Trigger Supervisor

Version 2

Ed Jastrzembski

11/10/2009

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 100 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 < 35 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 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 6-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(0-7)* 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. Any 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 such a branch when 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 corrupted. 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.

E. Other Features

An alternate mode of trigger operation exists. In this mode, all *Level 1 Accept* signals are driven promptly (10 ns delay) 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 functionality of a VME ROC Interface module is also included on the TS. This is useful when the TS crate contains modules that must be read out on an event-by-event basis. 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.

Two programmed events are available. 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. 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. Optionally, the synchronization flag may be also be asserted for these events. 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.

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 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 TS has 21 on-board 32-bit scalers to monitor the system. Each of the 12 *Level 1 Trigger* inputs has a dedicated scaler assigned to it. One scaler counts accepted events, and six additional scalers can be programmed to monitor various other important TS signals. A pair of scalers is specially configured to allow a measurement of the system live time. The pair count a free running 200 kHz clock, with one scaler being gated by the TS live signal. The ratio of these scalers is thus an estimate of the system live time.

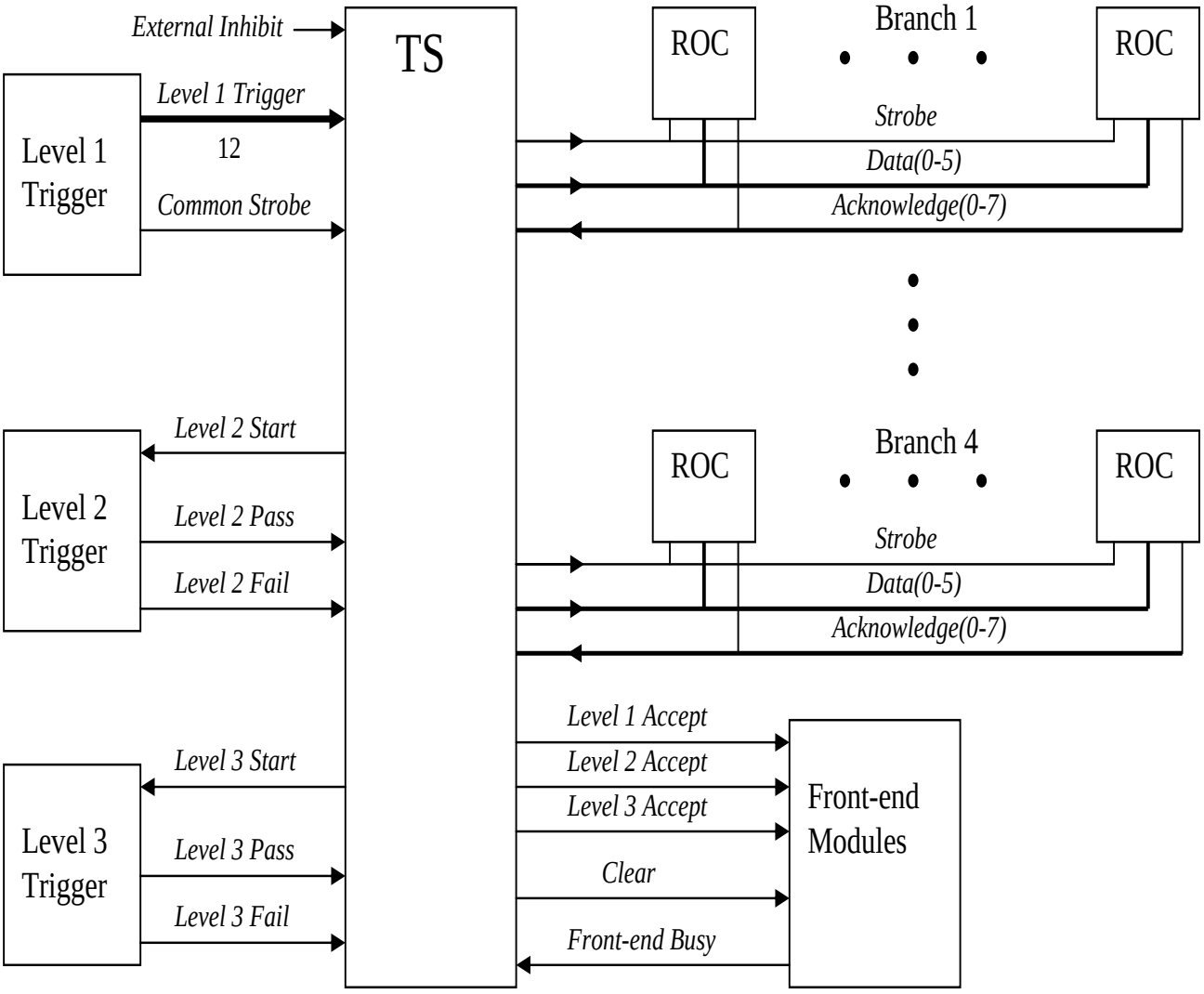

Figure 1: The Trigger Supervisor in a data acquisition system. ROC here includes both the readout controller and interface. Each branch may have up to 8 ROCs attached.

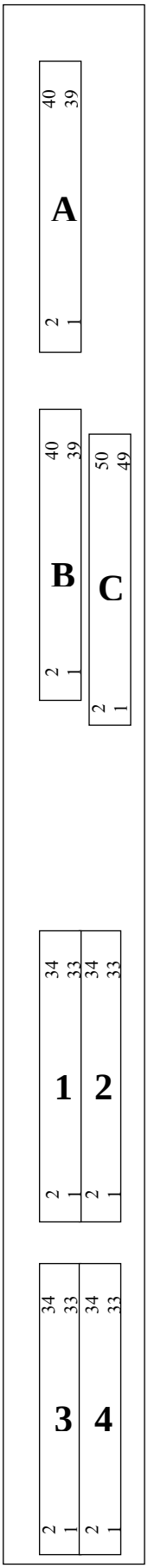

Figure 2: Trigger Supervisor front panel.

TS Facts

Level 1 Trigger Inputs	12
Prescalable Inputs	8
Prescale Factors	
Inputs 1-4	24 bits
Inputs 5-8	16 bits
Prescale Input Bandwidth	> 60 MHz
Level 1 Trigger Coincidence window	7-100 ns
Delay (Trigger to Level 1 Accept)	40 ns (min)
Level 1 Accept Jitter (RMS)	<35 ps
Maximum Event Rate	3 MHz
TS-ROC Communication	RS-485 parallel, Full handshake
Programming Interface	A24/D32 VME slave
Form Factor	‘D’ size VXI (340 x 367 mm)

Table 2.1 Connector A signal definition (listed top to bottom)

- All signals are differential ECL pairs
- All signal pairs are terminated into 100 ohms

Signal name	Direction	Pin # (Q,/Q)
Common Strobe	Input	39,40
Trigger 1	Input	37,38
Trigger 2	Input	35,36
Trigger 3	Input	33,34
Trigger 4	Input	31,32
Trigger 5	Input	29,30
Trigger 6	Input	27,28
Trigger 7	Input	25,26
Trigger 8	Input	23,24
Trigger 9	Input	21,22
Trigger 10	Input	19,20
Trigger 11	Input	17,18
Trigger 12	Input	15,16
External Event	Input	13,14
Level 2 Pass	Input	11,12
Level 2 Fail	Input	9,10
Level 3 Pass	Input	7,8
Level 3 Fail	Input	5,6
Front End Busy	Input	3,4
External Inhibit	Input	1,2

Table 2.2 Connector B signal definition (listed top to bottom)

- All signals are differential ECL pairs

Signal name	Direction	Pin # (Q,/Q)
Level 1 Accept (0)	Output	39,40
Level 1 Accept (1)	Output	37,38
Level 1 Accept (2)	Output	35,36
Level 1 Accept (3)	Output	33,34
Level 1 Accept (4)	Output	31,32
Level 1 Accept (5)	Output	29,30
Level 1 Accept (6)	Output	27,28
Level 1 Accept (7)	Output	25,26
Level 1 Accept (8)	Output	23,24
Level 2 Accept	Output	21,22
Level 3 Accept	Output	19,20
Clear	Output	17,18
Level 2 Start	Output	15,16
Level 3 Start	Output	13,14
Enable level 1	Output	11,12
TS Live	Output	9,10
TS Busy	Output	7,8
OR Level 1 Trigger	Output	5,6
GO TS	Output	3,4
Gate Out	Output	1,2

Table 2.3 Connector C signal definition (listed top to bottom)

- All signals are differential ECL pairs

Signal name	Direction	Pin # (Q,/Q)
Latched Trigger 12	Output	49,50
Latched Trigger 11	Output	47,48
Latched Trigger 10	Output	45,46
Latched Trigger 9	Output	43,44
Latched Trigger 8	Output	41,42
Latched Trigger 7	Output	39,40
Latched Trigger 6	Output	37,38
Latched Trigger 5	Output	35,36
Latched Trigger 4	Output	33,34
Latched Trigger 3	Output	31,32
Latched Trigger 2	Output	29,30
Latched Trigger 1	Output	27,28
(No Connection)		25,26
Prescaled Trigger 12	Output	23,24
Prescaled Trigger 11	Output	21,22
Prescaled Trigger 10	Output	19,20
Prescaled Trigger 9	Output	17,18
Prescaled Trigger 8	Output	15,16
Prescaled Trigger 7	Output	13,14
Prescaled Trigger 6	Output	11,12
Prescaled Trigger 5	Output	9,10
Prescaled Trigger 4	Output	7,8
Prescaled Trigger 3	Output	5,6
Prescaled Trigger 2	Output	3,4
Prescaled Trigger 1	Output	1,2

Table 2.4 Connector 1, 2, 3, 4 signal definition (listed top to bottom)

- All signals are differential RS-485 pairs
- The four connectors represent the four ROC branches, all having the same basic signal assignments shown below (ROC n represents the nth ROC on the given branch)

Signal name	Direction	Pin # (Q,/Q)
ROC 7 Acknowledge	Input	33,34
ROC 6 Acknowledge	Input	31,32
ROC 5 Acknowledge	Input	29,30
ROC 4 Acknowledge	Input	27,28
ROC 3 Acknowledge	Input	25,26
ROC 2 Acknowledge	Input	23,24
ROC 1 Acknowledge	Input	21,22
ROC 0 Acknowledge	Input	19,20
ROC Code Data Bit 5	Output	17,18
ROC Code Data Bit 4	Output	15,16
ROC Code Data Bit 3	Output	13,14
ROC Code Data Bit 2	Output	11,12
ROC Code Data Bit 1	Output	9,10
ROC Code Data Bit 0	Output	7,8
Late Fail Flag	Output	5,6
Synchronization Flag	Output	3,4
Strobe	Output	1,2

The Trigger Supervisor Registers

The Trigger Supervisor is programmed by the user through VME bus protocols (ANSI/IEEE STD1014-1987). The device meets all VME bus standards except in terms of board size. The form factor is that of a 'D' size (340 x 367 mm) VXI module (VME extension for instrumentation). This configuration allows the device to plug directly into the VXI based trigger system of one experimental hall as well as into standard 32-bit VME backplanes (with suitably modified card cage) of other experimental facilities.

Officially the Trigger Supervisor is categorized as an A24, D32 VME bus slave. All storage locations can be accessed as both Standard Supervisory and Standard Non-privileged data. In terms of its interrupt capability the module is classified as an I(1-7), D08(O), ROAK VME bus interrupter.

A brief discussion of the types of register access through VME is presented here because it impacts of the correct use of the Trigger Supervisor. The user is referred to the full VME bus specification for a complete description.

The smallest addressable unit of storage is the byte location, to which a unique binary address is assigned. There are four categories of bytes distinguished by the least significant two bits of their address:

xx...x00 BYTE(0)
xx...x01 BYTE(1)
xx...x10 BYTE(2)
xx...x11 BYTE(3).

A set of byte locations differing only in the two least significant bits defines a four byte BYTE(0-3) group. The VME data transfer protocols allow a master to access 1, 2, 3, or 4 byte locations from the same group simultaneously:

Single byte access - BYTE(0), BYTE(1), BYTE(2), BYTE(3),
Double byte access - BYTE(0-1), BYTE(1-2), BYTE(2-3),
Triple byte access - BYTE(0-2), BYTE(1-3),
Quad byte access - BYTE(0-3).

For example, the 32-bits of data stored in a 4-byte group of a D32 slave may be accessed by a D32 master in a single bus cycle via the BYTE(0-3) transfer, or in 4 bus cycles via the BYTE(n) transfer. However, a 16-bit master would require a minimum of 2 bus cycles (BYTE(0-1) & BYTE(2-3)) to accomplish the same data transmission.

We now turn to a general discussion of the Trigger Supervisor's registers in the above terminology. Each individual register or memory location of the Trigger Supervisor is defined to be a 4-byte group. This is somewhat inefficient usage of address space as the actual storage locations contain from 1 to 4 bytes of data. However, the decision results in a simplification of board design and possibly in programming (since all of the above access types are legal for all locations). The local address given in this document for each register is the address of BYTE(0) for that register. Data is stored in the 4-byte registers as follows:

D(31)-D(24) BYTE(0)
D(23)-D(16) BYTE(1)
D(15)-D(08) BYTE(2)
D(07)-D(00) BYTE(3).

Certain considerations concerning a register's function must be understood by the user before employing the different access schemes described above. For example, data written to some registers has to be loaded into counters. For simplicity we have coupled this loading action with the actual write of the register. But as discussed above, the entire data word may be assembled as several writes by user choice or hardware constraints. We therefore adopt the convention that only a write of BYTE(3) triggers such a loading process. This requires that BYTE(3) must be included in the group of bytes that is written last.

Similarly, some data to be read is dynamic in nature. Such data must be latched as a unit and then read. For simplicity we have coupled the latching action with the actual read of the register. But as discussed above, the full data word may be assembled only after several reads. We therefore adopt the convention that such dynamic data is latched only upon a read of BYTE(0). Reading of the remaining bytes causes no latching action to occur. This requires that BYTE(0) must be included in the group of bytes that is read first.

Of course, as long as the mode of access includes all bytes of interest, the above requirements are automatically met.

Many of the trigger supervisor's registers are designated as protected against writes while the device is active. The trigger supervisor is defined to be active if any of the following conditions hold:

- the GO bit of the Control/Status Register (CSR) is set,
- the TS main sequencer is active,
- the TS user synchronization or program event sequencers is active.

The write protection prevents the user from changing the TS operating conditions while events are being processed.

The Trigger Supervisor's memory is also protected against reads while the device is active. Reading the memory while the device is active would interfere with event driven memory accesses.

The base address of the module is set by 9 DIP switch elements corresponding to VME address bits A23-A15. An open switch defines a ‘1’. The base address (hex) can thus be set to XX0000 or XX8000. The module occupies 32 Kbytes (8K Word, 32-bit) of address space. The first half of this is for access to the Trigger Supervisor registers. The second half corresponds to the RAM that stores the trigger lookup table.

In what follows the local register and memory addresses (hex) are listed. The absolute VME bus address is given by:

VME Address = Module Base Address + Local Address.

The VME bus interrupt request level of the module is set by 3 DIP switch elements. An open switch defines a ‘1’. The level (1-7) is binary encoded.

TS Register List

(0x00) CSR 1
(0x04) CSR 2
(0x08) Trigger Control Register
(0x0C) ROC Enable Register

(0x10) Synchronization Interval Register
(0x14) Trigger Word Count Register
(0x18) Trigger Data Register
(0x1C) Local ROC (Branch 5) Data Register

(0x20) Trigger Input (1) Prescale Register
(0x24) Trigger Input (2) Prescale Register
(0x28) Trigger Input (3) Prescale Register
(0x2C) Trigger Input (4) Prescale Register

(0x30) Trigger Input (5) Prescale Register
(0x34) Trigger Input (6) Prescale Register
(0x38) Trigger Input (7) Prescale Register
(0x3C) Trigger Input (8) Prescale Register

(0x40) Clear Permit Timer Register
(0x44) Level 2 Accept Timer Register
(0x48) Level 3 Accept Timer Register
(0x4C) Front Busy Timer Register

(0x50) Clear Hold Timer Register
(0x54) Interrupt ID Register
(0x58) Branch (1 - 4) ROC Buffer Status Register
(0x5C) Local ROC (Branch 5) Buffer Status Register

(0x60) Branch (1 - 4) ROC Acknowledge Status Register
(0x64) Program 1 Event Data Register
(0x68) Program 2 Event Data Register
(0x6C) State Register

(0x70) Test Register
(0x74) Reserved
(0x78) Scaler (13 - 18) Assign Register
(0x7C) Scaler Control Register

(0x80) Scaler Trigger Input 1 Register
(0x84) Scaler Trigger Input 2 Register
(0x88) Scaler Trigger Input 3 Register
(0x8C) Scaler Trigger Input 4 Register

(0x90) Scaler Trigger Input 5 Register
(0x94) Scaler Trigger Input 6 Register
(0x98) Scaler Trigger Input 7 Register
(0x9C) Scaler Trigger Input 8 Register

(0xA0) Scaler Trigger Input 9 Register
(0xA4) Scaler Trigger Input 10 Register
(0xA8) Scaler Trigger Input 11 Register
(0xAC) Scaler Trigger Input 12 Register

(0xB0) Scaler 13 Register
(0xB4) Scaler 14 Register
(0xB8) Scaler 15 Register
(0xBC) Scaler 16 Register

(0xC0) Scaler 17 Register
(0xC4) Scaler 18 Register
(0xC8) Scaler Event Register
(0xCC) Scaler Live 1 Register

(0xD0) Scaler Live 2 Register
(0xD4) Version Register

(0x4000 – 0x7FFC) Memory

TS Registers

CSR 1 (0x0)

Writing 1 to bit N (N=0,…,9) sets the function associated with that bit, while writing 1 to bit N+16 clears that same function. Bits are Read/Write unless otherwise indicated.

(0) Go
(1) Pause on Next Scheduled Sync
(2) Sync and Pause
(3) Initiate Sync Event
(4) Initiate Program 1 Event
(5) Initiate Program 2 Event
(6) Enable Level 1 (drives output)
(7) Override Inhibit
(8) Test Mode
(9) Reserved
(10) - (13) Unused (Read as 1)
(14) Reset (Write only, Read as 1)
(15) Initialize (Write only, Read as 1)

(26) - (30) Unused (Read as 1)
(31) Clear Latched Status bits (16) - (23) (Write only, Read as 1)

Latched Status bits

(16) Sync Event occurred
(17) Program 1 Event occurred
(18) Program 2 Event occurred
(19) Late Fail occurred
(20) Inhibit occurred
(21) Write FIFO Error occurred
(22) Read FIFO Error occurred
(23) Reserved

CSR 2 (0x4) (Write protected when TS active)

(0) Enable Scheduled Sync
(1) Use Clear Permit Timer
(2) Use Front Busy Timer
(3) Use Clear Hold Timer
(4) Use External Front Busy
(5) Lock ROC Branch 1
(6) Lock ROC Branch 2
(7) Lock ROC Branch 3
(8) Lock ROC Branch 4
(9) Lock ROC Branch 5
(10) Enable Program 1 Front Panel Input
(11) Enable Interrupt
(12) Enable Local ROC (Branch 5)
(13) Reserved

(14) Disable Level 1 Trigger Inputs 9 - 12 from Triggering
(15) Use Fast Mode (no memory lookup for Level 1 Accept)

(16) - (31) Unused (Read as 1)

Trigger Control Register (0x8) (Write protected when TS active)

(0) Non-Common Strobe Mode
(1) - (12) Input Enables
(13) No Pulse Regeneration on Trigger Inputs 9 - 10
(14) No Pulse Regeneration on Trigger Inputs 11 - 12
(15) Open Prescales

(16) – (31) Unused (Read as 1)

ROC Enable Register (0xC) (Write protected when TS active)

(0) - (7) Enable ROC 0-7 on Branch 1
(8) - (15) Enable ROC 0-7 on Branch 2
(16) - (23) Enable ROC 0-7 on Branch 3
(24) - (31) Enable ROC 0-7 on Branch 4

Synchronization Interval Register (0x10) (Write protected when TS active)

(0) - (15) Number of Events between Scheduled Synchronizations

(16) - (31) Unused (Read as 1)

Trigger Word Count Register (0x14) (Read only)

(0) - (15) Number of Trigger Data words in Trigger FIFO

(16) - (31) Unused (Read as 1)

Trigger Data Register (0x18) (Read only)

(0) - (11) Next Trigger Data word from Trigger FIFO
(12) - (15) Unused (Read as 0)
(16) - (31) Unused (Read as 1)

Local ROC (Branch 5) Data Register (0x1C)

(0) Synchronization Flag
(1) Late Fail Flag
(2) - (7) ROC code (0) - (5)

(8) - (31) Unused (Read as 1)

Input Trigger (1 - 4) Prescale Registers (0x20, 0x24, 0x28, 0x2C)
(Write protected when TS active)

(0) - (23) Prescale Factor

(24) - (31) Unused (Read as 1)

Input Trigger (5 - 8) Prescale Registers (0x30, 0x34, 0x38, 0x3C)
(Write protected when TS active)

(0) - (15) Prescale Factor

(16) - (31) Unused (Read as 1)

Clear Permit Timer Register (0x40) (Write protected when TS active)

(0) - (15) Timer Value (40 ns/count)

(16) - (31) Unused (Read as 1)

Level 2 Accept Timer Register (0x44) (Write protected when TS active)

(0) - (15) Timer Value (40 ns/count)

(16) - (31) Unused (Read as 1)

Level 3 Accept Timer Register (0x48) (Write protected when TS active)

(0) - (15) Timer Value (40 ns/count)

(16) - (31) Unused (Read as 1)

Front Busy Timer Register (0x4C) (Write protected when TS active)

(0) - (15) Timer Value (40 ns/count)

(16) - (31) Unused (Read as 1)

Clear Hold Timer Register (0x50) (Write protected when TS active)

(0) - (7) Timer Value (40 ns/count)

(8) - (31) Unused (Read as 1)

Interrupt ID Register (0x54) (Write protected when TS active)

(0) - (7) Interrupt ID

(8) - (31) Unused (Read as 1)

Branch (1 - 4) ROC Buffer Status Register (0x58) (Read only)

(0) - (3) Branch 1 Buffer Count
(4) - (5) Unused (Read as 0)
(6) Branch 1 Empty Flag
(7) Branch 1 Full Flag

(8) - (11) Branch 2 Buffer Count
(12) - (13) Unused (Read as 0)
(14) Branch 2 Empty Flag
(15) Branch 2 Full Flag

(16) - (19) Branch 3 Buffer Count
(20) - (21) Unused (Read as 0)
(22) Branch 3 Empty Flag
(23) Branch 3 Full Flag

(24) - (27) Branch 4 Buffer Count
(28) - (29) Unused (Read as 0)
(30) Branch 4 Empty Flag
(31) Branch 4 Full Flag

Local ROC (Branch 5) Buffer Status Register (0x5C)
Bits 0 - 7, and 15 are Read only.

(0) - (3) Buffer Count
(4) - (5) Unused (Read as 0)
(6) Empty Flag
(7) Full Flag

(8) Local Acknowledge

(9) - (14) Unused (Read as 1)

(15) Local Event Strobe Status

(16) – (31) Unused (Read as 1)

Branch (1 - 4) ROC Acknowledge Status Register (0x60) (Read only)

(0) - (7) Branch 1 ROC Acknowledge 0 - 7

(8) - (15) Branch 2 ROC Acknowledge 0 - 7

(16) - (23) Branch 3 ROC Acknowledge 0 - 7

(24) - (31) Branch 4 ROC Acknowledge 0 - 7

Program 1 Event Data Register (0x64)

(0) - (5) Program 1 Event ROC code
(6) Reserved
(7) Program 1 Event Synchronization Flag

(8) - (31) Unused (Read as 1)

Program 2 Event Data Register (0x68)

(0) - (5) Program 2 Event ROC code
(6) Reserved
(7) Program 2 Event Synchronization Flag

(8) - (31) Unused (Read as 1)

State Register (0x6C) (Read only)

(0) Level 1 Accept
(1) Start Level 2 Trigger
(2) Level 2 Pass Latched
(3) Level 2 Fail Latched
(4) Level 2 Accept
(5) Start Level 3 Trigger
(6) Level 3 Pass Latched
(7) Level 3 Fail Latched
(8) Level 3 Accept
(9) Clear
(10) Front End Busy (external)
(11) External Inhibit
(12) Latched Trigger
(13) TS Busy
(14) TS Active
(15) TS Ready
(16) Main Sequencer Active
(17) Synchronization Sequencer Active
(18) Program 1 Event Sequencer Active
(19) Program 2 Event Sequencer Active
(20) - (31) Unused (Read as 1)

Test Register (0x70)

Bits (0) - (7) are write only and Read as 1.

(0) Level 1 Trigger Pulse
(1) Level 2 Pass Pulse
(2) Level 3 Pass Pulse
(3) Level 2 Fail Pulse
(4) Level 3 Fail Pulse
(5) - (7) Reserved

(8) Front End Busy Level
(9) External Inhibit Level
(10) Branch 1 ROC Acknowledge Level
(11) Branch 2 ROC Acknowledge Level
(12) Branch 3 ROC Acknowledge Level
(13) Branch 4 ROC Acknowledge Level
(14) Enable Test ROC Acknowledge for Branches 1 - 4
(15) Reserved

(16) - (31) Unused (Read as 1)

Scaler (13 - 18) Assign Register (0x78)

(0) – (3) Scaler 13 Assign Code

(4) – (7) Scaler 14 Assign Code

(8) – (11) Scaler 15 Assign Code

(12) – (15) Scaler 16 Assign Code

(16) – (19) Scaler 17 Assign Code

(20) – (23) Scaler 18 Assign Code

(24) – (31) Unused (Read as 1)

Assign Codes:	0 – OR TRIGGER	1 – LATCHED TRIGGER
	2 – LEVEL 1 ACCEPT	3 – LEVEL 2 ACCEPT
	4 – LEVEL 3 ACCEPT	5 – FAST RESET
	6 – CLEAR	7 – LEVEL 2 PASS
	8 – LEVEL 2 FAIL	9 – LEVEL 3 PASS
	A – LEVEL 3 FAIL	B – LATE FAIL
	C – SYNC SCHEDULED	D – SYNC FORCED
	E – PROGRAM 1 EVENT	F – PROGRAM 2 EVENT

Scaler Control Register (0x7C)

Bits 0 – 19 are write only, and Read as 1.

(0) – Reset Scaler 1
(1) – Reset Scaler 2
(2) – Reset Scaler 3
(3) – Reset Scaler 4
(4) – Reset Scaler 5
(5) – Reset Scaler 6
(6) – Reset Scaler 7
(7) – Reset Scaler 8
(8) – Reset Scaler 9
(9) – Reset Scaler 10
(10) – Reset Scaler 11
(11) – Reset Scaler 12
(12) – Reset Scaler 13
(13) – Reset Scaler 14
(14) – Reset Scaler 15
(15) – Reset Scaler 16
(16) – Reset Scaler 17
(17) – Reset Scaler 18
(18) – Reset Event Scaler
(19) – Reset Live Time Scalers

(20) - (22) Unused (Read as 1)

(23) – Latch and Hold all Scaler Registers

Scaler Registers (Read only)

0x80: (0) – (31) Trigger Input 1 Count

0x84: (0) – (31) Trigger Input 2 Count

0x88: (0) – (31) Trigger Input 3 Count

0x8C: (0) – (31) Trigger Input 4 Count

0x90: (0) – (31) Trigger Input 5 Count

0x94: (0) – (31) Trigger Input 6 Count

0x98: (0) – (31) Trigger Input 7 Count

0x9C: (0) – (31) Trigger Input 8 Count

0xA0: (0) – (31) Trigger Input 9 Count

0xA4: (0) – (31) Trigger Input 10 Count

0xA8: (0) – (31) Trigger Input 11 Count

0xAC: (0) – (31) Trigger Input 12 Count

0xB0: (0) – (31) Scaler 13 Count

0xB4: (0) – (31) Scaler 14 Count

0xB8: (0) – (31) Scaler 15 Count

0xBC: (0) – (31) Scaler 16 Count

0xC0: (0) – (31) Scaler 17 Count

0xC4: (0) – (31) Scaler 18 Count

0xC8: (0) – (31) Event Count

0xCC: (0) – (31) Live 1 Count

0xD0: (0) – (31) Live 2 Count (Live Time = (Live 1 Count) / (Live 2 Count))

Version Register (0xD4)

(0) – (15) Encoded information about circuit board and firmware revision.

Memory (0x4000 – 0x7FFC) (Write and read protected when TS active)

(0) Level 1 OK
(1) Class 1 Trigger
(2) Class 2 Trigger
(3) Class 3 Trigger
(4) - (7) Unused
(8) - (15) Level 1 Accept Pattern Output
(16) - (21) ROC code
(22) - (31) Unused

The VME address of a memory location is related to the input Trigger Pattern by the equation:

Memory Address (hex) = 4 * Trigger Pattern + 4000

where

Trigger Pattern (0) - (11) = Latched Level 1 Trigger Inputs (1) – (12).

Triggering

A Level 1 Trigger signal is admitted by the TS only if the input channel has been enabled. When operating in the *Common Strobe Mode* (Trigger Control Register (0x8), bit 0 = 0), *the Level 1 Trigger* input signals must be in coincidence with the *Common Strobe* signal.

Before entering the prescale circuitry the trigger signal is regenerated (on occurrence of its rising edge) as a 15 ns pulse. The prescale circuitry requires a pulse at least this wide to function properly. The regeneration rather than an input specification is necessary because when the *Common Strobe Mode* is used, the resulting overlap pulse that is fed to the prescale circuitry could be narrow enough to violate any specification.

Channels 9-12 have no prescaling feature but are otherwise identical to channels 1-8. This insures that the propagation delays will be roughly the same for all input channels.

After prescaling the 12 trigger channels are collected into a 12-fold OR signal. The leading edge of the OR signal sets a latch if the TS is ready. The TS is ready if all of the following are true: the GO bit is set (CSR 1 (0) = 1), no TS cycle is currently active, the *Front End Busy* input is not asserted, and the *External Inhibit* input is not asserted. From this latched trigger signal a trigger gate signal is generated. Trigger pulses from the 12 channels that are in coincidence with this gate are individually latched. The latch pattern determines the address that is applied to the look-up memory. The data from this memory location fixes the status of the trigger (accept or fast TS reset). If the trigger is accepted the data also defines the trigger class, the pattern of *Level 1 Accept* signals generated, and the *ROC Code* (i.e. event type) for the event.

The trigger gate width for the TS as supplied defines a simultaneous trigger resolution time of 10 ns. That is, if an input trigger signal on any channel has a leading edge within 10 ns of the first such trigger signal, it is included among those that are latched to determine the look-up memory address. Adjusting a timing resistor and positioning jumpers on the board permits the user to select resolution times of up to 100 ns.

When the data from the memory is valid it is strobed by a delayed signal derived from the trigger gate. If the trigger pattern is acceptable (*Level 1 OK* bit asserted) the *Level 1 Accept* signals (1)-(8) are driven out as programmed along with the Level 1 OK signal (*Level 1 Accept (0)*). These 9 signals are in time with each other, and are issued 42 ns later than the input trigger's leading edge ("insertion time"), assuming a 10 ns simultaneous trigger resolution time. (An increase in the trigger resolution time beyond 10 ns will increase the insertion time by the same amount.) The Level 1 OK signal also starts the main sequencer (50 Mhz) of the TS. The pattern of *Level 1 Accept* signals will remain asserted until the sequencer has completed its cycle. The latched trigger is then reset and the TS can accept new triggers.

If the trigger pattern is not acceptable (Level 1 OK bit not asserted), the TS resets its logic with no *Level 1 Accept* signals issued. The total time from the rejected input trigger to the trigger being re-enabled is the insertion time plus 15 ns.

Optional Trigger Configurations

There are optional ways to configure the triggering behavior of the module. These add some flexibility in its use. The options may be used simultaneously.

Option 1 - (Fast Mode) In this mode, all *Level 1 Accept* signals are driven promptly (10 ns delay) 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 (*ROC Code*), 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. Select the Fast Mode by programming CSR 2 (15) = 1.

Option 2 - Channels 9-12 are configured as a group not to contribute to the OR trigger signal. Input signals on these channels cannot themselves generate a latched trigger and start a TS cycle. However, they continue to be latched when in coincidence with a latched trigger originating from channels 1-8. This allows the signals of channels 9-12 to serve as in-time vetoes (by using the TS fast reset capability), or simply as additional in-time information with which to determine the *Level 1 Accept* pattern and *ROC Code* for the event. Select Option 2 by programming CSR 2 (14) = 0.

Option 3 - Channel pairs 9-10, and 11-12 can be configured to bypass pulse regeneration. For a channel pair so configured, an input *level* (rather than an *in-time edge*) may be used to tag latched triggers when Option 2 is used. Select Option 3 by setting the Trigger Control Register bits (13) or (14).

Using the Trigger Supervisor Scalers

The Trigger Supervisor has a set of 32-bit scalers on board. An individual scaler consists of a counter and register, each 32-bits wide. The action of reading a scaler latches the current counter value into its associated register, and enables the data from this register onto the VME bus. Reading a scaler in no way interferes with its counting.

When we read a set of scalers *sequentially,* we are latching their counter values at different times. We also have the capability of latching the counter values of all scalers *simultaneously*. Writing a '1' to bit 23 of the Scaler Control Register (0x7C) does this. While bit 23 of this register remains set, the latching action of any scaler read is disabled. Once we have read all the scalers of interest, we can clear this control bit. In this way we can take a single 'snapshot' of all the scalers. None of this interferes with the actual counting of the scalers.

We have the capability to reset the counts of any set of scalers by writing the appropriate bit pattern to the Scaler Control Register (0x7C). The resetting action occurs *simultaneously* for all scalers selected for reset when the write to this register occurs.

The six scalers of the Trigger Supervisor that are not dedicated can be assigned to monitor selected signals by programming the Scaler Assign Register (0x78). We can choose from a set of 16 important signals to monitor.