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]

[a]*Thomas Jefferson National Accelerator Facility, Newport News, VA, USA*
[b]*Sacley, France*
[c]*INFN, Milan, Italy*

Abstract

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

1. Overview

The CLAS12 Data Aqcuisition System (DAQ) collects data from all CLAS12 detector components. It is a network-based system collecting data from more then 100 front-end components that comprise the trigger system signals, assembling the component data into events, monitoring data quality, and recording data to tape. Typical event rates are 15-20 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 was used successfully during the first year of CLAS12 operations and is projected to be used for the entire CLAS12 life span with only minimal changes.

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 upgraded 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 20 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 pipeline-style, 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. Currently, the front-end components used are commercial VME/VXS crates, commercial Linux servers, and Jefferson Laboratory(JLAB)-designed VXS trigger processor boards. VTPs are installed in all of the VXS crates, but are read out by the DAQ as independent components. All components are running on a 250 MHz clock distributed over OM3-rated parallel optic fibers. The same fiber system is used to distribute thw synchronization reset and trigger signals, and to collect the busy signals from all front-end electronics.

All front-end components are connected to the Event Builder (EB), which is a multi-threaded program running on a multi-core Linux server. Most connections are 1 Gbit ethernet links. For several components that generate significant data rate, 10 Gbit ethernet connection is used.

Complete events are sent to the Event Transfer (ET) System. This multi-threaded program provides a data ring where various data processing programs can be attached to monitor data quality online. It can be run on

the same server as the EB, or can be used to create a sequential chain of servers to increase the 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, that 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.

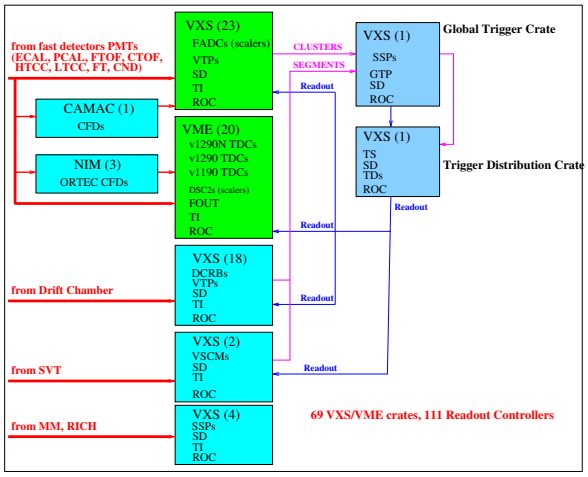

Figure 1: Diagram of the CLAS12 Data Aquisition System

4. Hardware Components

New instrumentation modules have been designed by JLab that 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 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 back-plane configuration is used, and one switch slot is used for the trigger processing and one switch slot is used to distribute the essential timing and synchronization signals to each of the front-end boards.

The trigger processor switch slot board 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 are manufactured by WIENER and the back-plane can support up to 8 Gb/s. The payload boards use Xilinx Virtex V technology, and these FPGAs have up to 6.25 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 trigger processor module.

The design challenges for reliable and successful transmission of gigabit serial data over the VXS backplane 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 circuit board routing simulation tools such as Mentor Graphics HyperLynx [**??**, 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 hardware 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.

4.1. Fiber Optic Trigger distribution system

As shown in Fig. 2, the digital sum value 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.

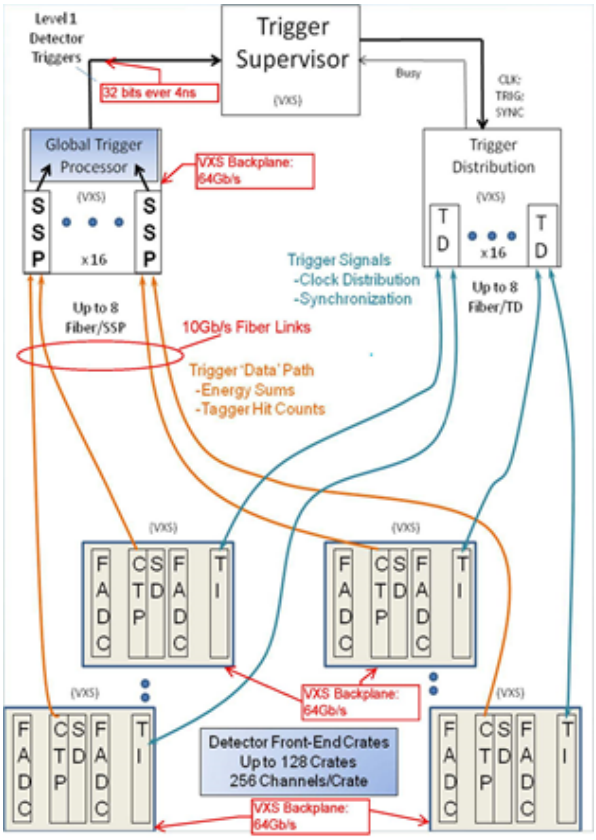

Figure 2: Hardware diagram with the implemented fiber links scheme.

4.2. VXS/VME crates

Previous experiments with the original CLAS spectrometer (see [1]) 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 acquired with FastBus ADC and TDC 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 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 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.

4.3. 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 100 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 an specified Linux operating system.

We also expect a certain minimum 10 year timeframe in which these controllers will be supported by the vendor with respect to the avilable parts, repair, and software updates. VME controller requirements are summerized in following list:

- Single slot VME form-factor - no required rear transition module
- Intel Core i7 dual or quad core embedded processer 2 GHz (or greater)
- Hyperthreading and 64 bit arch support
- 4 GBytes 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 XVR16 Intel forth Generation 4-core i7-based rugged VME single board computers from GE (see Fig. 3). They were installed in the 70+ crates for CLAS12 and have demonstrated excellent performance and reliability. Most of the controllers send data over their built-in 1 GBit link, while for few of them, a 10 GBit daughter board was installed to increase the bandwidth. The maximum data rate from a single crate in CLAS12 never exceeds 130 MB/s, and with that rate 4-core controllers are able to handle the VME data polling, data processing, and sending data over the network without any issues.

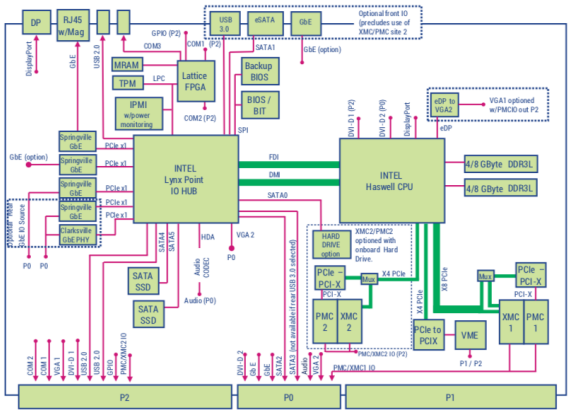

Figure 3: Block diagram of the XVR16 VME crate controller

4.4. *Trigger Distribution System Modules (TS, TD, TI)*

The TCS (TRIGGER, CLOCK, SYNC, and BUSY) distribution system [2] 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 frontend electronics' status (BUSY) and makes sure of the smooth data readout of the experiments. Fig. 4 shows a diagram of the trigger and TCS distribution.

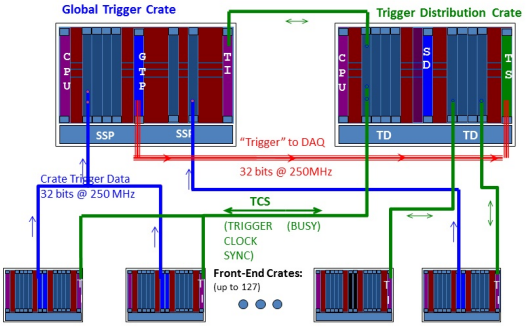

Figure 4: Diagram of the trigger and TCS distribution

The main hardware of the TCS distribution system includes a Trigger Supervisor (TS [5]) board (see Fig. 5 and 6), Signal Distribution (SD [6]) boards, Trigger Distribution (TD [4]) boards (see Fig. 7), Trigger Interface (TI [3]) boards (see Fig. 8 and 9), 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 and high-speed differential backplane connections are used to transmit signals at high speed over long distances.

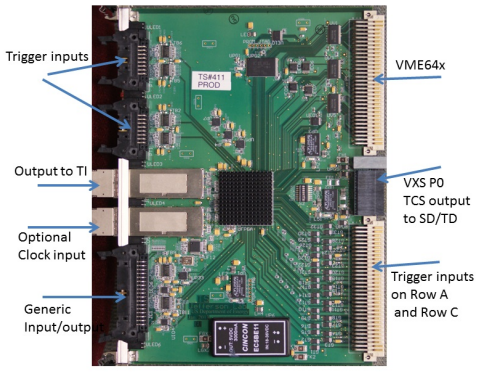

Figure 5: TS board

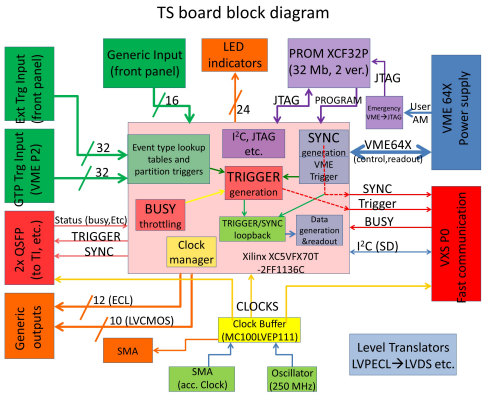

Figure 6: TS board diagram

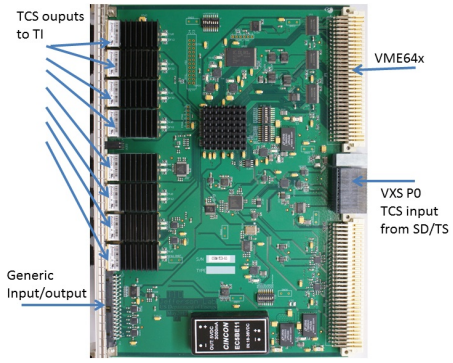

Figure 7: TD board

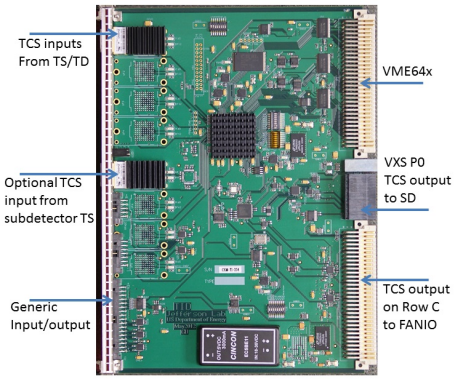

Figure 8: TI board

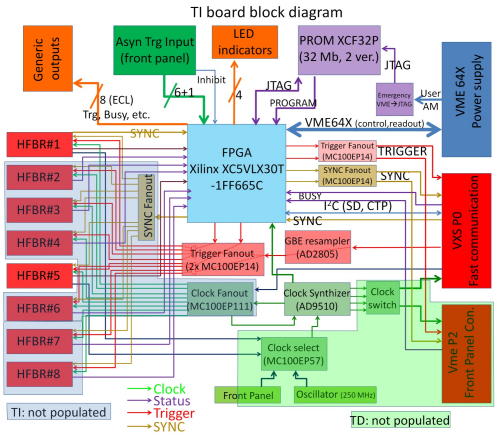

Figure 9: TI board diagram

4.4.1. *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 fronts 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 (TDC, ADC). 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 realignment command. The clock jitter is about one ps measured at the front-end electronics. The clock distribution skew can be adjusted by the SD clock delay if necessary.

4.4.2. *SYNC Distribution*

The TS generates and distributes the SYNC signal. The SYNC is an encoded 4-bit serialized command 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 '1's 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

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

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 FPGA's 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 meter fibres), the measurement on these two fibres can be used as the latency of the other fibres in the cable. After the SYNC latency compensation, all 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.

4.4.3. 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 **??**.

Both the fiber latency and trigger word serializer/ deserializer are compensated so that all 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. 10 shows the diagram of the compensated trigger distribution.

On the TI board, the deserialized trigger word is clocked into and clocked out of a FIFO using the

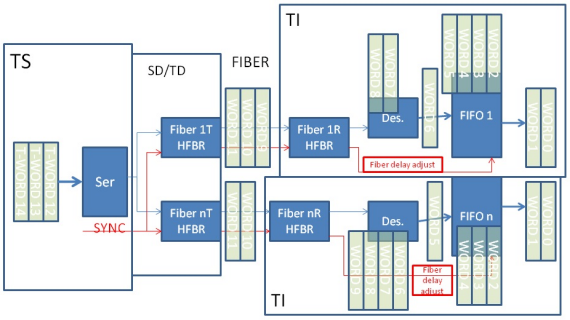

Figure 10: Trigger synchronization between TIs

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 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, so does the readout trigger. The clock phase can be adjusted by SD if the skew is critical to the system.

4.4.4. 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. 11 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).

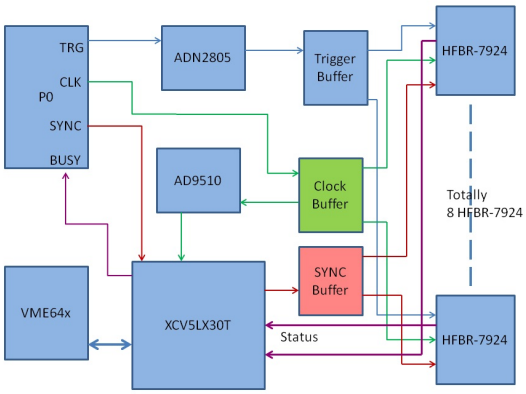

Figure 11: DAQ synchronization

4.5. Signal Distribution Module (SD)

The Signal Distribution Board (SD [6]) module (see Fig. 12) 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 (ADC 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.

4.6. Flash ADC Module (FADC250)

A 16-channel 250 MSPS pipelined flash ADC (FADC [7], see Fig. 13) with 12-bit precision was designed to digitize and process detector pulses for experiments at JLab. The FADC250 module (see Fig. 14) 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

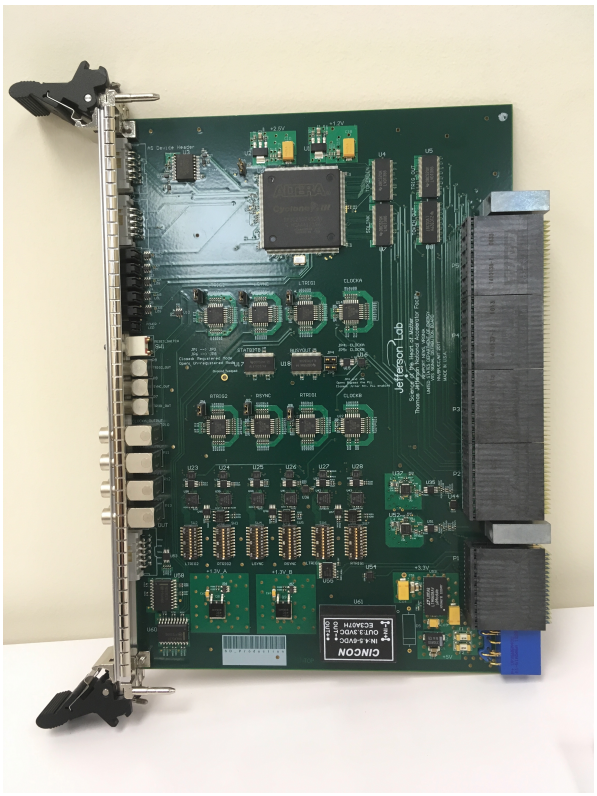

Figure 12: Signal Distribution module (SD)

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 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

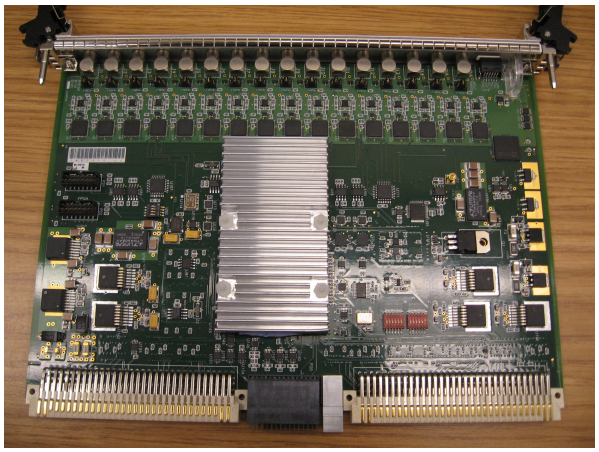

Figure 13: Flash ADC module (FADC250)

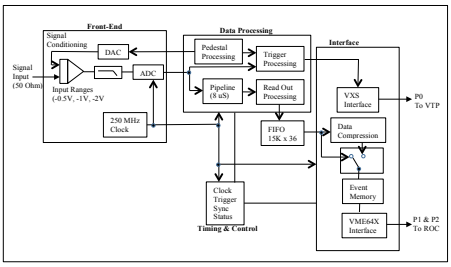

Figure 14: Flash ADC module diagram (FADC250)

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 read out 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 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.

4.7. Discriminator Scaler Module (DSC2)

The Discriminator Scaler Module (DSC2 [8], see Fig. 15) 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.

DSC2 Specifications

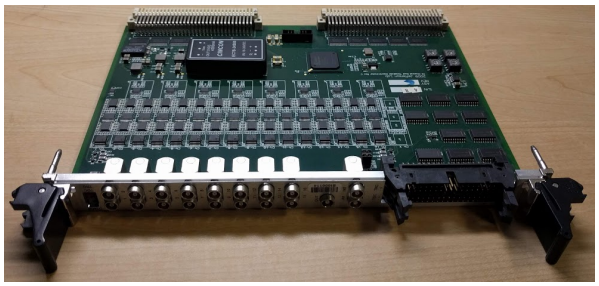

Figure 15: Discriminator Scaler module (DSC2)

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 us
Digital width maximum	1 us
Maximum count rate	125 MHz

Discriminator. Inputs are single-ended LEMO and leading-edge discrimated by two different thresholds. 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. 16 shows the typical jitter as a function of a variety of input slew rate signals and threshold overdrive conditions.

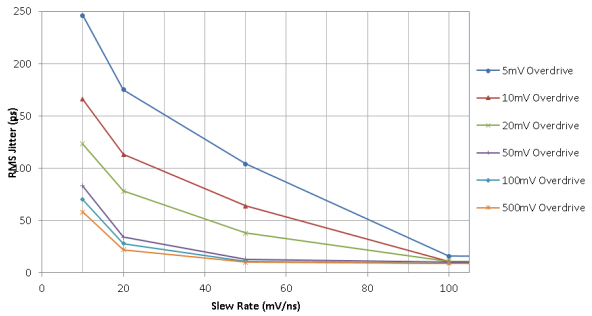

Figure 16: Output Jitter vs Input slew rate and overdrive

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 computing 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 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.

4.8. TDC Modiles (v1190/v1290)

Commercial CAEN V1190/V1290/V1290N TDC [11] boards are used for timing measurements in the PMT-based CLAS12 detectors (see Fig. 17). 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 then internal 40 MHz clock. The use of a different clock required compensation table remeasuring and reloading.

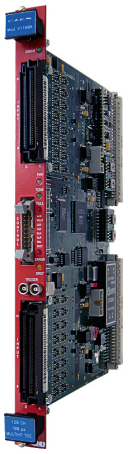

Figure 17: CAEN V1190 TDC module (V1190)

4.9. Drift Chamber Readout Board (DCRB)

The Drift Chamber Readout Board (DCRB [9], see Fig. 18) is a 96-channel amplifier, discriminator, and time-to-digital converter module used to digitize and readout hits from the CLAS12 Drift Chambers. A single VXS crate of DCRB modules can readout a full region

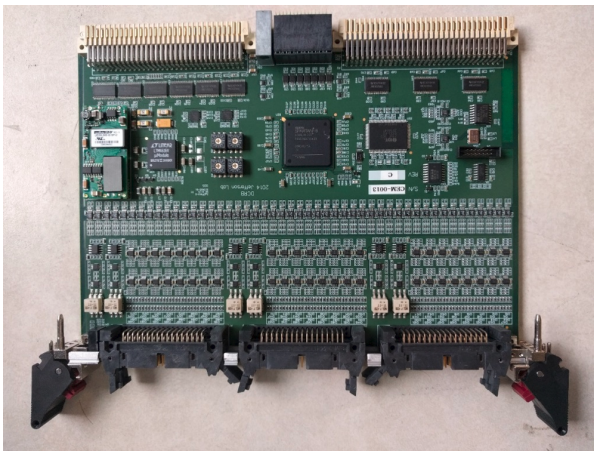

Figure 18: Drift Chamber Readout Board (DCRB)

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 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 of 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 then 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.

4.10. VXS Silicon Readout Module (VSCM)

The CLAS12 Silicon Vertex Tracker detector (SVT, [14]) front-end utilizes the data driven FSSR2 ASIC for digitization. The VXS Silicon Readout Module (VSCM [10], Fig. 19) 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.

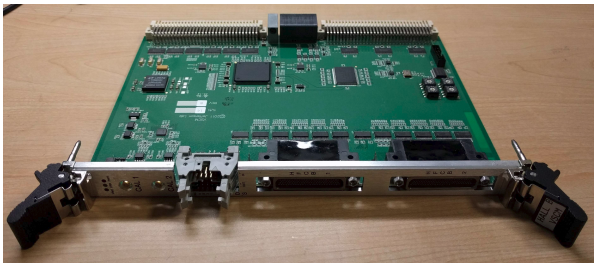

Figure 19: VXS Silicon Readout Module (VSCM)

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

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.

Low time resolution hit word

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

High time resolution hit word

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

Fig. 20 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.

4.11. SSP Board as ffber Readout Module

The SubSystem Processor (SSP [12]) was originally designed to work as part of CLAS12 Trigger System. It is used to readout some front-end electronics in the DAQ system as well. The board description can be found in the CLAS12 Trigger System article [15].

5. Software

5.1. CODA DAQ Software

The CLAS12 DAQ system is based on CODA ([13], see Fig. 21), short for CEBAF Online Data Acquisition.

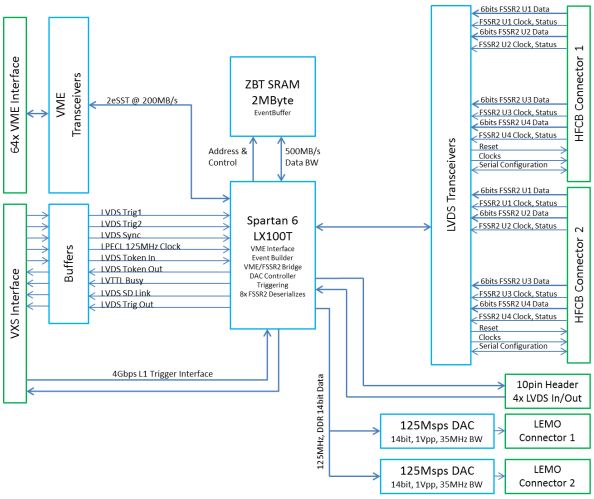

Figure 20: VSCM Hardware Diagram

It is a kit of parts 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 hardware components along with complementary software components.

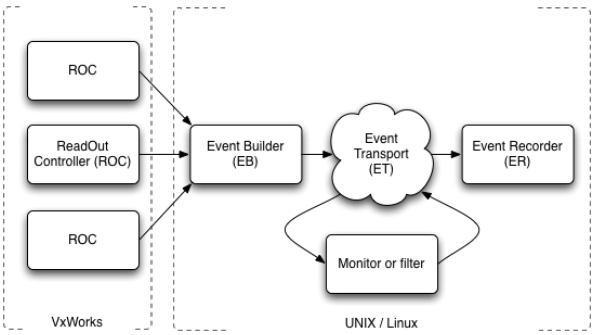

Figure 21: CODA System Diagram

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 that participate in the run. The latter are defined in run configuration files that the operator chooses at startup. The Run Control Graphic User Interface (see Fig. 22) 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.

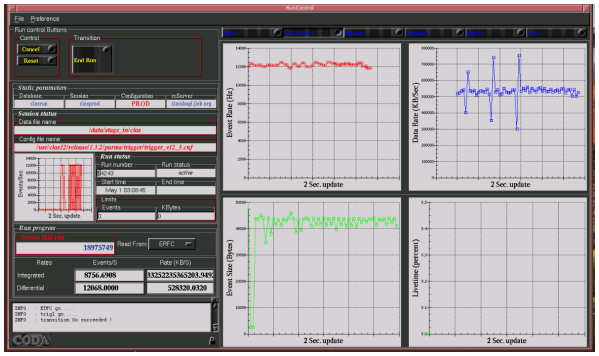

Figure 22: Runcontrol GUI

5.3. Front-End Libraries

The front-end libraries were developed mostly in JLab.

Just an outline ATM

GEFANUC driver:

- Customized Kernel Driver and Userspace Interface
- C API provides:
  - Setup of VME inbound and outbound windows
    - Permanent windows: CRCSR, A16, A24, A32
    - Map of these windows into userspace
  - Allocate of Physical Memory for DMA
    - Map of this memory into userspace

JVME Userspace Driver:

- C API provides
  - common userspace interface to GEFANUC driver and others.
  - Initialization of kernel driver and default VME windows
  - Maps VME bridge registers into userspace
    - 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. 23) software component is a 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 the DAQ startup, the ROC main program starts three threads, after that it just controls thread health and communicates with the run control process. Three threads (readout, processing, and network) pass data from one to another, communicating over circular buffers. The typical number of buffers in each circle is 8 and the size of every buffer is 4 MBytes, which defines how long the front-end electronics will be read out before feeling the effect of the 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 pooling mode which occupies an entire CPU core, or in interrupt mode. The CLAS12 readout primarely employs pooling 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 and data sanity checks. The results are placed into the second circular buffer. That component can create its own threads to increase processing power. The third thread (network) reads data from the second circular buffer and sends it over the network to the Event Builder.

The first and second threads have a user part that is compiled separately and downloaded dynamically, which allows users to develop experiment-dependent code without recompiling the CODA framework.

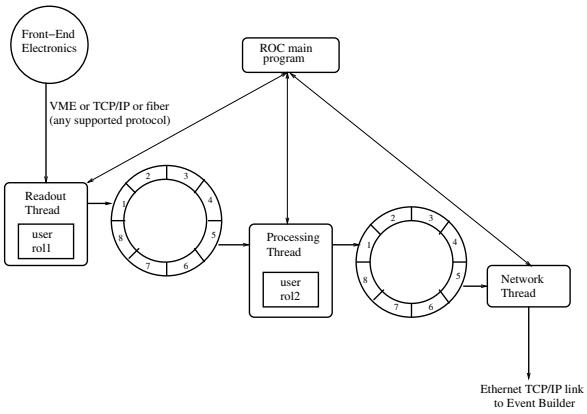

Figure 23: Readout Controller Diagram

5.5. Event Builder

The Event Builder (EB, see Fig. 24) 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 values 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.

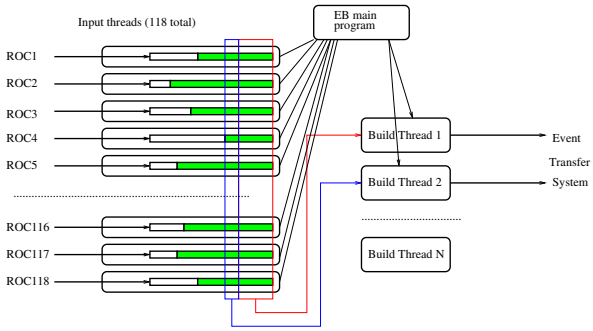

Figure 24: Event Builder Diagram

5.6. Event Transfer

The Event Transfer (ET, see Fig. 25) system provides the user with an efficient method for moving bulk data between processes. The ET was not designed as a messaging system, but it can rather 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. 25, 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.

5.7. Event Recorder

The Event Recorder (ER, see Fig. 26) is the program that receives data from the ET system and records the data files onto disk. The ER can write data in one or several files in parallel (so-called multi-stream mode). If multi-stream mode is used, the event order is preserved,

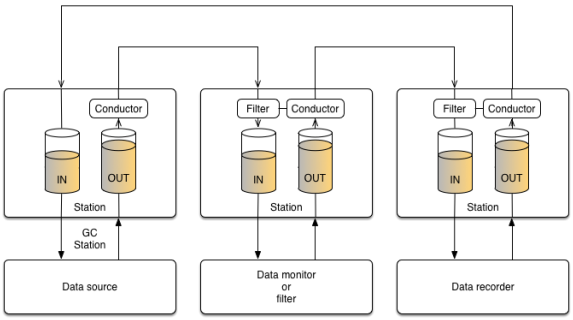

Figure 25: Event Transfer System Diagram

but it requires the allocated memory size to be bigger than the number of streams multiplied by the file size.

The ER structure in multi-stream mode is shown in Fig. 26. 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.

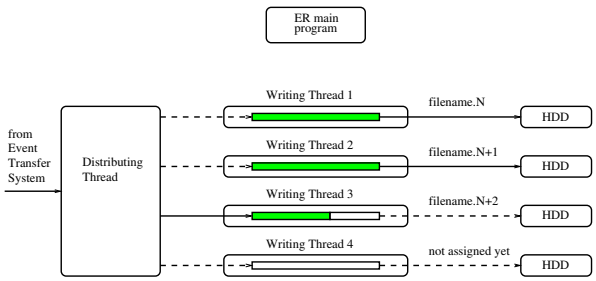

Figure 26: Event Recorder Diagram

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 web interface (see Fig.27) and provides an interface to various languages including C++ and Java.

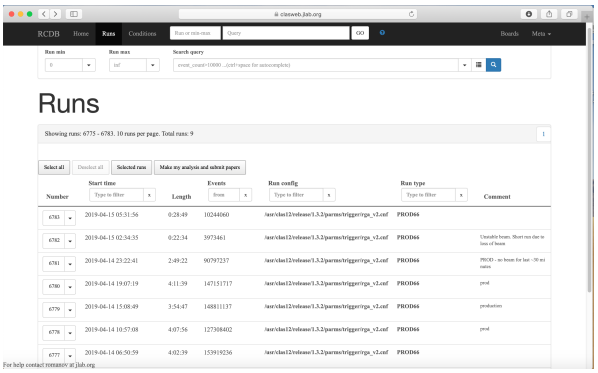

Figure 27: Runtime Database Web Browser example.

5.10. *Online Data Monitoring*

The CLAS12 online data monitoring is the set of programs attached to Event Transfer System and processing data in real time. One of such program's output is shown on Fig.28.

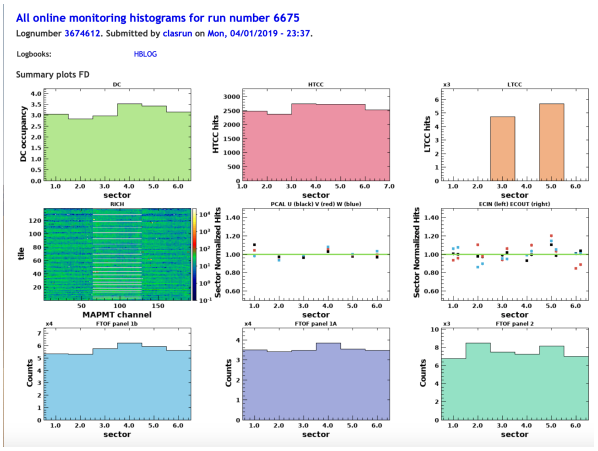

Figure 28: Online monitoring example.

5.11. *ROOT for DAQ (FT)*

A ROOT-based system was developed to display integrated quantities from the CLAS trigger system in forms 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

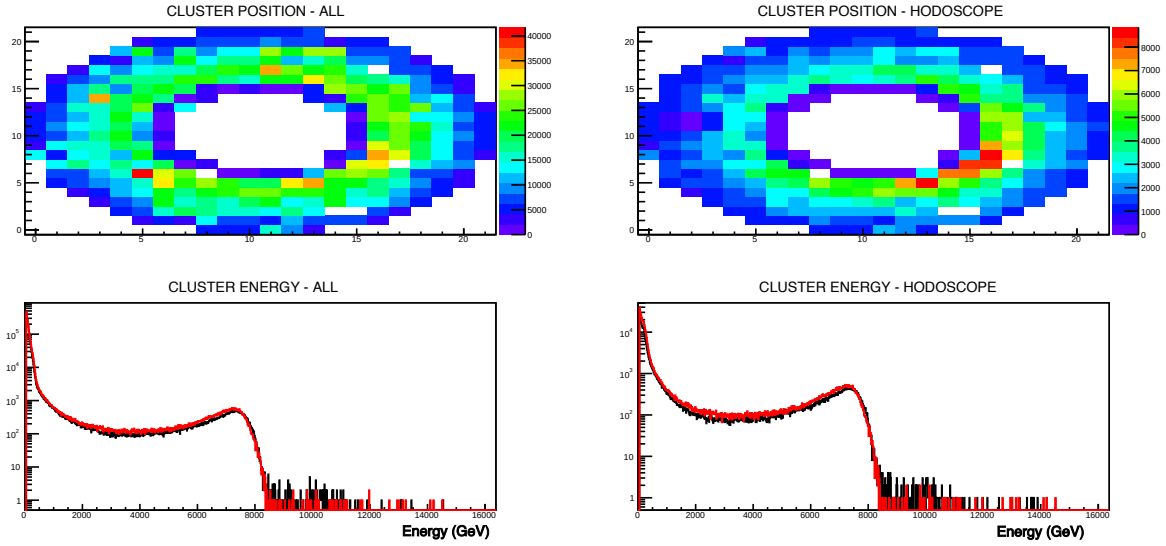

Figure 29: Example of a ROOT-based GUI to monitor trigger data, for the specific case of Forward Tagger detector. Top histograms report the distribution of electromagnetic clusters seed hits positions, while bottom histograms report the electromagnetic clusters energy distribution. Right (left) column report histograms for electromagnetic clusters with (without) a matching hit in the Forward Tagger Hodoscope.

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 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 throuh ROOT TSocket mechanism. Fig. 5.11 shows a GUI reporting histograms from the Forward Tagger system **??**: two 2D histograms showing the distribution of electromagnetic clusters seed hits in the Forward Tagger Calorimeter, and two 1D histograms showing the electromagnetic cluster energy distribution. The right (left) column report histograms for electromagnetic clusters with (without) a matching hit in the Forward Tagger Hodoscope.

5.12. CLAS Event Display

The CLAS12 Event Display (CED) is a full-function graphical application that displays on-line (and off-line) 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 on-line 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. 30.

6. Network

The CLAS12 network is shown in Fig. 31. Its main component is an Arista router that serves as the backbone for the entire system. The set of main DAQ servers

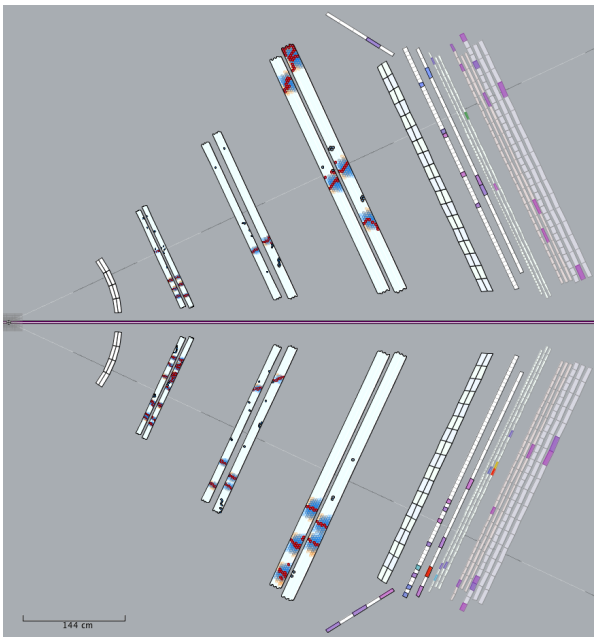

Figure 30: CED Event Example

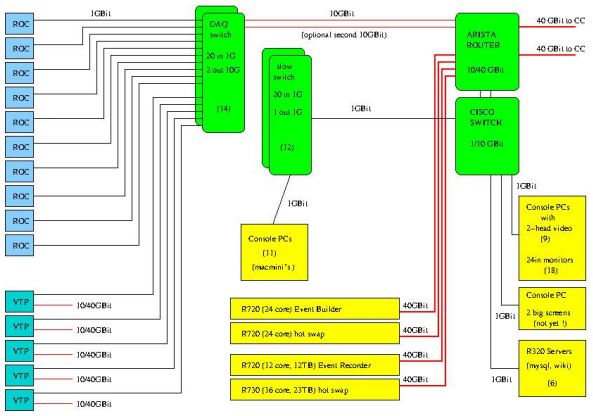

Figure 31: CLAS12 DAQ Network Diagram

is connected directly to the router by 40 Gbit links. Some front-end components with particularly high data rates are connected directly to the router by 10 Gbit links. Most of the front-end conponents, as well as the workstations are connected to the network switches using 1 Gbit links, while those switches are connected to the router by 10 Gbit links. Two 40 Gbit uplinks connect the entire system to the JLab Computer Center.

Most of the 1 Gbit links use copper wiring, while the 10 Gbit and 40 Gbit links use optic fibers, with the exception of short range server links where 40 Gbit wires 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 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 [**?**], and includes about 90 EPICS input-output controllers (IOCs) interfacing with approximately 50 different types of hardware via various communication protocols and 100K(?) 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 hardwares, flasher sytems, all of the VXS crates, trigger system performance, beamline motors, with sequencing for more complex operations like polarimetry and magnet hystereses. An example of the superconducting torus magnet’s nitrogen system and 10 kHz EPICS monitoring is shown in Fig.33, and one of the scaler displays covering multiple CLAS12 detector systems is shown in Fig.32.

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 [**?**].

For the graphical user interface we chose the modern Control Systems Studio (CS-Studio) developed at Oak Ridge National Laboratory [**?**]. CS-Studio is an Eclipse-based suite of tools for developing and monitoring large-scale control systems. It includes various

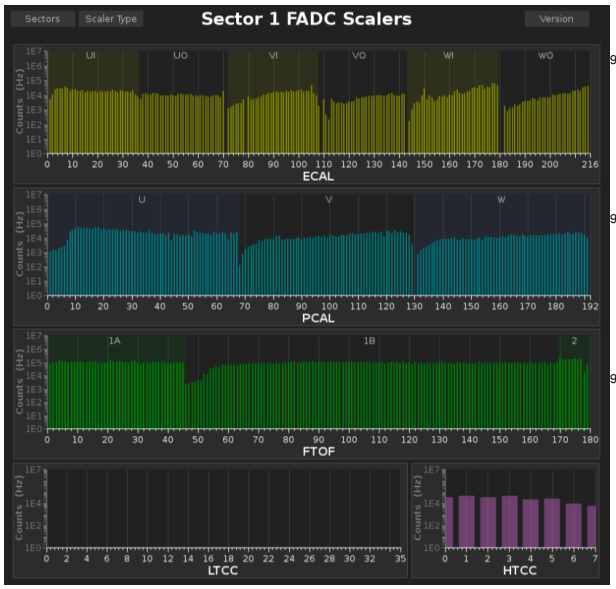

Figure 32: And example of the JLab FADC250 scalers for CLAS12 PMT-based detector systems.

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.

The operators of the controls system run directly on desktop PCs running RHEL7. 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 [**?**].

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 [**?**]. 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 provides email notifications about system errors [**?**].

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%. Data rate remains below 1000 MByte/sec (see Fig. 22).

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. Acknowledgements

We appreciate the contribution of ... We are grateful to administrative, engineering, and technical staff of ...

10. Conclusion

11. References

References

[1] B.A. Mecking *et al.*, Nucl. Inst. and Meth. **A503**, 513 (2003).
[2] XXX *et al.*, *"The YYY Boards"*.
[3] XXX *et al.*, *"The TI Board"*.
[4] XXX *et al.*, *"The TD Board"*.
[5] XXX *et al.*, *"The TS Board"*.
[6] XXX *et al.*, *"The SD Board"*.
[7] XXX *et al.*, *"The FADC Board"*.
[8] XXX *et al.*, *"The DSC2 Board"*.
[9] XXX *et al.*, *"The DCRB Board"*.
[10] XXX *et al.*, *"The VSCM Board"*.
[11] XXX *et al.*, *"The TDC Board"*.

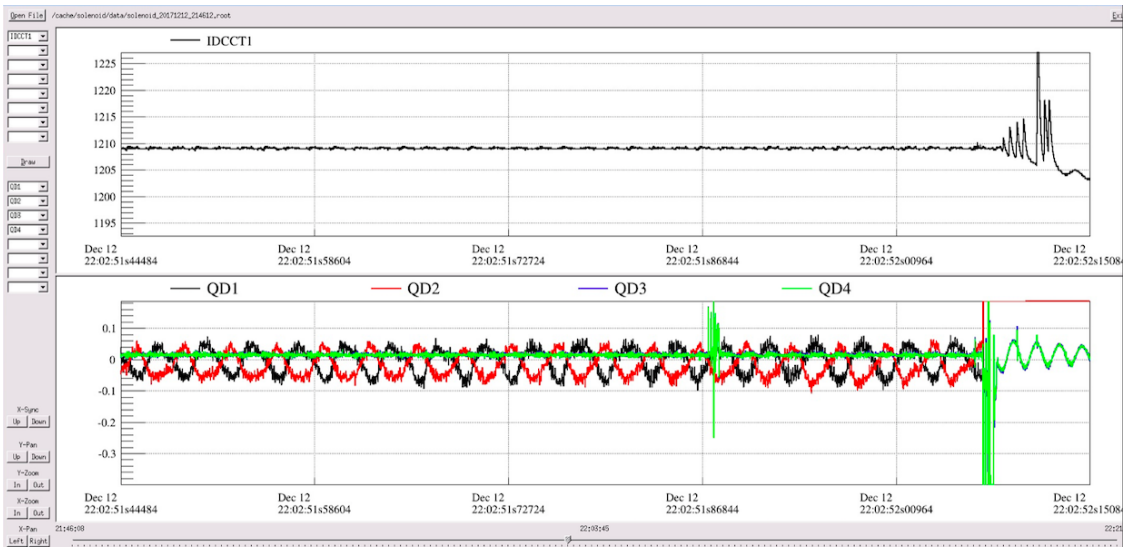

Figure 33: The 10 kHz EPICS-based readout of superconducting magnet's quench detection system.

[12] XXX *et al.*, *"The SSP Board"*.
[13] XXX *et al.*, *"The CODA System'*.
[14] XXX *et al.*, *"The CLAS12 SVT Detector'*, see this issue.
[15] B.Raydo *et al.*, *"The CLAS12 Trigger System'*, see this issue.