The CLAS12 Data Acquisition System

S. Boyarinov[a], B. Raydo[a], C. Cuevas[a], C. Dickover[a], H. Dong[a], G. Hayes[a], D. Abbott[a], V. Gyurjyan[a], C. Timmer[a], E. Jastrzembski[a], W. Gu[a], B. Moffit[a], I. Mandjavidze[b], N. Baltzell[a], D. Heddle[a], C. Smith[a], R. De Vita[c], A. Celentano[c], N. Baltzell[a], B. McKinnon[d], K. Livingston[d], W. Moore[a]

[a]*Thomas Jefferson National Accelerator Facility, Newport News, VA, USA*
[b]*Irfu, CEA, Université Paris-Saclay, 91191, Gif-sur-Yvette, France*
[c]*INFN, Sezione di Genova, Via Dodecaneso 33, I-16146 Genova, Italy*
[d]*University of Glasgow, Glasgow G12 8QQ, United Kingdom*

Abstract

The CLAS12 Data Acquisition System was designed and built as part of the CLAS12 detector project. This article contains a full descripton of the system, including requirements, design, hardware, and software descriptions, as well as the achieved performance. The associated systems such as the computing, network, and slow controls systems are also described.

1. Overview

The CLAS12 Data Acquisition system (DAQ) collects data from CLAS12 detector [1]. It is a network-based system obtaining data from more than 100 front-end components comprising the trigger system signals, assembling the component data into events, monitoring data quality, and recording data to tape. Typical event rates are 15-30 kHz and 500-1000 MByte/sec with an acquisition livetime above 90%. The CLAS12 DAQ software is written in C/C++ as an expandable multi-threaded system, allowing relatively easy upgrades in case higher performance is required or new components have to be included into the system. The CLAS12 DAQ has been successfully operated during the first year of the CLAS12 experiment and expected to be used in the future with minor modifications.

2. Requirements

The CLAS12 DAQ requirements were initially defined for electroproduction experiments with a 10 kHz event rate and a 100 MB/s data rate, with livetime above 90%. During detector construction the data rate requirement was increased to 200 MB/s. The data rate requirements were never officially changed, however the very first experiment showed that the initial design requirements were too low. This situation was anticipated by the developers and the system has been shown to be able to take data with an event rate of at least 30 kHz and a data rate of at least 1 GB/s, which meets the current experiment demand. The system has the potential for futher performance increases as well.

3. Design

The CLAS12 DAQ was designed as a pipelined, network-based system. The data-taking process starts from the front-end components. Those components can have different hardware and software implementations, but have to follow certain requirements to be compatible with the rest of the system. Front-end components are usually referenced as Readout Controllers (ROCs), although initially that term was used to name only the processing part of the front-end hardware and corresponding software (for example Intel-based VME controllers). Currently, the ROCs used are commercial VME/VXS crates with Intel-based controllers and custom VME/VXS boards, commercial Linux servers, and Jefferson Laboratory(JLab)-designed VXS Trigger Processor boards (VTPs). VTPs are installed in all of the VXS crates, but they are read out by the DAQ as independent ROCs. All components receive a common 250 MHz clock distributed over OM3-rated parallel optic fibers. The same fiber system is used to distribute both the synchronization reset and trigger signals and to collect the busy signals from all front-end electronics.

All front-end components are connected via TCP sockets over Ethernet to the Event Builder (EB) component. The EB is a multi-threaded program (C application) running on a multi-core Linux server. Most front-

end connections are 1 Gb links. However, for several components that generate significant data rate, a 10 Gb Ethernet connection is used.

The built events are output to the Event Transfer (ET) system. This multi-threaded program (C application) provides access to shared memory storage in a ring architecture where multiple data processing programs can be attached to process, filter, or monitor data quality online. The ET system is typically run on the same server as the EB, but it can also be used to distribute events to a sequential chain of ET servers to increase the aggregate data processing power.

The last component in the data chain is the Event Recorder (ER). This multi-threaded program receives data from the ET system and records it to the disk. A multi-stream mode is available, which allows several files to be written in parallel to the same or different disk partitions to increase writing performance. The event order in multi-stream mode is preserved. The CLAS12 DAQ system diagram is shown in Fig. 1.

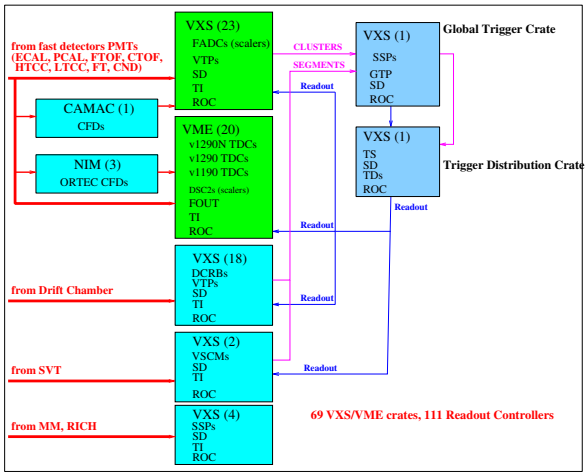

Figure 1: Diagram of the CLAS12 Data Aquisition system. It includes 69 VXS/VME64X crates with front-end electronics, trigger modules, and Intel-based controllers running Linux OS. Back-end components such as Event Builder, Event Transfer, Event Recorder, and Run Control, run on a designated server (not shown) connected to the front-end crates by 1 Gb and 10 Gb Ethernet links.

4. Hardware Components

4.1. General Information

Most of the front-end electronics for the CLAS12 DAQ System was designed and built in JLab (except CAEN v1190/v1290 TDCs). New modules take advantage of the higher performance and elegant back-plane connectivity of the VITA 41 standard or "VXS", defined as VME with serial extensions. VXS was selected as the 12 GeV data acquisition back-plane foundation for the front-end detector readout and trigger hardware interface because this standard not only supported legacy VME hardware but also offered a method to easily synchronize and pass signals between each of the payload slots to a central switch fabric slot. At JLab a dual-star configuration is used for the serial backplane. One switch slot is used for the trigger processing from payload boards, and a second switch slot is used to distribute the essential timing and synchronization signals to each of the front-end boards.

Following VXS modules were produced: TS (Trigger Supervisor), TD (Trigger Distribution), TI (Trigger Interface), SD (Signal Distribution), FADC250 Flash ADC), DCRB (Drift Chamber Readout), VSCM (VXS SIlicone Readout), SSP (Subsystem Processor), VTP (VXS Trigger Processor). Last two modules were designed for the CLAS12 Trigger system but eventually served for both Trigger and DAQ systems.

All those modules are described in details below in this section, while VTP description can be found in the CLAS12 Trigger System publication (ref. [2]). Shortly, VTP is installed in the VXS switch slot and manages the high-speed gigabit signaling from each of the payload slots, where eight differential pairs connect from the payload slots to the switch slots. The VXS crates used in JLab are manufactured by WIENER and the back-plane can support up to 8 Gb/s. The payload boards use Xilinx FPGAs and have up to 10 Gbps transceivers. The payload boards are designed to run these high-speed gigabit transceivers at a maximum of 5 Gpbs to transfer trigger data to the VTP module to produce Level 1 Trigger decision.

The design challenges for reliable and successful transmission of gigabit serial data over the VXS back-plane requires the investment of high-speed circuit board layout and routing tools. The FPGA selection requirements include at least four full duplex gigabit transceivers, user I/O pin count > 500, and fast integrated block memory with multi-rate FIFO logic. We use the circuit board routing simulation Hyperlynx tools from Mentor Graphics [3], which are invaluable for critical simulation and verification of circuit board signal integrity for the gigabit transmission paths before the manufacturing process. The FPGA devices that we use are capable of 6.25 Gb/s serial transfer, and we have designed our circuit boards with signal integrity techniques using standard FR4 circuit board material to achieve > 2.5 Gb/s, which meets the data transfer bandwidth requirements.

Another significant investment required for the hard-

ware verification of the gigabit transceivers was a digital signal analyzer with 8 GHz bandwidth to measure and record the backplane and fiber optic gigabit transceiver performance and to perform jitter analysis on the critical system clock and synchronization signals with at least 1 ps resolution. We used the Tektronix jitter analysis software, which is a critical tool for the verification of our system clock, and for measurements of the phase-controlled jitter attenuated clock provided by the Signal Distribution (SD) switch card in every crate. The investment of firmware development tools from FPGA industry leaders Xilinx and Altera were also taken into consideration for the upgrade path to VXS. We use the Xilinx Aurora protocol for serial transmission as it is robust and simple, and is included with the FPGA development tools.

4.2. Fiber Optic Trigger distribution system

As shown in Fig. 2, the trigger information from each VTP in the front-end crate, and the distribution of the global clock, synchronization, and trigger commands from the global trigger hardware, use a separate fiber optic cable. The crate sum fiber link is shown in orange, and the critical timing signals distributed to each front-end crate are blue. Each fiber optic link makes use of the Avago POP4 fiber optic transceivers and parallel OM3-rated glass fiber cable with MTP connections. These fiber optic transceivers operate at 3.125 Gb/s for an aggregate bandwidth of 10 Gb/s, which is ample enough for the summing information that is sent forward to the global trigger processing hardware. The fiber link used for the distribution of the global clock, critical timing signals, and trigger commands runs at 1.25 Gb/s.

4.3. VXS/VME Crates

Previous experiments with the original CLAS spectrometer (see [4]) used the VXI standard, which was a new extension of the original VME standard. VXI offered a method to distribute clock and other timing signals with low skews via the back-plane. 9U circuit boards were used that offered a large number of front panel input/output connections to handle the six sectors of the CLAS detectors that contributed to the level 1 trigger. The detector signals were digitized with FASTBUS ADC and TDC modules, or in some instances, from VME or even CAMAC modules.

During the initial design phase of the 12 GeV experiments, the requirements of a 200 kHz sustained trigger rate demanded that the front-end modules adopt a new method of handling precision timing and synchronization over dozens of front-end crates. The latest technology at the 12 GeV inception included FPGA devices

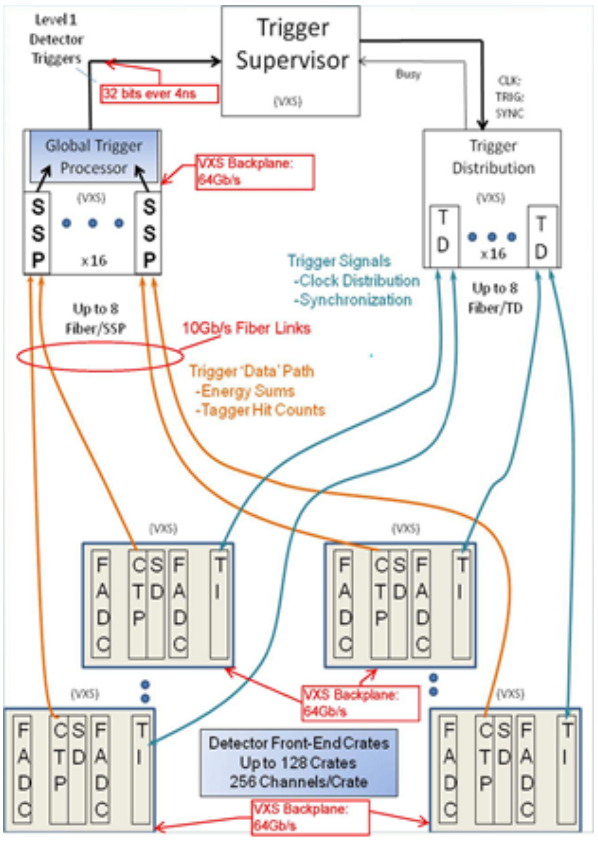

Figure 2: Hardware diagram with the implemented fiber links scheme. All critical signals such as clock, sync, trigger, and busy, are disctributed over optical fibers.

with high-speed serial transceivers built into the silicon fabric. A new VME extension was also emerging at the same time, which was labeled VXS, and defined a new high-speed gigabit connector with links between the VME slots and eight serial links to common switch slots. The VXS standard was declared as VITA 41 and several new standards have emerged in the past decade that expand the use of gigabit serial transmission via the crate back-plane. For the JLab 12 GeV experiment era, we now have thousands of custom VXS payload and switch slot modules and hundreds of VXS front-end crates. Complex experiments and high-channel-count detectors make use of these custom VXS boards designs for all four experimental halls at JLab.

4.4. VME Crate Controllers

The high-speed data physics acquistion and trigger systems for the JLab 12 GeV experiments have been standardized on the VME64X and VXS backplane and crate enclosure form-factor. In addition to the custom

electronics that reside in these crates, there must also be a single "controller" for each crate. Considering all four experimental halls, this exceeds 150 controllers.

There are many commercial off-the-shelf options for this type of controller, and our general requirements do not extend beyond what is currently commercially available. We do have some specific requirements that narrowed the viable choices.

We purchased VME controllers from several vendors for development purposes and made a significant investment in custom software that runs on all of the existing boards. We also benchmarked our code and have come to expect certain "minimum" requirements for performance from the chosen architecture within a specified Linux operating system.

We also expect a certain minimum 10 year time frame in which these controllers will be supported by the vendor with respect to the available parts, repair, and software updates. Our VME controller requirements are summarized as followings:

- Single-slot VME form-factor - no required rear transition module;
- Intel Core i7 dual- or quad-core embedded processor 2 GHz (or greater);
- Hyperthreading and 64-bit arch support;
- 4 GByte DDR3 (1066 MHz) ECC SDRAM (or greater);
- Front-panel gigabit Ethernet and serial port console;
- 1 x4 PCI Express XMC expansion slot (or greater);
- VME320-interface using the Tundra Tsi148 chip;
- Support for all VME transfer modes including 2eSST;
- VXS optional: interface supporting both VITA 41.3 (Gig E) and 41.4 (PCIe) standards.

After several different boards were evaluated, we purchased VX915 Intel 4th Generation quad-core i7-based VME single board computers from Concurrent Technologies, and some number of the XVR16 Intel i7-based VME single board computers from GE. They were installed in the 69 crates for CLAS12 and have demonstrated excellent performance and reliability. Most of the controllers send data over their built-in 1 Gb link, while for a few of them, a 10 Gb XMC daughter board was installed to increase the output bandwidth. The maximum data rate from a single crate in CLAS12 never exceeds 130 MB/s, and with that rate quad-core controllers are able to handle the VME data polling, event processing, and sending over the network to the EB without any issues.

4.5. Trigger Distribution System Modules (TS, TD, TI)

The TCS (TRIGGER, CLOCK, SYNC, and BUSY) distribution system [5] is the hardware interface to bridge the trigger and the DAQ. The TCS system receives the trigger decision from the trigger system, and initiates data readout for the DAQ system by distributing the readout trigger (TRIGGER) signal. Additionally, it distributes a 250 MHz system clock (CLOCK) to pipeline the system, and it distributes an encoded synchronous signal (SYNC) for the system synchronization. It monitors the front-end electronics status (BUSY) and makes sure of the smooth data readout of the experiments. Fig. 3 shows a diagram of the trigger and TCS distribution.

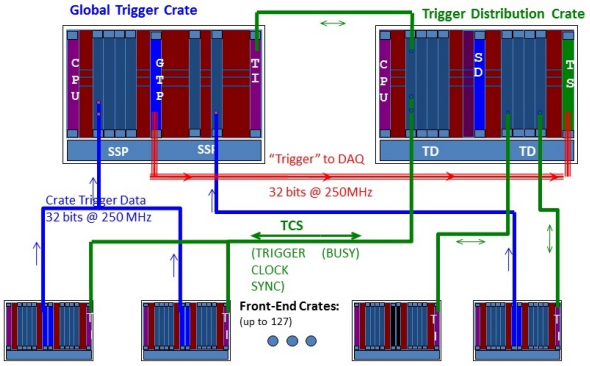

Figure 3: Diagram of the trigger and TCS distribution. The VTP boards generate triggers using detector signals from the front-end crates. The final trigger decisions, up to 32 trigger types, are sent from VTP to TS for data readout. The TS distributes the TCS to the TD through SD and the backplane, then to the front-end crate through optic fiber and the TI. The TI collects the front-end board busy signals and sends them to the TD, which throttles (disables) the readout trigger distribution on the TS.

The main hardware components of the TCS distribution system includes a Trigger Supervisor (TS [6]) board (see Figs. 4 and 5), Signal Distribution (SD [7]) boards, Trigger Distribution (TD [8]) boards (see Fig. 6), Trigger Interface (TI [8]) boards (see Figs. 7 and 8), VXS crates, and optic fibers. The TS board, one SD board, and up to sixteen TD boards are located in the global TCS distribution VXS crate. One TI board and one SD board and/or one FANIO board are located in each front-end crate. The electronics boards were custom designed and produced for the 12 GeV upgrade.

Field Programmable Gate Arrays (FPGA) are used for TCS generation, control, and decoding. Optical fibers are used to transmit signals at high speed over long distances and the high-speed serial backplane for internal crate distribution.

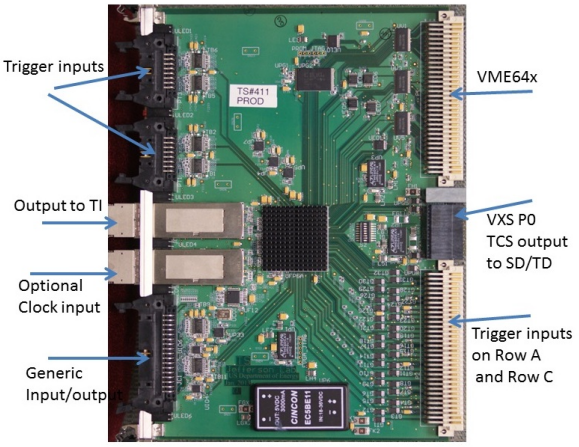

Figure 4: Trigger Supervisor (TS) board. The TS board is a 6U by 160 mm VME board with VXS connector. It generates and distributes the readout triggers, synchronization signals, and clock (either from the front-panel input or the on-board oscillator) to the TD boards via the SD through the VXS backplane.

4.5.1. *Clock Distribution*

The TCS system uses the 250 MHz clock that comes from the TS in the global trigger distribution crate. This clock is either generated by the TS on-board oscillator or its front-panel input. The clock is fanned out to the VXS P0 connector and then to the SD board. The SD fans out the clock to the TD boards via the VXS P0 backplane. The TD boards further fan out to the TI boards via optic fibers. The TI uses this clock to generate clocks with proper frequencies (250 MHz, 125 MHz, 62.5 MHz, 31.25 MHz and 41.67 MHz) and sends the clocks to the front-end crate SD board, and the SD fans out to the front-end DAQ modules (TDCs, ADCs). The fan-out buffer level is minimized on every board to limit the clock jitter. The slower clocks derived from the main system clock are phase aligned thanks to the Analog Devices AD9510 with a synchronous phase re-alignment command. The clock jitter is about one picosecond measured at the front-end electronics. The clock distribution skew can be adjusted by the SD clock delay if necessary.

4.5.2. *SYNC Distribution*

The TS generates and distributes the SYNC signal. The SYNC is an encoded 4-bit serialized command

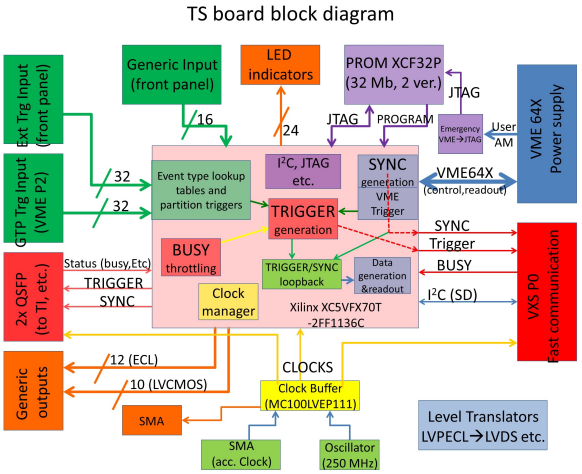

Figure 5: TS board diagram. The TS generates the readout triggers from up to 32 front-panel trigger inputs and up to 32 backplane trigger inputs (from the GTP), and sends out the triggers via encoded 16-bit words to the backplane.

transferred at 250 Mbps synchronized with the system clock. Normally, the serial SYNC line stays at logic high (or 1). When transferring a SYNC command, the SYNC goes to logic low for one bit, followed by the 4-bit command code. After the 4-bit SYNC command, the SYNC goes to logic high again. There is a minimum of four 1s before the next cycle begins. The SYNC start is phase aligned to the 62.5 MHz clock used for the trigger word transfer, the 41.67 MHz clock used for the CAEN TDC boards, and the 31.25 MHz clock used for Flash ADC boards. This phase relation is used to synchronize the slower clocks on the TI to the 62.5 MHz clock on the TS. This also limits the SYNC command to no more than one per 96 ns. To facilitate the AC coupled optical transceivers, the SYNC is Manchester encoded on the TS and the TD, and Manchester decoded on the TI and the TD.

The SYNC is phase aligned with the 250 MHz system clock on the TI boards using their FPGAs IODELAY. The SYNC is synchronized across the TI boards by applying different delays on the individual TI boards. The delays are determined by the fiber latency measurement.

The spare fibers between the TD and TI boards are used to measure the fiber latency. The TI sends a test signal to the TD through one fiber, and the TD loops back the signal through another fiber. The TI measures the delay between the test pulse and the looped-back test pulse using the FPGA counter and the carry chain in the FPGA. As the fiber skew is small (less than 1 ns for 100 m fibers), the measurement on these two fibers can be used as the latency of the other fibers in the ca-

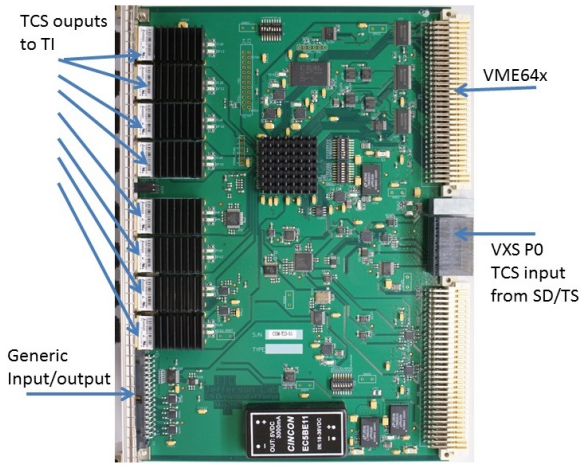

Figure 6: Trigger Distribution (TD) board. The TD board is a 6U by 160 mm VME board with VXS connector. It receives the TCS from the TS via the VXS backplane, and distributes the TCS via the front-panel QSFP optic links. It collects the BUSY inputs from up to eight TI boards, and generates the readout buffer busies for up to eight front-end crates. The TD sends the collective BUSY to TS to back pressure the trigger generation.

Bit 15:12	Bit 11:10	Bit 9-0	Comment
1001	Quadrant timing	Event type	GTP major trigger
1010	Quadrant timing	Event type	Ext major trigger
1011	Four TS partitions event types		TS partitioning (4, 3, 2, 1)
0110	Quadrant timing	Trigger source Event type	TImaster legacy Trigger (TS) VME trigger
0101	Trigger command/Control	VME command	
0100	TS timer (TS time bit(13:2))	TI Sync check	
0111	Trigger content	Additional trigger info	

Table 1: Trigger word definition. The TS encodes the readout trigger, event type, and the fine timing (4 ns quadrant) information into a 16-bit word, which is transferred every 16 ns. This 16-bit word can also be some system setting information or the system timer when there is no readout trigger in the 16 ns period.

ble. After the SYNC latency compensation, all of the TI boards receive the SYNC at the same time with the skew of one system clock period, which is 4 ns. The synchronized SYNC signals are used to synchronize the triggers as described next.

4.5.3. Trigger Distribution

The trigger words, which include the readout trigger signals and event information (event type, trigger timing, etc.), are generated and serialized on the TS. The serialized trigger word is fanned out by the SD board and the TD board, and deserialized by the TI board. The 16-bit trigger words are summarized in Table 1.

Both the fiber latency and trigger word serializer/ deserializer are compensated so that all of the TI boards send the readout trigger at the same time to the front-end data acquisition electronics. The SYNC is used in conjunction with a synchronous FIFO to enforce a fixed latency on the trigger distribution. Fig. 9 shows the diagram of the compensated trigger distribution.

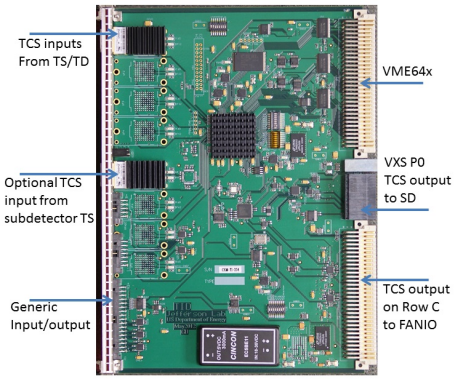

Figure 7: Trigger Interface (TI) board. The TI board is a 6U by 160 mm VME board with (or without) the VXS connector. It uses the same printed circuit board (PCB) as the TD board with modified component population. It receives the TCS from the TS/TD or a subsystem controller via fiber, and collects the BUSY and sends it to the TS/TD via the same fiber. The TI can also be used as a subsystem controller in the so-called master mode.

On the TI board, the deserialized trigger word is clocked into and clocked out of a FIFO using the 62.5 MHz clock. At the start-up, the FIFO is reset (0 words) and the FIFO read/write is disabled. The serial trigger link is idle words only. On trigger start, the TS starts trigger word transmission. The TI will write the deserialized data (valid data, that is a non-idle data word) to the FIFO. After some pre-set delay (VME register controlled), the TS issues a Trigger Start command on the SYNC link. When TI receives the Trigger Start, the TI resets the trigger FIFO readout address, and enables continuous readout of the FIFO. As the SYNC lines are fiber length adjusted and the 62.5 MHz clocks are phase aligned, the trigger words from the TI board FIFO are synchronized across the system. The trigger word also has the fine trigger timing information. By decoding that, the TI board distributes the trigger in 4 ns precision, although the trigger word is serialized every 16 ns. If the system clock phase is not adjusted, there will be a maximum of 4 ns skew among the clocks on the TI boards, and the same is true for the readout trigger. The clock phase can be adjusted by the SD if the skew is critical to the system.

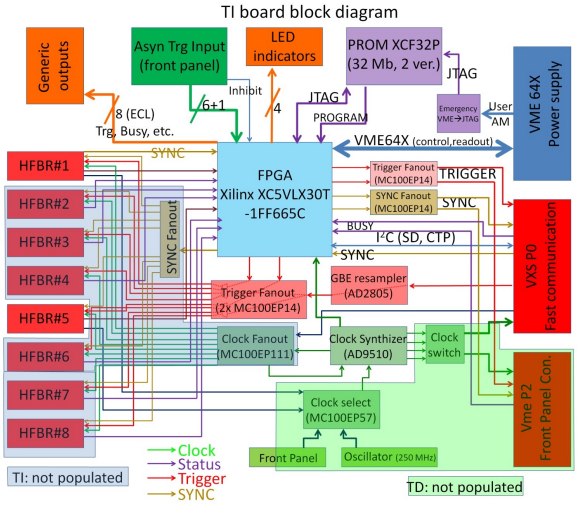

Figure 8: TI and TD board diagram. The TI and TD use the same PCB design, but different component population and FPGA firmware.

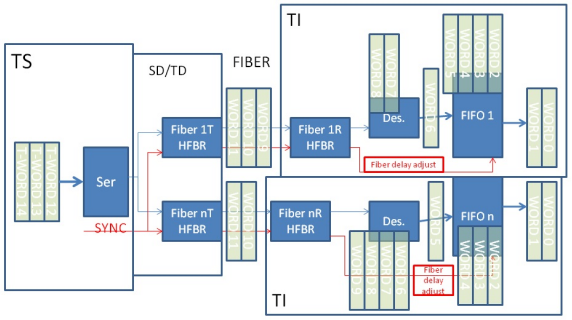

Figure 9: Trigger synchronization between TIs. The TI boards delay the decoding of the received readout trigger by the complement of the TS/TD to TI transfer latency, so that all the front end boards receive the readout trigger simultaneously.

4.5.4. *DAQ Synchronization (Trigger Throttling) Control*

Because of the finite memory size and the randomness of the triggers, it is possible for the memory to become overwhelmed somewhere in the system, which could cause DAQ problems. The TCS throttling mechanism is used to prevent possible memory overflows, and to keep the DAQ synchronized. Fig. 10 shows the DAQ synchronization logic implementation. Three methods are used to keep the DAQ synchronized. These include trigger rules and event limit setting, pipeline DAQ, and synchronization events (special events).

4.6. *Signal Distribution Module (SD)*

The Signal Distribution Board (SD [7]) module (see Fig. 11) occupies the B switch card slot as specified in VITA 41. The main purpose of this module is to distribute the signals received from payload slot 18 (Trigger Interface board) of a VXS crate to the 16 other payload slots occupied by FADC, DCRB, VSCM, or any other VXS front-end boards.

The SD module distributes the 4 LVPECL differential pair signals from payload slot 18 to 16 VXS payload slots within the crate. This is done using the high-speed, point-to-point connections from the switch slot to each payload slot. The four distributed signals are length-matched to minimize the output jitter seen on all of the payload slots. Three of the four remaining pairs are LVDS signals routed from the each payload module to the FPGA on the SD module. The last pair is an LVDS signal routed from the FPGA on the SD module to each payload module. Each of the 16 payload modules has a single-ended signal to the SD module and one from the SD module back to the payload module.

4.7. *Flash ADC Module (FADC250)*

A 16-channel 250 MSPS pipelined flash ADC (FADC [9], see Fig. 12) with 12-bit precision was designed to digitize and process detector pulses for experiments at JLab. The FADC250 module (see Fig. 13) conforms to the VITA 41 VME64x switched serial (VXS) standard. Each channel of the module accepts input signals on a LEMO style coaxial connector and has three user-selectable ranges (0.5 V, 1.0 V, 2.0 V). Differential signal conditioning scales the input signals to within the dynamic range of the ADC and a single-pole low pass filter limits the signal bandwidth to the Nyquist band of the converter (125 MHz). Individual channel offsets are accomplished by means of DACs under VME control. Each channel has its own dedicated ADC chip (Analog Devices AD9230). Both positive and negative polarity input signals are supported. The 250 MHz clock is distributed to the ADC chips through a low jitter (< 2 ps) network.

Digitized data from the 16 FADC chips is processed and formatted for readout in a pair of high performance Xilinx FPGAs. The digitized data from the ADCs follows two distinct paths. Logic in the trigger data path pre-processes data for the trigger algorithms of the VXS Trigger Processor (VTP). The FADC250 module continuously streams this data to the crate VTP located in VXS switch slot A via high-speed serial links of the

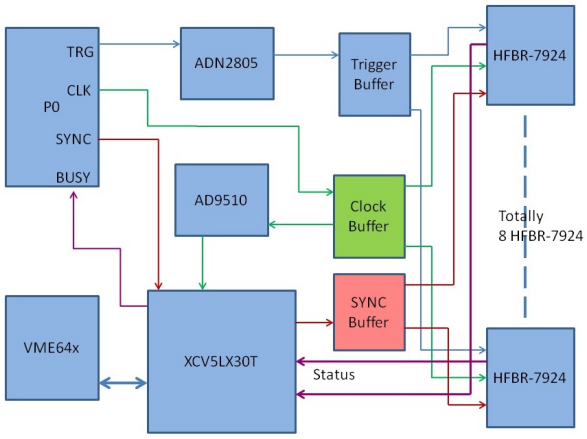

Figure 10: DAQ synchronization logic. Three methods are used to keep the DAQ synchronized: trigger rules and event limit setting, pipeline DAQ, and synchronization events.

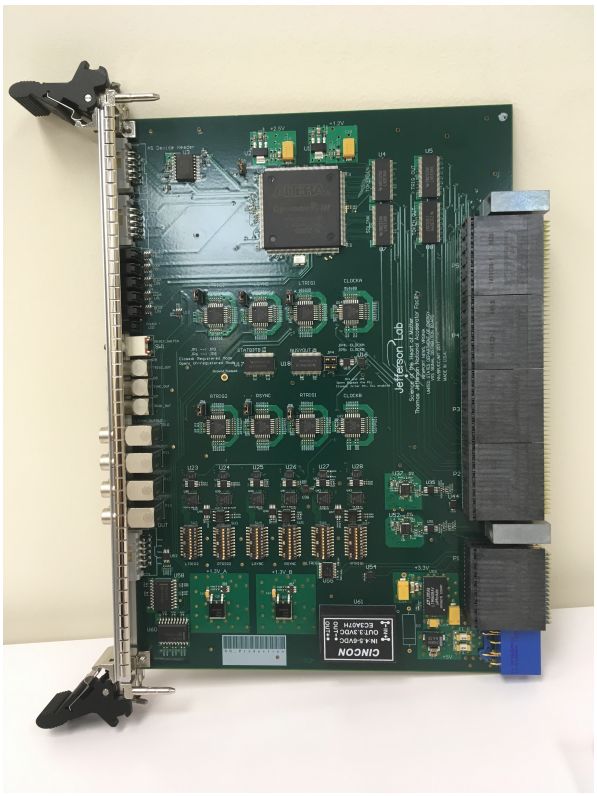

Figure 11: Signal Distribution module (SD). It distribute the signals received from payload slot 18 (Trigger Interface board) of a VXS crate to the 16 other payload slots.

VXS fabric. Data from multiple VTPs and other modules are used to form a global trigger signal that is returned to the crates to initiate data readout.

The readout data path continuously stores digitized data for each channel in circular buffers. When a trigger signal is received by the module a programmable window up to 2 μs wide of digitized data is extracted from the buffers for processing. The starting point of this window can be programmed up to 8 μs earlier than the arrival of the trigger signal to account for the time required to form the trigger signal. Zero suppression on the extracted data may be implemented for each channel using programmable thresholds. A lossless data compression algorithm can also be applied to the data of each channel with typical compression factors of 2 to 3. The design is pipelined so that data from multiple triggers can be processed simultaneously. Triggers separated in time by as little as 50 ns can be accepted by the module. The trigger number and trigger time (clock periods since last synchronization) are reported along with the channel data so that data from multiple modules can be correctly assembled into events.

Data associated with a programmable number "N" of triggers is packaged into a block of data for readout over VME. "N" can take values from 1 to 255, with 40 being a typical value chosen. Data is stored in an on-board 8 MB SRAM as 64-bit words to match the 64-bit high-speed (200 MB/s) dual edge VME Source Synchronous Transfer (2eSST) mode employed to read out the module.

In order to save the overhead of setting up a DMA transfer for each FADC250 module in the crate, a chained block readout mechanism with token passing is used. A common address range is enabled for all modules in the crate but only the module having the token will respond to a read request. A single logical DMA read is initiated by the VME crate controller and the first module in the chain supplies data from its block of event fragments. When the block data from the first module is exhausted, a token signal is passed to the next module in the chain and this module then proceeds to transmit its data from its block. When the block for the data from that module is exhausted, it transfers the token to the next module. This continues until the data from the last module in the chain is exhausted. Instead of passing the token, the last module asserts the VME bus error signal (BERR), which terminates the DMA cycle. The user returns the token to the first module and the process can begin again when the next block of events is ready for readout. The user does not have to query the modules in advance to discover the number of words to read out. The DMA is set up with a total number of words larger than any expected value for the entire crate. Data from each module is tagged with the slot number to identify

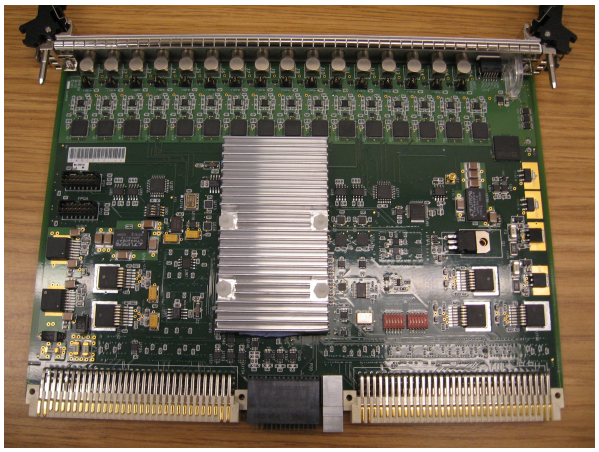

Figure 12: Flash ADC module (FADC250). 16-channel 12-bit 250 MHz digitizer in VXS format is capable of providing information to both the readout chain over VME bus and trigger logic over VXS serial lines.

Property	Value
Analog Discriminator	
Threshold	0 to -1023 mV
Pulse width	4 ns to 40 ns
Dead-time	4 ns typ. w/8 ns pulser width
Maximum input rate	>125 MHz
Ch-ch isolation	>65 dB
Threshold noise	1.3 mV RMS (typical)
Slew-rate delay disperson	<20 ps
Input-to-output delay	<5 ns
Digital Processing	
Digital delay step	4 ns
Digital delay maximum	1 μs
Digital width maximum	1 μs
Maximum count rate	125 MHz

Table 2: DSC2 Board Specifications.

its source. The token passes along a VXS signal line to VXS switch slot B, where a module there (SD) routes it to the next enabled module.

4.8. Discriminator Scaler Module (DSC2)

The Discriminator Scaler Module (DSC2 [10], see Fig. 14) is a 16-channel general purpose discriminator and scaler module designed as a 6U VME card. It replaces an older design, improving on jitter, noise, crosstalk, and adding new features. The board specifications are summarized in Table 2.

Discriminator. Inputs are single-ended LEMO and leading-edge discrimated by two different thresholds.

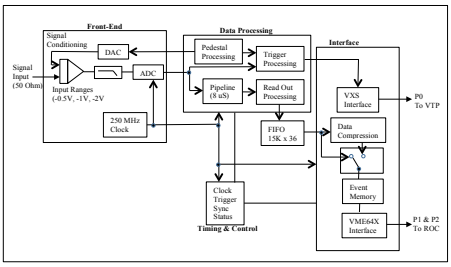

Figure 13: Flash ADC module diagram (FADC250). In addition to pulses readout and trigger information, it is equipped with scalers for every input channel.

Typically one threshold is used for time-to-digital applications and the other threshold is used for trigger applications. There are separate differential ECL outputs for each channel and threshold. Low jitter performance was an important goal of the design as this module will be used in high resolution applications. Fig. 15 shows the typical jitter as a function of a variety of input slew rate signals and threshold overdrive conditions.

Digital Processing. A Xilinx Spartan 3A FPGA is used to implement the VME interface, scalers, discriminator controls, and scaler event building features. Each channel and threshold has two scalers associated with it. The first scaler counts all threshold crossings for the input. The second scaler is gated using a front-panel input source, which can be useful to compute the dead-time of channels and many other applications. Additionally, reference scalers are accumulated (a gated and ungated version) that count the elapsed time, which can be used to normalize inputs scalers to Hz. All together there are 68 scalers, which can be slow to read over VME if using single-cycle transfers. An event builder is implemented that can synchronously read (and optionally

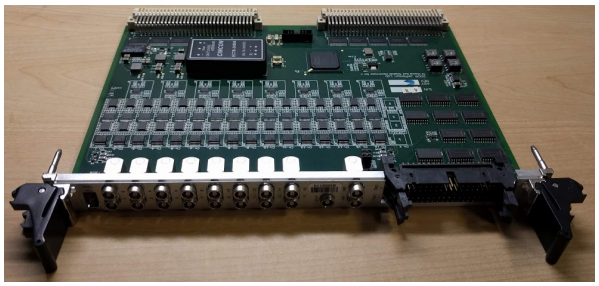

Figure 14: Discriminator Scaler module (DSC2). 16-channel general purpose discriminator and scaler module designed as a 6U VME card.

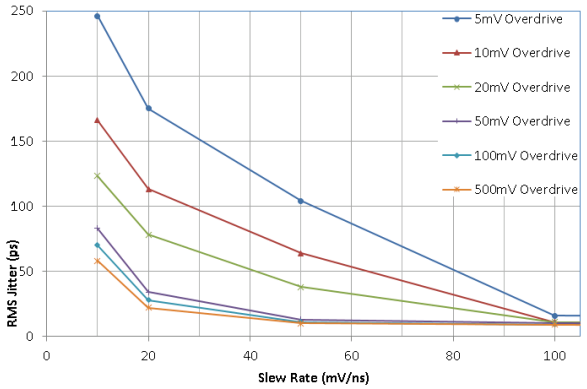

Figure 15: Output Jitter vs Input slew rate and overdrive for DSC2.

clear) all scalers and build an event with this data. Over 100 events can be buffered and readout using the VME 2eSST protocol at 200 MB/s.

4.9. TDC Modules (v1190/v1290)

Commercial CAEN V1190/V1290/V1290N TDC [11] boards are used for timing measurements in the PMT-based CLAS12 detectors (see Fig. 16). The V1190 has timing resolution about 100 ps and the V1290 about 35 ps.

All boards installed in CLAS12 run on an external 250/6=41.666 MHz clock rather than the internal 40 MHz clock. The use of a different clock required compensation table remeasuring and reloading. That table is implemented to compensate for the integral non-linearity (INL). The INL of the TDCs represents the accumulated error of the input-output characteristic of the TDC with respect to the ideal response and is defined by:

$$D(t) = \int \frac{t}{LSB}, \qquad (1)$$

where D is the output data, t is the input time, and LSB (least significant bit) is the intrinsic bin size. The compensation tables for the CAEN V1190A and VX1290A TDCs are stored as tables in the unit SRAM memory. Initial tables are measured at the factory and come preloaded on the modules. These tables are reasonably accurate when operating the module using its internal 40 MHz/25 ns period clock. However, in CLAS12, the modules are strobed with an external clock of a slightly larger frequency of 41.67 MHz. This difference in the frequency has a non-negligible affect on the INL tables. For our purposes we use a high frequency pulser to populate the full dynamic range of the TDC within the CLAS12 readout clock. The measured INL tables that were derived from this calibration were written into the TDC memory to replace the factory-loaded values. Details on the procedure and the residual non-linearity affects are given in Ref. [12].

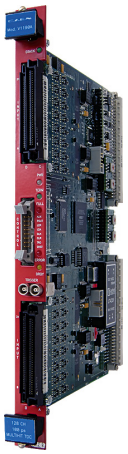

Figure 16: CAEN V1190 TDC module (V1190). 128-channel 100 ps LSB TDC board in VME format is shown. Another board in use is V1290 with 32 channels and 25 ps LSB.

4.10. Drift Chamber Readout Board (DCRB)

The Drift Chamber Readout Board (DCRB [13], see Fig. 17) is a 96-channel amplifier, discriminator, and time-to-digital converter module used to digitize and readout hits from the CLAS12 drift chambers [14]. A single VXS crate of DCRB modules can readout a full region of the drift chambers for a single sector resulting in 18 VXS crates of DCRBs to instrument 6 sectors each having 3 regions of drift chamber.

Analog Inputs. Each DCRB receives 96 differential analog signal pairs using twisted pair cabling from the drift chamber pre-amplifiers. The pre-amplfiers located on the detector provide a gain of 2.3 mV/μA. On the DCRB each analog input channel is amplified by

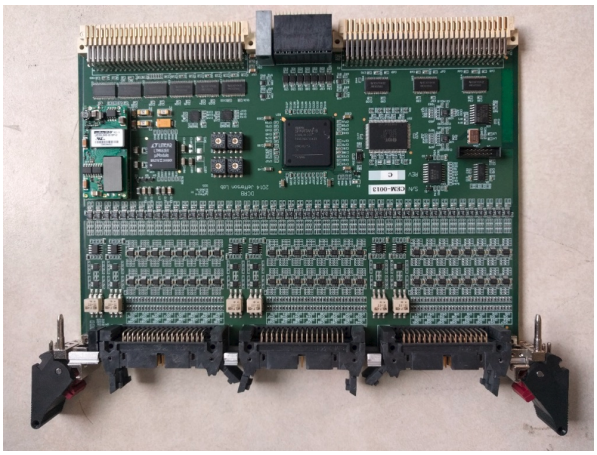

Figure 17: Drift Chamber Readout Board (DCRB). 96-channel amplifier, discriminator, and time-to-digital converter module in VXS format.

a voltage gain of 30 and then discriminated by a programmable threshold (common to all channels on the board with an effective chamber wire threshold range of 0 to 3.5/μA).

TDC Event Builder. All discriminated channels go to a Xilinx Spartan 6 FPGA where a 96-channel, 1 ns resolution time-to-digital converter (TDC) is implemented in firmware. The TDC is based on the ISERDES2 shift register FPGA primitive that directly samples the digital input with a single-data-rate (SDR) input register clocked at 1 GHz. The TDC sampling clock is synchronized to the CLAS12 master oscillator, making it easy to relate hit times in the drift chamber to the other detectors in CLAS12. The TDC inputs are buffered to support multiple hits, allowing for an average hit rate of 4 MHz per input before loss of data, which exceeds the chamber design hits rates by a few orders of magnitude. Hits from groups of 16 channels are written into a large buffer that a linked-list content addressable memory (CAM) tracks for 16 μs. When a L1A trigger signal is received, a time window of hits is extracted from the TDC hit buffer. The readout window times are supplied to the CAM, and the CAM provides the address of the last hit matching each readout time bin. The hit buffer is then read to extract the hit and also the address of the next hit in the buffer matching the time bin (this is the linked-list behavior). The result is an extremely fast event builder with natural zero suppression that does not require time-sorted data. Cleanup is accomplished by a timer that invalidates the CAM entries after time bins are older than 16 μs. Hits for an event are assembled and buffered in a 2 MByte external RAM, which is readout through the VME bus using the 2eSST protocol at 200 MB/s.

Calibration Support. A programmable amplitude pulse generator is implemented that can inject test pulses directly into the DCRB differential amplifier inputs, as well as to the pre-amplfiers that are on the detector. This provides a way to test points of failure, check channel gain, and check channel delays without any extra equipment. A scaler is implemented on each channel for slow control monitoring of all chamber wires.

4.11. VXS Silicon Readout Module (VSCM)

The CLAS12 Silicon Vertex Tracker detector (SVT, [15]) front-end utilizes the data driven FSSR2 ASIC for digitization. The VXS Silicon Readout Module (VSCM [16], Fig. 18) was designed to interface the FSSR2-based front-end to the CLAS12 DAQ system. This system is capable of reading out all 33,792 SVT channels in 3 VXS crates.

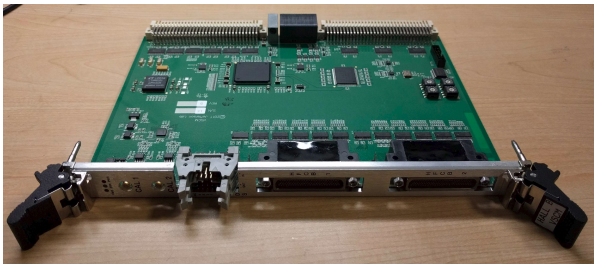

Figure 18: VXS Silicon Readout Module (VSCM). Designed to interface the FSSR2-based Silicon Vertex Tracker detector front-end to the CLAS12 DAQ system.

The main features of the VSCM include:

- Receives 8 FSSR2 streams, each at 840 Mbps;
- De-randomizes hits into an 8 μs buffer;
- 512k hit, multi-event buffer;
- Supports >1 MHz trigger rate;
- Programmable amplitude charge injector;
- 1 ns resolution time-to-digital converter (TDC);
- Per-channel hit scaler;
- FSSR2 synchronization, status, and control.

Property	Description
Hit Time	128 ns resolution
Channel	0-1023 strip ID
Charge	0-7 threshold

Table 3: VSCM data: low time resolution hit word

Property	Description
Hit Time	1 ns resolution
Channel	0-7 chip ID

Table 4: VSCM data: high time resolution hit word

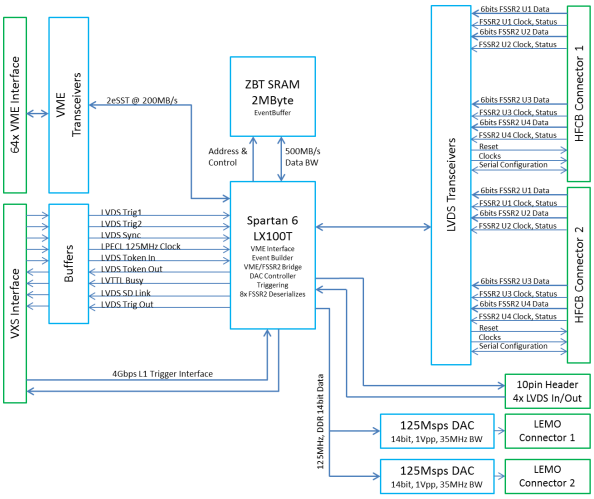

Figure 19: VSCM Hardware Diagram. Low-cost Xilinx Spartan 6 FPGA was used to implement board logic.

Event Builder. The VSCM deserializes the FSSR2 streams, checks for errors, and decodes the hits, which are stored in an 8 μs circular memory. The hits are not guaranteed to be time ordered, so the timestamp and channel number are used to form the circular memory address (rather than storing in the order received). The VSCM also implements an 8-channel, 1 ns time-to-digital converter (TDC) that measures the logic OR of hits from each FSSR2 ASIC. This high time resolution is significantly better than the FSSR2 serial stream hit time resolution and is required for improved out-of-time hit rejection. The L1A trigger signal time is used to look back a fixed amount of time and extract a time window of hits from the circular memory, which corresponds to the physics event. Non-zero hits are assembled as an event and buffered in a 2 MByte external RAM that is readout through the VME bus using the 2eSST protocol at 200 MB/s.

The event data contains primarily two hit word types that together provide high time resolution and spatial hit resolution while keeping the front-end complexity low (see Tables 3 and 4).

Fig. 19 shows the hardware block diagram of the module. Essentially, a single low-cost Xilinx Spartan 6 FPGA was used to implement the deserialization, buffering, event-building, monitoring, front-end configuration, time-to-digital conversion, and monitoring.

4.12. SSP Board as Fiber Readout Module

The SubSystem Processor (SSP [17]) was originally designed to work as part of CLAS12 Trigger System, but the optical transceivers, FPGA, and DDR2 memory are also suitable for remote front-end synchronization and readout. The SSP resides in a standard VXS crate configuration and acts as a bridge between custom front-end electronics (up to 32 remote front-ends per SSP) and the VME based CODA readout components. It collects event data from up to 32 fiber optic links, buffers events in the 4 GByte DDR2 memory, and readout by the crate controller using the 2eSST VME protocol at up to to 200 MB/sec. The RICH ([18]), MVT ([19]), and FT-Trk ([20]) CLAS12 subsystems utilize the SSP in fiber readout mode. The board description can be found in Ref. [2].

4.12.1. Readout of the Micromegas Trackers

The Micromegas Vertex Tracker (MVT) is a part of the CLAS12 Central Detector (see Refs. [19] and [20]). The 6 layers of the curved barrel detectors and the 6 forward discs account for about ~18000 and ~6000 readout channels, respectively. Two double-sided Micromegas discs form the FT-Trk tracker with some 4000 channels, which is a part of the CLAS12 Forward Tagger (FT, [21]). Under the harsh operating environment of the MVT and FT-Trk (stringent space limitations, imposed low material budget, high radiation environment, up to 5 T magnetic field), it was decided to implement readout systems with off-detector front-ends. Micro-coaxial cables of low 40 pF/m linear capacitance transfer non-amplified signals from the detectors to the front-end units (FEU) placed 1-2 m away. Fig. 20 shows the MVT/FT-Trk block diagram.

The FEUs (see Fig. 21) are equipped with the DREAM ASIC, specially designed for the CLAS12 application [DRM, [22]]. Each of the 64-channels of the DREAM chip includes a charge-sensitive amplifier adapted to a wide range of detector capacitances (up to 1 nF), a shaper with a wide range of pro-

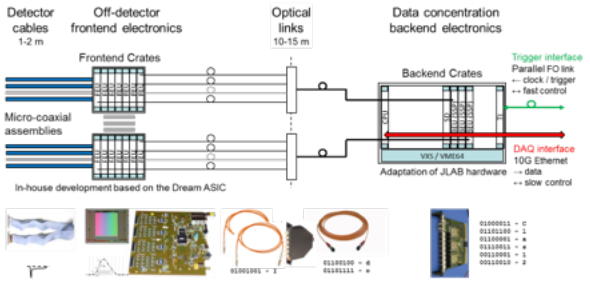

Figure 20: MVT/FT-Trk Readout System. Both front-end and back-end are shown.

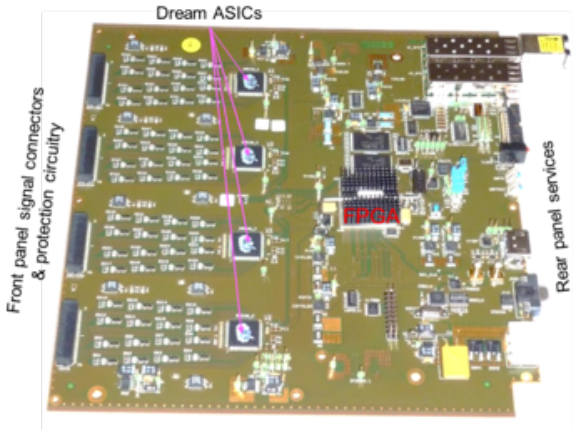

Figure 21: 512-channel front-end unit (FEU).

grammable peaking times (75 ns to 1 μs), and a 512-cell deep Switched Capacitor Array (SCA) used as the trigger pipeline memory and de-randomization buffer. The input signals are continuously sampled and stored in the SCA. Upon reception of the trigger signal, a programmable number of samples of all channels is fetched from the memory locations corresponding in time to the event and is read out serially. The sampling is not stopped during the readout process, allowing nearly dead-timeless operation. With the 25 MHz sampling and readout frequencies, and with 6 to 8 samples retained per event, the DREAM channel memory is enough to tolerate the targeted 20 kHz event rate with 8 μs trigger latency.

The 512-channel mixed analog-digital FEU includes 8 DREAM ASICs, an 8-channel serial AD9222 ADC from Analog Device, a Virtex-6 Xilinx FPGA, optical and electrical SFP modules, and several other services and monitoring circuits. The FEU is responsible for the configuration and the readout of the DREAMs, for analog to digital conversion of the signals, for the pedestal equalization, coherent noise subtraction, and zero suppression. The event fragments are then formed from the data of the retained channels and transmitted to the back-end electronics over a 2.5 Gbit/s optical link. A FEU consumes about 20 W. The 48 MVT FEUs and the 6 FT-Trk FEUs are housed in ad-hoc crates. Continuously flowing air ensures an ambient temperature of 35°C within the crates. The MVT front-end electronics operate in up to 1 T residual magnetic field of the solenoid.

The back-end electronics is based on the standard VXS crates including the crate controller SBC, TI and SD modules, and the SSP boards described in the previous sections. A dedicated SSP firmware has been developed to implement synchronous distribution of the system clock, trigger, and control signals from SSPs to FEUs over the 2.5 Gbit/s optical links. For each trigger, the SSPs perform local event building tasks assembling FEU event fragments. Multi-event buffers are formed and transferred from SSPs to the crate controller SBC over the VME64 backplane using the 2SST protocol. The transmission rates of 200 Mbyte/s are routinely achieved. A single SSP can communicate with up to 24 front-end units.

The SBC runs the Readout Controller (ROC) application of the CLAS12 CODA data acquisition framework [23]. It completes the data integrity checks performed in the SSP firmware, disentangles multi-event buffers, forms MVT and FT-Trk events, and sends them to the event builder over a 10 GB/s Ethernet link. The MVT and FT-Trk readout electronics are continuously surveyed by the CLAS12 detector monitoring system using the EPICS framework [24].

4.12.2. Readout of the Ring-Imaging Cherenkov Detector

The CLAS12 Ring-Imaging Cherenkov Detector (RICH, [18]) contains 391 multi-anode PMTs (MAPMTs), corresponding to 25024 channels of readout. The front-end electronics are equipped on-detector, forming compact modules called “tiles”, see Fig. 22, comprised of the MAPMT, adapter, ASIC, and FPGA boards. There are 138 tiles needed to readout a single sector of the detector (currently there is only a single sector installed). Each tile contains either 128 or 192 channels and uses the MAROC3A ASIC for gain equalization, shaping, and discrimination of PMT pulses. Each tile also has an FPGA board that performs a 1ns time-to-digital conversion measurement for leading and trailing edges of the amplified and discriminated PMT pulses.

All 138 tiles are distributed across 5 SSP located in a single VXS crate using ~20 m 2.5 Gbps optical links.

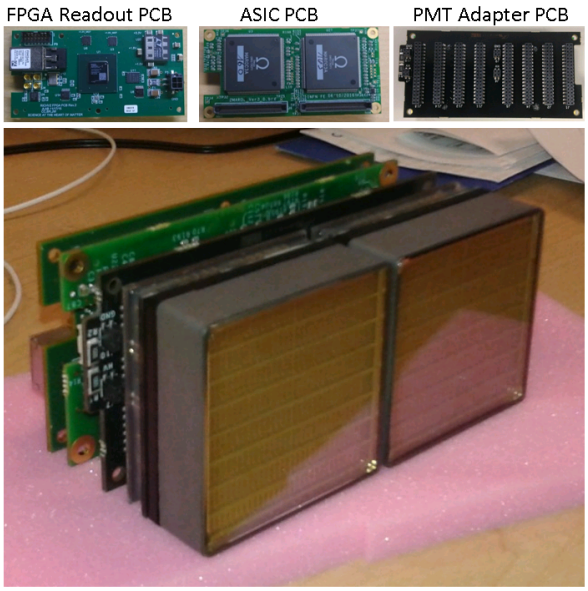

Figure 22: RICH front-end tile (128 channel version).

The tile FPGA buffers the TDC hits in on-chip memory waiting up to ~8 μs for a fixed latency trigger. Given the low hit occupancy of events, the RICH DAQ can accept trigger rates close to 100 kHz before contributing to significant deadtime. The data rate for the single RICH sector is typically under 10 MB/s for CLAS12 trigger rates around 15 kHz.

5. Software

5.1. CODA DAQ Software

The CLAS12 DAQ system is based on the CODA ([23] system developed at JLab (see Fig. 23), short for CEBAF Online Data Acquisition. It is a software toolkit of applications and libraries that allows the implementaion of a data acquisition system. The scale of the system can range from a few detector channels in a test stand to tens of thousands of channels in a large detector installation in one of the experimental halls. CODA achieves this scaling through modularity and provides a set of supported hardware components along with complementary software components.

5.2. Run Control

The CODA DAQ system includes a run control facility consisting of a back-end run control supervisor and a front-end graphical operator display that connects to the supervisor and controls its operation. The supervisor in turn controls operation of the many CODA components

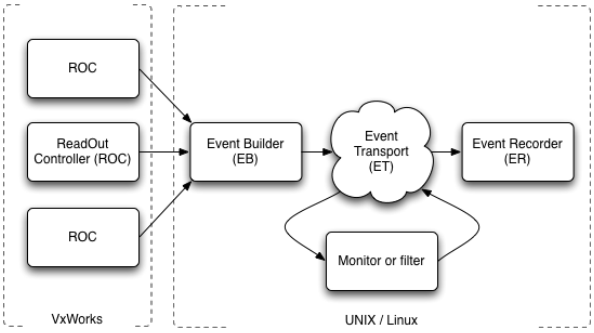

Figure 23: CODA System Diagram. Every box represents a multi-threaded program running on corresponding controller or server. All communication between programs is done over Ethernet using TCP/IP protocol.

that participate in the DAQ configuration. The latter are defined in run configuration files that the operator chooses at startup. The Run Control Graphic User Interface (see Fig. 24) presents the operator with a choice of possible actions that depend on the current state of the run. The supervisor translates the operator choice into appropriate commands to the individual components. Alternatively, limited communication with the supervisor can be performed via command-line scripts. In addition, the supervisor monitors the health and operation of the CODA components and warns the operator or pauses the run if problems are detected.

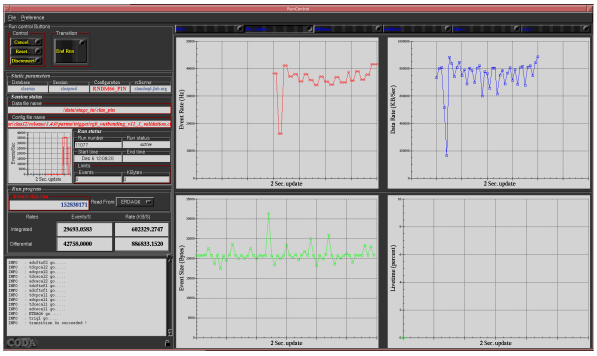

Figure 24: Run Control GUI. Main control and monitoring screen for the CLAS12 DAQ system. The tabs on the right side allows switching between different control screens, such as the configuration editor, current DAQ components status, and DAQ rates.

5.3. Front-End Libraries

The front-end libraries were developed mostly at JLab.

Just an outline ATM

GEFANUC driver:

- Customized kernel driver and user space interface
- C API provides:
  - Setup of VME inbound and outbound windows
    - Permanent windows: CRCSR, A16, A24, A32
    - Map of these windows into user space
  - Allocate Physical Memory for DMA
    - Map of this memory into user space

JVME User space Driver:

- C API provides
  - common user space interface to GEFANUC driver and others
  - Initialization of kernel driver and default VME windows
  - Maps VME bridge registers into user space
    - Provides userspace configuration and operation of DMA
  - Initialization of and access to shared memory mutex for intra-process cooperation during DMA

Front-end Libraries:

- C APIs provide
  - thread safe
  - module register mapping in VME windows to memory structures.
  - configures modules for readout via
    - programmed i/o
    - Single module DMA
    - Multiple module DMA
      - Token passing (P0/VXS and CBLT) with common A32 address range
      - Linked List DMA

5.4. Readout Controller

The Readout Controller (ROC, see Fig. 25) software component is a C program running on the front-end controllers such as the Intel-based VME/VXS crate controllers, VTP trigger boards, or regular Linux servers - essentially any hardware receiving data from the front-end electronics. On DAQ startup, the ROC main program starts three threads. After this it just controls the thread health and communicates with the Run Control process. Three threads (readout, processing, and network) pass data from one to another, communicating via circular buffers. The typical number of buffers in each buffer queue is 8 and the size of every buffer is 4 MB, which defines how long the front-end electronics can be read out before blocking triggers in the case of back-end busy conditions.

The first thread (readout) receives data from the front-end electronics and places it into the first circular buffer. That thread can run in polling mode, which occupies an entire CPU core (or optionally in interrupt mode). The CLAS12 readout primarily employs polling mode, which has adequate performance on multi-core controllers. The second thread (processing) reads data from the first circular buffer and performs all needed data processing. In particular it performs the so-called "disentangling" as well as data sanity checks. The results are placed into the second circular buffer. The processing component can create its own worker threads to increase processing power if necessary. The final thread (network) reads data from the second circular buffer and sends it over the network to the Event Builder.

The first and second threads can load user code that is compiled separately and downloaded dynamically at runtime, which allows users to develop experiment-dependent processing without recompiling the enitre ROC application.

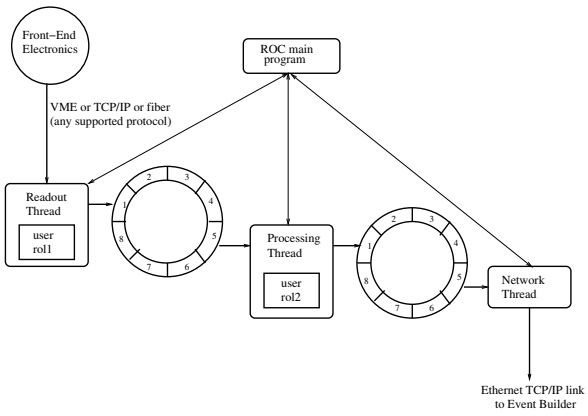

Figure 25: Readout Controller (ROC) Diagram. The ROC program typically runs on a 4-core VME Intel Controller. It includes 3 main threads performing VME readout, data processing, and networking.

5.5. Event Builder

The Event Builder (EB, see Fig. 26) is the program that receives the data fragments from all readout controllers and assembles it into events. The building process is based on event number, event type, and timestamp of the data fragments: for each event all three val-

ues have to be identical for all data from all Readout Controllers. In case of any differences, the DAQ will be stopped and the error reported.

The Event Builder consists of receiving and building parts. The receiving part contains a set of independent threads, one per Readout Controller connected to it by TCP protocol. Every thread receives data and places it into an internal buffer. If the buffer becomes full, the thread stops receiving data, effectively propagating the busy condition back to the Readout Controller. The building part has a number of identical building threads, which take turns by getting data from the receiving part of the internal buffers, building events, and placing them into Event Transfer System.

The total number of threads in the receiving part of the Event Builder in CLAS12 DAQ is currently 118, which therefore represents the number of network connections from the Readout Controllers. Because of this the DAQ has to run on a powerful server, with many CPU cores, large memory, and a high bandwidth network card. CLAS12 is using a Dell R730 server with 32 cores, 64 GByte memory, and 40 Gbit network card. This server is adequate for the present CLAS12 DAQ requirements.

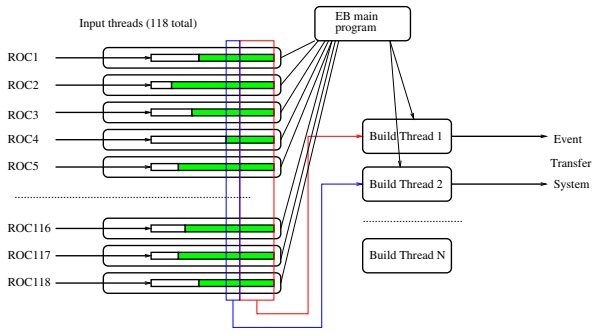

Figure 26: Event Builder (EB) Diagram. The EB program typically runs on a multi-core Linux server. It collects event fragments from the ROCs, builds events, and sends events to the Event Transfer system.

5.6. Event Transfer

The Event Transfer (ET, see Fig. 27) system provides the user with an efficient method for moving bulk data between processes. The ET was not designed as a messaging system but a shared memory system that can be visualized as data containers moving around a circular railway track. In this metaphor, a producer of data (Event Builder) requests an empty container, fills it with data, tags it with metadata describing the contents, and places it at the start of the track. Stations along the track are assigned algorithms for testing the metadata and selecting the containers of interest. The user has the option of making a station blocking or non-blocking. A container stopped at a blocking station holds up all of the other containers behind it on the track. A data consumer attached to the station processes the data in the container and has the option of either putting the container back on the track, where it proceeds to the next station, or returning the container to the station at the start of the track, effectively discarding the data. In the simple example diagram shown in Fig. 27, a data monitoring consumer attaches to a non-blocking station that samples the data of interest. The Event Recorder attaches to a blocking station and records the content of every container to disk. The container is then returned to the start of the track.

Using these simple concepts, complicated data pathways can be constructed. The ET package supports remote consumers connecting to stations over the network, allowing load sharing between multiple machines.

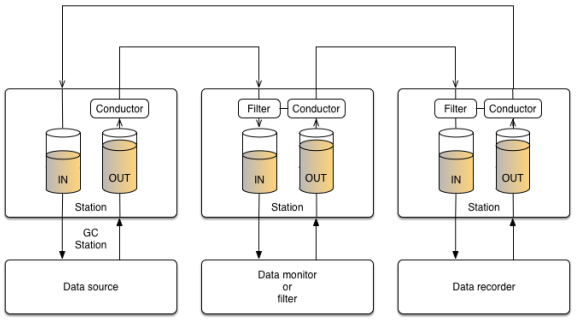

Figure 27: Event Transfer (ET) System Diagram. The ET program typically runs on a multi-core Linux server. It accepts events from the Event Builder and passes them through the set of stations, allowing attached programs to perform data processing. The last station typically has the Event Recorder attached to it.

5.7. Event Recorder

The Event Recorder (ER, see Fig. 28) is the software component that receives data from the ET system and records the data in files onto a disk. The ER can write data to one or several files in parallel (so-called multi-stream mode). If multi-stream mode is used, the event order is preserved, but it requires the ER allocated memory size to be larger than the number of streams multiplied by the individual file size.

The ER structure in multi-stream mode is shown in Fig. 28. The distribution thread receives data from the Event Transfer system, searches for the free writing

thread, and grabs its semaphore. It starts filling up the buffer of the thread with data until it becomes full. After that it signals the writing thread to start writing the entire buffer to the specified output file name. The writing thread marks itself "busy" and writes data to the file, and after that, it becomes "free" again. While the writing process is in progress, the distribution thread grabs another free writing thread and the process repeats. The writing performance of the ER can be increased by increasing the number of writing threads.

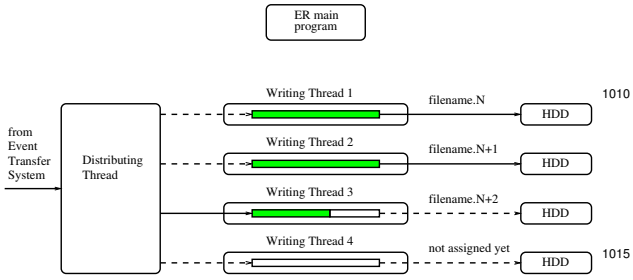

Figure 28: Event Recorder (ER) Diagram. The ER program typically runs on a multi-core Linux server. It receives events from the Event Transfer system and writes them to the disk. It is possible to write several files in parallel to increase writing performance; event order is preserved in a process.

5.8. Messaging System (ActiveMQ)

The messaging system in the CLAS12 DAQ is based on the ActiveMQ library and its C++ extension. Two ActiveMQ servers are used to route all communiations. The number of connections to ActiveMQ is several hundred, and the number of messages sent every second is several tens of thousands, with data volumes of a few tens of MBytes per second. The messaging system is used in particular to monitor and control the DAQ components, in addition to the Run Control process.

5.9. Runtime Database (RCDB)

A JLab-designed MYSQL-based Runtime Database (RCDB) is used to store the run parameters and statistics. It has a web interface (see Fig. 29) and provides an interface to various languages including C++ and Java.

5.10. Online Data Monitoring

The CLAS12 online data monitoring is the set of programs attached to the Event Transfer system that processes data in real time. One of such output of the program is shown in Fig. 30. The monitoring system allows shift takers to insert histograms into the electronic logbook. Every CLAS12 detector has a designated set of histograms that is closely monitored by shift takers and detector experts to ensure data quality.

5.11. Electronic Logbook

The CLAS12 shift personal use the JLab Electronic Logbook system [25]. This web-based system provides an interface to browse, search, and create electronic logbook entries. The system has been built using the Drupal content management system and utilizes a MySQL back-end database to store its entries. The system has been designed to consolidate and replace the functionality of the multiple electronic logbook systems that existed at Jefferson Lab between 1996 and 2012 .

5.12. ROOT for DAQ (FT)

A ROOT-based system was developed to display integrated quantities from the CLAS trigger system in the form of 1D and 2D histograms. The system is made by three different applications: a histogram sender running on each VTP, a histogram receiver running on a DAQ server, and a user-configurable GUI client. Each histogram sender application defines a set of 1D and 2D histograms to report trigger data specific to the CLAS12 subsytem handled by the VTP it is running on. Histograms are refreshed at a fixed rate and streamed to the DAQ server in the form of JavaScript Object Notation (JSON) messages, exploiting the previously described ActiveMQ infrastructure. Each message contains: the histogram name, the number of bins, and a data array with the number of counts in each bin. The message receiver application is responsible for decoding these messages, and of creating ROOT histograms from them. Finally, the client application displays ROOT histograms to the user through a customizable GUI. The GUI is composed of a programmable number of independent frames, with different histograms in each of them. Typically, each frame contains histograms related to the same CLAS12 subsystem. The GUI structure is specified through a configuration file passed as a command-line option when running the client. Communication between the message receiver/ROOT histogram produced application and the user client is handled through the ROOT TSocket mechanism. Fig. 31 shows a GUI reporting histograms from the Forward Tagger system [21]: two 2D histograms showing the distribution of electromagnetic cluster seed hits in the Forward Tagger Calorimeter, and two 1D histograms showing the electromagnetic cluster energy distribution. The right (left) column reports histograms for electromagnetic clusters with (without) a matching hit in the Forward Tagger Hodoscope.

5.13. CLAS Event Display

The CLAS12 Event Display (CED) is a full-function graphical application that displays online (and offline)

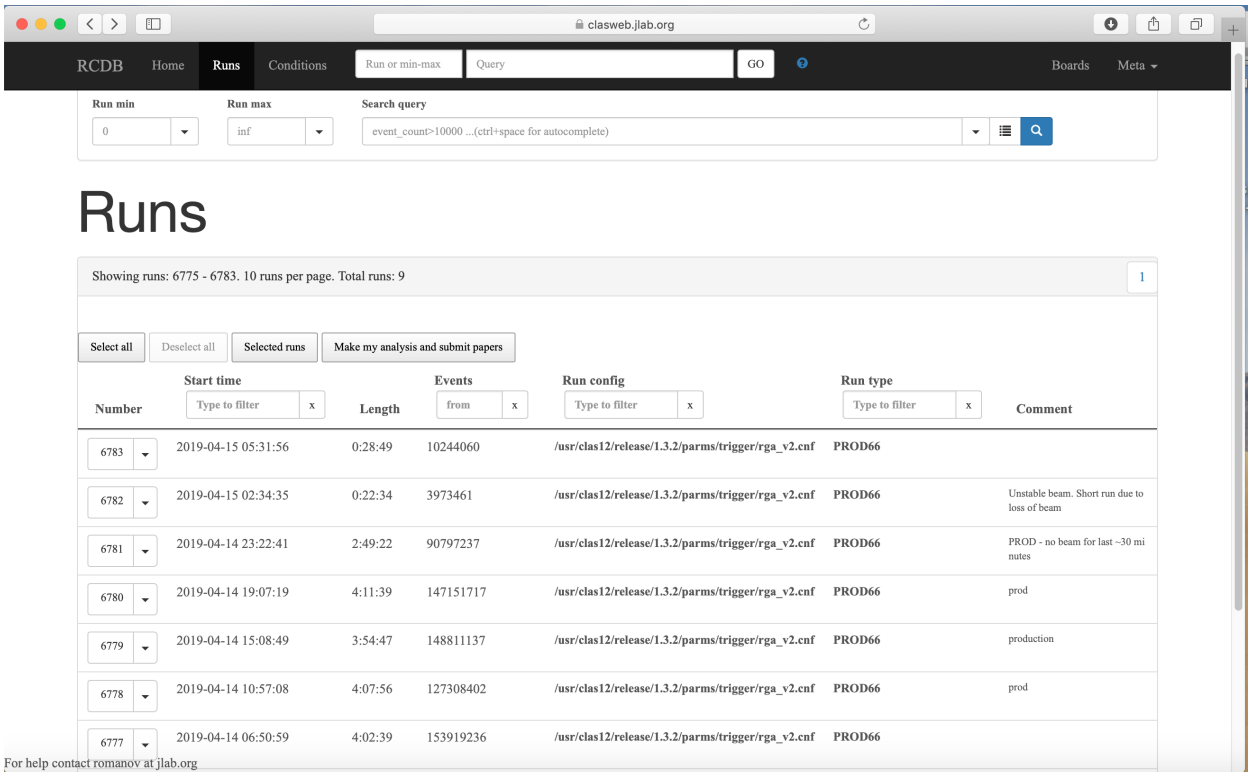

Figure 29: Runtime Database Web Browser example that allows users to search for run(s) with certain parameters.

events using various representations of CLAS12 called views. The views are independent windows that the user can pan, zoom, scroll, etc. Some of the views are geometrically faithful, and some are designed for maximal information content as opposed to realism. The primary purpose and utility of CED, when used online as part of the DAQ system, is for additional data monitoring. While running, CED will display an event from the live stream at a selectable rate, typically one every two seconds. A quick glance at CED will confirm, for example, that there are data in the drift chambers that appear to form tracks. In this way it serves as an early warning of problems with detectors and/or the data stream. It is also possible to operate CED in a mode where it creates graphical histograms or occupancy overlays. A typical view from CED is shown in Fig. 32.

6. Network

The CLAS12 network is shown in Fig. 33. Its main component is an Arista router that serves as the backbone for the entire system. The set of primary DAQ servers is connected directly to the router by 40 Gb links. Some front-end components with particularly high data rates are connected directly to the router by 10 Gb links. Most of the front-end components, as well as the workstations, are connected to multiple "leaf" network switches using 1 Gb links. These leaf switches are connected to the router by 10 Gb links. Two 40 Gb uplinks connect the entire CLAS12 system to the JLab Computer Center where data are sent for permanent storgage to tape.

Most of the 1 Gb links use standard CAT6 copper cables, while the 10 Gb and 40 Gb links use optical fibers, with the exception of the shorter range server links where SFP and QSFP passive copper twinax cables are used.

The CLAS12 network shows adequate performance and a high level of reliability. With the projected CLAS12 data rates, it can be used "as is" for the foreseeable future.

7. Slow Controls

The CLAS12 slow controls system was heavily upgraded for the 12 GeV era at JLab. It incorporates legacy and modern hardware and standalone controls

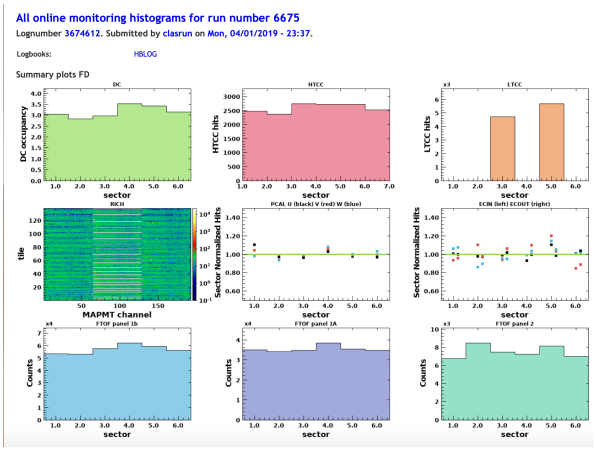

Figure 30: Online monitoring example. Only a small portion of the histograms is shown. The full set includes hundreds of histograms used by shift takers to monitor various detectors and subsystems.

systems into full EPICS integration, such that everything is accessible from a single interface and on any computer in the CLAS12 system. Another important aspect is leveraging standard infrastructure tools supported by central JLab computing resources, for a manageable and reliable controls system.

The CLAS12 slow controls system is based on Experimental Physics Industrial Control System (EPICS), currently version 3.14.12.5 [24], and includes about 100 EPICS input-output controllers (IOCs) interfacing with approximately 50 different types of hardware via various communication protocols and over 800K process variables (PVs).

The controls system monitors and controls all aspects of the CLAS12 detector, beamline, and magnet systems, with monitoring on the DAQ system. This includes power supplies from a variety of manufacturers, programmable logic controllers, cryogenic and gas systems, multiple scaler hardware, flasher sytems, all of the VXS crates, trigger system performance, and beamline motors, with sequencing for more complex operations like polarimetry and magnet hystereses. An example of the superconducting torus magnet nitrogen system and 10 kHz EPICS monitoring is shown in Fig. 34, and one of the scaler displays covering multiple CLAS12 detector systems is shown in Fig. 35.

Most of the IOCs run on standard rack-mounted servers running Red Hat Enterprise 7 (RHEL7) in the Control Room. A few IOCs run in more specialized systems in the experimental hall, such as VxWorks 5.X real-time operating systems on Motorolla VME controllers, primarily for high-rate, synchronous beam monitoring and support of legacy components. All IOCs and associated processes are managed with procServ for interactive access when necessary and cronjobs for automatic startup and recovery [26].

For the graphical user interface we chose the modern Control Systems Studio (CS-Studio) developed at Oak Ridge National Laboratory [27]. CS-Studio is an Eclipse-based suite of tools for developing and monitoring large-scale control systems. It includes various features supporting quick interface development, such as templating and dynamic generation.

We also use the CS-Studio alarm system. It is based on a MYSQL database for storing alarm configurations, status, and message history, combined with a server monitoring the IOCs, updating the database, and communicating with clients. The clients include a graphical interface tightly integrated with the rest of the controls system, with visible feedback and heirarchical view of the alarm system, an annuciator service running in the background in the Control Room, and a notifier service sending emails to on-call experts on particular alarm conditions.

Controls systems interfaces are all run locally on any desktop PC running RHEL7 in the Counting House. Full remote access is also provided via 2-factor authentication and X-forwarding or VNC, as well as VMWare virtual machines supported by JLab. Read-only access in a web browser is provided via WebOPI, another product affiliated with CS-Studio.

We utilize two internal EPICS channel access gateways, one read-only for the web interface, and the other for minimizing connections to superconducting magnet controls systems. Wherever appropriate, the standard autosave feature of EPICS is utilized to automatically preserve settings across IOC reboots, and the standard Burt save/restore mechanisms. The controls system also utilizes many custom hardware and software interlocks.

An important component of the slow controls system is archiving data. For this we utilize the EPICS archiving system called Mya, a product developed by and in support of accelerator operations at JLab and shared with the experimental halls. This provides storage and access to many previous years data of all CLAS12 EPICS PVs [28].

Installation and configuration of the associated desktop and server machines, including custom service daemons, cron jobs, IOCs, network disk mounting, and deployment of upgrades and software changes, is managed via puppet and a central server maintained by JLab [29]. The resulting maintainence is almost hands-free, and recovery from hardware failures or power outages is largely automated. Monitoring of the servers is performed with Nagios, also provided by JLab, which pro-

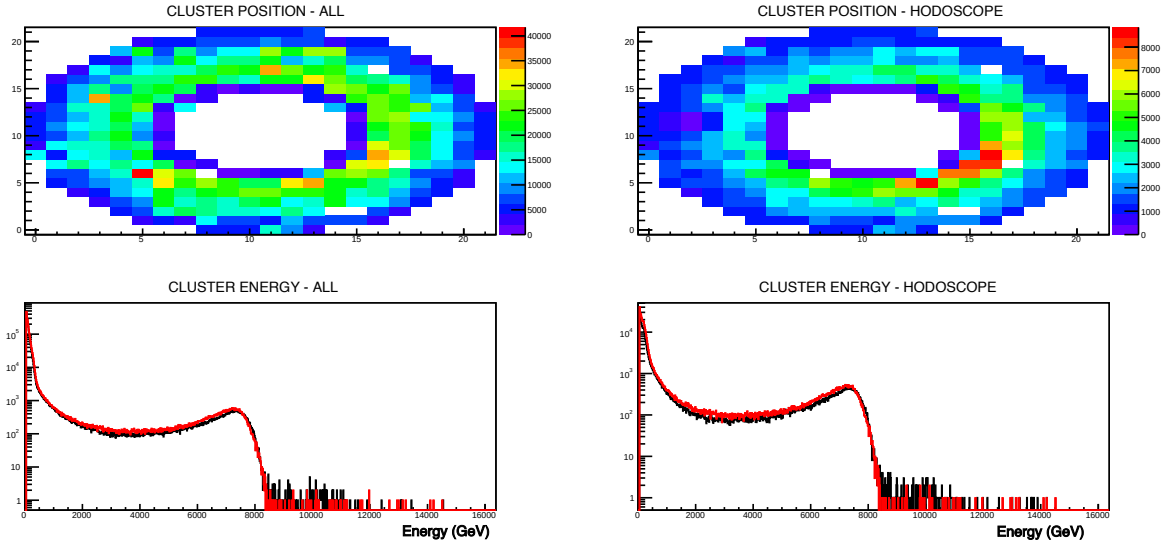

Figure 31: Example of a ROOT-based GUI to monitor trigger data for the specific case of the Forward Tagger detector. Top histograms report the distribution of electromagnetic cluster seed hit positions, while the bottom histograms report the electromagnetic cluster energy distribution. Right (left) column reports histograms for electromagnetic clusters with (without) a matching hit in the Forward Tagger Hodoscope.

vides email notifications about system errors [30].

8. Performance

The CLAS12 DAQ performance was adequate during the first year of running. Thanks to high trigger purity, the event rate never exceeded 20 kHz, which was easily handled by the DAQ with livetime above 93%. The data rate remains below 1000 MByte/sec (see Fig. 24).

Several performance tests were conducted to check the data rate limits of the back-end components. With the front-end and Event Builder excluded, the Event Transfer and Event Recorder showed a data rate exceeding 2 Gbyte/sec on a single Dell R730 server. This demonstrates that the CLAS12 DAQ data rate limitation is much higher than required for all anticipated CLAS12 experiments.

9. Conclusion

The work on the CLAS12 DAQ system started in 2008. The system was designed and implemented in the years from 2008 to 2017. The system has been successfully used during the phase of development, testing, and commissioning of all of the CLAS12 detectors. In December 2017, the CLAS12 DAQ was ready for the first beam experiment and has been in full operation mode since that time. The performance achieved by the CLAS12 DAQ allows it to be used without significant changes for the entire CLAS12 physics program.

10. Acknowledgements

We are grateful to the administrative, engineering, and technical staff of JLab for constant support. In particular, we appreciate the hard work of the JLab Fast Electronics Group technical personal: Mark Taylor, Armen Stepanyan, Jeff Wilson, and William Gunning. Our special thanks to the CLAS12 project leaders Volker Burkert and Latifa Elouadrhiri for leadership and setting goals. This work was supported in part by DOE Contract DE-AC05-84ER40150.

[1] V. Burkert, et al., The CLAS12 Detector, see this issue.
[2] B.Raydo, et al., The CLAS12 Trigger System, see this issue.
[3] Hyperlynx - PCB Analysis and verification. URL `https://www.mentor.com/pcb/hyperlynx`
[4] B. A. Mecking, et al., CLAS Detector, Nucl. Inst. and Meth. **A503**, 513 (2003).
[5] J. Gu, et al., The TRIGGER/CLOCK/SYNC Distribution for TJNAF 12 GeV Upgrade Experiments. URL `https://coda.jlab.org/wiki/index.php/Trigger_distribution_overview(2014,May)`
[6] J. Gu, et al., Description and technical information for the Trigger Supervisor (TS) module. (TJNAF, VA, 2013). URL `https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/TS/TS.pdf`
[7] Fast Electronics Group, Jefferson Lab, 2010, The SD Board. URL `https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/SD/SD_Description_VerA5_Updated_Nov13.pdf`
[8] J. Gu, et al., Design of the Trigger Interface and Distribution Board for TJNAF 12 GeV Upgrade, IEEE Trans. Nucl. Oct. 2013.
[9] F.J.Barbosa, et al., A VME64x 16-channel, Pipelined, 250MSPS

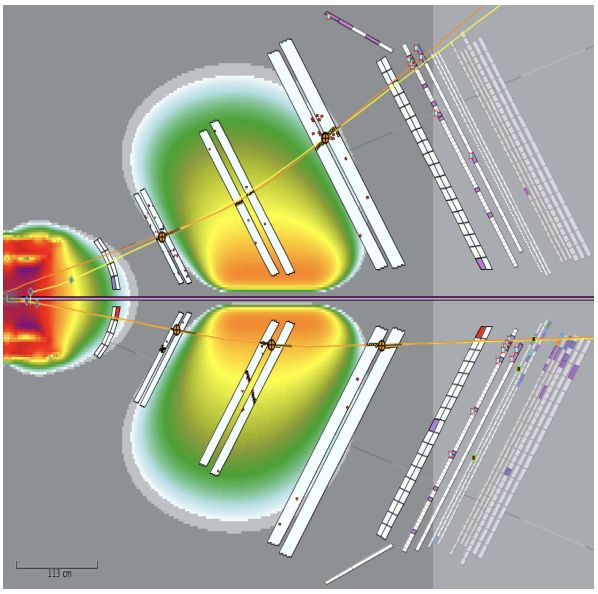

Figure 32: CED event example that includes a schematic view of all detectors with a realistic geometry.

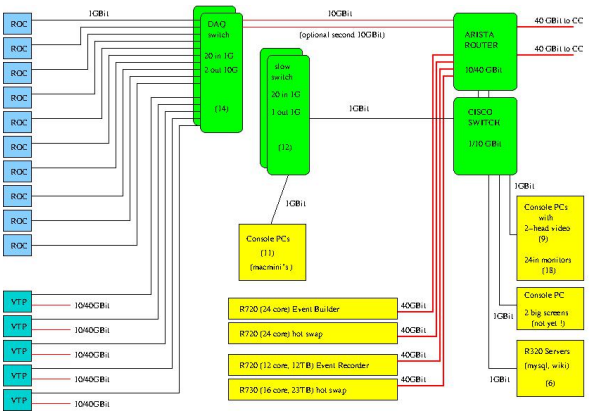

Figure 33: CLAS12 DAQ Network Diagram. All components and connections have adequate performance for the currently running and planned experimenmts.

Flash ADC With Switched Serial (VXS) Extensions, IEE-NSS 2007, Hawaii.

[10] Fast Electronics Group, Jefferson Lab, Feb 11th, 2011, The DSC2 Board. URL https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/vmeDSC/VME16Chan_DiscScaler_Specifications_revD.pdf

[11] CAEN, Italy, The TDC Boards. URL https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/CAEN/V1190_REV10.pdf(Revision10,30August2011),https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/CAEN/V1290_REV12.pdf(Revision12,30August2011)

[12] E.Jastrzembski, et al., CAEN TDC INL correction.

[13] Sergey Boyarinov, Chris Cuevas, Benjamin Raydo, The DCRB Board. URL https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/DCRB/DCRB_Manual_RevA.pdf

[14] M. Mestayer, et al., The CLAS12 Drift Chamber System, see this issue.

[15] M. A. Antonioli, et al., The CLAS12 Silicon Vertex Tracker, see this issue.

[16] Chris Cuevas, Benjamin Raydo, 12 December 2011, The VSCM Board. URL https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/VSCM/D00000-16-08-S004-VSCM_Module.pdf

[17] Benjamin Raydo, 14 January 2014, The SSP Board. URL https://coda.jlab.org/drupal/system/files/pdfs/HardwareManual/SSP/SSP_Module_HallD_v1.2.pdf

[18] M. Contalbrigo, et al., The CLAS12 RICH Detector, see this issue.

[19] A. Acker, et al., The CLAS12 Micromegas Vertex Tracker, see this issue.

[20] M. Bashkanov, et al., The Forward Tagger for CLAS12, see this issue.

[21] M. Bashkanov, et al., The Forward Tagger for CLAS12, see this issue.

[22] C. Flouzat, et al., Dream: a 64-channel Front-end Chip with Analog Trigger Latency Buffer for the Micromegas Tracker of the CLAS12 Experiment.

[23] JLab CODA Group, The CODA Data Acquisition System. URL https://coda.jlab.org

[24] Experimental industrial physics control system. URL https://epics-controls.org

[25] , The JLab Electronic Logbook. URL https://logbooks.jlab.org

[26] procserv. URL https://github.com/ralphlange/procServ

[27] Control Systems Studio. URL http://controlsystemstudio.org

[28] C. J. Slominski, A MySQL Based EPICS Archiver, Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems. URL http://accelconf.web.cern.ch/AccelConf/icalepcs2009/papers/wep021.pdf

[29] Puppet. URL https://puppet.com

[30] Nagios. URL https://www.nagios.org

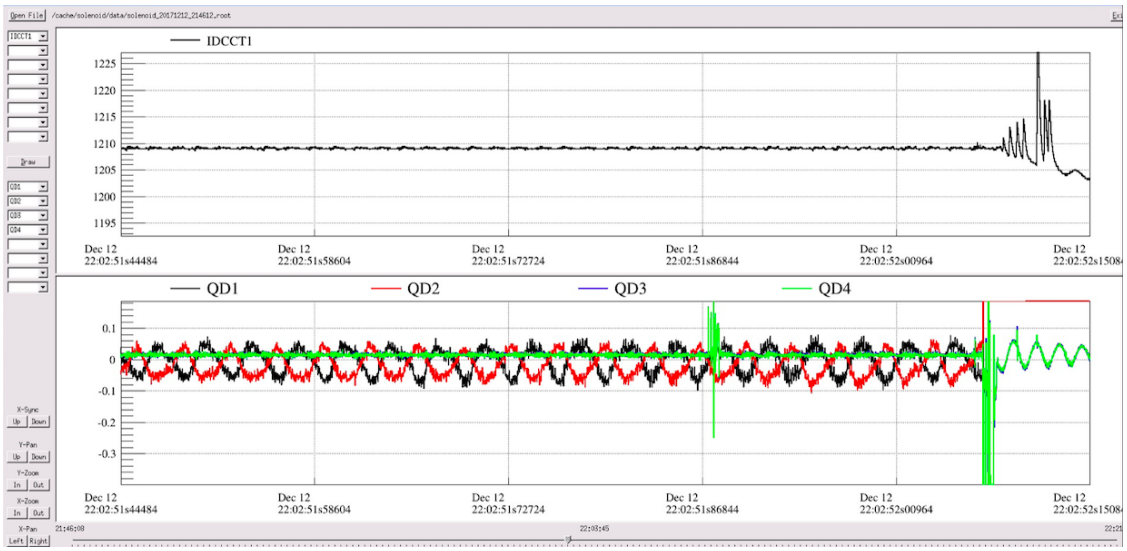

Figure 34: The 10 kHz EPICS-based readout of the superconducting magnet quench detection system.

Figure 35: An example of the JLab FADC250 scalers for the CLAS12 PMT-based detector systems.