The Jefferson Lab Trigger Supervisor System

E. Jastrzembski, D.J. Abbott, W.G. Heyes, R.W. MacLeod, C. Timmer, E. Wolin
Thomas Jefferson National Accelerator Facility, 12000 Jefferson Ave., Newport News, VA 23606

Abstract

We discuss the design and performance of a Trigger Supervisor System for use in nuclear physics experiments at Jefferson Lab. We also discuss the enhanced features of a new Trigger Supervisor Module now under construction.

I. Introduction

The Continuous Electron Beam Accelerator at Jefferson Lab is delivering up to 5 GeV electrons to three experimental halls. A common data acquisition architecture capable of high event rates (5 kHz) is in use [1]. The data acquisition system is based primarily on Fastbus [2] front-end electronics (ADCs, TDCs), with data collected in parallel by high-speed programmable readout controllers (ROCs). Event fragments from the ROCs are assembled and may be routed to processors for analysis before storage.

To reduce dead time in a high rate environment the system is designed to support buffered front-end modules. Having on-board buffering decouples the readout dead time from the conversion dead time, allowing a properly integrated system to run as fast as the front-end operation permits. The Jefferson Lab data acquisition system is designed to take advantage of on-board buffering for up to 8 events.

The Trigger Supervisor System is the link between the experiment specific triggering system and the ROCs. The system consists of a single Trigger Supervisor Module and a separate interface card for each ROC. The Trigger Supervisor Module acts as the central control point for acquisition activity. It accepts and prescales multiple sources of triggers, both physics and calibration types. The module maintains system busy while an accepted trigger is being processed. It generates signals for gating and timing of front-end electronics. The module coordinates the actions of multi-level triggering systems. It also keeps track of the number of events currently in the front-end module buffers. The module communicates trigger information to the ROCs via a parallel link to the ROC interface cards.

II. Trigger Supervisor Operation

A diagram of the Trigger Supervisor module (TS) and its connections to the trigger system, front-end electronics, and readout controllers is shown in Figure 1.

A. Triggering

The TS allows for three logical trigger levels that can be associated with three stages of gating the front-end electronics. Any number of physical trigger levels may make up a logical level. Twelve independent *Level 1 Trigger* streams can be accepted simultaneously. Four of these inputs can be prescaled by 24-bit factors, while another four can be prescaled by 16-bit factors. The *Level 1 Triggers* can be required to be in coincidence with a *Common Strobe*. Each Level 1 input can be individually disabled within the TS.

The TS is ready to accept triggers when all of the following conditions are satisfied: it is user enabled, no trigger is currently latched, and the inputs *Front-end Busy* and *External Inhibit* are not asserted. *Front-end Busy* is asserted when one or more of the front-end modules of the system are unable to accept new data. *External Inhibit* vetoes all input triggers while it is asserted. When the TS is ready to accept triggers, a *Level 1 Trigger* that passes through its prescale circuitry will latch up the TS and start a coincidence time window (adjustable 7 to 50 ns). *Level 1 Triggers* that occur on any of the 12 input channels during this time interval are latched. After the coincidence interval expires, this 12-bit latched trigger pattern addresses a fast trigger memory that is programmed by the user. Eight output bits of the memory are used to generate a pattern of *Level 1 Accept* signals. These signals have a precise time relationship to the input trigger (minimum 40 ns delay, jitter < 40 ps), and are used by external circuitry to generate ADC gates and TDC starts/stops. In this way, front-end modules can be selectively gated depending on the trigger type. Another output bit of the memory is available to tag unacceptable trigger patterns. When this bit is asserted an internal fast clear feature will reset the TS in 50 ns with no issuance of *Level 1 Accept*.

The trigger memory also determines how the TS will utilize higher-level trigger logic that may be present. Each latched trigger pattern is programmed as one of three possible classes. A Class 1 pattern does not require a higher-level trigger decision to have the event read out. A Class 2 pattern requires a pass response from the Level 2 logic to read out the event. A Class 3 pattern requires passes from both Level 2 and Level 3 logic for event readout. For single-hit trigger patterns the Class assignments refer directly to the Level 1 input streams. A calibration type input might be assigned to Class 1, while a complex physics input may be categorized as Class 3.

We first discuss the operation of the TS for a Class 3 trigger pattern. The issuance of *Level 1 Accept* results in the start of the TS main sequencer. At the same time the TS issues *Level 2 Start* to initiate the Level 2 trigger system. The TS will wait for a pass or fail response from Level 2. If *Level 2 Fail* is returned the TS issues *Clear*, which is used to externally generate a fast clear signal to the front-end electronics. In this case the TS will wait until the front end is no longer busy. Then a new *Level 1 Trigger* may be accepted. If *Level 2 Pass* is returned, then *Level 2 Accept* and *Level 3 Start* are issued. *Level 2 Accept* may be used to generate additional control signals to the front end (e.g. digitize data), while *Level 3 Start* initiates Level 3 trigger logic. The TS will wait for a pass or fail response from Level 3. If *Level 3 Fail* is returned, the sequencer will issue *Clear* and wait until the

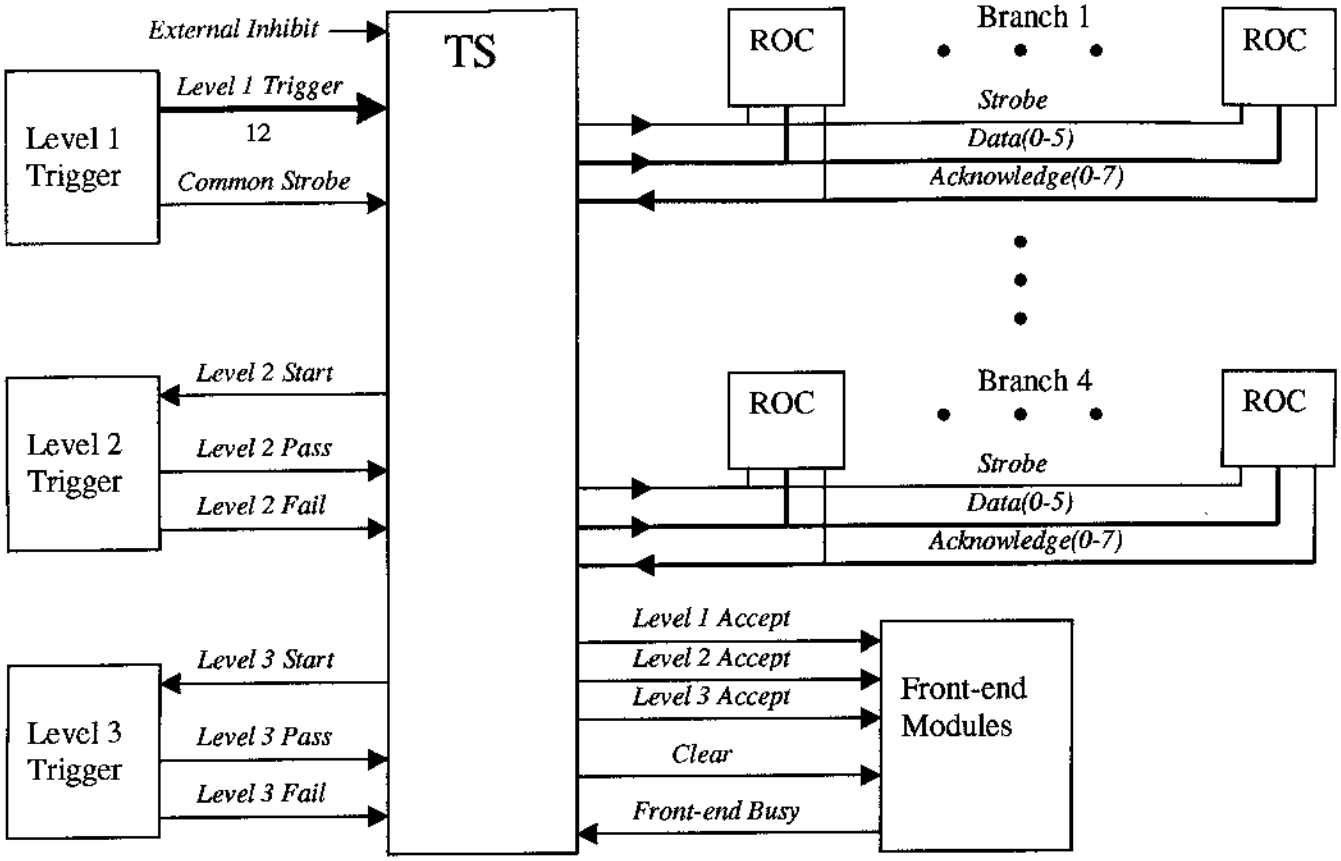

Figure 1: The Trigger Supervisor in a data acquisition system. ROC here includes both the readout controller and interface. Each branch may have up to 8 ROCs attached.

front end is ready to accept a new event. Then a new *Level 1 Trigger* may be accepted by the TS. If *Level 3 Pass* is returned, the event is to be read out. *Level 3 Accept* is issued by the TS and can be used to generate additional control signals to the front end (e.g. buffer data).

For Class 1 and Class 2 trigger patterns the activity of the TS is somewhat different. For Class 1 no higher-level trigger decision is required so *Level 2 Start* and *Level 3 Start* are never issued. The TS will nevertheless generate *Level 2 Accept* and *Level 3 Accept* signals that may be necessary for front-end control. These are issued at programmable delays after *Level 1 Accept*. For Class 2 trigger patterns the TS generates *Level 2 Start* and waits for the Level 2 decision. If *Level 2 Fail* is returned, the TS asserts *Clear* and prepares for a new trigger. If *Level 2 Pass* is returned, the TS issues *Level 2 Accept* but not *Level 3 Start*. *Level 3 Accept* is generated by the TS at a programmable time as for Class 1 trigger patterns. The delays are programmable in 40 ns increments up to a maximum of 2.6 milliseconds.

B. TS-ROC Communication

If *Level 3 Accept* has been issued the event is to be read out. The TS makes information about the trigger available to the ROCs in the form of a 4-bit Event Type from the trigger memory. Like data from the front-end modules, this data is presented to the ROCs in buffered fashion. The TS supports a total of 32 ROCs on its four independent ROC branches. Each branch consists of an 8-deep buffer memory (FIFO), buffer counter, and read sequencer on the TS, and an external cable that links up to 8 ROC interface cards. The Event Type and two status flags (described below) form the data transmitted along the branch. When *Front-end Busy* is no longer asserted the TS will load the Event Type and status data into the next location in each of the buffers. All buffer counters are incremented upon this load. If none of the buffers are full, the TS will negate *Level 1 Accept*, *Level 2 Accept*, *Level 3 Accept*, and re-arm itself to accept a new *Level 1 Trigger*. Otherwise, the sequencer will hold its state and wait for space to become available for the next event before negating the signals and re-arming itself.

The communication of trigger information occurs concurrently and independently on all four branches. Once the read sequencer on a branch notices that its buffer is not empty, the transmission of trigger information for the event to the ROC interface cards begins. The read sequencer places the Event Type and status data from the first valid buffer location onto data lines *Data(0-5)* and strobes these with *Strobe*. The ROCs along the branch receive this data and proceed to read out the event fragments according to the function defined by the Event Type. Each ROC is assigned a unique *Acknowledge* line on the branch cable. When a ROC on the branch is finished processing the event it asserts *Acknowledge*. Upon receipt of *Acknowledge* from all enabled ROCs on the branch, the read sequencer negates *Strobe* and decrements its buffer counter. Each ROC negates its

Acknowledge upon detecting the negation of *Strobe*. The read sequencer cycle is completed when it detects that *Acknowledge* has been negated by all enabled ROCs on the branch. These cycles on the branch will continue as long as there is valid trigger data available in the buffer.

For systems that have front-end modules with no buffering capability, the depth of the branch buffers can be set to 1. In this configuration the TS is able to accept new triggers only when the previous event has been completely read out. The TS also supports a mixed system of buffered and non-buffered front-end modules. One Branch can be set to have a buffer depth of 1 independent of the other branches. All non-buffered front-end modules must have their ROC reside on this branch when such a mixed system is used.

C. System Synchronization

When buffering is enabled the ROCs may be several events behind the trigger. The position in the parallel buffers links the fragments of data as an event. There is a danger that if a hardware error occurs resulting in a misalignment of data (e.g. a module misses a gate), all subsequent events will be corrupted. To avoid this potentially large loss of data the TS system can be enabled to periodically test for synchronization. The idea is to occasionally stop accepting triggers and check that all front-end module buffers have been emptied. Missing or excess data in any front-end module is indicative of a synchronization problem. The TS implements this by using a special status flag and a programmable synchronization counter. When the TS has accepted this programmed number of events, it asserts and writes the synchronization flag along with the Event Type to the branch buffers. As long as the sync flag remains asserted the TS will hold its state, unable to accept new triggers. The read sequencers on the branches continue to send trigger information for events that remain in the branch buffers, and the ROCs continue to read the event fragments from their front-end modules. A 'no data' response from any module indicates a loss of synchronization and the block of events since the last successful synchronization must be marked as corrupt. The last event in all branch buffers is the sync tagged event, and the sync flag appears as a Status Data bit on the branch cable when the read sequencer reaches this event. The ROCs read the associated data from the front-end modules but do not yet respond with *Acknowledge*. At this point all front-end module buffers should be empty. Each ROC attempts one more read to assure that no additional data is found. When this process and any bookkeeping associated with it is completed, the ROC issues *Acknowledge*. When all ROCs have responded, the synchronization flag and counter are reset and the TS cycle is completed with the re-arming of triggers. The synchronization counter may be programmed to a maximum of 64K events.

A synchronization can also be forced at any time. When the request is made the TS disables new triggers and waits for any current trigger cycle to finish. The TS then establishes a zero Event Type and asserts and writes the sync flag along with the Event Type to the branch buffers. When the ROCs recognize a zero Event Type along with the sync tag, they know that the front-end module buffers should be empty. As for a scheduled synchronization, each ROC attempts one more read to assure that no additional data is found. The ROC then issues its *Acknowledge*. When all ROCs have responded the synchronization flag and counter are reset and the TS cycle is completed with the re-arming of triggers.

D. Compatibility with Front-end Modules

This basic scheme assumes that the front-end modules will be told explicitly when to load the digitized data into their buffers. However, some commercially available buffered front-end modules do not operate in this way. Consider as an example the LeCroy 1882 Fastbus ADC [3]. When a gate is issued to this device it will sample and hold the analog input. The hold time is programmable and when it elapses the data will be digitized and the results loaded into its buffer. Only a *Clear* during the hold time will successfully purge the data. A *Clear* issued by the TS is the result of a *Level 2 Fail* or *Level 3 Fail* from the trigger system. If a fail signal comes after the hold interval expires, the data is destined for the buffers and the TS must declare to the ROCs that this event has been accepted. To do this the TS has a programmable clear enable timer that is started on the issuance of *Level 1 Accept*. The TS will behave as described earlier as long as this timer has not expired. If this clear enable period expires before the highest level decision required of the trigger occurs, the *Clear* signal is disabled and the TS continues to wait for a decision. If it is a fail, the TS tags the event by asserting and writing the Late Fail flag along with the Event Type to the branch buffers. The ROCs can use the Late Fail flag to flush the data for this event from the front-end module buffers if desired. The clear enable timer is programmable in 40 ns increments up to a maximum of 2.6 milliseconds. The value programmed should be less than the smallest front-end module hold time in the system.

III. TS IMPLEMENTATION

The Trigger Supervisor module is constructed as a 'D' size VXI card [4]. This form factor was chosen so that the device could plug directly into the complex VXI based trigger system of Hall B at Jefferson Lab. The large card size (340 x 367 mm) also allows the complete design to reside on a single printed circuit board. The Trigger Supervisor module is programmed through VME and is characterized as an A24/D32 slave. As an alternative to a VXI crate, we designed a custom VME/VXI chassis in which the Trigger Supervisor may be stationed. This mixed crate supports twelve standard 6U VME modules and three 9U VXI type modules. All slots share a common 32-bit VME backplane. A custom power distribution backplane supplies additional voltages (-5V, -2V) to the VXI slots.

High speed input and output signals for the Trigger Supervisor module are single-ended ECL levels. These are carried on 50-ohm coaxial cables (RG-178) terminated by a 2-pin receptacle. These mate with standard 0.10 inch pitch multi-pin headers to allow for high density I/O.

All signals on the TS branches are RS-485 levels. This differential mode of transmission was chosen for its superior noise immunity and long distance drive capability. The signals are carried on a 17-element twisted-pair cable terminated by a standard 34-position 0.10 inch pitch receptacle. Flat ribbon-type cable can be used, but round

shielded cable is preferable for long distance runs. Branch cables as long as 500 ft are in use at Jefferson Lab. All branch cables have a characteristic impedance of 100 ohms.

The Trigger Supervisor module is fabricated as a single 10-layer printed circuit board using high-speed ECL and TTL components. Six of the layers carry signals on 8 mil wide traces. The internal layers are organized as a pair of dual-stripline configurations and have a characteristic impedance of 50 ohms. High-speed ECL signals were placed exclusively on these impedance controlled layers. Slower TTL signals could use any of the six signal layers. Extensive use is made of programmable logic devices (PLD). With over 300 integrated circuit components comprising the design, space considerations led to the use of surface mount devices.

IV. ROC Interface Cards

Each readout controller in the system has an associated interface card. The interface card serves as the physical connection of the ROC to the TS branch. Jumpers on the card assign the ROC to its unique *Acknowledge* line on the branch cable. Communication between the ROC and the interface enables the TS branch protocol described earlier to be executed. The form factor of the interface card and the communication protocol used is tailored to the specific type of ROC.

Besides enabling the ROC to execute the TS branch protocol, the interface cards have additional functionality built in. Each has circuitry on board to allow for stand-alone crate operation. When programmed for this mode, the interface card performs Trigger Supervisor like functions for the single crate. It ignores any activity on the TS branch port and instead is enabled to accept one or more trigger streams into an alternate port. When a trigger occurs, the trigger bits and other input data bits are latched, subsequent triggers are held off, and prompt signals are issued that can be used in the gating of the modules in the crate. Communication between the interface and ROC is the same as for operation with the TS. The busy state is maintained until cleared by the ROC after readout. Interface cards also have some general purpose I/O capability.

Three types of ROCs have been in general use at Jefferson Lab: the STR340 Fastbus controller [5], the FSCC Fastbus controller [6], and VME single board computers.

The STR340 is a Fastbus readout controller that is managed by a VME single board computer mounted onto it. Because it has good performance as a Fastbus master and its compute engine can be upgraded, the STR340 is the preferred Fastbus ROC at Jefferson Lab. The ROC interface for it is implemented as a standard Fastbus Auxiliary card, with communication through the controller's Auxiliary port register. At our request the manufacturer modified the standard STR340 Auxiliary port to permit the two-way communication required for our interface. The TS branch *Strobe* line is mapped to an STR340 signal capable of interrupting of the associated VME computer when a new event is present. Alternatively, the status of this signal may be polled to detect a new event. In stand-alone mode the interface can accept 4 independent trigger streams and 2 data lines. While operating with the TS, these 6 inputs serve as a general input port, being latched on a read of the Auxiliary port register. An 8-bit output port is available in either mode.

The FSCC is a Fastbus readout controller with an embedded 68020 processor. The ROC interface for the FSCC is implemented as a standard Fastbus Auxiliary card, with communication through the controller's Auxiliary port register. The standard FSCC's Auxiliary port was customized to permit the two-way communication required for the interface. The TS branch *Strobe* line is mapped to an FSCC signal capable of interrupting the on-board processor when a new event is present. Alternatively, the state of this signal may be determined by polling. In stand-alone mode the interface can accept a single trigger stream. A 2-bit output port is available in either mode. In addition, the interface drives a 32-bit pipelined data port (RS-485) capable of 40MB/s rates. This high-speed data channel connects by cable to a custom built dual-ported VME memory module.

For a VME single board computer as a ROC, we chose to construct the interface as a VME module. Communication between the ROC and interface is done using standard VME protocols. When the TS branch *Strobe* line is asserted, the interface is capable of generating a VME interrupt to gain the attention of the ROC when a new event is present. Alternatively, the interface may be polled to detect the arrival of an event. In stand-alone mode the interface can accept 4 independent trigger streams and 8 data lines. While operating with the TS, these 12 inputs serve as a general input port, being latched on a read of the input port register. An 8-bit output port is available in either mode.

V. Performance

Trigger Supervisor systems at Jefferson Lab have been performing reliably since 1994. There are currently four separate installations of the system at the laboratory. The systems range from small (3 ROCs) to large (23 ROCs). At this time, two of the systems utilize the multi-level trigger capability of the TS module.

One installation is unique in that it has a separate TS module assigned to each of its two spectrometer arms. One of the TS modules is normally selected to control the entire system, but under some circumstances the spectrometer arms are run independently. We have provided additional hardware that makes switching between these modes simple and largely under program control.

VI. The New Trigger Supervisor

Requests for additional Trigger Supervisor modules at the laboratory could not be satisfied because several IC components used in the module were no longer available. Since a redesign was necessary, we had the opportunity to enhance the board's functionality and performance. The ability to include many new features is directly related to the significant improvements in the density and speed of PLDs since the original Trigger Supervisor was built.

An alternate mode of trigger operation has been added to the new TS. In this mode, all *Level 1 Accept* signals are

driven promptly when the initial trigger latches up the TS. The trigger pattern latched during the coincidence window still serves as the address of the trigger memory, but its outputs only determine the Event Type, event Class, and veto status. Because the *Level 1 Accepts* have already been driven, a true fast veto of an unacceptable trigger pattern is not possible. The TS instead issues *Clear* to the front-end modules. In this mode the ability to drive different patterns of *Level 1 Accept* signals is sacrificed to enable adjustment of the coincidence window without impact on front-end module timing.

The Event Type communicated to the ROCs has been expanded from 4 to 6 bits, thus allowing up to 64 separate readout controller functions.

Two programmed events have been added. These are special events in which the TS communicates with the ROCs but not with the front-end modules (i.e. no *Level 1 Accepts*, etc.). In this way they resemble the forced sync event feature of the original TS. They can be useful in signaling the ROC to execute a special action (e.g. readout of scalers). The Event Type for each special event is stored in a register and identifies it. Both events can be triggered to occur by software, and one can be triggered by a front-panel input. A programmed event is guaranteed to be seen by the ROCs, as the latching of *Level 1 Triggers* is suspended until the event is inserted into the TS-ROC pipeline.

On the new TS the pattern of twelve latched *Level 1 Trigger* bits for accepted events is stored in an onboard FIFO that can be read out at any time. The latched bits are also driven promptly from the board so that they may be used in higher-level trigger logic.

The functionality of a VME ROC Interface module has been included on the new TS. The single board computer in the TS crate can thus act as an event driven ROC without requiring an external ROC Interface module to be installed in the crate.

On the new TS each of the ROC branches may now be independently configured to have a buffer depth of 1. The number of events currently stored in each ROC branch buffer can be read at any time. The status of each ROC *Acknowledge* signal can also be monitored.

The new TS has eight on-board 32-bit scalers that can be configured to count signals input to the TS as well as signals generated by the TS.

Trigger related I/O (*Level 1 Trigger* inputs, *Level 1 Accept* outputs) are configurable on the new TS, in groups of four, as either single-ended ECL or differential ECL.

In terms of performance, the critical timing paths in the front end of the new TS have been implemented using very high-speed Motorola ECLinPS Lite (10EL series) and ECLinPS (10E series) components. All state machines and control logic are now done on the TTL side of the board using high speed Altera 7000, 9000, and 10000 series PLDs.

The design of the new Trigger Supervisor is completed. A prototype should be ready for testing in mid-summer 1999, with production units available in the fall.

VII. ACKNOWLEDGEMENTS

This work was supported by DOE Contract #DE-AC05-84ER40150.

VIII. REFERENCES

[1] G. Heyes, *et. al.*, "The CEBAF On-line Data Acquisition System", Proceedings of the CHEP Conference (Apr. 1994) , pp. 122-126.

[2] IEEE Std 960-1993.

[3] LeCroy Corporation, 700 Chestnut Ridge Road, Chestnut Ridge, NY 10977.

[4] IEEE Std 1155-1992.

[5] Bastian Technology, GmbH & Co.KG, Backerbarg 6, 22889 Tangstedt.

[6] Bi-Ra Systems, 2404 Comanche N.E., Albuquerque, NM 87107.