NUCL SCI TECH (2017) 28:139
DOI 10.1007/s41365-017-0296-6

Design of a NIM-based DAQ system

Wen-Xiong Zhou[1,2] · Yan-Yu Wang[3] · Liang-Ming Pan[1]

Received: 21 September 2016 / Revised: 16 November 2016 / Accepted: 19 January 2017 / Published online: 6 September 2017

Abstract In order to satisfy the requirements of beam measurement in the heavy ion medical machine and other small nuclear physics experiments, we designed and built a nuclear instrumentation module-based data acquisition system. This is composed of a set of functional modules and a purpose-built bus. One of the modules operates as a master, collecting data from the other slave modules. It then sends the data to the back-end computer via Ethernet. In addition to the hardware, dedicated software has been designed and implemented. In this paper, we provide a detailed description of the architecture of the system, the data frame, and the software. The bus is the central part of the system. It can transmit data from the slave modules to the master at 33 MB/s. The frame used to transmit the data also ensures its integrity and monitors the hardware architecture. The client software is designed to process data in real time and store data on a hard disk for later analysis.

Keywords DAQ system · Purpose-built bus · ARM · FPGA · NIM

This work was supported by the Fundamental Research Funds for the Central Universities of China (2015CDJXY).

✉ Wen-Xiong Zhou
zhouwenxiong@cqu.edu.cn

[1] College of Power Engineering, Chongqing University, Chongqing 400044, China

[2] Postdoctoral Station, Chongqing University, Chongqing 400044, China

[3] Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou 730000, China

1 Introduction

Data acquisition (DAQ) is a key activity in science and technology, particularly in nuclear physics experiments. In our case, beam measurement in the heavy ion medical machine (HIMM) built by the Institute of Modern Physics in China [1–3]. The signal in this facility is first processed using the nuclear instrumentation module (NIM) front-end electronics. Then the DAQ system sends the data to the back-end computer. If the measurement system is implemented using the NIM system and the DAQ system is implemented using VME eXtensions for Instrumentation (VXI) or PCI eXtensions for Implementation (PXI), for example [4–9], the overall system would be very complex. Furthermore, commercial portable DAQ systems such as the openDAQ [10], the ORTEC model 926, and the ORTEC EASY-MCA-2K/8K do not have the capacity for measuring enough parameters or provide the function of a synchronous trigger for the whole system. Furthermore, in some parts of the HIMM, space restrictions impose the need for a compact measurement and detection system.

In order to overcome these problems, we developed a NIM-based DAQ (NDAQ) system with a low-cost, compact structure, and flexible architecture. In this system, the measurement, control, and DAQ modules in the NIM crate are connected through a purpose-built bus. The maximum data transport speed of the bus is about 33 MB/s, which is faster than the speed of the General Purpose Interface Bus (GPIB) [11–13]. Due to the need for control of the bus, the achieved speed of bus is less than 33 MB/s. The maximum speed we have reached in a real experiment is about 4.8 MB/s. However, this is adequate for the data transport speed in beam measurement and accelerator control

applications. It is also faster than the speed of the General Purpose Interface Bus (GPIB).

The ROOT data analysis framework is highly flexible and stable. It is used in many projects, for example, the pixel detector DAQ system for high energy physics (HEP), the High Energy Spectroscopic System (HESS) DAQ system and so on [14–16]. Hence, the ROOT framework is well suited for the implementation of the client software for the NDAQ. One version was created for the Linux operating system, and another convenient and flexible version for Microsoft Windows has also been developed based on Visual C++.

2 NDAQ hardware architecture

2.1 System architecture

Figure 1 shows the hardware architecture of the NDAQ system [17–19]. All the modules within the system are linked through the purpose-built bus. In the system, one of the modules operates in master mode while the other modules operate in slave mode. The master module reads the data from the slave modules through the bus and sends it to the NDAQ client software via Ethernet. The communications are implemented using an Advanced Reduced Instruction Set Computer (RISC) Machine (ARM) rather than a field programmable gate array (FPGA) [18]. The module can also operate independently as a minimal DAQ system. In this situation, the module can read the data from the front-end electronics (FEE) and send it to the NDAQ client software directly.

The system is composed of various functional modules including analog-to-digital converter (ADC), time-to-digital converter (TDC), scaler (SCA) module, and some measurement and control modules. After completing the design of the ADC modules, the TDC and SCA modules were designed and tested. For these two modules, it was only necessary to design the time measurement and scaler functions because the same architecture as the bus is used in these two systems and the stability of the bus has already been tested. Thus ensuring correct communication in the TDC and SCA modules. There are two types of ADC module. One is implemented using the ARM (S3C6410) and FPGA (EP3C25Q240) with four input channels. It can operate in both master and slave modes with a switch to change between operational modes. The other is designed using a FPGA with eight input channels. It can only operate in slave mode.

2.2 Purpose-built bus

The bus is used for data transfer between the master module and the slave modules. It uses 40 signals as shown in Table 1. The frequency of the synchronous clock is 16.5 MHz. As the width of the data bus is 2 bytes, the maximum data transport speed of the bus is about 33 MB/s calculated through the formula as follows:

$$S = F \times W.$$

where F is the frequency of the clock, W is the width of the data bus, and S is the transfer speed of the bus. This speed is used because it was originally designed to communicate with the PCI bus in the former system, which had a clock frequency of 16.5 MHz. The bus enables the number of measurement parameters to be varied in accordance with the requirements of a particular experiment, and the synchronous trigger signal of the system is propagated from the master module to the slave modules through the bus.

After power up, the master module will detect any slave modules connected to it by means of the ADDR, AS, and nASK bus signals. There are 255 slave module addresses. After the AS has been set, the master module will wait for a nASK signal. If the master module does not receive this signal within 10 μs, the slave module at that address will not be recorded. The longest duration of the detection section is about 2.55 ms.

The slave modules can request the master module to read their data by sending a nIRQ signal. After receiving a nIRQ signal, the master module will read data from all the

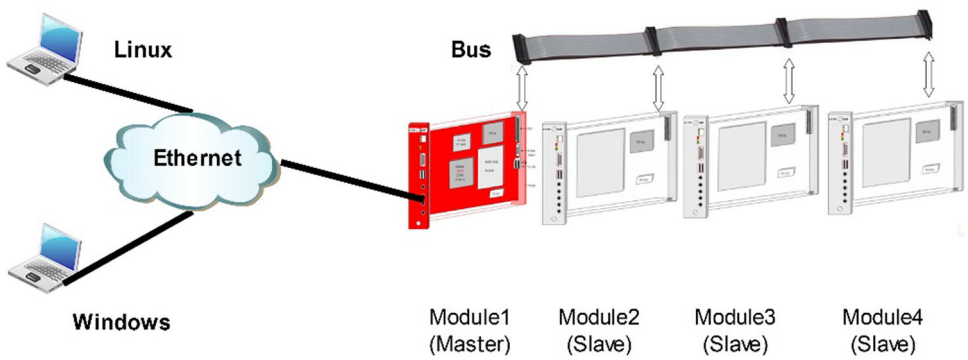

Fig. 1 (Color online) NDAQ hardware architecture

Table 1 Definition of the bus signals

Name	Number	Function
DATA	16	Transmit data and register address
ADDR	8	Transmit module address
CLK	2	Synchronization clock signal driven by the master module
nRST	1	Reset signal driven by the master module
TRIGGER	1	Synchronization trigger signal driven by the master module
nIRQ	1	Interrupt from the slave modules
ICK	1	Interrupt response from the master module
AS	1	Module address bus select
nASK	1	Module address bus select response
DS	1	Data bus select
nDSK	1	Data bus response
RW	1	Read or write
AOD	1	Address or data

slave modules in sequence. Additionally, the master module can write data to the slave modules directly.

3 NDAQ software architecture

The NDAQ software consists of firmware, server software (SS), and client software (CS). The firmware is used to configure the FPGA. The server software, including the driver software (DS) and the application software, is implemented on the embedded Linux operating system on the ARM. The client software is used to display spectral data and store data to disk. The main functions of the NDAQ software are as follows.

3.1 Requirement of the data transport

Since the volume of data typically encountered in small nuclear physics experiments and beam measurements is not large, the Ethernet has a maximum speed of 100 Mb/s. Thus, the transport speed between the ARM and the FPGA must be higher than this speed (about 12.5 MB/s), and the speed of the bus must also be greater than this speed because if the two speeds are less than the Ethernet speed, it will lead to a data transport bottleneck in the NDAQ system.

The communication methods between the FPGA and the ARM can be divided into three types: serial communication, user-defined parallel communication, and standard parallel communication. Standard parallel communication is the fastest. Therefore, the FPGA is designed as a read-only memory (ROM) device to utilize the memory bus controller of the ARM [18], so that the transport speed can satisfy the requirements of the system.

3.2 Dataflow

The dataflow, buffers, frame, and other architectural features are also important for the DAQ system. Figure 2 shows the flow of data in the NDAQ system. The NDAQ hardware consists of a master module and several slave modules. If a module operates as a master, it can transmit data to the NDAQ client software. If the module operates as a slave, the ARM is not operational. Only the FPGA is operational to obtain the data from the front-end electronics (FEE) and send the data to the master module through the bus.

The transmission of data can be divided into three procedures:

1. The FPGA in the module gathers the data of the entire system. This procedure is undertaken by FPGA only. Two buffers are implemented in the FPGA. Buffer-A is used to gather data from the FEE, while Buffer-B is used to store the data from Buffer-A in all modules. Buffer-B is only present in the master module. When the data stored in Buffer-A of the slave modules exceed the threshold, the FPGA sends an interrupt signal to the master module through the bus. Then, the master module reads the data from the corresponding slave module via the bus.
2. The FPGA in the master module transports the data to the ARM server. If the volume of data in Buffer-B of the master module exceeds the threshold, the FPGA sends an interrupt signal to the ARM. Then the dedicated driver software reads the data from the FPGA in response to the interrupt. When the volume of data in the buffer of the driver (Buffer-C1 or Buffer-C2) exceeds the threshold, the driver sends an interrupt to the server software.

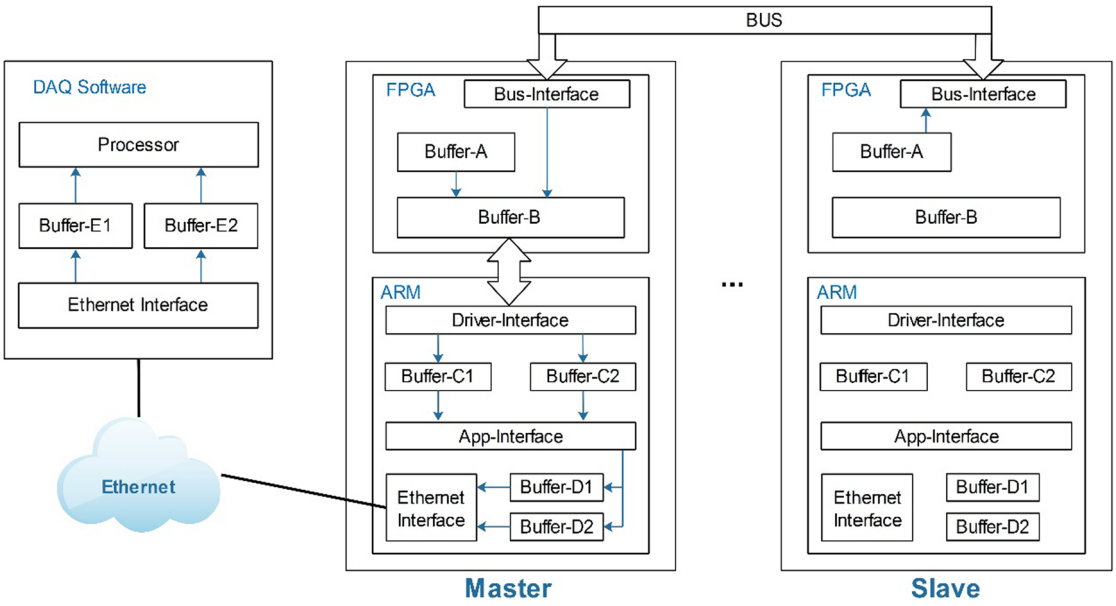

Fig. 2 (Color online) NDAQ dataflow

Table 2 Capacity and type of the buffers

	Buffer-A	Buffer-B	Buffer-C	Buffer-D	Buffer-E
Capacity (KB)	0.5	16	40	120	480
Buffer type	FIFO	FIFO	DB	DB	DB

3. The ARM in the master module transports data to the NDAQ client software in one of the two ways. One method is that the server software sends the data to the client actively when the volume of data in the server's buffer software exceeds the threshold. The other method is that the NDAQ client software requests the data from the server software. In the second method, the server software will force the driver software to read data from the FPGA, and the FPGA in the master module will also force the slave module to send the data to the master module.

3.3 Data buffers

In a DAQ system, the data need to be acquired, packaged, and transported. This can result in dead time between events [20]. In order to reduce the dead time, five levels of buffers (Buffer-A to Buffer-E) are used as shown in Fig. 2. Double-buffers, i.e., Buffer-C1 and Buffer-C2, are collectively referred to as "Buffer-C", so are Buffer-D1 and Buffer-D2, and Buffer-E1 and Buffer-E2. The capacity and type of the buffers are shown in Table 2. Buffer-A and Buffer-B are first-in-first-out (FIFO) buffers implemented in the FPGA. Buffer-C, Buffer-D, and Buffer-E are double-buffers (DB) implemented using software on the ARM. All

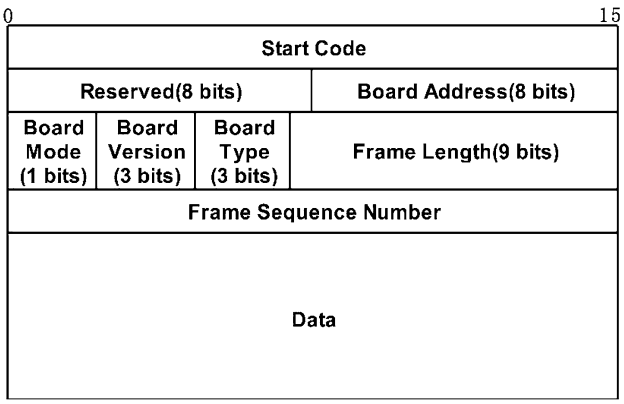

Fig. 3 Data frame used in the DAQ system

the buffers have their own thresholds. The initial threshold values are half of the corresponding buffer capacity.

An interrupt signal can be sent to the reader if the data volume exceeds the threshold. When that happens, the data are read out from the buffer and new data are written to the FIFO or DB at the same time. This mechanism effectively solves the problem of dead time but there are two new problems.

1. If the event rate is very low, it will take a long time to exceed the threshold of the five-level buffers. For example, if the event rate is 0.1 kHz, it would take

Table 3 Relationship between frame lengths and the buffer thresholds

Event rate (KHz)	Buffer-A (KB)	Buffer-B (KB)	Buffer-C (KB)	Buffer-D (KB)	Frame length (B)
≤0.1	0.064	0.256	0.256	0.256	64
0.1–0.5	0.064	0.256	1	1	64
0.5–1	0.128	0.512	1	1	128
1–2	0.128	0.512	2	2	128
2–5	0.256	1	4	4	256
5–8	0.256	2	4	8	256
8–10	0.256	4	8	16	256
10–20	0.256	4	8	32	256
20–40	0.256	8	32	64	256
≥40	0.256	8	80	120	256

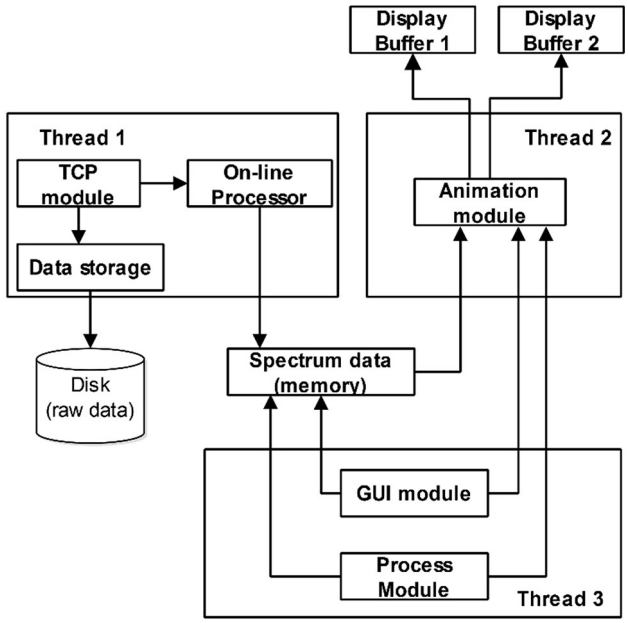

Fig. 4 Architecture of the client software (CS)

2,400 s to fill half of Buffer-E. To address this, an event counter is incorporated in the FPGA. The trigger threshold can be changed according to the event rate measured by the counter. Then, the dead time between two read interrupts can be shorted.

2. The system could lose the data stored in the buffers when the acquisition is stopped. In an experiment that has a low event rate, the lost data can be so significant that the experimental result may be affected.

Thus, a mechanism is implemented to force the controller to read out all the data in the buffers when it is needed. This mechanism not only solves the data loss problem, but can also be used to avoid long waiting times between interrupts.

3.4 Data frames

In order to ensure correct data transmission, a data frame is designed as shown in Fig. 3. The frame can be used to identify the architecture of the hardware. The data are packed before they are transported from Buffer-A to Buffer-B as shown in Fig. 2. The function of each field is as follows:

1. The start code marks the start of the frame. The software can distinguish the frames from a TCP package with the help of the start code. Since 0 is not used in the data, it can be used as the start code.
2. The board address, board mode, board version, and board type are used to transmit information about the modules installed. The board version can identify the data analysis method if the module is upgraded in the future. The board type identifies the data analysis method for different functional modules (TDC, ADC, and SCA).
3. The frame length is the length of the entire frame, which can be varied. That is useful when the event rate changes. The frame length can also be varied when different buffer thresholds are assigned. The relationship between the frame lengths and the buffer thresholds is shown in Table 3.
4. The frame sequence number is used to verify that data have been transported correctly. Each frame has a different number from 1 to 65,536. Thus, the client software can tell whether any data have been lost.

3.5 DAQ client software

The client software (CS) is implemented using Visual C++ on Microsoft Windows and using the ROOT framework on Linux. The CS can automatically identify

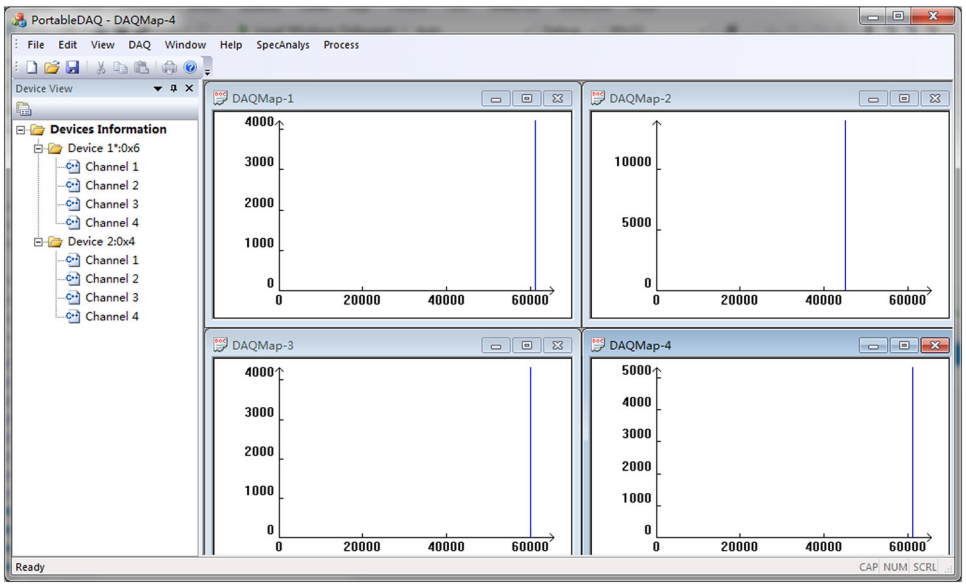

Fig. 5 (Color online) Interface of NDAQ client software in Windows operation system

the hardware structure of the NDAQ system through the data frame used in this system. The information obtained from the data frame can be used to decode and analyze the experimental data. Three threads are used to ensure the stable and efficient operation of the CS as shown in Fig. 4. The functions of the three threads are as follows:

1. The first thread is used to receive data from the server software, store data on the disk, and unpack the data in real time. The use of an independent thread helps to avoid the loss of data packets when the data transport speed is very high.
2. The second thread is used to process the spectrum in real time with a range of processing methods, for example, the background elimination method, the smoothing method, and the peak searching method. After processing, the spectrum is displayed in real time. The processing methods are derived from a paper by Miroslav Morhac and Vladislav Matousek [21].
3. The third thread is designed for human–computer interaction. The data transmission, decoding, unpacking, processing, and online display consume a large amount of CPU time. This affects the human–computer interaction. Thus, an independent thread is implemented to solve this problem.

Figure 5 shows the NDAQ client software implemented using Visual C++ on the Windows operating system. The NDAQ client software can display up to eight spectra simultaneously. In Fig. 5, four spectra are shown. The device hardware information is also displayed to the left of the spectra. A version of the client software based on ROOT is also available for the Linux operating system. Both versions have the same functionality.

4 Conclusion

This paper gives a detailed description of a portable DAQ system for small particle and nuclear physics experiments. The main problems, encountered during the design, and their solutions are introduced. The maximum data transmission speed of the purpose-built bus is about 33 MB/s, which is sufficient for small particle and nuclear physics experiments. The NDAQ client software can operate on both Microsoft Windows and Linux operating systems with similar interfaces. All the operations of the software are visual making it very easy for operators to use. A ROOT-based NDAQ client software is provided as an example. Thus, it is also convenient to customize for developing special purpose DAQ software.

References

1. J.C. Yang, J. Shi, W.P. Chai et al., Design of a compact structure cancer therapy synchrotron. Nucl. Instrum. Methods A **756**, 19–22 (2014). doi:10.1016/j.nima.2014.04.050
2. M. Li, Y.J. Yuan, R.S. Mao et al., The control system of the multi-strip ionization chamber for the HIMM. Nucl. Instrum. Methods A **776**, 21–26 (2015). doi:10.1016/j.nima.2014.12.021
3. W. Chai, J. Yang, J. Xia et al., Stripping accumulation and optimization of HIMM synchrotron. Nucl. Instrum. Methods A **763**, 272–277 (2014). doi:10.1016/j.nima.2014.05.117

4. M. Bhuyan, V.B. Chandratre, S. Dasgupta et al., VME-based data acquisition system for the India-based Neutrino Observatory prototype detector. Nucl. Instrum. Methods A **661**, S73–S76 (2012). doi:10.1016/j.nima.2010.08.075
5. Y. Chen, F. Wang, S. Li et al., Application of HDF5 in long-pulse quasi-steady state data acquisition at high sampling rate. Fusion Eng. Des. **89**, 721–725 (2014). doi:10.1016/j.fusengdes.2013.12.048
6. J. Kong, H. Su, Z.-Q. Chen et al., Development of multi-channel gated integrator and PXI-DAQ system for nuclear detector arrays. Nucl. Instrum. Methods A **622**, 215–218 (2010). doi:10.1016/j.nima.2010.07.030
7. M. Tecchio, J. Ameel, M. Bogdan et al., The data acquisition system for the KOTO detector. Phys. Proc. **37**, 1940–1947 (2012). doi:10.1016/j.phpro.2012.02.523
8. S.-H. Seo, J.S. Hong, M. Kwon, CAMAC, VXI, and PXI hybrid data acquisition system with MDSplus. Fusion Eng. Des. **71**, 141–144 (2004). doi:10.1016/j.fusengdes.2004.04.025
9. C. Li, J. Wang, K. Xuan et al., Event-driven timing system based on MRF cPCI hardware for HLS-II. Nucl. Sci. Tech. **26**, 060401 (2015). doi:10.13538/j.1001-8042/nst.26.060401
10. F.J. Ferrero Martín, M. Valledor Llopis, J.C. Campo Rodríguez et al., Low-cost open-source multifunction data acquisition system for accurate measurements. Measurement **55**, 265–271 (2014). doi:10.1016/j.measurement.2014.05.010
11. C. He, L. Ma, Y. Wu, et al. in *Application of VI Technology in DSO Measurement System with GPIB Interface, Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, Shenzhen, China, 12–14 June 2015
12. G. Liu, Q. Kong, Design of virtual oscilloscope based on GPIB interface and SCPI, in *The 11th IEEE International Conference on Electronic Measurement and Instruments*, HarBin, China, 16–19 Aug 2013
13. A. Wang, Y. Liu, Interface design of GPIB based on ARM RISC microprocessor, electronic measurement and instruments (ICEMI), in *2013 IEEE 11th International Conference*, Nanjing, China, 27–29 June 2013
14. M. Battaglia, D. Bisello, D. Contarato et al., A DAQ system for pixel detectors R&D. Nucl. Instrum. Methods A **611**, 105–110 (2009). doi:10.1016/j.nima.2009.09.008
15. A. Balzer, M. Füßling, M. Gajdus et al., The H.E.S.S. central data acquisition system. Astropart. Phys. **54**, 67–80 (2014). doi:10.1016/j.astropartphys.2013.11.007
16. G.A. Cox, E. Armengaud, C. Augier et al., A multi-tiered data structure and process management system based on ROOT and CouchDB. Nucl. Instrum. Methods A **684**, 63–72 (2012). doi:10.1016/j.nima.2012.04.049
17. G. Nan, Y. Wang, J. Zhang, Design of multi-channel pulse amplitude acquisition card based on NIM system. Nucl. Electron. Detect. Technol. **31**, 1250–1254 (2011)
18. W. Zhou, Y. Wang, G. Nan et al., Design of data transmission for a portable DAQ system. Nucl. Sci. Tech. **25**, 010404 (2014). doi:10.13538/j.1001-8042/nst.25.010404
19. J. Zhang, Y. Wang, G. Nan et al., Trigger signal pre-processing in nuclear physics experiment data acquisition systems. High Power Laser Part. Beams **24**, 2727–2730 (2012)
20. G.N. Perdue, L. Bagby, B. Baldin et al., The MINERnA data acquisition system and infrastructure. Nucl. Instrum. Methods A **694**, 179–192 (2012). doi:10.1016/j.nima.2012.08.024
21. M. Morháč, V. Matoušek, Library of sophisticated functions for analysis of nuclear spectra. Comput. Phys. Commun. **180**, 1913–1940 (2009). doi:10.1016/j.cpc.2009.04.025