Elsevier Editorial System(tm) for NIMA Proceedings
Manuscript Draft

Manuscript Number: NIMA_PROCEEDINGS-D-10-00193R1

Title: VME-based Data Acquisition System for the India-based Neutrino Observatory prototype detector.

Article Type: Special issue:RPC 2010

Section/Category: Special issue: RPC2010

Keywords: Special issue:RPC2010; TIFR; VME, DAQ, Qt, QtRoot

Corresponding Author: Dr. Deepak Samuel, Ph.D

Corresponding Author's Institution: Tata Institute of Fundamental Research

First Author: Deepak Samuel, Ph.D

Order of Authors: Deepak Samuel, Ph.D

Abstract: The India-based Neutrino Observatory (INO) collaboration has proposed to build a 50 kton Iron-Calorimeter (ICAL) to study neutrino oscillations. About 28,800 Resistive Plate Chambers will be used as active detector elements in this experiment. Preliminary studies are currently underway and as a part of it, a prototype detector was developed which now serves as a cosmic-ray telescope and as a test-bench to study the indigenously built RPCs. A VME-based data acquisition system was designed for this prototype system. Modern software tools were used in the designing of the DAQ software. The design and development of this DAQ system are discussed.

Detailed Response to Reviewers

Response to Reviewer #1

comment 1: None of your figures is referenced anywhere in the article.

This has been rectified in this revised submission.

comment 2: The subsection 2.1 does not describe (also does not give external reference) the analog and digital front end in enough detail (e.g. are the PA discriminators also controlled from the DAQ, etc.) and it should reference Fig 1.

Since this paper focusses on the DAQ part, I delibrately avoided dwelling much into the details of AFE and DFE. AFE and DFE related publications are in the process and therefore no suitable reference could be cited. Nevertheless, I agree that the section needed some clarity and therefore it has been suitably edited. I could not explain some things in details owing to the limit of 4 pages.

comment 3: In sec. 3.2: It would be nice if you could describe how the other threads are prioritized (apart from the IST), a relevant event/data rate is not mentioned at all.

Section 3.3 has been modified to address the issue of thread prioritization also. event rate is reported in the conclusion.

comment 4: Please add relevant items in your URL references where applicable.

References now have the author and the title of the page.

Thanks and Regards,

Deepak Samuel.

VME-based Data Acquisition System for the India-based Neutrino Observatory prototype detector.

M.Bhuyan [a] V.B.Chandratre [b] S.Dasgupta [a] V.M.Datar [c] S.D.Kalmani [a] S.M.Lahamge [a] N.K.Mondal [a] P.Nagaraj [a] S.Pal [a] S.K.Rao [a] A.Redij [a] D.Samuel [a,*], M.N.Saraf [a] B.Satyanarayana [a] R.R.Shinde [a] S.S.Upadhya [a]

[a] *Department of High Energy Physics, Tata Institute of Fundamental Research, Mumbai 400005, India.*
[b] *Electronics Division, Bhabha Atomic Research Centre, Mumbai 400085, India.*
[c] *Nuclear Physics Division, Bhabha Atomic Research Centre, Mumbai 400085, India.*

Abstract

The India-based Neutrino Observatory (INO) collaboration has proposed to build a 50 kton Iron-Calorimeter (ICAL) to study neutrino oscillations. About 28,800 Resistive Plate Chambers will be used as active detector elements in this experiment. Preliminary studies are currently underway and as a part of it, a prototype detector was developed which now serves as a cosmic-ray telescope and as a test-bench to study the indigenously built RPCs. A VME-based data acquisition system was designed for this prototype system. Modern software tools were used in the designing of the DAQ software. The design and development of this DAQ system are discussed.

Key words: VME, DAQ, Qt, QtRoot,

1. Introduction

A detailed description of the INO project can be found in other articles in this proceeding. A prototype of ICAL has been developed at the Tata Institute of Fundamental Research (TIFR). The detector consists of 12 layers of 1m x 1m RPC's with 32 strips on either planes. The gap between the layers is 15.8cm and the total height of the stack is 1.76m. During the initial phase of the prototype design, a CAMAC based DAQ was used. Due to the inherent limitations of the CAMAC hardware, it was later decided to develop a technically superior VME based DAQ system. The DAQ software was revamped to augment the new hardware. The software has a multi-threaded structure running behind an intuitive GUI. The DAQ supports concurrent plotting and analysis, thanks to QtRoot [1], a plug-in developed at the Brookhaven National Laboratory, which embeds ROOT's plotting canvas inside our Graphical User Interface (GUI) framework. The software was written in C++ with Qt libraries for the front-end GUI design. The software runs on a machine with 2.80 GHz Intel Xeon CPU with 1 GB RAM with an open-source 32 bit Linux operating system.

2. Hardware

2.1. *Signal Processing and Latching*

Figure 1 is the block diagram of the signal processing unit for the prototype.

Since the RPC's operate in the avalanche mode, the strip signals are amplified (80x) before being processed by the Analog Front End (AFE) boards. Each RPC has a dedicated AFE and a Digital Front End (DFE). The AFE has a common threshold adjustable upto 500 mV for discrimination of the signals (ECL output). The AFE also houses a trigger logic where the discriminator output of four channels (0,8,16,24; 1,9,17,25; 2,10,18,26; etc.,) are shaped to 50ns width and logically "OR"ed to get 8 pre-trigger signals called T0.

The discriminated signals and the T0 signals then pass through the DFE where the discriminated signals are first translated to TTL and stretched to a width of 700ns. The DFE board has four sections in it:

* *Email address:* samuel@tifr.res.in (D.Samuel).

(i) The Decoder which accepts the hand-shake signals from the Control and Data Router (CDR) and controls the mode of operation (i.e., Event / Noise Monitoring).
(ii) The Event Section which handles the latching of the strip-hit information. These latched data are flushed out serially to the CDR on receipt of an appropriate signal from the Decoder unit.
(iii) The Noise Rate Monitoring Section which latches the noise rate of the active strip or calibration signal on receipt of a clock signal from the Decoder unit and switches over to the next strip. At the end of the cycle (32^{nd} strip), the cycle is reset and the monitoring continues from the first strip. The clocking is handled by the DAQ system.
(iv) The Trigger Logic where various fold signals (1-Fold, 2-Fold, 3-Fold, 4-Fold) [1] are generated from the T0 signals.

The CDR routes the control signals and the data signals (Event Data/Noise Rate Data) while the Trigger and Timing router (TTR) routes timing signals to external measuring and recording systems. The DAQ can access and control the DFE sections only through these modules. The DAQ does not have any access to the AFE board.

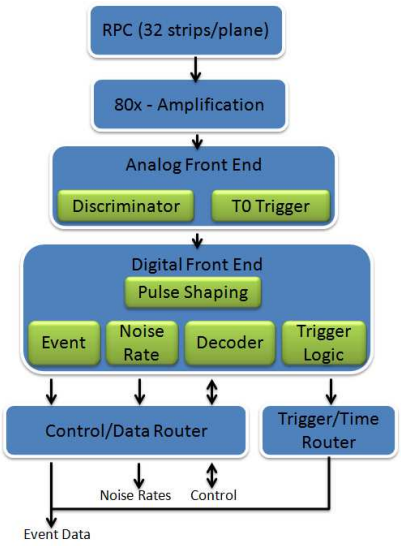

Fig. 1. The DAQ signal processing units.

2.2. *VME Modules*

The DAQ is mainly designed to track cosmic-muons that pass through the 12 layer stack. Using this data, various RPC parameters like efficiency, timing resolution, cluster size etc., are estimated. Apart from this, other analysis like zenith angle distribution and velocity distribution [3] of cosmic-muons are also made.

[1] 1-Fold $= T0_0 + T0_1 + T0_2 + T0_3 + T0_4 + T0_5 + T0_6 + T0_7$
2-Fold $= T0_0 \cdot T0_1 + T0_1 \cdot T0_2 + T0_2 \cdot T0_3 + T0_3 \cdot T0_4 + T0_4 \cdot T0_5 + T0_5 \cdot T0_6 + T0_6 \cdot T0_7$ and so on.

The noise rate of the individual strips is also monitored at regular intervals to ensure the stability of the detectors during data acquisition.

Thus, there are two types of data are received by the DAQ from the signal processing hardware discussed above. They are:

– **Event Data**- This contains:
  · The strip hit information, i.e., the x,y hit patterns in the layers through which the particle has passed through. This is read out by a I/O Module (Customized CAEN V1495). This I/O Module also manages the latching of data from the appropriate DFE boards using the control lines in the CDR.
  · The timing information from each of these layers, from the TTR unit. This is read out by a multi-hit TDC (CAEN V1190).
– **Noise Rates** - This contains the noise rate of the individual strips. Four calibration signals and fold signals are also received here. This is read out by a 24 bit scaler (CAEN V1190) at regular intervals.

2.3. *Trigger and Interrupts*

For Cosmic-ray related studies, an 8-layer coincidence i.e., the 1-Fold signals from 8 selected layers are logically "AND"ed to generate a trigger. This coincidence trigger also generates an interrupt via the TDC module, to read out the Event Data.

The I/O module manages the periodic read out of noise rates and also generates an interrupt via the Scaler for the same.

3. Software

During the primitive stages of an experiment, debugging and troubleshooting of the detector is a frequent requirement. Plotting and display of relevant parameters like hit information, noise rates, time spectrum etc., concurrently during the acquisition of data is one main requirement in such a scenario. A GUI is helpful in such cases in intuitively guiding an user to the information that is sought, hiding all the complexities in the background. The following software and tools were used in building our DAQ software:

– Qt: It is a cross-platform application and UI framework from Nokia Corporation (LPGL License). Qt offers useful C++ class libraries, which were used extensively in developing the code [2].
  · Qt Designer: This tool is a part of Qt SDK, which was used to design our GUI.
  · Qt Creator: Also a part of Qt SDK, this is an Integrated Development Environment (IDE), used for coding and project management.
– ROOT: Used for graphing and data analysis in the DAQ framework. Saving data as ROOT objects helps in easier analysis and plotting and therefore ROOT's TTree class was used for this purpose.

– QtRoot: This plug-in helps embed ROOT's plotting canvases inside our GUI framework. This is our key to implementing an integrated plotting and analysis environment in our DAQ application.

3.1. *The GUI Front-end*

The GUI was designed with the aim of providing the requested information as fast as possible, hiding all complexities from the user. The GUI incorporates a side-bar populated with often used functions and text and number entry fields. The ROOT plotting canvas is set in the center. A menu-bar allows users to switch between various canvases available. The settings of VME modules can also be changed in a separate window. The access to this window is also through the menu-bar. For custom plotting (For example, Time spectra, Trigger rates etc.,),the entry fields are implemented inside a docking widget. These entry fields take the inputs required for the "Draw" function in the TTree class namely the branch name and the condition, thus integrating all plotting and analysis features that ROOT offers, into our framework. Figure 2 is a screen-shot of the GUI.

3.2. *Back-end: Threads*

As discussed in section 2.3, two types of interrupts are generated. The Event Data read-out interrupt generated by the TDC and the Noise Rate read-out interrupt generated by the Scaler. The Event interrupt is random in nature as it is a consequence of a particle passing through the detector satisfying the coincidence condition.

The Noise Rate interrupt is periodic as it samples each strip at a pre-defined frequency.

The DAQ back-end process was designed to have a multi-threaded structure to support concurrent execution of various jobs in a prioritized manner thus enabling the application to offer the advantage of a full-fledged live analysis and plotting framework to the user. Although POSIX (Portable Operating System Interface) threads are widely used in C++, we have used the QThread class offered by Qt to implement the threads. The QThread class supports the "signal and slot" mechanism making the communication between a thread and the GUI much simpler [4]. There are four main threads that run in the background. They are:

– Interrupt Service Thread (IST)
– Event Thread
– Noise Rate Monitoring Thread and the
– Event Plotting Thread

The IST carries out the following jobs sequentially (cf. Fig. 3):

(i) Waits on an interrupt signal.
(ii) On receiving an interrupt, all interrupts are disabled via software.
(iii) Identifies the source of interrupt (TDC/Scaler) by reading out the Interrupt Vector Register.
(iv) Acknowledges the interrupt.

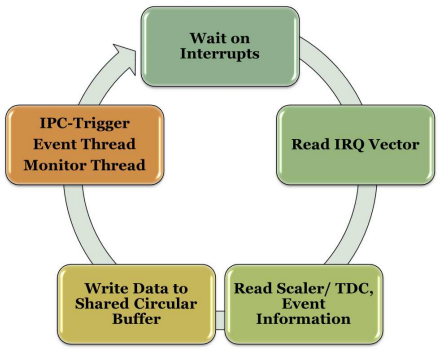

Fig. 3. The sequence of the Interrupt Service Thread.

(v) Reads data from the relevant modules and appends them in a shared memory and triggers either the Event Thread or the Noise Rate Monitoring Thread.
(vi) Enables interrupts.

The shared memory is a circular buffer which is allocated in the heap during the initialization phase of the program. Two shared memories are used, one for the Event data and another for the Noise Rate data.

Both the Event Thread and the Noise Rate Monitor Thread are semantically similar to each other in the sense that they do same job but on different data. As soon as the IST sends a trigger (sequence 5 of IST), the latest data from the shared memory is appended to a file and the file is finally saved at an appropriate moment.

The graphical display of strip hits of various layers is helpful in cases where the detector and the electronics are being tested. The display of the particle tracks through the detector in real-time also gives a feeling of the status of the detector. This is accomplished in the Event Plotting Thread. Plotting of each and every event is not desirable and therefore this thread is set to auto-trigger every n seconds, where n is set by the user in the GUI. This thread runs exclusively to plot the strip hit information in the layers. The thread reads the latest event, decodes them and plot on the canvas. Simultaneously, the same data is sent to a server. Users can access the plot of the strip hit information from our web page [5].

3.3. *Thread Prioritization, Synchronization and Plotting*

As mentioned earlier, the QThread class has been used to implement all the threads. Each thread is scheduled for execution based on its priority. Qt offers several types of priorities which can be assigned to the threads by the user. The priorities assigned to the threads discussed above and their description is illustrated in Figure 4. The IST is assigned the maximum priority as it executes time-critical operations. The Event Thread and the Noise Rate Monitoring Thread are assigned normal priority as transfer of data from the shared memory is not a time-critical operation, provided a proper thread synchronization mechanism is used. The plotting of events is a process that can be safely stopped or executed at a later point of time without hin-

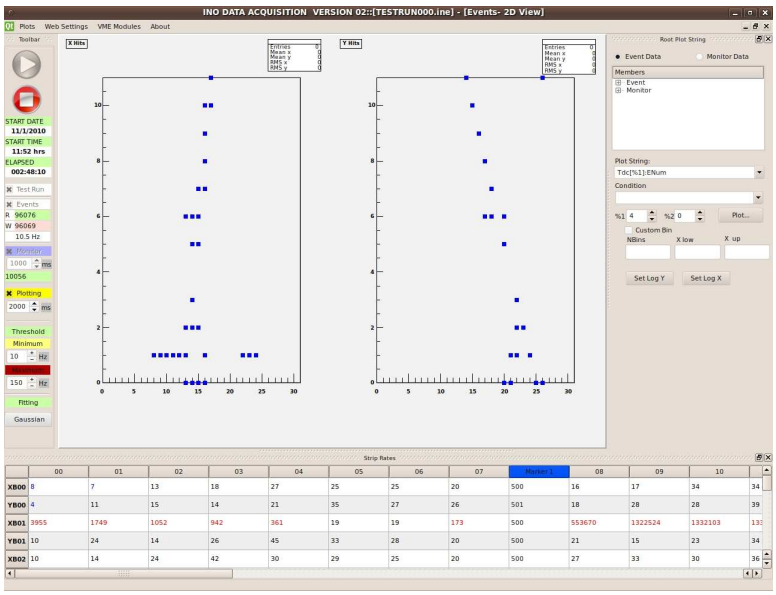

Fig. 2. Screen shot of the DAQ GUI, showing the embedded ROOT plotting canvas.

dering the data acquisition. Therefore, the Event Plotting Thread is assigned a low priority.

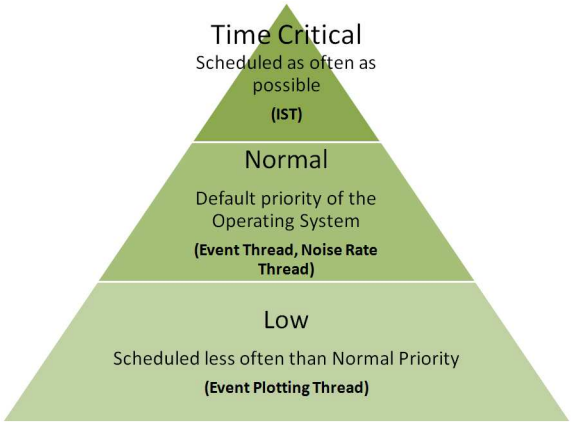

Fig. 4. Prioritization of the threads and their description.

The threads run at their own time scheduled by the Operating System. However, when a file is being accessed by one thread, for example, to plot the timing distribution or the noise rate distribution, another access to the same file should be avoided. To overcome such situations, the threads are synchronized and files are protected within Inter Process Communication (IPC) objects called Mutex. Here again, we have used the QMutex class by Qt to implement the mutexes. The threads check the status of the Mutex objects before writing to the file. In case the file is being accessed by the user, the thread waits until the job is completed and then proceeds with the writing.

4. Conclusion

The newly developed VME-based data acquisition system for the INO prototype has shown an overall stable and satisfactory performance. The transition from the CAMAC bus to the VME bus was clearly felt with the improved trigger rates. With this set-up, the DAQ can handle a trigger rate of about 300Hz. Though this can be improved even further, with a cosmic-muon trigger rate of only about 10Hz for top and bottom layer coincidence, this was deemed to be unnecessary for the prototype. Though varieties of tool are available in the market, we have been successful in developing our DAQ using the best available Open-Source tools. Qt has not only helped in designing professional GUI design for our DAQ but also in developing a multi-threaded program. The role of QtRoot in getting an integrated analysis and plotting environment in our applications is definitely worth a mention.

Acknowledgments

The INO project is funded by the Department of Atomic Energy (DAE) and the Department of Science and Technology (DST), Government of India. Crucial contributions from many INO collaborators to this paper are gratefully acknowledged.

References

[1] V.Fine, The Qt ROOT Version for Unix, Windows and MacOS, http://root.bnl.gov/, June 2010.
[2] Nokia Corporation, Qt - Cross-platform application and UI framework, http://qt.nokia.com/, June 2010.
[3] S. Pal et al, Velocity Measurement of Cosmic Muons using the India-based Neutrino Observatory Prototype Detector, In this proceeding, 2010.
[4] Nokia Corporation, Threads and QObjects, http://doc.qt.nokia.com/4.6/threads-qobject.html, June 2010.
[5] TIFR-INO Collaboration, Real-Time Cosmic Muon Events, http://www.hecr.tifr.res.in/~ino/c217_daq/daqweb/html/basic.html, June 2010.