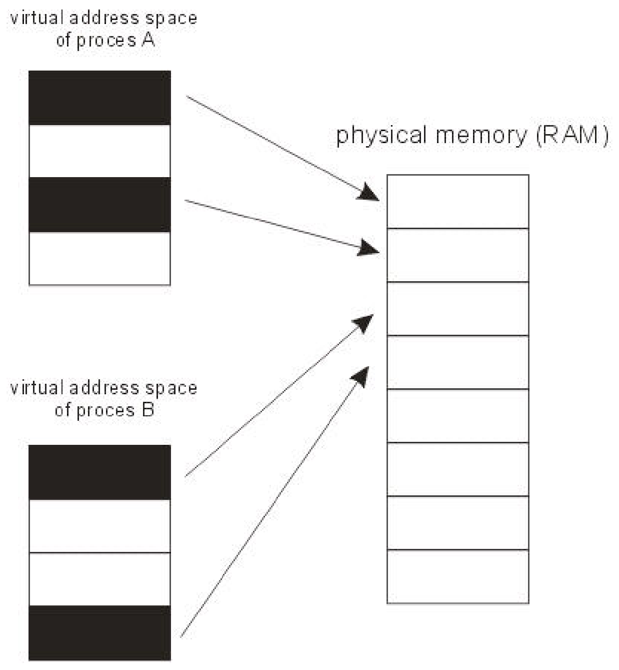

virtual address space of proces A
physical memory (RAM)
virtual address space of proces B

**Figure 4.1 Example of the relationship between the virtual address spaces of two processes and physical memory**

To get access to the memory of the VME cards, the process has to map that space of address to its own virtual memory. Since the virtual memory system deals with the memory blocks called page, the size of memory to be mapped has to be equal to the page size times the number of page blockes in use. This is shown in Figure 4.2.

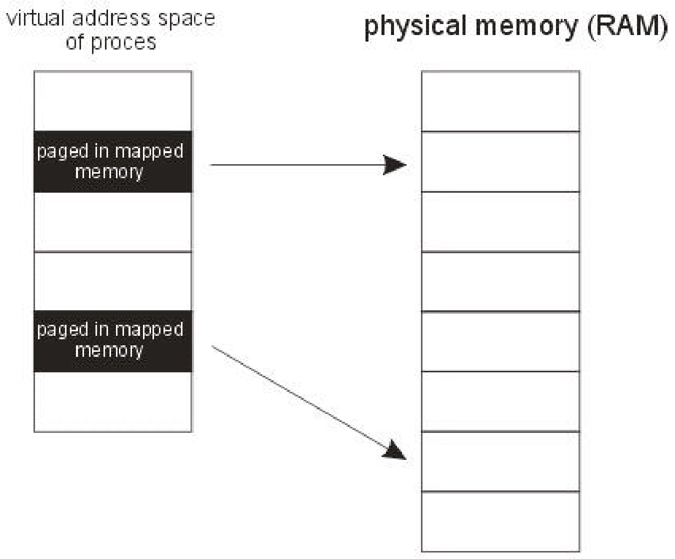


**Figure 4.2 Mapping of the Virtual Memory**

The Linux operating system running on our VME CPU has two functions to map and clear the memory mmap() and munmap() respectively. The device file "/dev/mem" is used to get access to the physical memory and kernel memory by these functions.

In our drivers a few functions are used to initialize, open and map these are: vme_init, VME_Open, VME_MstMap and a few functions are used to end, unmapping and close memory: end_vme, VME_MstUnmap and VME_Close.
The vme_init function consist of VME_Open and VME_MstMap funcitons. The VME_Open function opens the "/dev/mem" device and the VME_MstMap mapping of the VME cards memory according to the parameters such as Base Addres and size of memory which are given by the user.

The vme_end function consist of VME_Close and VME_MstUnmap functions. The VME_Close function closes the "/dev/mem" device and the VME_MstUnmap, clears the memory which was mapped by the VME_MstMap routine.
The size of memory is the CR/CRS space depends on the VME card.

### 4.1.2 The VMEbus error detections

Different set of drivers are the VMEbus error detection functions. As part of the general error reporting and error handling system in DAQ it is required to have software facility which allows to identify the source of errors at the level of the I/O module (IOM).

An IOM library function returns an error code as the function value [9]. The error code has a value of zero if the function call was successful, otherwise a non-zero value is returned.

An error defined within a package has two components: a package identifier (PID) and an error number (ERRNO) which is local to the package. The use of a package identifier allows defining package errors which are unique across libraries:
*package error == package identifier + error number*
In case of no error, the package error is equal to zero, which implies that the package identifier component of the package error is not defined, i.e. if there was no error, you do not need to know the package identifier.

The IOM error package has been designed such that all information needed by the error system is set up at run-time. This implies that each package has to link with the IOM error library and includes the global IOM error include file, but there are no links to other IOM libraries. The main features are as follows:

- The package identifiers and names are defined in a global include file (iom_error.h) within the error package. This file is included by all the library include files which allows those to define unique error codes based on the package IDs and the error numbers defined locally within the library.
- The package error text strings are defined within each package and can be retrieved via an "error get" function. Each package has to provide such a function which obtains the package error strings from the local include file. The address of this function is transmitted to the IOM error library (iom_error_init) when the package is opened. This function is called internally only.
- The IOM "error get" function in the error library analyses, the major and minor package IDs and error numbers encoded in the error code returned from a library function. The corresponding package name strings are directly accessible in the global error file (iom_error.h) while the package error text strings are retrieved via calls to the appropriate package "get error" functions.

## 4.2 Online program structure

The main purpose of the online program is to read the data from the VME cards. To fulfil that task the online program has to synchronize the operations between the following VME cards:

- SEQUENCER;
- C-Rams
- TDC

The offline program calculates the position of hits and the time response of the TGC chambers uses the online binary output file.
The online program produces during the test an output binary file and three ASCII files. These files serve as an input to a slow control code which monitors the behaviour of the system during the data acquisition process.
The online program is built with four main blocks:

- Initialize the memory of the VME cards;
- Setting the VME cards;
- Setting the pedestals and thresholds;
- Data acquisition.

For the memory initialization the online code uses the drivers (see § 4.1). After that, the VME memory is mapped to the virtual memory of the process. It means that the online program is able to read, write the data from the VME cards, to the ouput files.

The VME cards are set and initialized by the online code according to the requirement of the system.

### 4.2.1 The SEQUENCER setting

Several parameters of the SEQUENCER are set:

- Status Register;
- Test Register;
- Number of Channels Register;
- T1, T2, T3, T4, T5 Registers

The zero value is written to the Status register. Since this module should not have an internal delay, no veto state while the common trigger is used to prevent the auto trigger mode. At the end of a sequence the auto trigger mode calls the next sequence immediately. In that case the system will not wait for the common trigger signal and that mode is rejected.

Since the module works in the normal mode a zero value is written to Test register.

The number of channels has to be the same as the number of channels in the C-RAMS. In our case 18 GassiPlex are used, which results in 864 channels and this value is written to the Number of Channels register.

The value of t1 should be around the 700 ns. However a value of 500 ns was set where the cables, discriminators, logical coincidence contribute additional 200 ns between the real trigger and the trigger signal which reaches the input of the SEQUENCER. Therefore the online code is writing 500 ns value to the T1 register.

The Gassiplex produces oscillations on its output after the HOLD pulse. Therefore a delay of at least 2000 ns between the HOLD and the first CLOCK pulse is required and this value is written to the T2 register.

The minimum value of the t3 time is 150ns. The GassiChip can not detect the CLOCK pulse if it's width is less than 150ns. Therefore T3 register is set to 400 ns.

The CLOCK period should be long enough that the delay of the GassiChip chain (1728 ns) will be inside one CLOCK period. With the occurrence of the CONVERT pulse the C-RAMS samples the input signal, that moment the input signal should be stable. To met these conditions a 4000 ns CLOCK period was chosen and T4 register is filling this value. Since the 864 channels per one Precision Chamber are read, the single readout sequence last 3.5 ms (see Figure 4.). If this time is shorter then 3.5 ms it means that the CLOCK is not passing throught all GassiChips (see Figure 4.5).

The minimum value of the t5 is 1728 nano seconds. Since each GassiChip channel makes a two ns delay of the CLOCK pulses. The last CLOCK pulse is delayed by 1728 nano seconds. Thus the t4 and the t5 time periods have to be bigger then 1728 nano seconds. Otherwise the CONVERT pulse will come before the right CLOCK pulse (the CLOCK and the CONVERT should be coupled, that couple can not be disconneted) see Figure 4.6. A value of 3360 ns for t5 was choosen and this value is the entry of the T5 register.
The time configuration is shown in Figure 4.3.

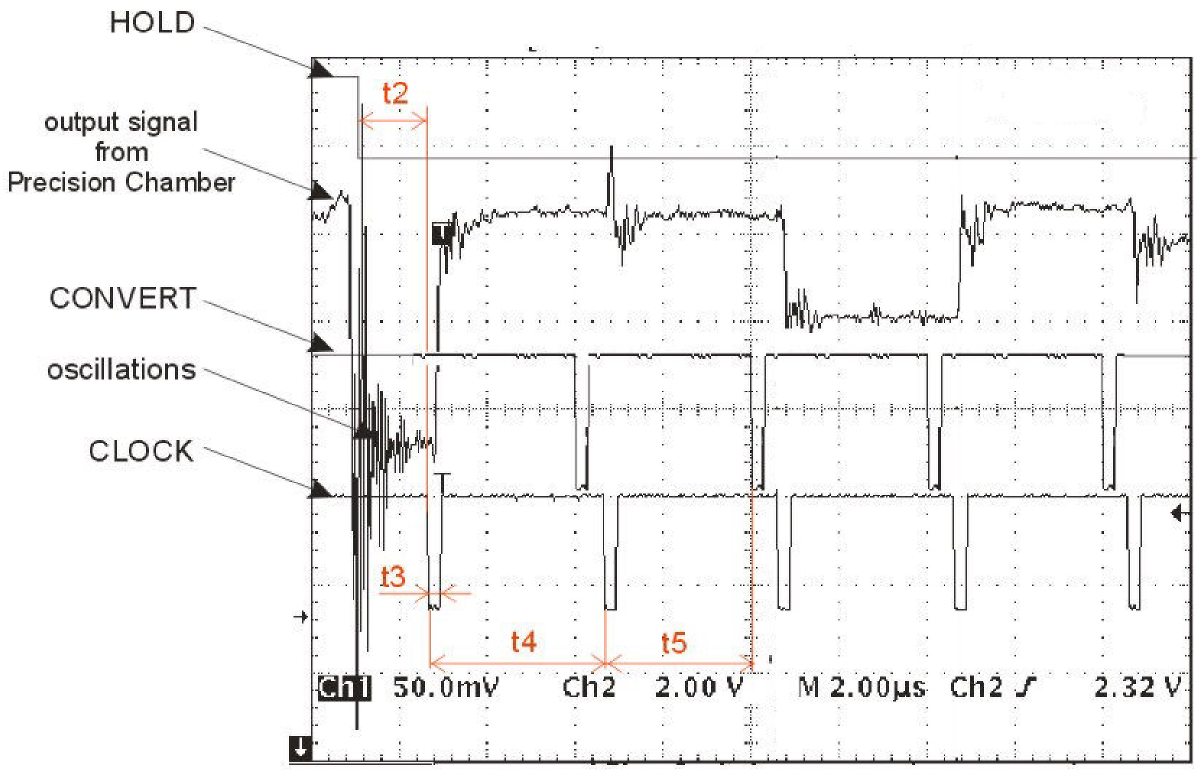
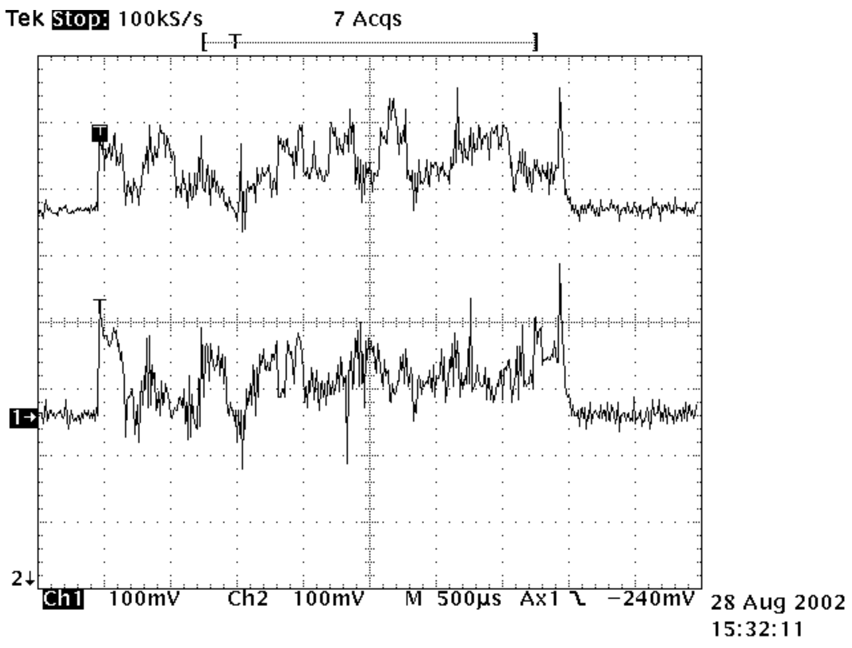


**Figure 4.3 The SEQUENCER timing oscilogram**



Ch 1: The signal from the top Precision Chamber
Ch 2: The signal from the bottom Precision Chamber

**Figure 4.4 The oscilogram of the signals from the Precision Chambers.**

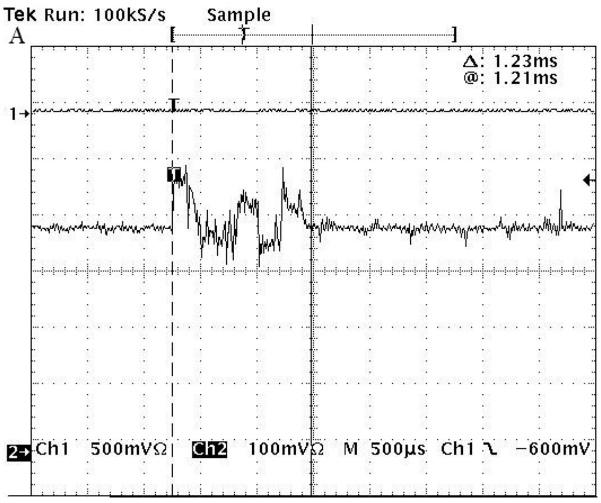


**Figure 4.5 The signal from the PC when the CLOCK pulse is not passing all GassiChips.**

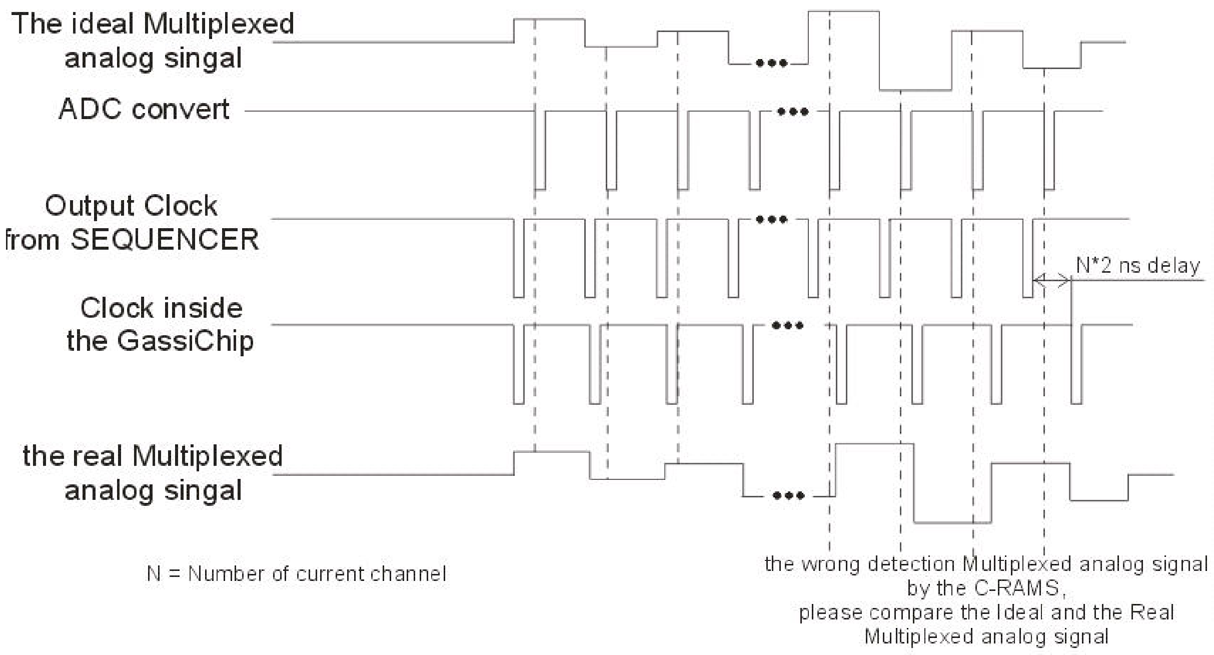

The ideal Multiplexed analog singal
ADC convert
Output Clock from SEQUENCER
Clock inside the GassiChip
the real Multiplexed analog singal
N*2 ns delay
N = Number of current channel
the wrong detection Multiplexed analog signal by the C-RAMS, please compare the Ideal and the Real Multiplexed analog signal

**Figure 4.6 Readout sequence inside GassiChips**

## 4.2.2 The C-RAMS setting

A few parameters are set like:

- Status Register
- Number of Channels Register
- Pedestal and Threshold Memories

A zero value is written to the Status register since the module does not work in the test mode and the VME is owner of pedestal and threshold memories.

An entry of Number of Channels register is the number of GassiPlex channels divided by 32. In a case when the number of channels is not divided by 32 the larger value has to be written. In our case the 864 channels are read so the 27 is written to the Number of Channels register.

The Pedestal and Threshold Memories are cleared in order to take the pedestals and calculates the thresholds.

### 4.2.3 The TDC setting

A few parameters are set like:

- Control Register 2
- Disable memory test
- Acquisition mode
- Window width
- Window offset
- Substraction of trigger time
- Data ready mode
- Channels enable

A zero value is written to the Control Register two, this indicates that module is set to a normal mode.

Since the module is set to the normal mode a memory is disabled for testing.

Since the module has to work in the *Common Stop Emmulation*, that operation is not foreesen by software setting, the Stop Trigger Matching mode has to be chosen.

It is required that all single TGC signals have to be within the Window width (see Figure 4.) A width equals to 625 ns was chosen to meet this condition.

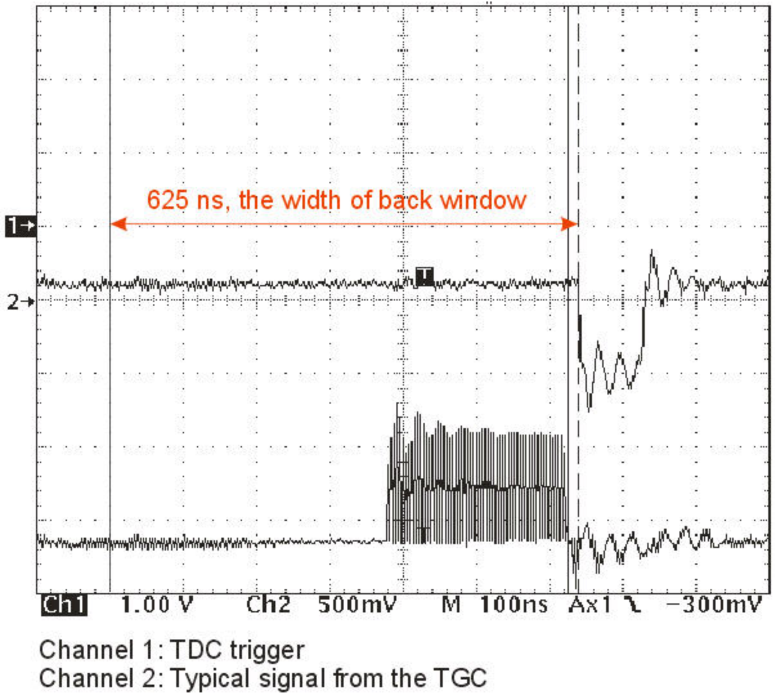


**Figure 4.7 The back window width of the TDC**

Since the module works in the *Common Stop Emulation* the window offset has to be equal to the Window width.

An offline program uses the top and the bottom trigger time to calculate the time response of the TGC. That time has to be independent of the TDC trigger time. Thus the substraction of trigger time should be disabled.

The data in an output buffer of the TDC for the *Stop Trigger Matching* mode is organized in an event structure. Thus the event ready was chosen to indicate that the data is ready.

The channels enable setting is optional. The input channels of the TDC, which are connected to the ASDTDC, can be enabled and the rest of them disabled.

## 4.3 Pedestals and Thresholds

The Pedestals and Thresholds memory of C-RAMS has to be set before the acquisition data process (see Figure 4.12). However during that operation the pedestals and thresholds memory has to be clear. A sample of 2,000 events are taken to calculate the average value of pedestal and the threshold value. The threshold is the sum of the average pedestal plus sigma. Each channel has its own pedestal and sigma value. The average pedestal is calculated using the routines MINUIT, an analysis tool developed at CERN.
A threshold is calculated according to the following equation:

$$Thr_{ch} = < p_{ch} > + M * \Gamma_{ch} \qquad \text{5-1}$$
$$\text{where } M \text{ is a selectable constant usually} \geq 3$$

After the 2000 events are accumulated the routine has enough data to calculate pedestals and thresholds. In our case the online has to calculate the 1728 pedestal and threshold values. The distributions of the Precision Chamber channels are shown in Figure 4.8 and Figure 4.9.Since the number of Precision Chember channels for one coordinate is not multiplication of 48 (number of channels of one GassiPlex) one or two GassiPlex are not full connected to the Precision Chamber. However the CLOCK pulses go through all the GassiPlex so the channels which are not connected to the Precision Chamber has to be disabled. It is performed by setting the threshold values of these channels to maximum (0xFFF). In that case the online code will never take data from them. The example of an event in the Precision Chamber is shown in Figure 4.9.

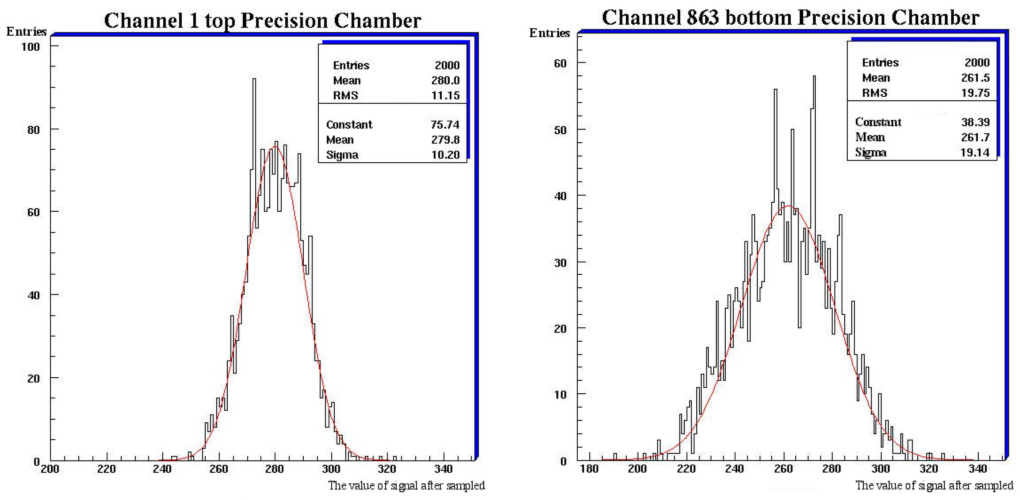


**Figure 4.8 Typical distribution of pedestals of top Presicion Chamber**

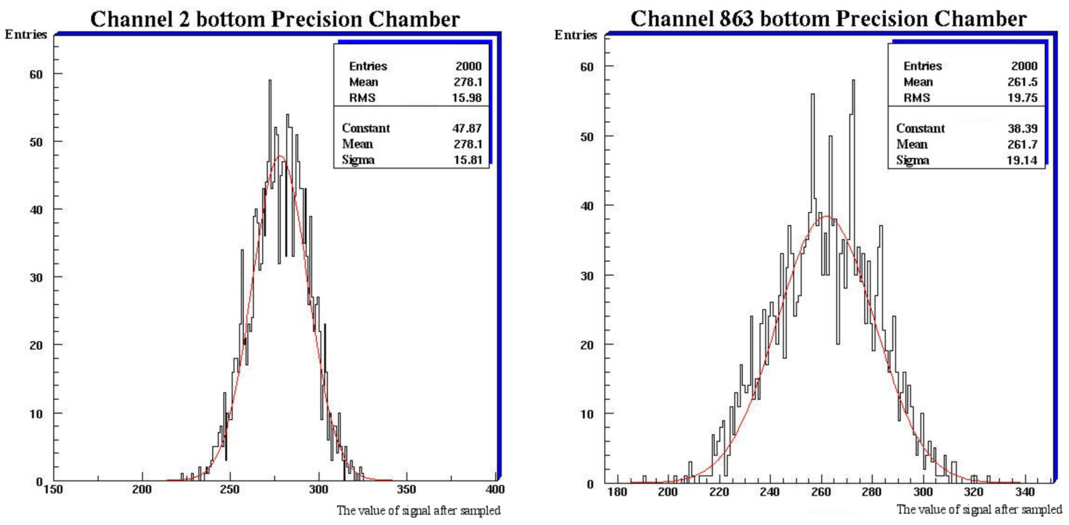
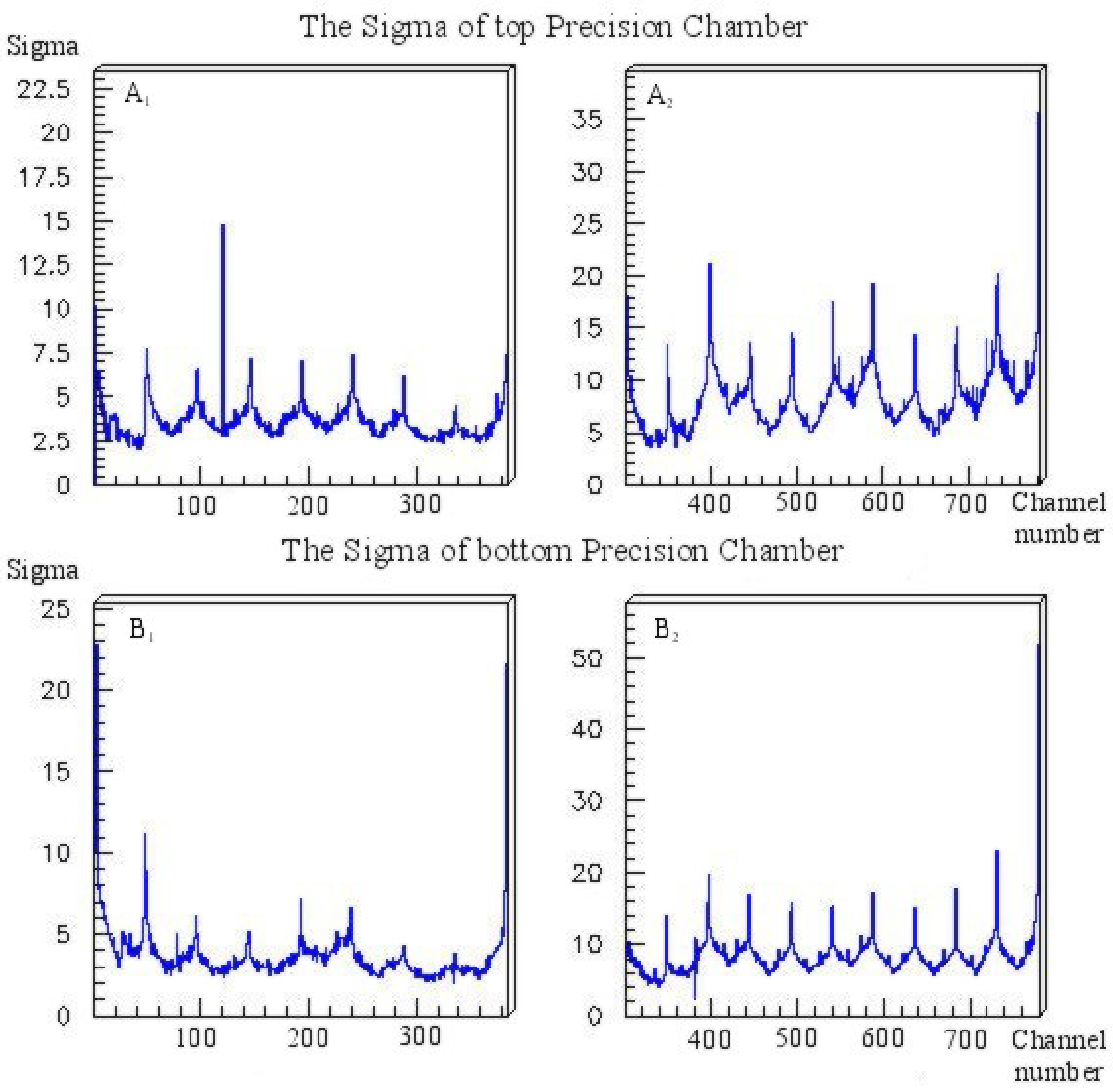


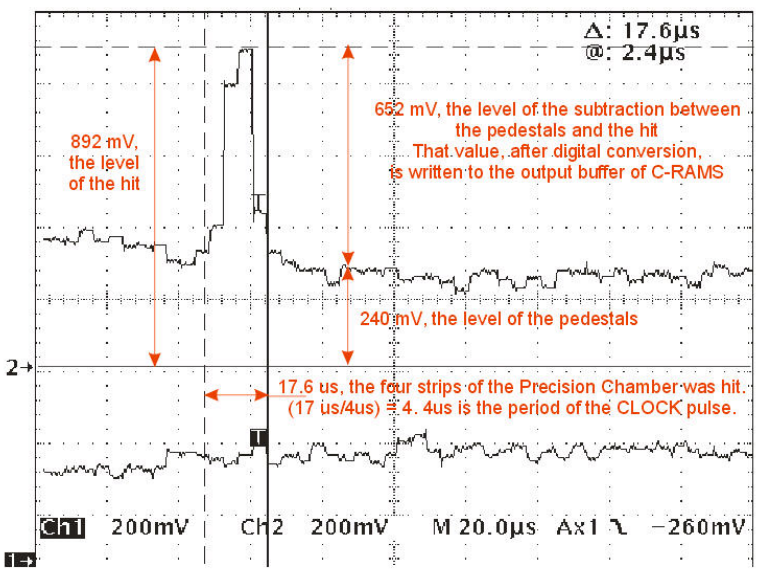
**Figure 4.9 Typical distribution of pedestals of the bottom Presicion Chamber**



The x axis represents the number of channels. The y axis represents the sigma value.
A1 – X coordinate top Precision Chamber
A2 – Y coordinate top Precision Chamber
B1 – X coordinate bottom Precision Chamber
B2 – Y coordinate bottom Precision Chamber

**Figure 4.10 The sigma for all channels of both Precision Chambers**

In Figure 4. we can see that the value of sigma is bigger for the channels from 384 to 864 then channels from one to 384 on the both Precision Chambers. It means that the y coordinate of the Precision Chambers is more noisy then the x coordinate.



Channel 1: signal from the top Precision Chamber
Channel 2: signal from the bottom Precision Chamber

**Figure 4.91 The example of the hit of the Precision Chamber**

## *4.4 The Data Acquisition program*

That routine controls and synhronizes the reading data from the VME cards. The number of event data is set by the user. This funcitons can be divided to three time steps as demonstrated in Figure 4.11

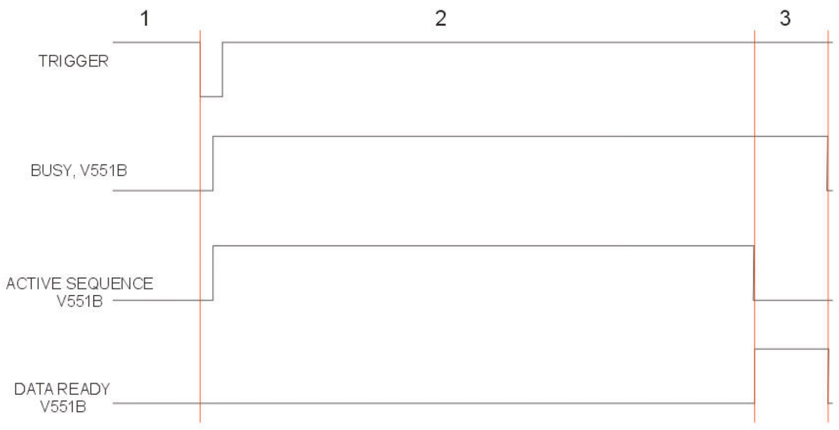


**Figure 4.112 The structure of data acquisition routine.**

In section one the routine waits for certain time for a data ready in the TDC (see $P_1$ on Figure 4.12). When the routine detects data the program moves to stage two. If in the TDC there is no data after that time, the routine clears the TDC and C-RAMS and jumps to step one (see $P_2$). The program checks the status of SEQUENCER, C-RAMS, TDC at the beginning of that operation (see $P_3$). The status should be the following:

- status of SEQUENCER: busy, active sequence, no data ready;
- status of C-RAMS: no half full buffer ADC0 and ADC1, no full buffer ADC0 and ADC1;
- status of TDC: data ready, no busy, no buffer empty, no buffer almoust full, no buffer full.

If at least one status is not correct the program will clear the TDC and the C-RAMS and jump to the step one.

In step two the program is waiting for data in the C-RAMS (see $P_4$). While waiting for data ready in the C-RAMS the program checks the status of the SEQUENCER (see $P_5$). If status is no active sequence and no data ready, it means that the C-RAMS has not data, the routine clears the TDC and the C-RAMS and jumps to step one. If the C-RAMS has data the function moves to step one. At the beginning of that operation the routine checks the status of the SEQUENCER, C-RAMS, TDC (see $P_6$). The status should be the following:

- status of SEQUENCER: busy, no active sequence, data ready;
- status of C-RAMS: data ready in ADC0, data ready in ADC1, no buffer empty in ADC0 and ADC1, no buffer half full ADC0 and ADC1, no buffer full ADC0 and ADC1;
- status of TDC: data ready, no busy, no buffer empty, no buffer almoust full, no buffer full.

If one or more status is wrong the program will clear the TDC and C-RAMS and jump to step one. Otherwise the routine reads the data from the TDC and the C-RAMS (see $P_7$ and $P_9$). Sometimes the buffer of the TDC has only Header and EOB (see § 3.3.1), in that case the routine clears the C-RAMS and jumps to step one (see $P_8$). If the buffer of the C-RAMS has the data which indicates overrange, the routine clears the C-RAMS and jumps to step one (see $P_{10}$). If all the required conditions are fulfilled the program writes data to the file (see $P_{11}$), increments the number of accepted event (see $P_{12}$) and jumps to step one.

The clearing of the TDC and the C-RAMS has to be done, otherwise the system would stop since data which are in the buffer C-RAMS generates the data ready and the SEQENCER will be in a busy mode permanently. That means that trigger will be rejected by the busy and system will not get the next trigger.

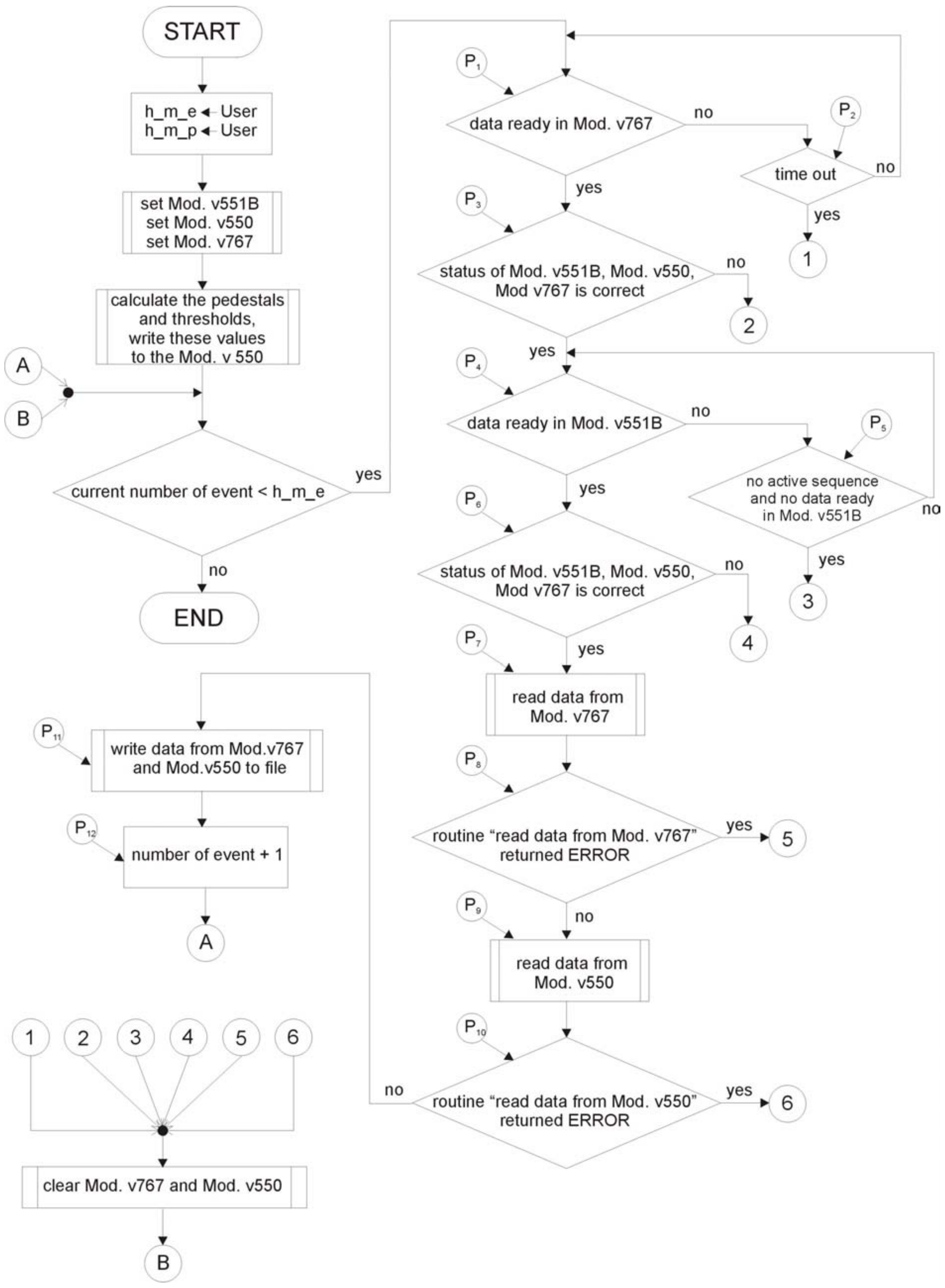

START
h_m_e ← User
h_m_p ← User
set Mod. v551B
set Mod. v550
set Mod. v767
calculate the pedestals and thresholds, write these values to the Mod. v 550
A
B
current number of event < h_m_e
yes
no
END
P1
data ready in Mod. v767
no
P2
time out
no
yes
1
yes
P3
status of Mod. v551B, Mod. v550, Mod v767 is correct
no
2
yes
P4
data ready in Mod. v551B
no
P5
no active sequence and no data ready in Mod. v551B
no
yes
3
yes
P6
status of Mod. v551B, Mod. v550, Mod v767 is correct
no
4
P7
yes
read data from Mod. v767
P8
routine "read data from Mod. v767" returned ERROR
yes
5
P9
no
read data from Mod. v550
P10
routine "read data from Mod. v550" returned ERROR
yes
6
no
P11
write data from Mod.v767 and Mod.v550 to file
P12
number of event + 1
A
1
2
3
4
5
6
clear Mod. v767 and Mod. v550
B

**Figure 4.123 Algoritm of the online code.**

## 4.5 Data acqusition system monitoring

During the data acqusition the online program creates a few files which are used to monitor the behaviour of the system while running.

The following pictures are an example of the monitoring online plots.

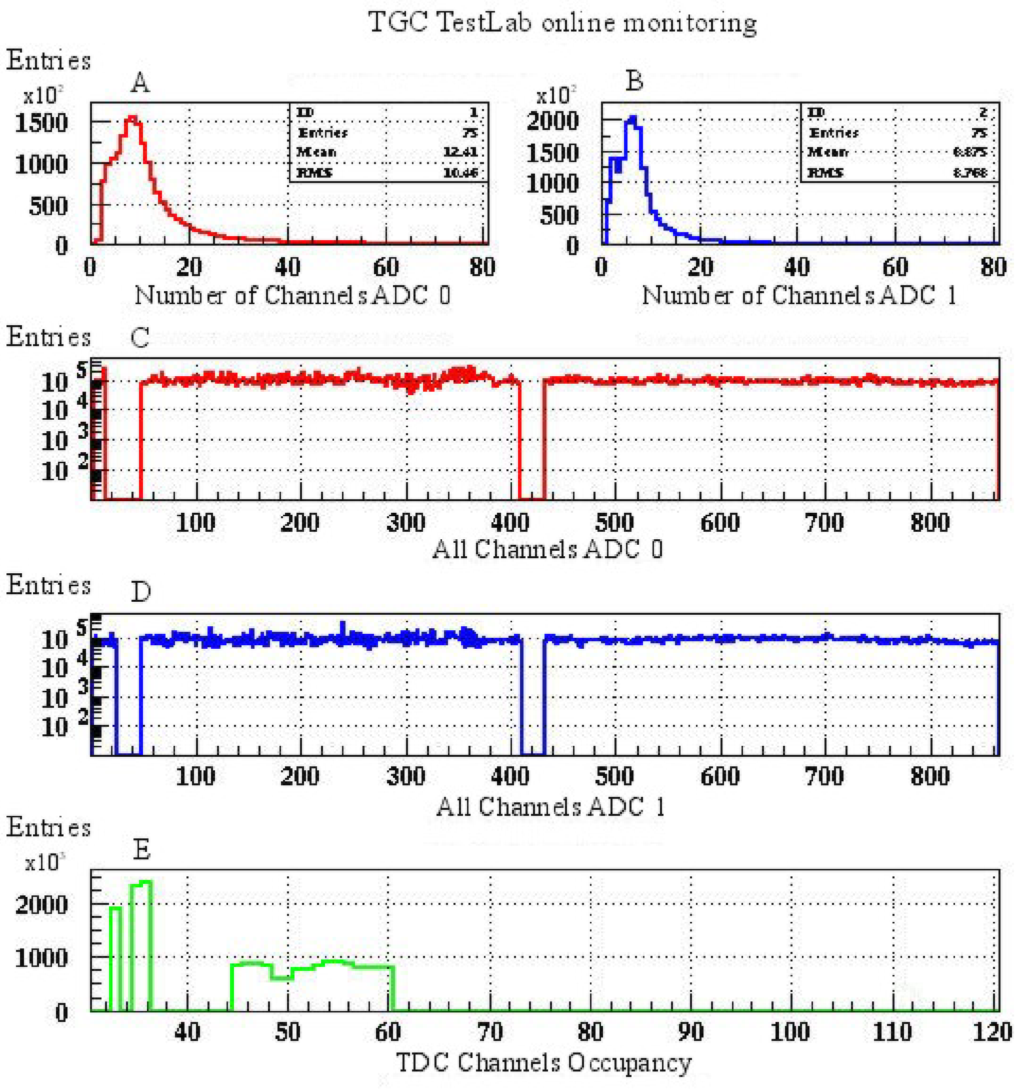


**Figure 4.14 The monitoring**

The pictures A and B show histogram of the number of strips produced per one event. The y axis presents the number of entries in a linear scale. The x axis presents the number of strips which were hit by one muon. Histogram A has larger mean value (12.41) then histogram B (mean value 9.8). It means that this Precision Chamber is more sensitive than the second one, which allows a better position measurement with the muon intersection point.

The pictures C and D present the number of entries in each channel of both Precision Chambers. The y axis presents the number of entries in the logarithmic scale and the x axis is equivalent to the number of channels. We can see that there are regions with no entries. These channels are not connected to the Precision Chambers and they are disabled by the online code.

Picture E gives us the test pattern of the TDC. The y axis represents the number of entries in the linear scale, and the x axis is equal to the number of channels in the TDC. The first peak is the TDC trigger. Second and third peaks are the top and bottom scintillators hits. The trigger obviously has fewer number of entries than the top and bottom hits. This is the result of the trigger being formed by an AND coincidence between the top and bottom hits. The rest of the entries in that histogram are the data from the TGCs.

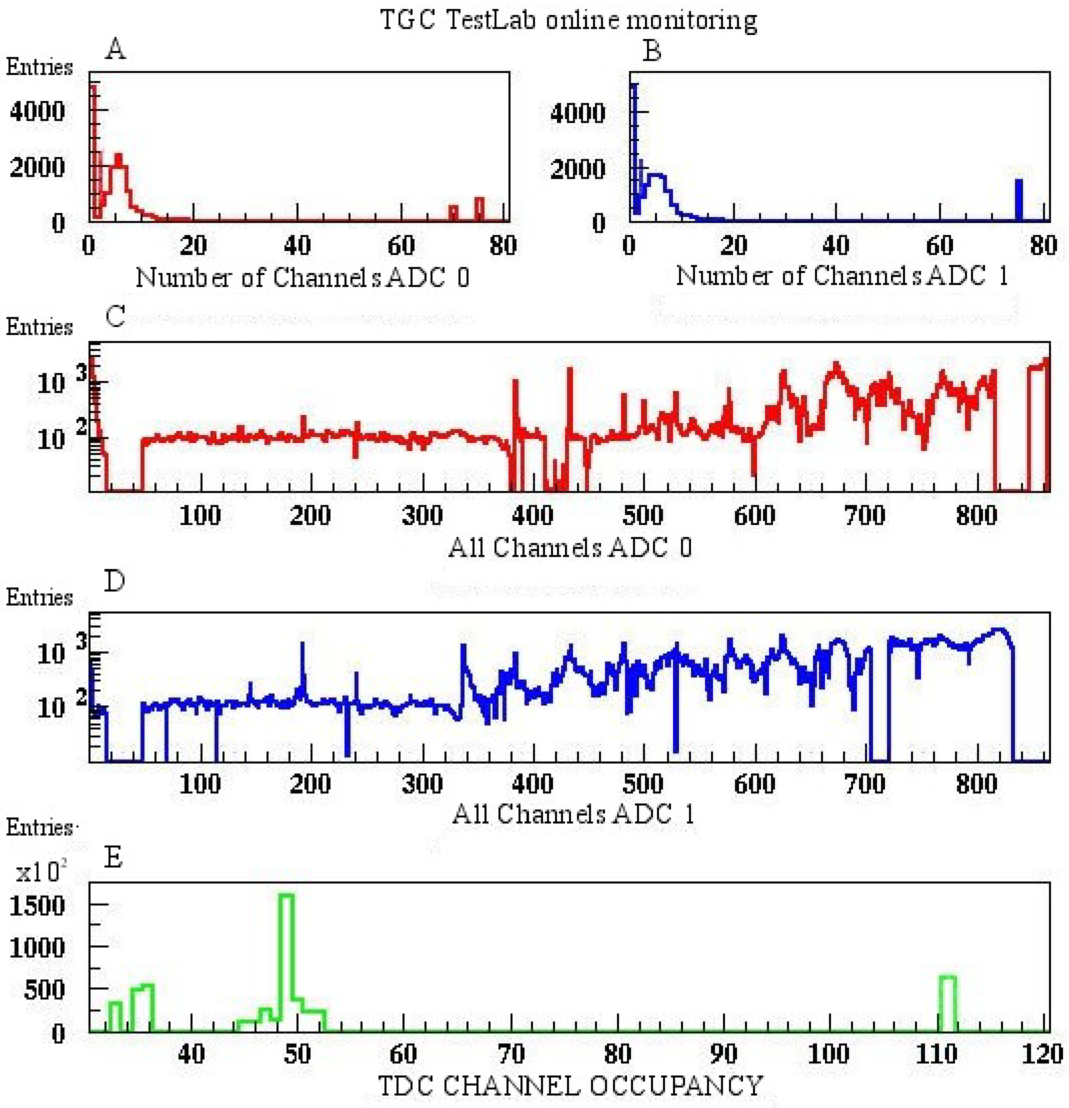


**Figure 4.15 Early stage monitoring plots**

When starting operating there were several topics which required tuning and fixing. It could be demonstrated in the online monitoring plots of early runs. For example poor results of the monitoring are shown in Figure 4.15. The Precision Chambers have a lot of events which have only one entry. This can happen when the Precision Chamber voltage is too low, it has poor micture of gas or threshold value is set too high. The peak shown around channel 75 means that the Presicion Chambers have some events which have plenty of entries (more then 75). This may be a result of the threshold set too low (see picture A and B). It can be fixed by raising the threshold values. All the channels of the Precision Chambers are shown in the picture C and D. The difference between channels from zero to 350 can be observed. The channels from zero to 350 are channels of the x coordinate and the rest of them are the y

coordinate of the Precision Chamber. The y coordinates have more entries then the x coordinate. This indicates that the readout from the y coordinate is more unstable and the threshold value might be too low.
The TDC channels status is shown on figure E. A large peak can be noticed on channel 49. It maybe a result setting ASD threshold too low or a problem with the ASDB.

# 5 Results

The efficiency of the TGC chambers is calculated an offline code. That C++ analysis program calculates the trajectory of muon and estimates the point muon crossing through the TGC. The TGC surface is divided into small square bins (1cm$^2$). The efficiency of each square is equal the number of muons extrapolated to go through it and detected by the TGC divided by all muons extrapoolated to go through that bin. The plots of efficiency maps and the timing are done by ROOT An Object-Oriented Data Analysis Framework.

The scale of colors are shown on the right side of the map plots. The white color represent a 100% efficiency and the black one is equal to efficiency under 50%. The TDC doublet has two detectors each of which with one plane of wires and one plane of strips. The criteria of a detector to pass the cosmic ray test is when the 95% of its surface has the efficiency over then 95%.

## 5.1 The timing of the Scintillators

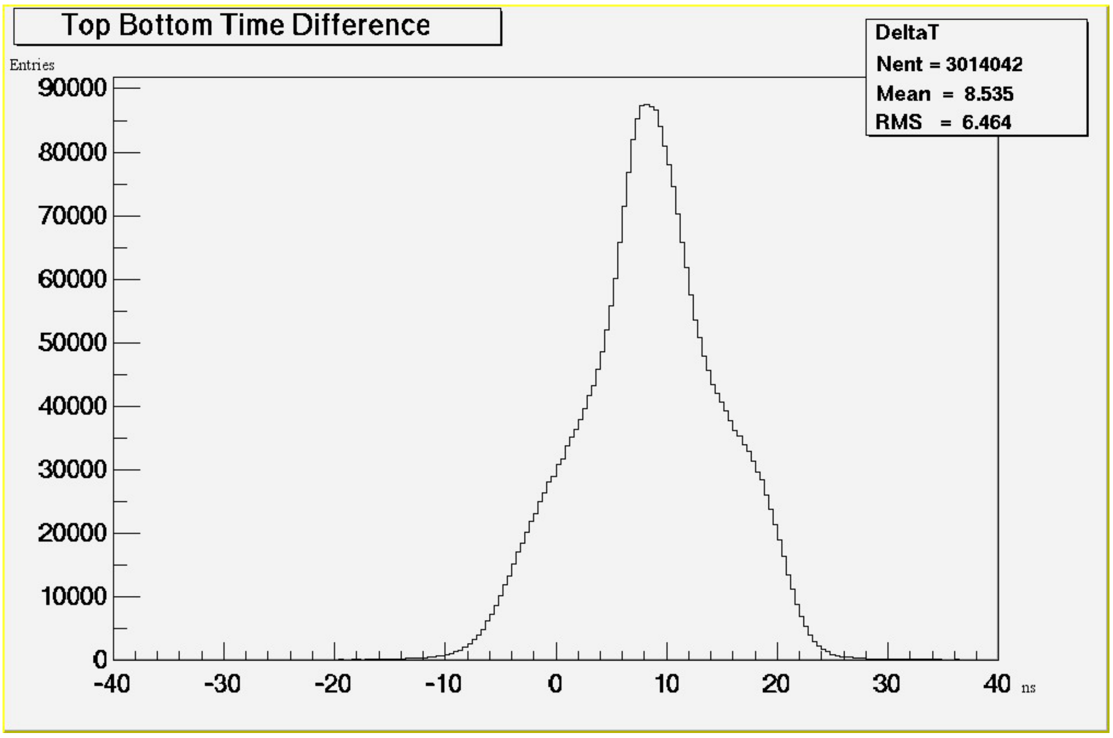


**Figure 5.1 The timing of the Scintillators.**

The Figure 5.1 shows the distribution of the time difference between the arrival time of the top and the bottom hits. This distribution is added to the timing of the TGCs unit. The distribution width depends on several factors:

- The muon crossing point in the Scintillators;
- The time response of the PMTs;

- The voltage of the PMTs;
- The logical gates which are used to generate the trigger.

To get more precise timing of the TGCs this distribution should be narrower.

## 5.2 Example - The results of unit U08F2I-521.0

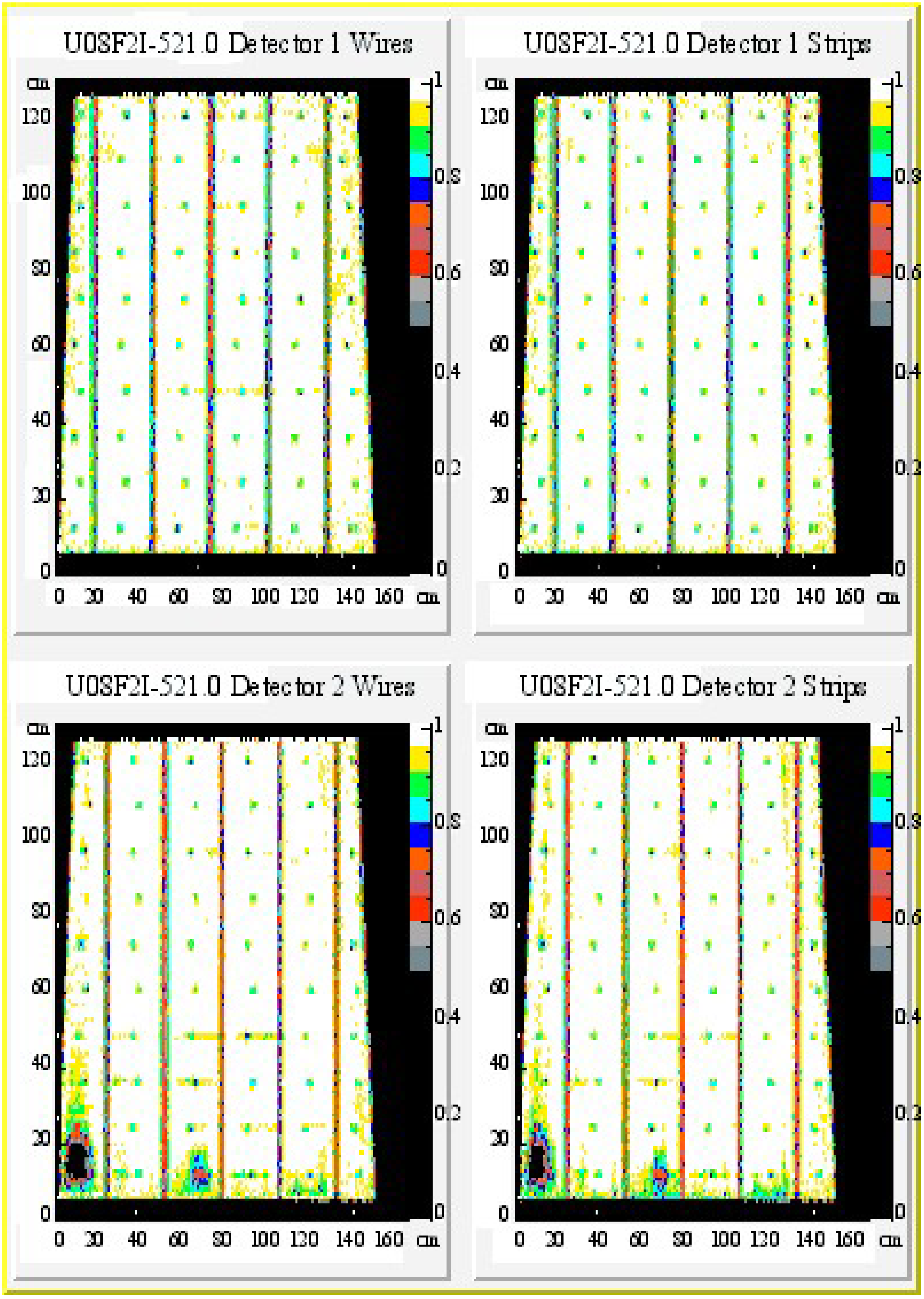


**Figure 5.2 The efficiency map of unit U08F2I-521.0**

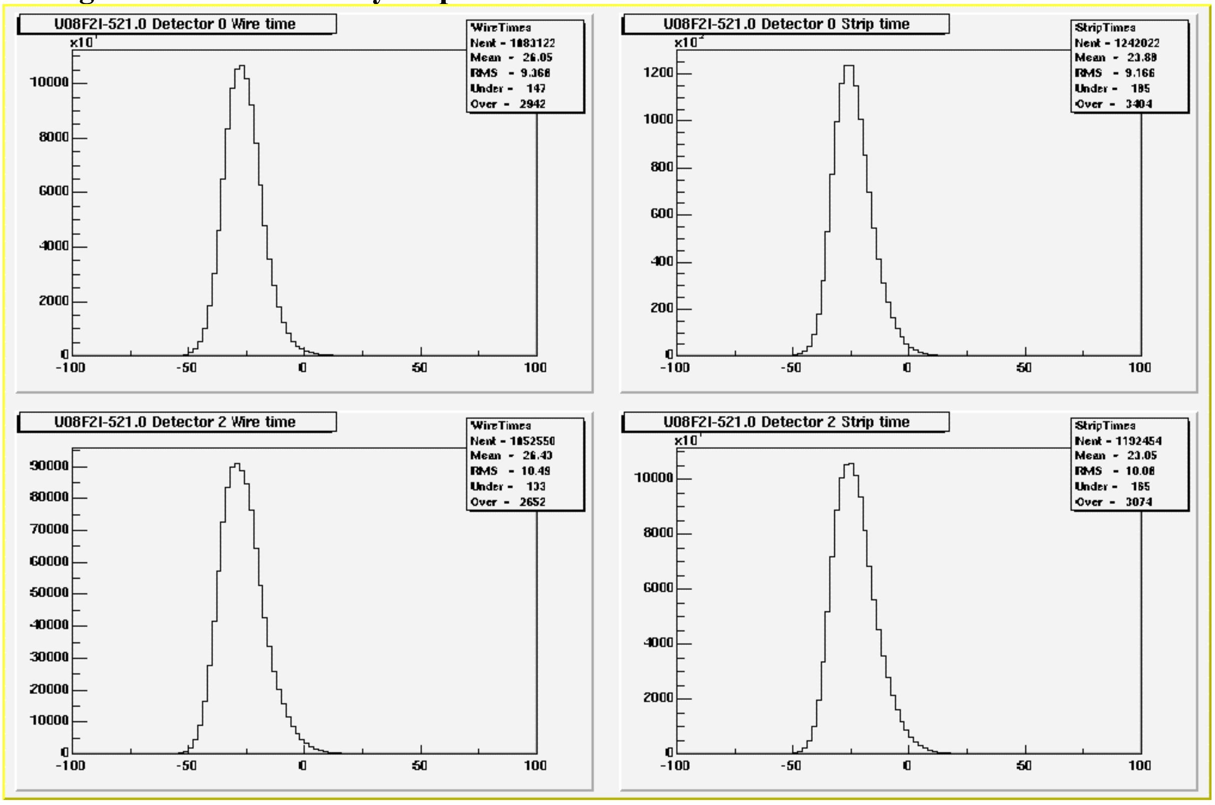


**Figure 5.3 The timing of the unit U08F2I-521.0**

The efficiency map of this unit is shown on the Figure 5.2. The plot clearly shows the structure of the tested TGC chamber. The lines and separators which have the efficiency under the 50 % can be seen on the surface of the TGC. They are the supports and separators of the TGC. We obviously expect low efficiency in these regions.
The first detector is efficienent uniformly on the all its surface but the second detector has few unefficient areas. These unefficient regions are probably a result of gluing problems during the construction of the chamber. However the edges of the TGCs are sharp and there is no hits outside the chambers. It seems that all surface is well covered by the Precision Chambers and the Scintillators and the offline code determines the trajectory and the hit points in the TGC correctly. To obtain good resolution of the efficiency map a few milion events are reguired. Each square of the TGC surface should have at the order of 50 hits for precision of about 5%. The online code collects this amount of data during about five days of run. Table 5.1 is showing the efficiency of this unit.

| **Unit** | **U08F2I-521.0** | | | |
|---|---|---|---|---|
| **Layer** | detector 1 Wires | Detector 1 strips | detector 2 wires | detector 2 strips |
| | **percent of surface** | | | |
| 90% eff. | 99.3% | 100% | 97.8% | 97.8% |
| 95% eff. | 92.7% | 93.9% | 89.2% | 89.2% |

**Table 5.1 The efficiency of the unit U08F2I-521.0**

The timing of this unit is shown on the Figure 5.3. All chambers a similar width distribution. The distribution should be narrower within the 25 ns window. However one needs to have in mind the wide distribution of the  Scintillators, logical units, TDC which affect these plots. We are currently working on further improving the performance of the scintillators timing measurement.

## 5.3 *Another example - The result of unit U08F2I-548.9.*

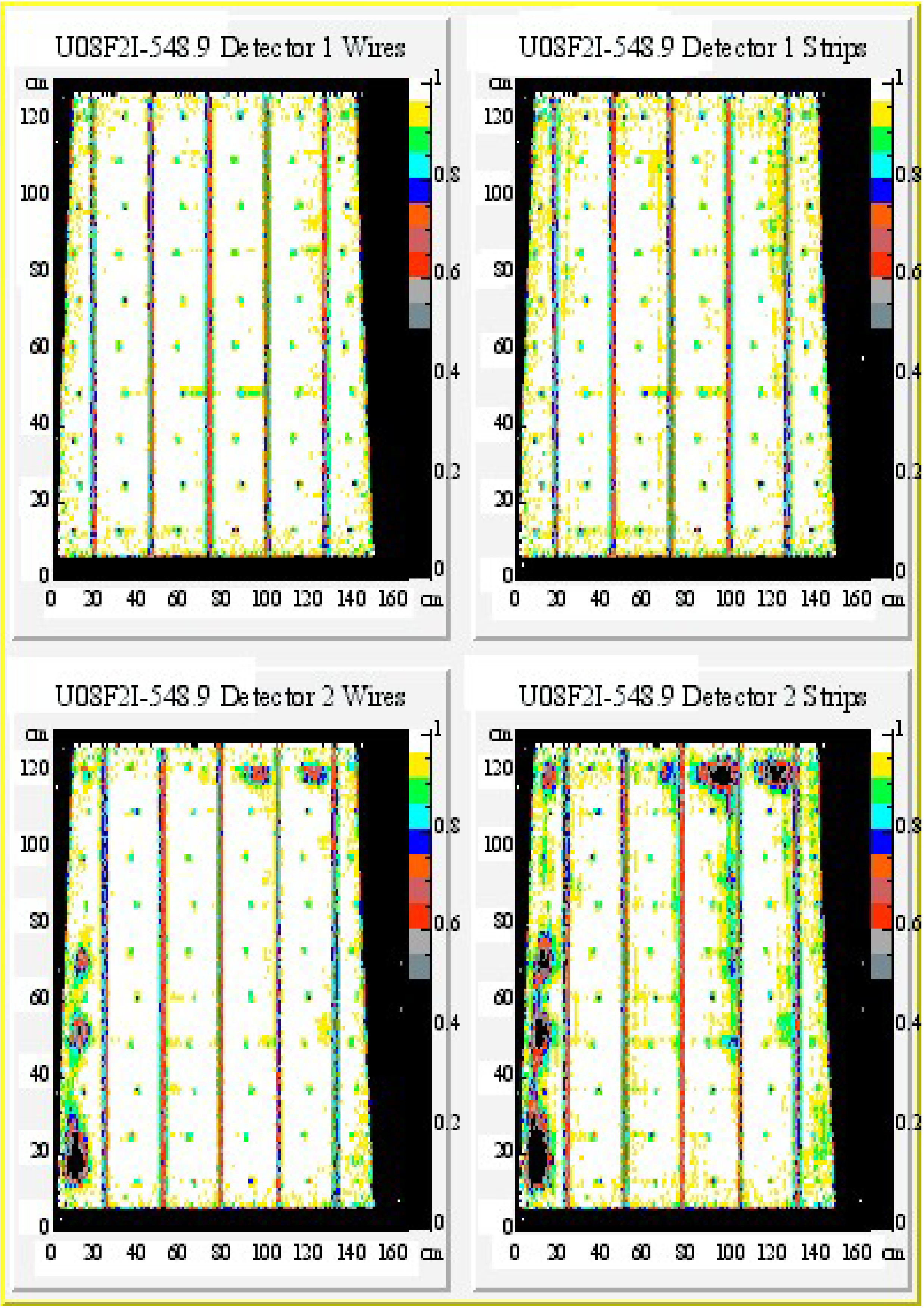


**Figure 5.4 The efficiency map of unit U08F2I-548.9**

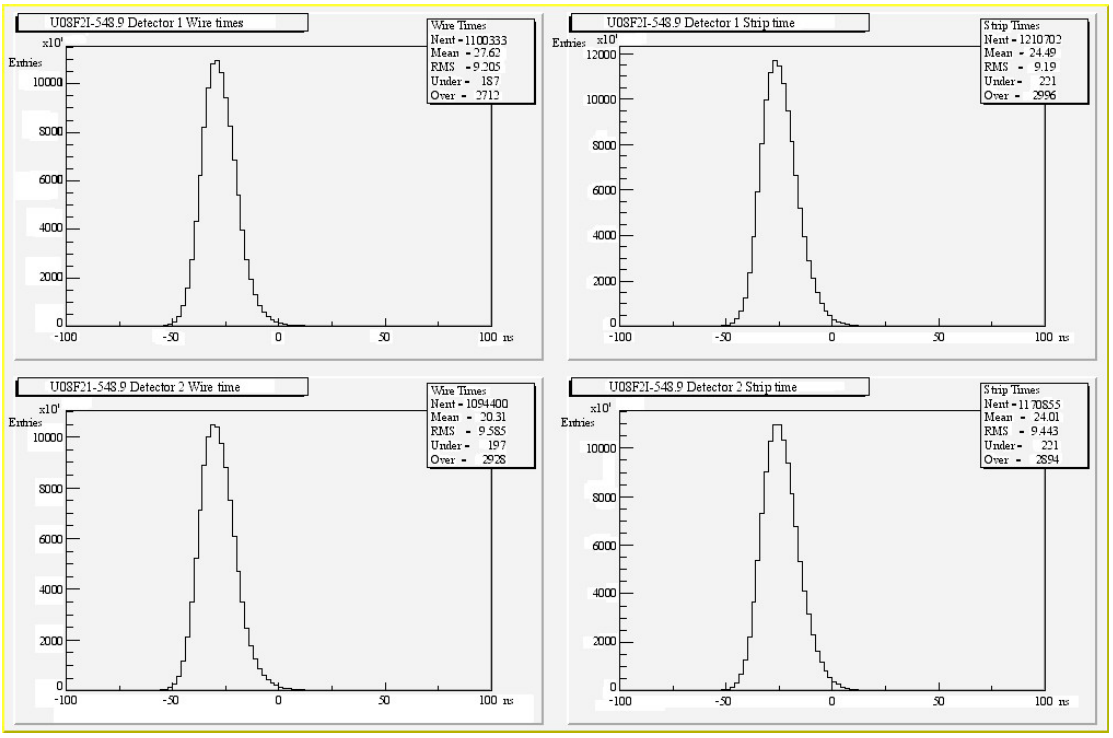


**Figure 5.5 The timing of the unit U08F2I-548.9**

In Figures 5.4-5.5 we presnt another example of tested TGC. This chamber performed worse than the first chamber since it has some regions of lowere efficiency at its edges. The supports and the sepparators are clearly showen. The edges of unit are sharp.
The few black spots which have efficiency under 50% can be observed on the first detector of this unit. The timing of that unit is similar to the previous chamber Table 5.2 is showing the efficiency of that unit. This is an example of a cahmber that fails our QA and is returned to production line for further tests.

| **Unit** | **U08F2I-548.9** | | | |
|---|---|---|---|---|
| **Layer** | detector 1 Wires | detector 1 strips | detector 2 wires | detector 2 strips |
| | **percent of surface** | | | |
| 90% eff. | 98.1% | 97.2% | 94.1% | 86.1% |
| 95% eff. | 90.3% | 85.2% | 84.7% | 70.6% |

**Table 5.2 The efficiency of the unit U08F2I-548.9**

# 6 Summary

This work has describes the Test Bench that was built in order to test the muon endcap trigger chambers of the ATLAS detector. We already successully managed to test five TGC chambers. The results are sufficient, providing detailed mapping of the efficiency of the tested chambers. There are still some open topics that need further tunning, but at this point we go to production mode and any further improvement will be done in parallel to actual TGCs testing. This Test Bench is now starting to take data and it will continue testing about 1000 chambers in the next few years.

# 7 Bibliography

[1] *ATLAS Detector and physics performance Technical Design Report, CERN/LHCC/95-15, 25 May 1999*

[2] *Techniques for Nuclear and Particle Physics Expermients, William R. Leo, 9 September 1994*

[3] *ATLAS First-level trigger Technical Design Report, CERN/LHCC/98-14, 20 June 1998*

[4] M*ultipurpose Analog Multiplexed Read Out System For Pad/Strip Detectors: Description And instruction Manual, CERN 24/1/1977*

[5] *Technical Information Manual, Version 1.1, 9 July 1999, MOD. V 551B C-RAMS SEQUENCER, CAEN*

[6] *Technical Information Manual, Version 1.0, 9 April 1999, MOD. V500 2 CHANNELS C-RAMS, CAEN*

[7] *Technical Information Manual, Revision n.2 ,1 June 1999, MOD. V767/V767B 128 CH. GENERAL PURPOSE MULTIHIT TDC, CAEN*

[8] *ATLAS THIN GAP CHAMBER, Amplifier-Shaper-Discriminator Ics, Production Readines Review Report, 12 November, ATLAS TGC Collaboration*

[9] *Error reporting in the I/O module libraries, Note number: 51, Version: 3.2, 29 April 19*
*http://rd13doc.cern.ch/Atlas/Notes/051/Note051-1.htm*

# Appendix I: VMEBUS Specification

## 1.1 VMEBUS Specification Objectives

The VMEbus specification defines an interfacing system used to interconnect microprocessors, data storage, and peripheral control devices in a closely coupled hardware configuration. The system has been conceived with the following objectives:

- **a.** To allow communication between devices on the VMEbus without disturbing the internal activities of other devices interfaced to the VMEbus.
- **b.** To specify the electrical and mechanical system characteristics required to design devices that will reliably and unambiguously communicate with other devices interfaced to the VMEbus.
- **c.** To specify protocols that precisely define the interaction between the VMEbus and devices interfaced to it.
- **d.** To provide terminology and definitions that describe system protocol.
- **e.** To allow a broad range of design latitude so that the designer can optimize cost and/or performance without affecting system compatibility.
- **f.** To provide a system where performance is primarily device limited, rather than system interface limited.

## 1.2 VMEBUS Interface System Elements

**Definitions**

The VMEbus structure can be described from two points of view: its mechanical structure and its functional structure. The mechanical specification describes the physical dimensions of subracks, backplanes, front panels, plug-in boards, etc. The VMEbus functional specification describes how the bus works, what functional modules are involved in each transaction, and the rules which govern their behavior. This section provides informal definitions for some basic terms used to describe both the mechanical and the functional structure of the VMEbus.

## 1.3 Terms Used to Describe the VMEbus Mechanical Structure

**VMEbus Backplane** — A printed circuit (PC) board with 96 or 160 pin connectors and signal paths that bus the connector pins. Some VMEbus systems have a single PC board, called the J1 backplane. It provides the signal paths needed for basic operation. Other VMEbus systems also have an optional second PC board, called a J2 backplane. It provides the additional 96 or 160 pin connectors and signal paths needed for wider data and address transfers. Still others have a single PC board that provides the signal conductors and connectors of both the J1 and J2 backplanes.

**Board** — A printed circuit (PC) board, its collection of electronic components, with either one or two 96 or 160 pin connectors that can be plugged into VMEbus backplane connectors.

**Slot** — A position where a board can be inserted into a VMEbus backplane. If the VMEbus system has both a J1 and a J2 backplane (or a combination J1/J2 backplane) each slot provides a pair of 96 or 160 pin connectors. If the system has only a J1 backplane, then each slot provides a single 96 or 160 pin connector.

**Subrack** — A rigid framework that provides mechanical support for boards inserted into the backplane, ensuring that the connectors mate properly and that adjacent boards do not contact each other. It also guides the cooling airflow through the system, and ensures that inserted boards do not disengage themselves from the backplane due to vibration or shock.

## *1.4 Terms Used to Describe the VMEbus Functional Structure*

Figure AI.1 shows a simplified block diagram of the functional structure, including the VMEbus signal lines, backplane interface logic, and functional modules.

**Backplane Interface Logic** — Special interface logic that takes into account the characteristics of the backplane: its signal line impedance, propagation time, termination values, etc. The VMEbus specification prescribes certain rules for the design of this logic based on the maximum length of the backplane and its maximum number of board slots.

**Functional Module** — A collection of electronic circuitry that resides on one VMEbus board and works together to accomplish a task.

**Data Transfer Bus** — One of the four buses provided by the VMEbus backplane. The Data Transfer Bus allows Masters to direct the transfer of binary data between themselves and Slaves. (Data Transfer Bus is often abbreviated DTB.)

**Data Transfer Bus Cycle** — A sequence of level transitions on the signal lines of the DTB that result in the transfer of an address or an address and data between a Master and a Slave. The Data Transfer Bus cycle is divided into two portions, the address broadcast and then zero or more data transfers. There are 34 types of Data Transfer Bus cycles. They are defined later in this chapter.

**Master** — A functional module that initiates DTB cycles in order to transfer data between itself and a Slave module.

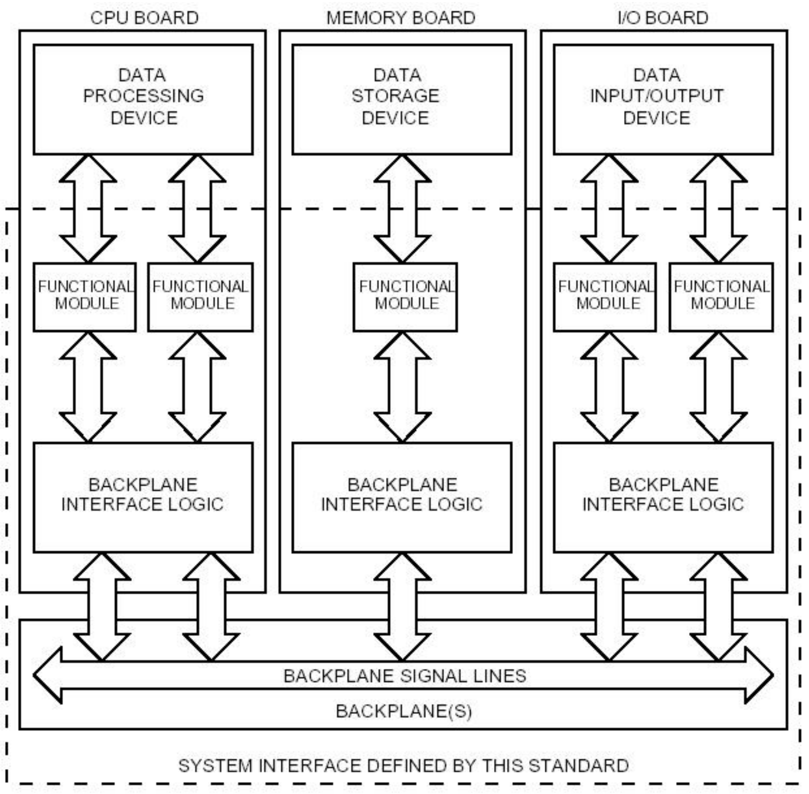


**Fig. AI.1** System Elements Defined.

**Slave** — A functional module that detects DTB cycles initiated by a Master and, when those cycles specify its participation, transfers data between itself and the Master.

**Interrupter** — A functional module that generates an interrupt request on the Priority Interrupt Bus and then provides Status/ID information when the Interrupt Handler requests it.

**Interrupt Handler** — A functional module that detects interrupt requests generated by Interrupters and responds to those requests by asking for Status/ID information.

**CR/CSR --** A functional module that provides Configuration ROM information and Control and Status Registers. The module provides manufacturing and board ID and other important board information. The CSRs are used for software configuration of a VMEbus system.

## *1.5 Types of Cycles on the VMEbus*

**Read Cycle** — A DTB cycle used to transfer 1, 2, 3, 4 or 8 bytes from a Slave to a Master. The cycle begins when the Master broadcasts an address and an address modifier. Each Slave captures the address modifier and address and checks to see if it is to respond to the cycle. If so, it retrieves the data from its internal storage, places it on the data bus and acknowledges the transfer. The Master then terminates the cycle.

**Write Cycle** — A DTB cycle used to transfer 1, 2, 3, 4 or 8 bytes from a Master to a Slave. The cycle begins when the Master broadcasts an address and address modifier and places data on the DTB. Each Slave captures the address and address modifier and checks to see if it is to respond to the cycle. If so, it stores the data and then acknowledges the transfer. The Master then terminates the cycle.

**Block Read Cycle** — A DTB cycle used to transfer a block of 1 to 256 bytes from a Slave to a Master. This transfer is done using a string of 1, 2, or 4 byte data transfers. Once the block transfer is started, the Master does not release the DTB until all of the bytes have been transferred. It differs from a string of read cycles in that the Master broadcasts only one address and address modifier (at the beginning of the cycle). Then the Slave increments this address on each transfer so that the data for the next transfer is retrieved from the next higher location.

**Block Write Cycle** — A DTB cycle used to transfer a block of 1 to 256 bytes from a Master to a Slave. The block write cycle is very similar to the block read cycle. It uses a string of 1, 2, or 4 byte data transfers. The Master does not release the DTB until all of the bytes have been transferred. It differs from a string of write cycles in that the Master broadcasts only one address and address modifier (at the beginning of the cycle). Then the Slave increments this address on each transfer so that the data from the next transfer is stored in the next higher location.

**Address-Only Cycle** — A DTB cycle that consists of an address broadcast, but no data transfer. Slaves do not acknowledge ADDRESS-ONLY cycles and Masters terminate the cycle without waiting for an acknowledgment. No data strobes or acknowledge strobes are asserted in an ADDRESS-ONLY cycle.

**Interrupt Acknowledge Cycle** — A DTB cycle, initiated by an Interrupt Handler, which reads a STATUS/ID from an Interrupter. An Interrupt Handler generates this cycle whenever it detects an interrupt request from an Interrupter and it has control of the DTB.

# Appendix II: VMEbus Application Program Interface

This apendix describes the VME drivers and the online data acqusition program (DAQ). These programs will be used to control the VME cards and to take data from these card. The drivers contains functions related to the following uses of the VMEbus:

- Master mappings and single cycles;
- Bus error handling;

The drivers further contains type definitions, functions to handle the return codes and general functions for the use of the VMEbus.
The DAQ contains type definitions, functions to set the VME cards and functions for taking data from these cards.

## *1.1 Design Issues.*

### 1.1.1 Names.

The drivers uses readable and meaningful names for all of its fuctions, as well as the common prefix "vme_" for the mappings and the "iom_" for the error handling. The DAQ also uses the lucid names for its functions. A structure of DAQ program is organised that one functions has its own file.

### 1.1.2 Identifiers.

The drivers uses the identifiers of type "int" for the master mapping, slave mapping. However the DAQ uses the type "unsigned" for the reading/writing data to VME cards.

### 1.1.3 Return codes

All functions return an unambiguous return code. The return code is equal to 0 if the function has terminated without error. The return code is different from 0 in case the function terminated with error. A textual representation of the return code can be printed to "stdout" or to a string by the application program.

## *1.2 Implementation Issues.*

### 1.2.1 Drivers implemetation

The drivers uses the low-level device "/dev/mem" to make a memory of VME cards visible to DAQ. However the drivers work in the level of User. A connections between the VME cards memory and a User aplications memory is done by the system function "mmap" of Linux.

### 1.2.2 Bus error handling

The implementation of the drivers allows User to detect the erros which occur while opening and mapping the memory. However the read/write access to the memory is without checking the VME bus error, since these operations works in the memory of user aplications.

### 1.2.3 DAQ implementation

The DAQ uses the VME cards CR/SCR space to read and write data.

### 1.2.4 Language binding

C was chosen for the language binding of the drivers and the DAQ program.

### 1.2.5 Data types

A few types are used by the drivers and the DAQ program. The types are proposed for the following descriptions:

- "unsigned char" or "Uchar" or "Data8" for data
- "unsigned short" or "Ushort" or "Data16" for data or "Word16" for addresses;
- "unsigned int" or "Uint" or "Data32" for data or "Word32" for addresses;

The User knows the types of values and defines them in the application program. The compiler shall be used to enforce type safety fot the function calls.

## *1.3 Driver*

### 1.3.1 Overview

The following list is an overwiew of all type and functions definitions in the VMEbus drivers:

1. Bus Error Handling functions
- iom_error_err_get;
- iom_error_open;
- iom_error_close;
- iom_error_init;
- iom_error_print;
- iom_error_get;
- vmemap_err_get;
2. General Functions
- Init_vme;
- End_vme;
- VME_Open
- VME_Close;
3. Master Mapping functions
- VME_MstMap;
- VME_MstUnmap
4. Single Cycles read, write from, to VME cards functions
- vme_write16;
- vme_write32;
- vme_read16;
- vme_read32;

### 1.3.2 VMEbus error handler

#### 1.3.2.1 iom_error_open()

**Synopsis:**

err_type iom_error_open(void);

**Parameters:**

none

**Descriptions:**

The iom_error_open() function opens an iom error package.

**Return Values:**

| | |
|---|---|
| EIOM_OK | The library was successfully opened |

#### 1.3.2.2 iom_error_close()

**Synopsis:**

err_type iom_error_close(void);

**Parameters:**

none

**Descriptions:**

The iom_error_open() function closes an iom error package.

**Return Values:**

| | |
|---|---|
| EIOM_OK | The library was successfully closed |
| EIOM_NOTOPEN | The library was not open |

#### 1.3.2.3 iom_error_init()

**Synopsis:**

err_type iom_error_init( err_pid pid, err_type (*f)(err_pack,err_str, err_str) )

**Parameters:**

| | | |
|---|---|---|
| err_pid pid | in | error code |
| err_type (*f)(err_pack,err_str, err_str) | in | pointer to function |

**Descriptions:**

The iom_error_init is part of the IOM error library and is called from an IOM package to establish a connection between the package identifier and the package specific "err_get" function within the IOM error system. This function is typically called from within the "package open" call.

**Return Values:**

| | |
|---|---|
| EIOM_OK | The initialize was done successfully |

#### 1.3.2.4 iom_error_print()

**Synopsis:**

err_type iom_error_print(FILE* stream, err_type err)

**Parameters:**

| | | |
|---|---|---|
| FILE *stream | in | Stream |
| err_type err | in | error code |

**Descriptions:**

The iom_error_print() functions prints a textual representation of error code to the stream.

**Return Values:**

| State | MAJOR and MAINOR error |
|---|---|
| EIOM_STREAM | The stream was not opened |

#### 1.3.2.5 iom_error_get()

**Synopsis:**

err_type iom_error_get(err_type err, err_str Maj_Pid_str, err_str Maj_en_str, err_str Min_Pid_str, err_str Min_en_str)

**Parameters:**

| Err_type err | in | error code |
|---|---|---|
| err_str Maj_Pid_str | out | array to store a textual representation of major error code |
| err_str Maj_en_str | out | array to store a textual representation of major error string |
| err_str Min_Pid_str | out | array to store a textual representation of mainor error code |
| err_str Min_en_str | out | array to store a textual representation of mainor error string |

**Descriptions:**

The iom_error_get retrieves the package ID names and error text strings associated with an error code returned by an IOM library function and iom_error_print allows to print the messages

**Return Values:**

| | |
|---|---|
| EIOM_NOINIT | The iom_error_init function was not called |
| EIOM_NOCODE | The code of error is not know |
| EIOM_OK | The iom_error_init function was finish succesfully |

#### 1.3.2.6 vmemap_err_get()

vmemap_err_get;
*packxyz_err_get* is a function which has to be provided by every IOM library. This function is totally internal to the package and is referenced only via its function pointer from within the IOM error system.

### 1.3.3 General functions

#### 1.3.3.1 init_vme()

**Synopsis:**

unsiged int init_vme(Word32 base_add, Word16 size, Data8 mode)

**Parameters:**

| | | |
|---|---|---|
| Word32 base_add | in | Base address of VME card |
| Word16 size | in | Size of memory to be mapped |
| Data8 mode | in | The access mode to the mapped memory |

**Descriptions:**

The init_vme() function initialize the driver. That function calls the VME_Open(), VME_MstMap() routines.

**Return Values:**

| | |
|---|---|
| VMEMAPE_OK | The VMEbus device was successfully opened |
| Others | Specific to the implementation |

### 1.3.3.2 end_vme()

**Synopsis:**

void end_vme(Word32 log_adr, Word16 size)

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add | in | virutal address of mapped memory of VME card |
| Word16 size | in | Size of memory to be unmapped |

**Descriptions:**

The init_vme() function finishes the driver. That function calls the VME_MstUnmap(), VME_Close() routines.

**Return Values:**

none

### 1.3.3.3 VME_Open()

**Synopsis:**

err_type VME_Open(void)

**Parameters:**

none

**Descriptions:**

The VME_Open() function initialize the error VMEbus functions and opens the memory device.

**Return Values:**

| | |
|---|---|
| VMEMAPE_OK | The VMEbus device was successfully opened |
| Others | Specific to the implementation |

### 1.3.3.4 VME_Close()

**Synopsis:**

err_type VME_Close(void)

**Parameters:**

none

**Descriptions:**

The VME_Close() function closes the memory device.

**Return Values:**

| | |
|---|---|
| VMEMAPE_ISNOTOPEN | The VMEbus device was not opened |
| VMEMAPE_OK | The VMEbus device was successfully closed |

## 1.3.4 Master Mapping

### 1.3.4.1 VME_MstMap()

**Synopsis:**

int VME_MstMap(Word32 &log_adr, Word32 base_add, Word16 size, Data8 mode)

**Parameters:**

| | | |
|---|---|---|
| Word32 log_adr | out | virtual address mapped memory |
| Word32 base_add | in | Base address of VME card |
| Word16 size | in | Size of memory to be mapped |
| Data8 mode | in | The access mode to the mapped memory |

**Descriptions:**

The VME_MstMap() function mapped the VME cards memory to the virtual memory of proces according to the input parameters.

**Return Values:**

| | |
|---|---|
| VMEMAPE_OK | The memory was successfully mapped |
| Others | Specific to the implementation |

#### 1.3.4.2 VME_MstUnmap()

**Synopsis:**

int VME_MstUnmap(Word32 log_adr, Word16 size)

**Parameters:**

| | | |
|---|---|---|
| Word32 log_adr | in | virtual address mapped memory |
| Word16 size | in | Size of memory to be unmapped |

**Descriptions:**

The VME_MstUnmap() function breaks the connection between VME cards memory and proces virtual memory.

**Return Values:**

| | |
|---|---|
| VMEMAPE_OK | The memory was successfully unmapped |
| Others | Sepcific to the implementation |

### 1.3.5 Single cycles read, write from, to VME cards

#### 1.3.5.1 vme_write16()

**Synopsis:**

void vme_write16(Word32 address, Data16 data)

**Parameters:**

| | | |
|---|---|---|
| Word32 address | in | virtual address of memory |
| Data16 data | in | data to be written to the memory |

**Descriptions:**

The vme_write16() function writes the 16 bits of data to the specific address of memory.

**Return Values:**

none

#### 1.3.5.2 vme_write32()

**Synopsis:**

void vme_write32(Word32 address, Data32 data)

**Parameters:**

| | | |
|---|---|---|
| Word32 address | in | virtual address of memory |
| Data32 data | in | data to be written to the memory |

**Descriptions:**

The vme_write32() function writes the 32 bits of data to the specific address of memory.

**Return Values:**

None

#### 1.3.5.3 vme_read16()

**Synopsis:**

Data16 vme_read16(Word32 address)

**Parameters:**

| | | |
|---|---|---|
| Word32 address | in | virtual address of memory to be read |

**Descriptions:**

The vme_read16() function reads the 16 bits of data from the specific address of memory.

**Return Values:**

| | |
|---|---|
| data16 | Value was read from the memory |

#### 1.3.5.4 vme_read32()

**Synopsis:**

Data32 vme_read32(Word32 address)

**Parameters:**

| | | |
|---|---|---|
| Word32 address | in | virtual address of memory to be read |

**Descriptions:**

The vme_read32() function reads the 32 bits of data from the specific address of memory.

**Return Values:**

| | |
|---|---|
| data32 | Value was read from the memory |

## *1.4 The online functions.*

### 1.4.1 Overview

The following list is an overwiew of all type and functions definitions in the DAQ code:

1. General Functions
- set_thr_ped;
- control_vme_cards;
- v551B_set;
- v550_set;
- v767_set;
- get_event_767;
- get_event_550;
- write_output_binary_file;
- clear_v767_v550;

### 1.4.2 Type Definitions

1. Return Code Type
- The DAQ returns 0 when operations were done without errors. Othewise the returned value is different from 0.

### 1.4.3 General functions

#### 1.4.3.1 Set_thr_ped()

**Synopsis:**

int set_thr_ped(Word32 log_add_v550, Word32 log_add_v551B, Data16 h_m_p);

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v550 | in | base address of Mod. V550 CRAMS |
| Word32 log_add_v551B | in | base address of Mod. V551B SEQUENCER |
| Data16 h_m_p | in | number of events required to calculate the pedestals and thresholds value |

**Descriptions:**

The set_thr_ped() function takes the number of events given by User and calculate the pedestals and thresholds values. These values are written to a Pedestals and Thresholds memory of  Mod. V550 CRAMS

**Return Values:**

| | |
|---|---|
| int | Error code, specific to the implementation |

#### 1.4.3.2 Control_vme_cards()

**Synopsis:**

int control_vme_cards(Word32 log_add_v550, Word32 log_add_v551B, Word32 log_add_v767, unsigned int h_m_e)

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v551B | In | Base address of Mod. V551B SEQUENCER |
| Word32 log_add_v550 | In | base address of Mod. V550 CRAMS |
| Word32 log_add_v767 | In | base address of Mod. V767 TDC |
| Unsigned int h_m_e | In | number of events required |

**Descriptions:**

The control_vme_cards() function synchronize the data acqusition procedure.

**Return Values:**

| | |
|---|---|
| Int | Error code, specific to the implementation |

#### 1.4.3.3 v551B_set()

**Synopsis:**

int v551B_set(Word32 log_add_v551B);

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v551B | In | base address of Mod. V551B SEQUENCER |

**Descriptions:**

The v551B_set() function sets the Mod. V551B SEQUENCER according to the demands are given by User.

**Return Values:**

| | |
|---|---|
| Int | Error code, specific to the implementation |

#### 1.4.3.4 v550_set()

**Synopsis:**

int v550_set(Word32 log_add_v550);

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v550 | in | base address of Mod. V550 CRAMS |

**Descriptions:**

The v550_set() function sets the Mod. V550 CRAMS according to the demands are given by User.

**Return Values:**

| | |
|---|---|
| Int | Error code, specific to the implementation |

#### 1.4.3.5 v767_set()

**Synopsis:**

int v767_set(Word32 log_add_767);

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v767 | in | base address of Mod. V767 TDC |

**Descriptions:**

The v767_set() function sets the Mod. V767 TDC according to the demands are given by User.

**Return Values:**

| | |
|---|---|
| int | Error code, specific to the implementation |

#### 1.4.3.6 get_event_v767()

**Synopsis:**

int get_event_767(Word 32 log_add_v767, Data32 event_count, tdc_event &tdc)

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v767 | in | base address of Mod. V767 TDC |
| Data32 event_count | in | value of current number of events |
| tdc_event &tdc | out | storage structure for data |

**Descriptions:**

The get_event_767() function reads the output buffer of TDC and writes that data to the structure tdc_event.

**Return Values:**

| | |
|---|---|
| Int | Error code, specific to the implementation |

#### 1.4.3.7 get_event_v550()

**Synopsis:**

int get_event_550(log_add_v550, event_count, adc_event &adc)

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v550 | in | base address of Mod. V550 CRAMS |
| Data32 event_count | in | value of current number of events |
| adc_event &adc | out | storage structure for data |

**Descriptions:**

The get_event_550() function reads the output buffer of CRAMS and writes that data to the structure adc_event.

**Return Values:**

| | |
|---|---|
| Int | Error code, specific to the implementation |

### 1.4.3.8 write_output_binary_file()

**Synopsis:**

int write_output_binary_file(int out_bin_file, tdc_event &tdc, adc_event &adc, Data32 event_count)

**Parameters:**

| | | |
|---|---|---|
| int out_bin_file | in | description of output file to be written |
| adc_event &adc | in | structure of data from CRAMS |
| tdc_event &tdc | in | structure of data from TDC |
| Data32 event_count | in | value of current number of events |

**Descriptions:**

The write_output_binary_file() function writes the data from structures to a binary output file.

**Return Values:**

| | |
|---|---|
| int | Error code, specific to the implementation |

### 1.4.3.9 clear_v767_v550

**Synopsis:**

int clear_v767_v550(Word32 log_add_v550, Word32 log_add_v767, Word_32 log_add_v551B)

**Parameters:**

| | | |
|---|---|---|
| Word32 log_add_v550 | in | description of output file to be written |
| Word32 log_add_v767 | in | structure of data from CRAMS |
| Word32 log_add_v551B | in | structure of data from TDC |

**Descriptions:**

The clear_v767_v550 clears the output buffers of the TDC and the C-RAMS.

**Return Values:**

| | |
|---|---|
| int | Error code, specific to the implementation |

# Appendix III: A structure of binary output file

The offline program to calculate the TGC efficiency uses the information stored in the binary online output file. The online program writes the following details:

- TGC signal timing information - in order to evaluate their response time;
- HV information - in order to monitor the condition of high voltage (not implemented);
- ADC (C-RAMS) information - in order to calculate the position of the muons when crossing the tested TGCs.

All the entries of that file are four bytes long. Before each block of the data there is a pointer, which indicates the number of event, the type of the data and number of the 4 bytes words. There are two typess of pointers:

- The pedestal data block ;
- The TDC event data block .

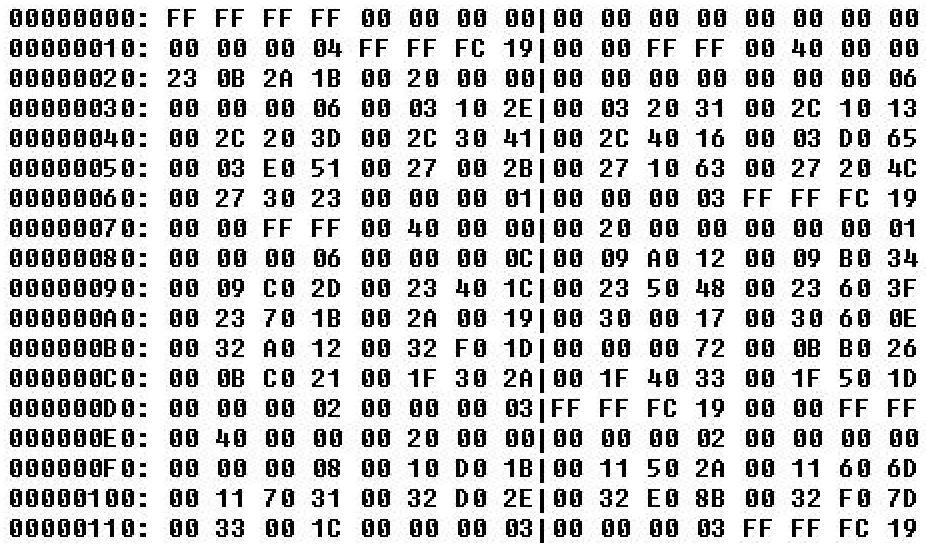
In the following the structure of the output file:
One word (usually four bytes) is represented by:
FF FF FF FF in the hexadecimal base.

The pedestals[1] information is written as:
FF FF FF FF nn nn nn nn mm mm mm mm xx xx xx xx .. .. .. ..
where:
FF FF FF FF –is the pointer that indicates the beginning of the pedestals data block.
nn nn nn nn – is the number of pedestal words of ADC in channel1[2];
mm mm mm mm – is the number of pedestal words of ADC in channel0;
xx xx xx xx –is the pedestals data. It has structure of ADC output buffer.

The following presents the structure of the TDC event information block:
zz zz zz zz vv vv vv vv FF FF FC 19 kk kk kk kk yy yy yy yy .. .. .. ..
where:
zz zz zz zz –is the number of current event, starting at zero;
vv vv vv vv –is the number of the TDC data words;
kk kk kk kk – is the HV status word. Each of the bits represent a status of one HV unit channel , 0 – means the channel is in normal mode, 1 – means the channel is in a trip mode;
yy yy yy yy – is the TDC event data. It has structure of TDC output buffer, including header and eob (end of buffer).
The following explanations the structure of the ADC (C-RAM) event:
zz zz zz zz hh hh hh hh pp pp pp pp oo oo oo oo .. .. .. ..
where:
zz zz zz zz – is the number of the current event;
hh hh hh hh – is the number of the ADC(C-RAMS) channel0 data words;
pp pp pp pp – is the number of the ADC(C-RAMS) channel1 data words;
oo oo oo oo –is the ADC(C-RAMS) event data. It has structure of the ADC(C-RAMS) output buffer.

**Table AIII.1** shows a typical output file. (This is a figure not a table.. where the table of explanation after that IS a table)

[1] The pedestal is the data of ADC channels that include noise and signals.
[2] The channel1 and channel0 are representing the serial information from the upper and the lower precision chambers.



```
00000000: FF FF FF FF 00 00 00 00|00 00 00 00 00 00 00 00
00000010: 00 00 00 04 FF FF FC 19|00 00 FF FF 00 40 00 00
00000020: 23 0B 2A 1B 00 20 00 00|00 00 00 00 00 00 00 06
00000030: 00 00 00 06 00 03 10 2E|00 03 20 31 00 2C 10 13
00000040: 00 2C 20 3D 00 2C 30 41|00 2C 40 16 00 03 D0 65
00000050: 00 03 E0 51 00 27 00 2B|00 27 10 63 00 27 20 4C
00000060: 00 27 30 23 00 00 00 01|00 00 00 03 FF FF FC 19
00000070: 00 00 FF FF 00 40 00 00|00 20 00 00 00 00 00 01
00000080: 00 00 00 06 00 00 00 0C|00 09 A0 12 00 09 B0 34
00000090: 00 09 C0 2D 00 23 40 1C|00 23 50 48 00 23 60 3F
000000A0: 00 23 70 1B 00 2A 00 19|00 30 00 17 00 30 60 0E
000000B0: 00 32 A0 12 00 32 F0 1D|00 00 00 72 00 0B B0 26
000000C0: 00 0B C0 21 00 1F 30 2A|00 1F 40 33 00 1F 50 1D
000000D0: 00 00 00 02 00 00 00 03|FF FF FC 19 00 00 FF FF
000000E0: 00 40 00 00 00 20 00 00|00 00 00 02 00 00 00 00
000000F0: 00 00 00 08 00 10 D0 1B|00 11 50 2A 00 11 60 6D
00000100: 00 11 70 31 00 32 D0 2E|00 32 E0 8B 00 32 F0 7D
00000110: 00 33 00 1C 00 00 00 03|00 00 00 03 FF FF FC 19
```

**Table AIII.1: The structure of the binary online output file.**

**Table AIII.1** has the following structure:

| Offset | Value | Description |
|---|---|---|
| 0x000 | 0xFF FF FF FF | Start of pedestal block |
| 0x004 | 0x00 00 00 00 | Number of words of the ADC channel0 |
| 0x008 | 0x00 00 00 00 | Number of words of the ADC channel1. |
| 0x00C | 0x00 00 00 00 | Number of the event. |
| 0x010 | 0x00 00 00 04 | Number of the TDC event data. |
| 0x014 | 0xFF FF FC 19 | Start of the TDC event block. |
| 0x018 | 0x00 00 FF FF | HV status, dummy value. |
| 0x01C | 0x00 40 00 00 | Header of the TDC event. |
| 0x020 | 0x23 0B 2A 1B | The TDC event data. |
| 0x024 | 0x00 20 00 00 | Eob of the TDC event |
| 0x028 | 0x00 00 00 00 | Number of the event. |
| 0x02C | 0x00 00 00 06 | Number of words of the ADC channel0 data. |
| 0x030 | 0x00 00 00 06 | Number of words of the ADC channel1 data. |
| 0x034 – 0x048 | 0xnn nn nn nn | The ADC channel1 data. |
| 0x04C – 0x064 | 0xnn nn nn nn | The ADC channel0 data. |
| 0x068 | 0x00 00 00 01 | Number of the event. |

A double line is marking one block of the event data.