The Advanced Photon Source Event System*

Frank R. Lenkszus and Robert Laird
Argonne National Laboratory

ABSTRACT

The Advanced Photon Source, like many other facilities, requires a means of transmitting timing information to distributed control system I/O controllers. The APS event system provides the means of distributing medium resolution/accuracy timing events throughout the facility. It consists of VME event generators and event receivers which are interconnected with 100Mbit/s fiber optic links at distances of up to 650m in either a star or a daisy chain configuration. The systems event throughput rate is 10Mevents/s with a peak-to-peak timing jitter down to 100ns depending on the source of the event. It is integrated into the EPICS-based APS control system through record and device support. Event generators broadcast timing events over fiber optic links to event receivers which are programmed to decode specific events. Event generators generate events in response to external inputs, from internal programmable event sequence RAMs, and from VME bus writes. The event receivers can be programmed to generate both pulse and set/reset level outputs to synchronize hardware and to generate interrupts to initiate EPICS record processing. In addition, each event receiver contains a time stamp counter which is used to provide synchronized time stamps to EPICS records.

INTRODUCTION

The Advanced Photon Source Event System provides a means of distributing medium resolution timing to distributed control system I/O controllers. A timing event is a numeric code broadcast over a fiber optic cable plant by event generators. An event generator can generate timing events in response to external inputs, from programmable internal sequence RAMs and upon command from the I/O controller processor. The numeric code is decoded and acted upon by event receivers located in the distributed I/O controllers. Event receivers, under software control, may generate hardware outputs and/or generate an interrupt to the I/O controller processor. Both the event generator and event receiver are A16/D16 single-width VME modules.

GENERAL

A timing event is an 8-bit number that is broadcast over a cable plant. Since serial data transmission is used, a single fiber is sufficient to connect an event receiver to an event generator. Event generators may be cascaded to expand capability. Event receivers may be connected in either a star or a daisy chain configuration. There is, however, a limit to the number of event receivers that can be daisy chained since the serial bit stream out of a receiver is not decoded and regenerated but merely repeated through a buffer/driver. Figure 1 shows an example of how event generators and event receivers may be interconnected.

The event system uses the TAXIchip™ (Transparent Asynchronous Transmitter/Receiver Interface [1] to provide serial links between event generators and event receivers. The TAXIchips provide the parallel-to-serial, serial-to-parallel conversions and link management functions. The serial links run at 100Mbits/s, but since an 8-bit event is encapsulated in a 10-bit packet, the maximum event transfer rate is 10Mevents/s. Our tests show the system works reliably at distances of 650m. We have not tested longer link lengths.

THE EVENT GENERATOR

The event generator is a single-width VME module which generates timing events in response to several stimuli. Figure 2 shows a simplified block diagram of the event generator. A 10MHz oscillator sets the TAXI transmitter's operating frequency. The TAXI transmitter multiplies the operating frequency by a factor of 10 to create the 100MHz bit rate. The buffered 10MHz clock output of the TAXI transmitter is used as the clock for the event generator's synchronous internal logic.

Events are generated in response to external inputs, sequence RAM contents and register writes from the VME bus. In addition, a fiber-optic input is provided which accepts the output of an upstream event generator and is used to cascade event generators. Each potential source of events has an independent enable/disable control.

Eight edge-triggered maskable inputs are provided. Each of these inputs has an associated mapping register which is loaded with the event code to be generated upon receipt of the input.

Two 32k event sequence RAMs are included. Each sequence RAM accepts an external clock and external start. The external clock rate, 1MHz maximum, determines the time resolution and maximum time duration of an event sequence. Upon receipt of the external trigger, the sequence RAM is stepped through sequentially at the external clock rate. The contents of each location may contain an event code. Non-null codes (contents not equal to "0") are transmitted and the null code, 0, is ignored. The time interval between the start input and a generated event code is determined by the clock rate and the RAM address at which the event code is stored. A reserved code, the "end sequence" event code, may be placed in the RAM to define the end of the sequence. Another reserved code, the "freeze sequence" code, when encountered, will cause the sequence to suspend. The sequence will resume upon receipt of another "start." Neither the "end sequence" nor the "freeze sequence" codes are transmitted.

*Work supported by U.S. Department of Energy Office of Basic Energy Science under Contract No. W-31-109-ENG-38

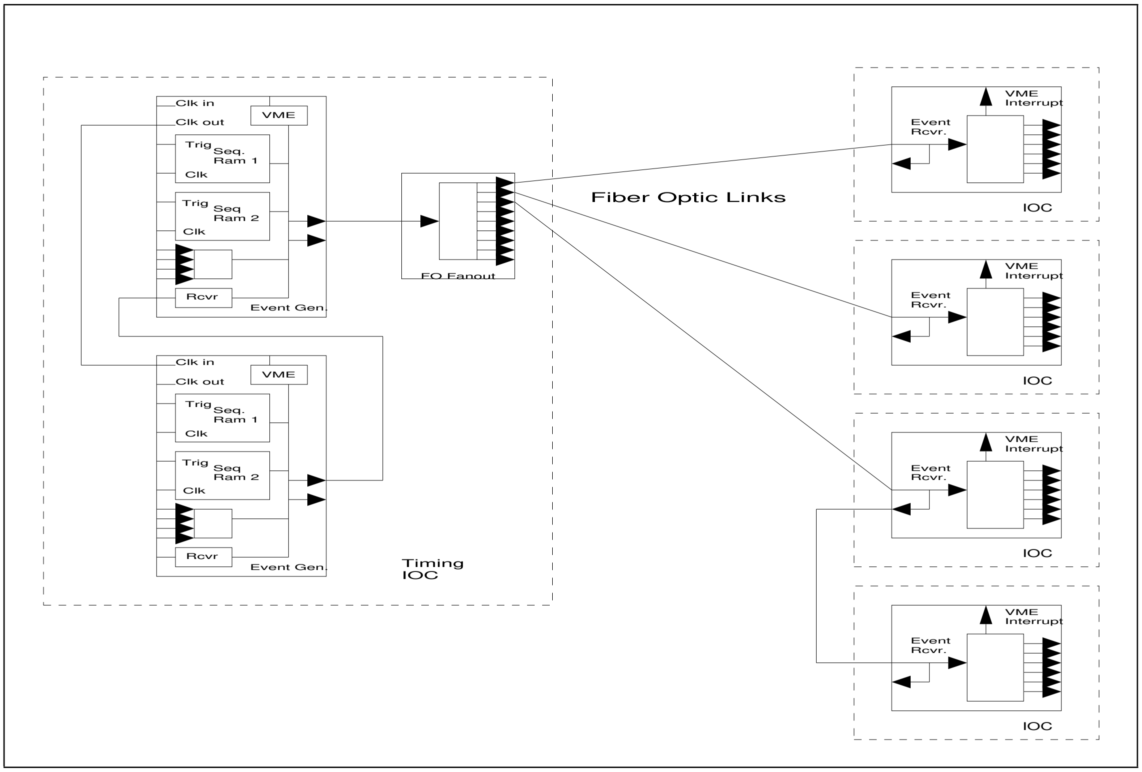

Figure 1 Example Event System Interconnection

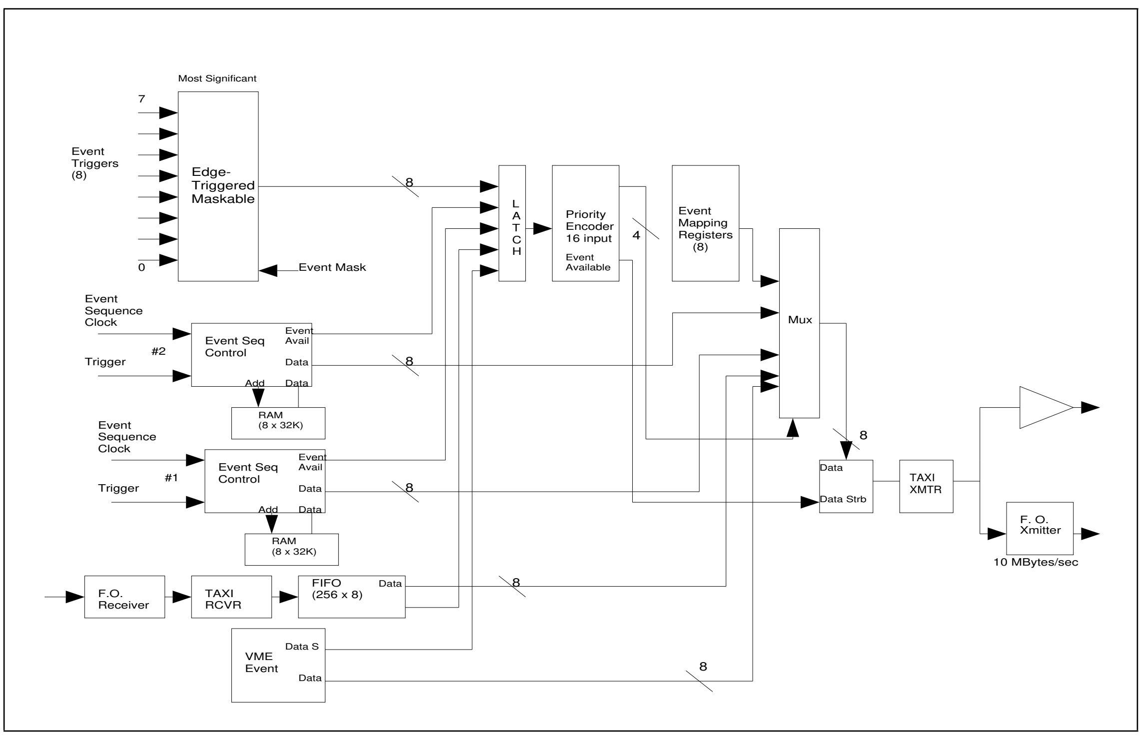

Figure 2 Event Generator Block Diagram

Each event sequence may be placed in one of four modes: normal, single sequence, recycle sequence, and alternate. In normal mode, the sequence is executed once for each start. In the single sequence mode, the sequence is executed once upon receipt of a start. Upon completion of the sequence, the sequence enable bit is reset, causing subsequent starts to be ignored. The recycle mode causes the sequence to continuously repeat after receipt of an initial start. The alternate mode connects the start and clock of sequence RAM 2 to the respective inputs of sequence RAM 1 so that both sequence RAMs receive a common clock and a common start. The two sequence RAMs are used in a foreground/background mode. The background RAM sequence may be modified, while the foreground RAM generates the sequence. The role of the two RAMs is then switched between the end of the sequence and the next start. Thus, in this mode, seamless timing changes may be made to the event sequence. In fact, a software slider can be attached to an event code and used to move the event code's time of occurrence. As the slider is moved, the background RAM is updated with a new position for the event and switched to the foreground. The two RAMs "alternate" between foreground and background as the slider is moved.

Each event RAM is randomly addressable via its own address register. An auto-increment control bit is provided for each of the address registers. When this bit is set, the address register is automatically incremented upon each VME access. Conversely, if the bit is reset, the address register is not incremented on each VME memory access. A null fill command is provided for each of the RAMs which zeros the entire RAM. The event generator has an on-board lithium battery which preserves RAM contents on power down.

In practice, the RAMs tend to be sparsely filled. Our worse case to date is 15 event codes in a sequence. We considered using a scheme based on storing delta time/event code pairs, but felt that the additional hardware and software complication was not worth any potential benefit.

The third way to generate an event code is via a VME write of an event code to the VME event register. The value written to this register will be transmitted as an event code. An example use of this method is the event system heartbeat. The heartbeat event, another reserved event code, is used by the event system to verify operation and link continuity. The heartbeat event is generated by the VME processor periodically writing the heartbeat event code to the VME event register.

A fiber optic input is provided to receive the serial event code stream from other additional event generators. This provides the mechanism for cascading event generators. The incoming event stream is converted to an 8-bit byte by a TAXI receiver. Received event codes are queued in a 256-deep fifo stack for retransmission. Through this mechanism the event streams of multiple event generators are combined into a single event stream. The TAXI receiver is phase locked to the upstream TAXI transmitter. Thus, since the onboard TAXI transmitter uses its local oscillator for its time base, the onboard TAXI receiver and onboard transmitter are not normally phase locked. As a result, an additional peak-to-peak jitter of 100ns is incurred for each unsynchronized event generator that an event passes through. For event generators in close physical proximity, this jitter can be eliminated by driving the TAXI transmitter clock on each "slave" from the clock of a "master." The event generator has a front panel clock input and clock output to permit multiple event generators to be run synchronously.

Since events are generated from multiple asynchronous sources, collisions will occur. The event generator uses a priority encoder to resolve collisions. The order of priority with highest first is: externally triggered events, RAM sequence 1, RAM sequence 2, TAXI receiver, VME-generated events. A collision causes the lower priority event to be delayed by up to 100ns to allow the higher priority event to complete transmission.

THE EVENT RECEIVER

The event receiver, a single-width VME module, receives the serial event stream over a fiber optic cable. A simplified block diagram is shown in Figure 3. The event stream is buffered and retransmitted via a fiber optic transmitter. This output may be used to cascade event receivers. The on-board TAXI receiver converts the serial event stream to an 8-bit parallel stream.

The event receiver has fourteen 100ns-pulse outputs, seven set/reset flipflop outputs, and four programmable delayed-pulse outputs. The delayed-pulse outputs offer programmable delay or width of up to 1.6 seconds. The actual range of programmable delay and width of a channel depends on which of three different versions of the delay generator PLD is socketed for that channel. Any and all of these outputs may be generated upon receipt of any event code. In addition, the event receiver may generate a VME interrupt in response to any event. The response of the event receiver to a particular event is determined by a mapping RAM that maps event codes into responses.

Incoming events are applied to the address lines of two 16-bit by 256 word mapping RAMs. The contents of the location corresponding to an event code determines what if any action will be taken by the event receiver upon receipt of that event code. Only one mapping RAM is active at a time. The inactive RAM may be modified and then selected as the active RAM. This permits bumpless modification of the mapping RAM on a running system. Both RAMs are battery backed up with an on-board lithium battery.

The format of the mapping RAM is shown in Figure 4. The 14 least significant bits determine what hardware outputs will be generated upon receipt of an event. The bits are multi-functioned. For example, setting bit 1 in the map location for a particular event code can cause the 100ns-pulse output number 1 to be generated, set flip-flop number 1, and/or trigger delay pulse output number 1. Each of the outputs has an enable/disable bit. Which output(s) will be generated when the event is received depends on the state of the enable bits. In this case, if the 100ns-pulse output number 1 is disabled, and flip-flop number 0 is disabled, but delayed-pulse number 1 output is enabled, the delayed-pulse output but neither the flipflop nor the 100ns-pulse outputs will be generated when this particular event is received. The set/reset of the seven flipflops are controlled by odd/even pairs of mapping RAM bits. Since there are only four delayed-pulse outputs available, only the four least significant bits of the mapping RAM are used to control delayed-pulse outputs.

The event receiver provides hardware to support synchronized time stamps. Two reserved event codes are decoded internally to control the time stamp counter: an increment time stamp counter code and a reset time stamp counter code. Upon receipt of the reset time stamp counter code, all event receivers reset their time stamp counters. Similarly, upon receipt of an increment time stamp counter event code, all event receivers advance their time stamp counters by one. Thus all event receivers (assuming they are all downstream of the event generator issuing the time stamp events) maintain a synchronized relative time stamp which may be read via VME. We presently are using a 1kHz clock to generate the increment time stamp event. The time stamp counter may be incremented by a 1MHz internal clock derived from the TAXI receiver's clock instead of the increment time stamp event. Since all the TAXI receivers are phase locked to the incoming bit stream clock, the 1MHz clocks of all event receivers are phase locked.

As mentioned earlier, each event can cause a VME interrupt to be generated. The most significant bit in the mapping RAM location for a particular event code determines if an interrupt is generated. If that bit is set for a particular event code, upon receipt of that event an interrupt is generated and the event code and the 24 least significant bits of the time stamp counter are placed in a 32-bit by 256-word event/time fifo. The interrupt service routine reads the fifo to determine the number of the event generating the interrupt and the value of the time stamp counter when that event occurred.

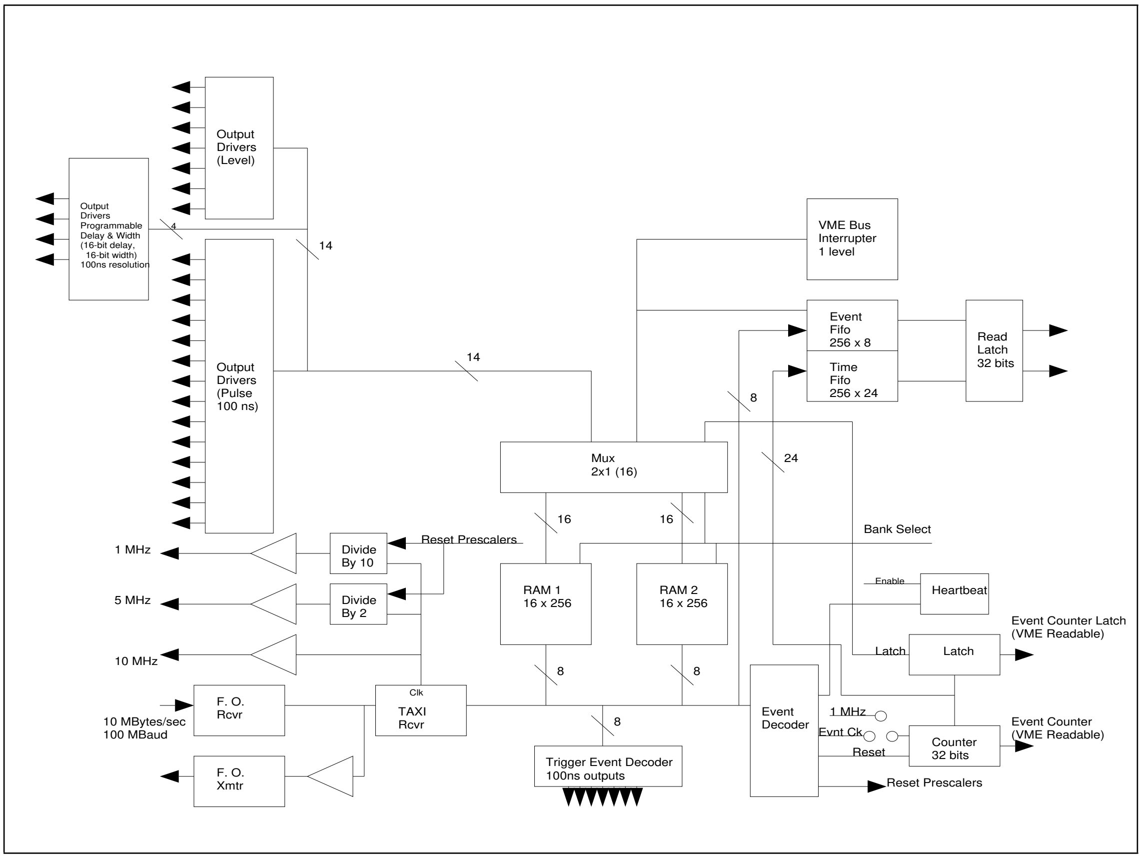

Figure 3 Event Receiver Block Diagram

In addition to generating an interrupt or generating outputs, an event may cause the current value of the time stamp counter to be latched in a VME-readable 32-bit latch. We have not made use of this feature to date.

The event receiver provides three frequency outputs: 10MHz, 5MHz, and 1MHz. These frequencies are derived from the TAXI clock which, as mentioned previously, is phase locked to the incoming bit stream. The 5MHz and 1MHz frequencies are generated from the 10MHz TAXI clock by synchronous counters. A reserved event code, the "reset event receiver prescalers" event code, causes the prescalers to synchronously reset. Thus the three frequency outputs are phased the same across all receivers. There will, however, be time skew due to differing propagation delays resulting from different signal path lengths.

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VME Interrupt	Latch Time	Pulse 13 Set FF 6	Pulse 12 Reset FF 6	Pulse 11 Set FF 5	Pulse 10 Reset FF 5	Pulse 9 Set FF 4	Pulse 8 Reset FF 4	Pulse 7 Set FF 3	Pulse 6 Reset FF 3	Pulse 5 Set FF 2	Pulse 4 Reset FF 2	Pulse 3 Set FF 1 Dlyd Puls 3	Pulse 2 Reset FF 1 Dlyd Puls 2	Pulse 1 Set FF 0 Dlyd Puls 1	Pulse 0 Reset FF 0 Dlyd Puls 0

Figure 4 Event Receiver Mapping Ram Data Format

The module can generate VME interrupts for event received, heartbeat missed, event fifo full, and TAXI receiver error.

EXPERIENCE AND USE

The first event receivers and generators were installed in the APS control system in January of 1994. Presently, we have 52 event receivers and 6 event generators in use.We have not experienced any hardware failures to date. A test was run over a seven-day period, transmitting and receiving an event at 250kHz over a 625m fiber optic cable. No errors were detected before the test was terminated. The event was transmitted and received over 1.5 x 10^{11} times without error. Another test involved using the time stamp counters to verify event system reliability. This test was run on the installed system which had its normal event traffic. The time stamp counters in four event receivers were reset (with the reset time stamp counter event code) and allowed to accumulate time stamp events. The time stamp event was generated at 1kHz. The test was terminated after approximately 90 hours of running time. The time stamp counters of each of the four event receivers agreed to the tic (326,187,516 tics to be exact). While these tests are not rigorous, we believe that the system reliably transmits and receives events.

Presently, the APS event system uses 37 of the 255 possible event codes. The event rate is approximately 5.2k events/s. The bulk of that rate is due to two sources: the 1kHz time stamp event and a 4kHz orbit feedback synchronization clock.

The hardware outputs have proved useful for a number of different purposes. They have been particularly useful for quickly providing timing for a new or unforeseen application. In most instances, a properly timed event was already available, so all that needed to be done was to program the appropriate event receiver's mapping RAM to generate the desired type of output. No cable pulling or termination was required.

The event receiver and generator are fully supported under EPICS. Five record types are available for use with the event system. The event generator record provides the basic interface to the module. Event generator RAM records provide the means to program the event sequence RAMs. The event receiver is controlled through its own record type with an additional record type to specify mapping RAM contents. The fifth record type available is the standard EPICS event record. This record has proved to be extremely useful for synchronizing database record processing with external processes. The event record provides the mechanism for triggering record processing upon receipt of an event. The event receiver is programmed via an event receiver RAM record to generate an interrupt upon receipt of the desired event. The EPICS event record is set to process upon an I/O interrupt with the desired event. Under these conditions, the EPICS driver causes the event record to process upon receipt of the desired event. Any desired set of database records can then be caused to process through the EPICS forward link mechanism.

The synchronized time stamps are also fully supported under EPICS. Three modes are available for time stamping the processing of an EPICS record when the event system is used. Under the "normal" mode, the record's time stamp is obtained from the VME processor's 60Hz clock. In this mode, the time stamps across multiple processors will be within one tic of 60Hz. The second mode is the "event" mode. In this mode, a record's time stamp is the last time of occurrence of a selected timing event. The third mode is the high resolution mode wherein the record obtains its time stamp by reading the current value of the event receiver's time stamp counter. This mode causes a VME access each time the record is processed and so should be used judiciously. As an aside, in the absence of an event receiver, EPICS uses Network Time Protocol (NTP) to synchronize timestamps across multiple processors. It's been our experience that timestamps are synchronized to no better than 100ms across multiple processors when the EPICS NTP is used.

We plan to use the high resolution time stamps to help unfold the sequence of hardware trips that may occur when beam is lost. A likely scenario is that an initiating occurrence will cause beam to be lost and also cause a cascade of trips. EPICS monitors all trip sources. Inspection of the high resolution time stamps should help to determine the failure sequence.

SUMMARY

The APS Event Timing System has been in production use for nearly two years. We have found it to be extremely useful for generating timing outputs at distributed locations. The ability to trigger EPICS record processing upon occurrence of a timing event has been extremely useful.

REFERENCES

[1] Advanced Micro Devices, Inc., Am7968/AM7969 TAXIchip Handbook, Publication BAN-13.5M-5/94-0, Sunnyvale, CA, April 7, 1994