AUSTRALIA TELESCOPE
NATIONAL FACILITY
PC EVENT GENERATOR

Last Updated: 7 July 2000

CONTENTS

1.0 General.

The PC based Event Generator is the next generation event generator based on the earlier Q-Bus version. As a result of advances in technology, programmable gate arrays are used to provide enhanced features. Key features are:

* Inbuilt Frame Grabber (entire frame),
* Inbuilt Waveform generator.
* Reduced power consumption.
* Pending Event purging.
* Near 100% connector pinout compatibility with Q-Bus version.
* Accepts either complementary (9 pin D) or single ended (BNC) distributed BAT clock signal.
* 1 second (whole) interrupt tick.
* Timed interrupt (software selectable) from any one of the 16 standard Event output lines.
* Waveform generator enable control (software selectable) from any one of the 16 standard Event output lines.
* Host Event write and read.
* Selectable free run Event Strobe.
* AT ISA bus standard supporting both 8 bit and 16 bit transfers.
* ORed interrupt and DMA Request.
* Compactness and 2 layer PCB.
* Reduced cost. The Waveform generator components may be left off to produce a minimal (low cost) version.

2.0 Specifications:
Timed Event Generator. Any 16 bit Event can be programmed to occur at any microsecond count within 2e45 microseconds from the time of loading.

Waveform Generator. Any single waveform with frequency from 500KHz down to ~0.06 Hz. or any group of 8 binary waveforms with maximum frequency presetable from 500KHz to 15.26Hz.

All output signals and all input signals TTL compatable.
The PC Event Generator requires a full 16 bit slot on the ISA Bus.

2.1 BAT Input.
LK2: 2-3
9 pin D male Connector:

Pin	Signal
1	+Bat (TTL).
6	-Bat (TTL).
5	Gnd.

LK2: 1-2
BNC Connector:

Pin	Signal
Inside	+BAT (TTL).
Outside	Gnd.

2.2 Outputs.

2.2.1 EVENT SIGNAL 34 Pin Header Connector.

PIN	Description	PIN	Description
1	EVENT 0 +	2	EVENT 0 -
3	EVENT 1 +	4	EVENT 1 -
5	EVENT 2 +	6	EVENT 2 -
7	EVENT 3 +	8	EVENT 3 -
9	EVENT 4 +	10	EVENT 4 -
11	EVENT 5 +	12	EVENT 5 -
13	EVENT 6 +	14	EVENT 6 -
15	EVENT 7 +	16	EVENT 7 -
17	EVENT 8 +	18	EVENT 8 -
19	EVENT 9 +	20	EVENT 9 -
21	EVENT 10 +	22	EVENT 10 -
23	EVENT 11 +	24	EVENT 11 -
25	EVENT 12 +	26	EVENT 12 -
27	EVENT 13 +	28	EVENT 13 -
29	EVENT 14 +	30	EVENT 14 -
31	EVENT 15 +	32	EVENT 15 -
33	EVENT STROBE +	34	EVENT STROBE -

2.2.2 TIMING SIGNALS 50 Pin Header Connector.

PIN	Description	PIN	Description
1	8MHz Clock	2	GND
3	2MHz Clock	4	GND
5	1MHz QCLK	6	GND
7	1MHz TCLK	8	GND
9	1MHz DCLK	10	GND
11	500 KHz Clock	12	GND
13	250 KHz Clock	14	GND
15	125 KHz Clock	16	EVENT 8
17	62 PPMS	18	EVENT 9
19	31 PPMS	20	EVENT 10
21	16 PPMS	22	EVENT 11
23	8 PPMS	24	EVENT 12
25	4 PPMS	26	EVENT 13
27	2 PPMS	28	EVENT 14
29	XFS	30	EVENT 15
31	MSCLK (N/I)	32	GND
33	X1SEC	34	GND
35	X1SCLK (N/I)	36	GND
37	COUNTER-OUT (N/I)	38	COUNTER-IN (N/I)
39	+5V *****	40	EXT-TRIG0 *****
41	EXT-INTERRUPT	42	GND
43	+5V *****	44	EXT-TRIG1 *****
45	UNUSED I/P *****	46	GND
47	BUS-EVENT *****	48	GND
49	CLK-DATA	50	GND

***** - Different function from Q-Bus version of Event Generator.
N/I - Not Implemented.

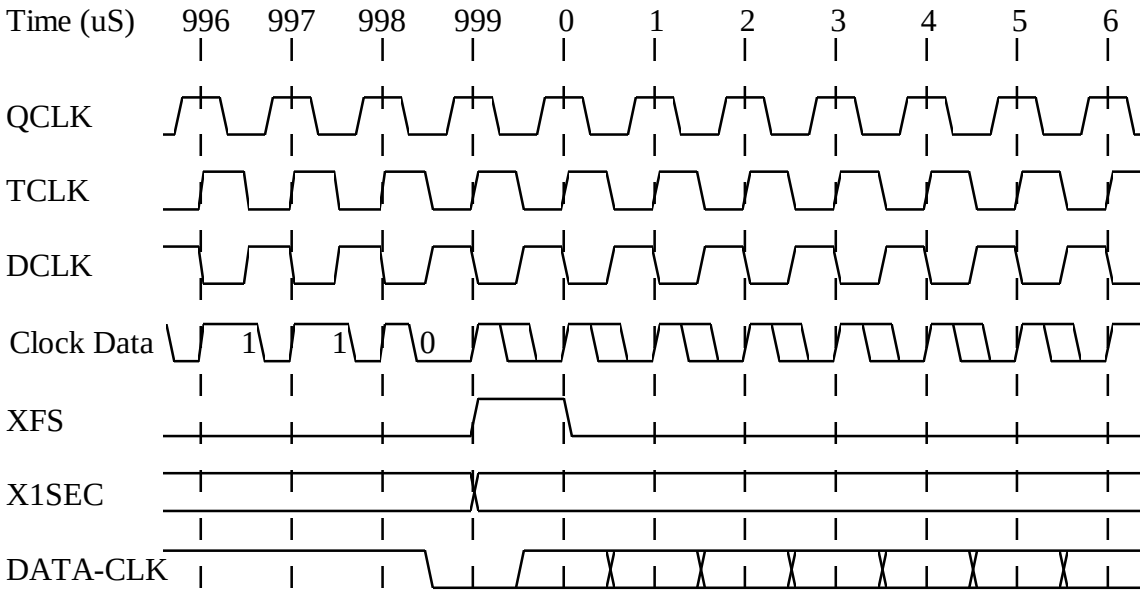
2.2.3 WAVEFORM SIGNAL 16 Pin Header Connector.

PIN	Description	PIN	Description
1	WAVEFORM 0 +	2	WAVEFORM 0 -
3	WAVEFORM 1 +	4	WAVEFORM 1 -
5	WAVEFORM 2 +	6	WAVEFORM 2 -
7	WAVEFORM 3 +	8	WAVEFORM 3 -
9	WAVEFORM 4 +	10	WAVEFORM 4 -
11	WAVEFORM 5 +	12	WAVEFORM 5 -
13	WAVEFORM 6 +	14	WAVEFORM 6 -
15	WAVEFORM 7 +	16	WAVEFORM 7 -

2.3 Signal Timing.

2.3.1 Frame Timing.

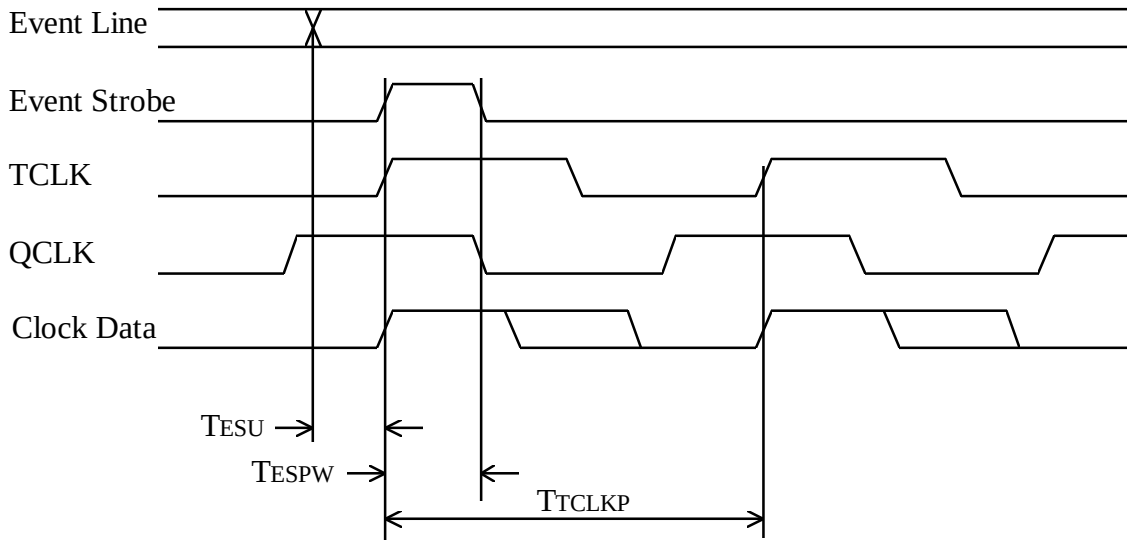
The fundemental clock in the event generator is TCLK. The rising edge represents the start of each microsecond period. DCLK is the inversion of TCLK, and is used to clock in the Clock Data, ie the data sent by the BAT Clock. QCLK is TCLK shifted by -250nS and hence is the quadrature of TCLK. XFS is the external Frame Sync indicator and occurs every clock frame which is 1ms. Frame Sync is characterised by a sequence of at least 24 consecutive 1's and ending with the first 0 in the clocks data stream. X1SEC is the external whole second indicator and is characterised by a sequence of at least 40 consecutive 1's and ending with the first 0 in the clocks data stream. As this definition also includes the Frame Sync characteristic, X1SEC goes true at the same time as XFS and since it lasts 1 frame period, will go false the next time XFS goes true, each second. DATA-CLK is the Clock data captured by DCLK.



FRAME TIMING

2.3.2 Event Timing.

Events become valid precisely 192nS before rising edge of Event Strobe. Rising edge of Event Strobe is concurrent with rising edge of TCLK, and lasts 250nS. Events may be programmed to occur at a prescribed time or by an asynchronous write from the host processor. Event Strobe may be also programmed to occur continuously, to allow, for example, external events to be piped into existing targets. In this case timing of Event Strobe with respect to TCLK is maintained, and occurs every microsecond.



EVENT TIMING

Time	Description	Value	Units
TESU	Event Setup	192	nS
TESPW	Event Strobe Pulse Width	250	nS
TTCLKP	TCLK Period	1000	nS

2.3.3 Waveform timing.

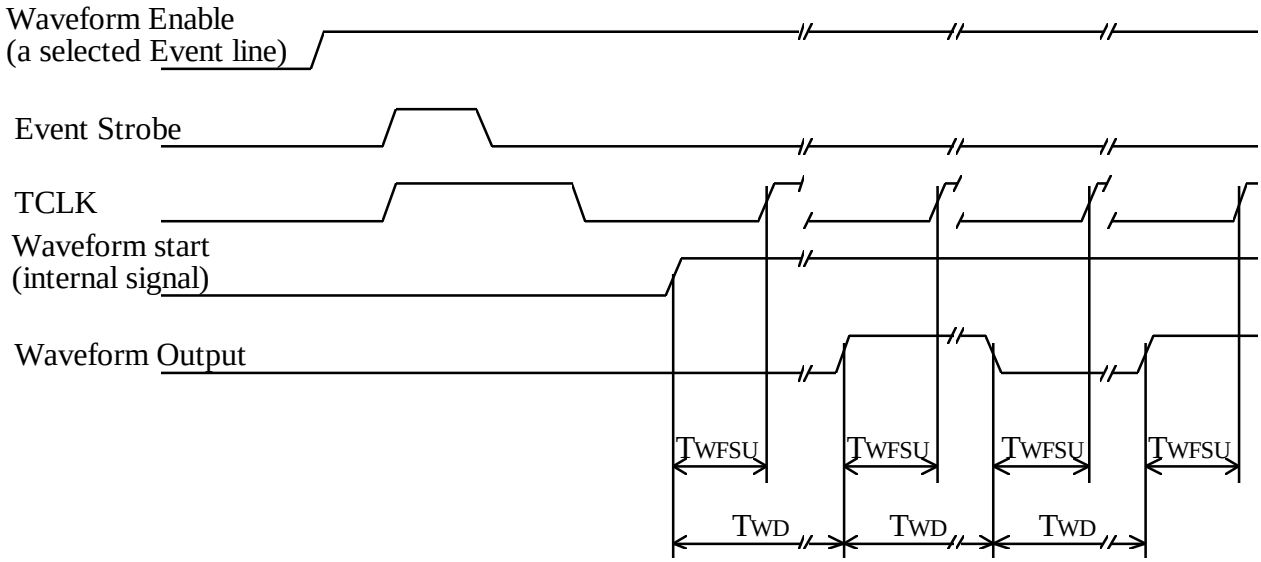
Waveform generation starts whenever both the chosen event line controlling the Waveform Generator goes Hi and that control line is enabled. As all Waveform generator signals are timed to occur 250 nS before TCLK, the process of waveform generation does not start until the following microsecond after the event line goes Hi. Similarly, Waveform generation does not stop until the microsecond after the chosen event line goes Lo. Whenever the Waveform Generator is not enabled, all outputs are Lo.

The duration of each waveform pulse (ie the time the signal is either Hi or Lo) is set by the value loaded into the Prescale register. This 16 bit register allows values from 0 to 65535. The Waveform Pulse duration may be calculated as:

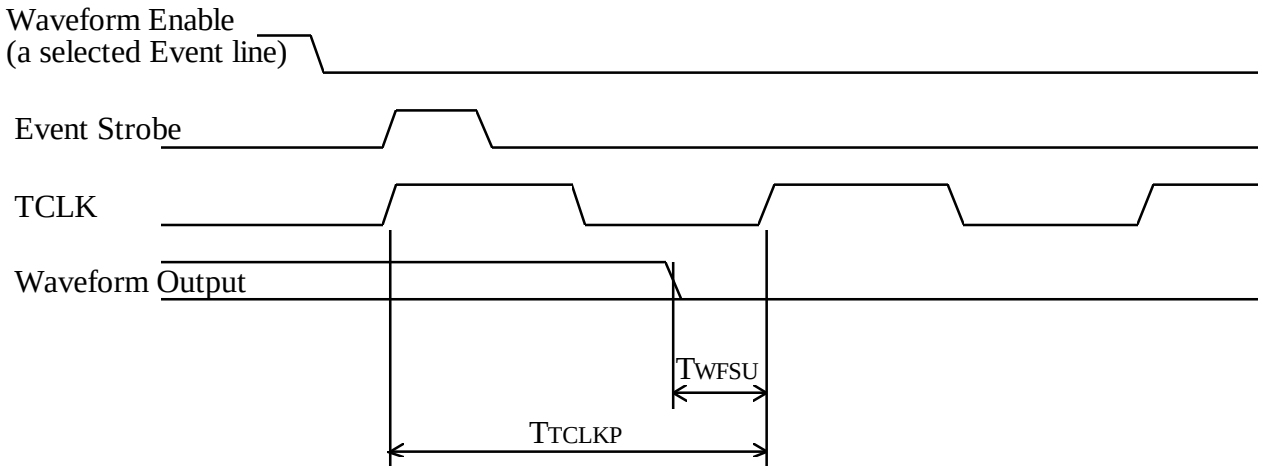
Waveform pulse duration = (Prescale value + 1) uS.

and

Waveform Period = 2 X Waveform pulse duration.



WAVEFORM TIMING (Starting)



WAVEFORM TIMING (Stopping)

Time	**Description**	**Value**	**Units**
TWFSU	Waveform Setup	250	nS
TTCLKP	TCLK Period	1000	nS
TWD	Waveform Duration	(Prescale + 1)	uS

2.4 Interrupts:

IRQ setting:

3, 4, 5, 6, 7, 9, 10, 11, 12, 15. Jumper selection.

Interrupt sources:

Reference FIFO empty, Reference FIFO half empty, Reference FIFO full, Late Event, Checksum error, False Sync, Missed Sync, Frame loaded, DMA Done, External interrupt, Event interrupt, Phase Locked Loop dropout, 1 Second tick. Event interrupt is software selected and enabled to come from any one of the 16 Event lines.

Control:

Bitwise selectable along with a master interrupt enable bit.

2.5 Serial Number PROM.

Each Event generator has a serial number PROM. This contains the serial number and other relevent information for each board. Data is held as a preamble followed by a NULL terminated ASCII string (as per the C language). The preamble is any number of 1's, followed by a 0 and then seven other bits (don't cares). The next bit is the start of the data. Each ASCII character is 8 bits and is read from the PROM least significant bit first.

2.6 Waveform Generator waveform selection table.

Each of the 8 outputs of the Waveform Generator may select any of the following signals, by setting the corresponding code in the appropriate field of the Waveform Selection Register.

Waveform Selection Table

Code	Waveform Signal
0	0
1	1 / (PS * 2) MHz
2	1 / (PS * 4) MHz
3	1 / (PS * 8) MHz
4	1 / (PS * 16) MHz
5	1 / (PS * 32) MHz
6	1 / (PS * 64) MHz
7	1 / (PS * 128) MHz
8	1 / (PS * 256) MHz
9	1 / (PS * 32) MHz Quadrature
A	250KHz
B	500KHz
C	0
D	0
E	1
F	1

NOTE: PS = "Prescale Register contents + 1"

2.7 Configuring the Waveform Generator.

The following registers must be configured to successfully run the Waveform Generator.

a). Event Output Control/Status Register.

Set the Waveform Generator Event Line Select field with the binary code of the event line that is chosen to control the Waveform Generator.

Set the Waveform Generator Event Line Enable bit to allow the selected event line to control the WFG. With this bit set, when the selected event line goes hi, the WFG will be run.

If controlling the WFG by non-timed events, it will also be necessary to set the Expert Mode bit in this register to allow the host processor to write directly to the Host Event Register.

b). Waveform Selection Register 1 & 2.

Set each 4 bit field corresponding to each Waveform Generator output with the binary code of the signal type desired. If a signal that uses the prescaler is chosen, then it will be necessary to set the prescaler register.

c). Generator Prescale Register.

Set the MS byte and LS byte prescale registers with the desired prescale value. This register need not be set if prescaler signals are not required.

See section 5.3 for more detail.

3.0 Some general considerations.

3.1 I/O address assignment.

The PC's I/O address bus is a 16 bit bus, which gives a maximum space of 64K bytes. Historically, the early PC (known as XT's) only provided 10 bits of the 16 bits for I/O devices to decode thus giving an I/O address space of 1K bytes. Consequently any older I/O device (including standard PC items of hardware) appear at the same spot in each of the 64, 1K blocks as they do not decode the top 6 bits of this bus. Thus the selection of addresses to locate the Event Generator should be done with some care. When multiple Event Generators are used, or are used in conjunction with other full 16 bit decoded boards, then these boards should possibly be located at addresses of 0x400 steps, for example, 0x0300, 0x700, 0xb00.

Address setting switch SW1.

Pole	Address line
1	A16
2	A15
3	A14
4	A13
5	A12
6	A11
7	A10*
8	A9*
9	A8
10	A7

*For default address 0x300, these switches OFF (1), all others ON (0).

Each Event Generator uses all 16 I/O address lines, and decodes the top 10 to occupies a block of I/O space of 64 bytes or 32 words (16 bits).

3.2 Interrupts.

Each Event Generator requires its own interrupt line. The earlier versions did allow sharing of interrupts, but this has proved un-reliable with some brands of CPU card.

3.3 Event generator response times.

It should be noted that due to a shortage of I/O pins on the Xilinx IC, some status signals are multiplexed into the various controller chips. The signals effected are not critical and are mostly only those that return status to the host processor. The delay between when the actual signal changes to when the signal in the XILINX follows can be upto 12 PC system clock cycles later. For most PC AT's the clock rate is around 8.0 to 8.33 MHz (usually the CPU clock rate divided by an integer value to give somewhere around this value). Any external signal feeding the edge triggered interrupt circuitry must last at least 12 PC system clock cycles. **This includes Timed Interrupt signals generated by programming the Event Generator output.** Any signal so affected is called a slow signal in this manual.

3.4 Miscellaneous.

Some bits in the various registers have a dual function in that the write value has a different meaning to the read value. Consequently, the modifications to a bit or bits in these registers by first reading its contents then ORing or ANDing a mask with the current contents, may lead to spurious results. Make sure the program first masks out these dual function bits.

3.4 PLL Lock warning.

PLL lock warning is done by monitoring the VCO voltage level. A window type monitor circuit is used where an upper and lower limit is set. To set this window perform the following operations.

1. Connect BAT signal to CLOCK input connector J2.
2. Connect a CRO to test points TP1 and TP2 and check if both signals are locked. Use a time base of around 100nS/div - 500nS/div and trigget off the rising edge of the REF signal. The VCO signal will be a falling edge, several nanoseconds later.
3. Turn the UPPER adjustment pot at least 10 turns clockwise or until the front PLL red LED goes out.
4. Turn the LOWER adjustment pot at least 10 turns anti-clockwise or until the front PLL red LED goes out.
5. Turn the UPPER adjustment pot anti-clockwise until the front PLL red LED turns on. Back this pot off about 2 turns (clockwise) from this point. The LED should be now off.
6. Turn the LOWER adjustment pot clockwise until the front PLL red LED turns on. Back this pot off about 2 turns (anti-clockwise) from this point. The LED should be now off.

7. Test PLL monitoring by disconnecting Clock signal. Front PLL red LED should light.

4.0 Register descriptions.

4.1 I/O Map.

Each Event Generator occupies a 32 byte address space on the I/O bus. Full 16 bit address decoding is done. There is the possibility of address conflict if older style boards which have only 10 bit address decoding are used in conjunction with this one. Care should be used when selecting the address. When used in conjunction with other Event Generators or with the Correlator interface board addresses should be set to 0x400 steps.

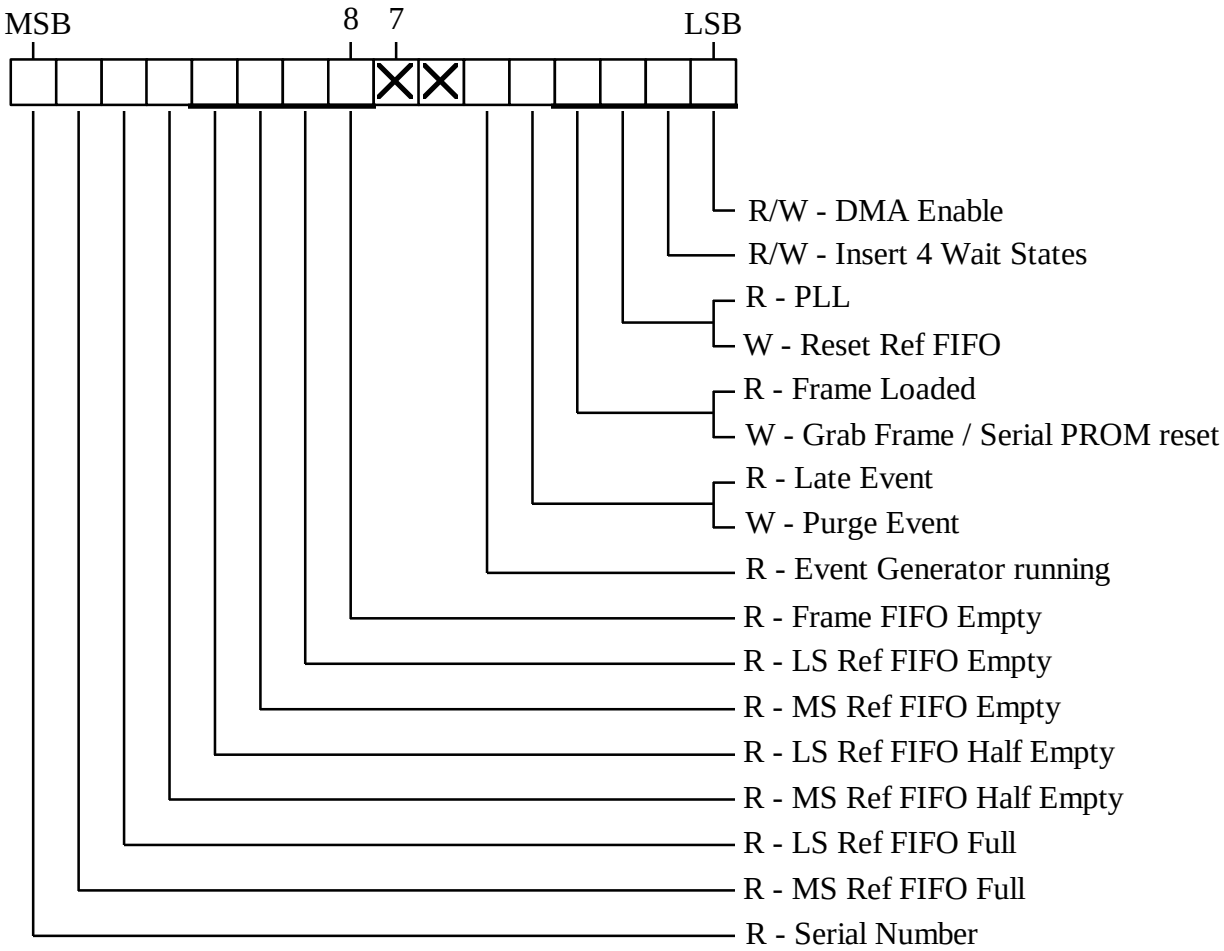
There are eleven, 16 bit non contiguous registers in the Event Generator. The following table shows their mapping reference to the boards base address.

Address	**Description**
Base	Master Control/Status register (Read / Write).
Base + 0x2	Interrupt Control register (Read / Write).
Base + 0x4	Interrupt Status (Read).
Base + 0x6	Reference FIFO (Write), Frame FIFO (Read).
Base + 0x8	Event Output Controller control/status register (Read/Write).
Base + 0xA	Event Output Controller current event (Read) and asynchronous event (Write).
Base + 0xC	Reserved.
Base + 0xE	Reserved.
Base + 0x10	Waveform Generator status register (Read/ Write).
Base + 0x12	Waveform Generator LS Byte Prescale count (Write).
Base + 0x14	Waveform Generator MS Byte Prescale count (Write).
Base + 0x16	Waveform Generator outputs 0 - 3 source select. (Read/Write).
Base + 0x18	Waveform Generator outputs 4 - 7 source select. (Read/Write).
Base + 0x1A	Reserved.
Base + 0x1C	Reserved.
Base + 0x1E	Reserved.

4.2 Master Control/Status Register.

This register allows control, setting and reading of basic control points within the Event Generator board.

MASTER CONTROL/STATUS REGISTER (Offset 0x0)



Mask	Type	Description.
0001	R/W	DMA Enable. When set, will cause the Event Generator to request the CPU to perform a DMA transfer on this board, according to the DMA controller setup. The DMA channel is selected by the jumpers on the Event Generator board. DMA will proceed in Single mode until a Terminal Count signal is sent from the CPU, at which time the DMA will halt.
0002	R/W	Insert 4 Wait States. When set the controller will insert a 4 system clock wait period into the access process. This is only necessary when accessing the serial number PROM. Thus for normal operations this bit should be cleared.

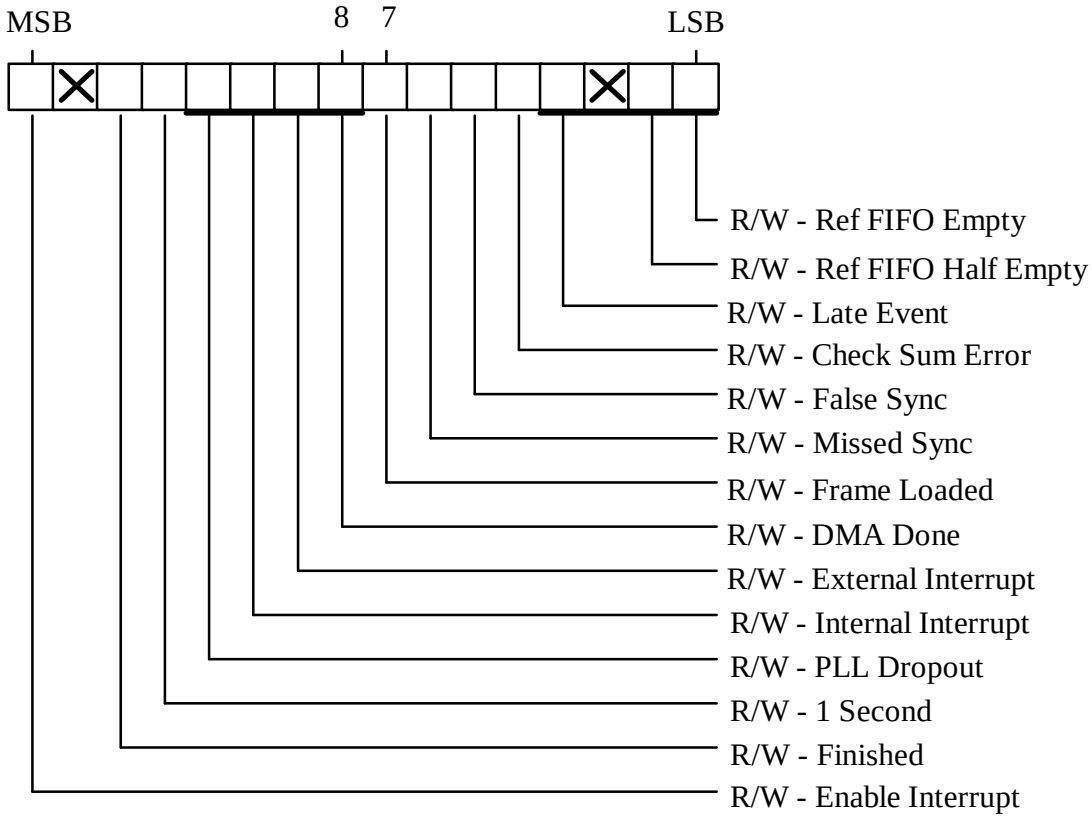
0004	R	PLL. This bit is the status of the Phase Locked Loop. When Lo the PLL is locked, when Hi the PLL is out of lock. The PLL lock thresholds must be adjusted correctly for this indicator to be valid. See section 3.4.
	W	Reset Reference FIFO. Writing a 1 to this location resets the reference FIFO.
0008	R	Frame Loaded. This bit indicates that a Time Frame has been read and loaded into the Frame FIFO. It is reset on issuing a Grab Frame and remains lo until the frame has successfully been retrieved. (Note a frame is only successfully retrieved when the frame CRC matches, there is no Missed Sync and no False Sync errors detected). This is a slow signal.
	W	Grab Frame/Serial Number PROM Reset. Writing a 1 to this location causes a frame to be grabbed from the time signal and resets the serial number PROM. When performing operations with the serial number PROM, set the Insert 4 Wait States bit.
0010	R	Late Event. This bit is the current state of the Late Event bit. When high the time in the Time Comparator is passed. This bit should be read in conjunction with the Late Event bit in the interrupt register, as the latter is only set by a positive edge of the Late Event signal.
	W	Purge Event. Writing a 1 to this location causes the pending event to occur and the next event held in the reference FIFOs to be loaded into the output queue.
0020	R	Event Generator running. This bit when hi indicates that the event generator is running. This means that there is an event pending and the time controller is waiting for the scheduled time to become valid.
0100	R	Frame FIFO Empty. When LO this bit indicated the Frame FIFO is empty. This is a slow signal.
0200	R	LS Reference FIFO Empty. When LO this bit indicated the least significant Reference FIFO is empty. This is a slow signal.
0400	R	MS Reference FIFO Empty. When LO this bit indicated the most significant Reference FIFO is empty. This is a fast signal.
0800	R	LS Reference FIFO Half Empty. When LO this bit indicated the least significant Reference FIFO is more than half full. This is a slow signal.
1000	R	MS Reference FIFO Half Empty. When LO this bit indicated the most significant Reference FIFO is more than half full. This is a slow signal.
2000	R	LS Reference FIFO Full. When LO this bit indicated the least significant Reference FIFO is full. This is a slow signal.
4000	R	MS Reference FIFO Full. When LO this bit indicated the most significant Reference FIFO is full. This is a fast signal.
8000	R	Serial Number. This bit is the current bit in the serial number ROM. The serial number for each board is stored in a serial ROM and can be read at almost any time. To read the serial number the ROM must first be reset, then the serial bit is read, then the ROM internal address counter incremented. The last two operations are repeated until the required number of bits is

read. To reset the ROM a Grab Frame is issued, and to increment its internal address register a Read Frame FIFO is done As the serial number PROM is a slow access device set the Insert 4 Wait States bit in this register. (Don't forget to reset it on completion of the PROM read operation). This is a slow signal.

4.3 Interrupt Control register.

This register allows enabling of individual interrupts, as well as disabling all interrupts. A Hi written to each bit enables that interrupt, while a Hi read from each bit indicates that that interrupt is enabled. No interrupt will occur if the 'Enable Interrupt' bit is not set. Disabling the individual interrupts only disables the connection to the interrupt circuitry. The capture logic is still operational and the result can be seen in the Interrupt Status register.

INTERRUPT CONTROL REGISTER (Offset 0x2)



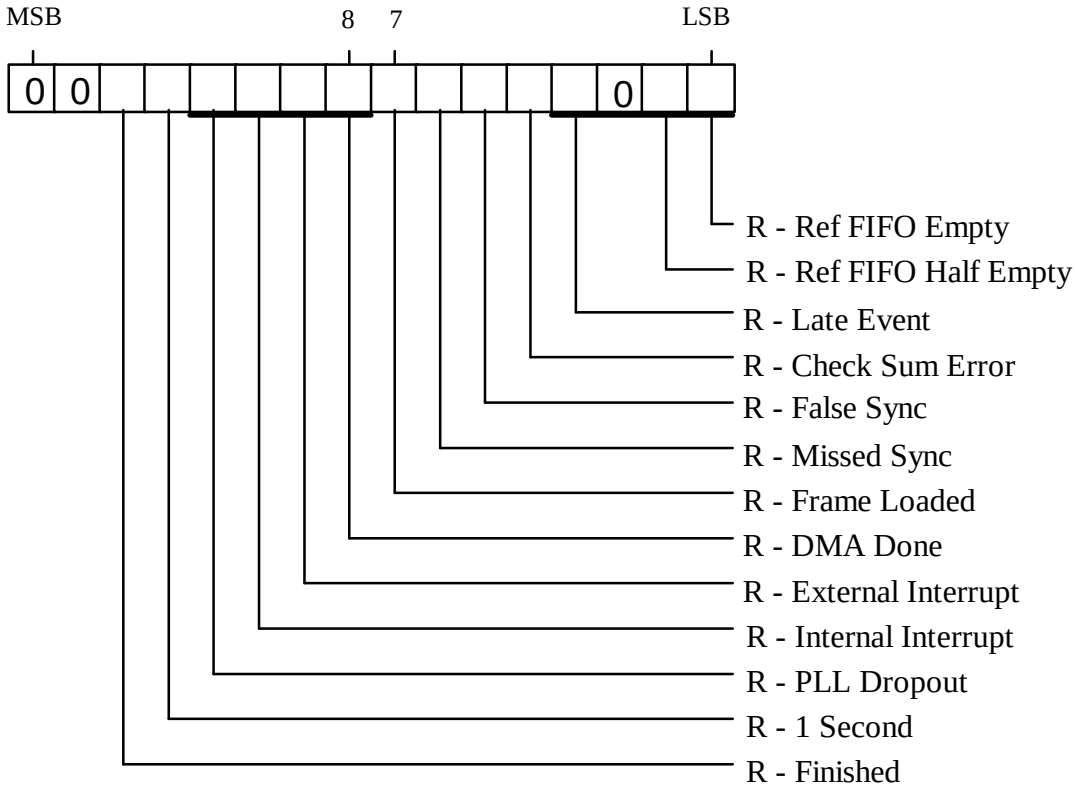
Mask	Type	Description.
0001	R/W	Reference FIFO Empty. When set, a Hi - Lo transition on either reference FIFO empty signals will cause an interrupt.

0002	R/W	Reference FIFO Half Empty. When set, a Hi - Lo transition on either reference FIFO half empty signals will cause an interrupt.
0004	R/W	Reserved.
0008	R/W	Late Event. When set, a Lo - Hi transition on the Late Event signal will cause an interrupt. The Late Event bit in the Status Register should also be checked to see if this line is not stuck in the high state.
0010	R/W	Checksum Error. When set, a Lo - Hi transition on the Checksum Error signal will cause an interrupt.
0020	R/W	False Sync. When set, a Lo - Hi transition on the False Sync signal will cause an interrupt.
0040	R/W	Missed Sync. When set, a Lo - Hi transition on the Missed Sync signal will cause an interrupt.
0080	R/W	Frame Loaded. When set, a Lo - Hi transition on the Frame Loaded signal will cause an interrupt.
0100	R/W	DMA Done. When set, a DMA completion will cause an interrupt.
0200	R/W	External Interrupt. When set, a Lo - Hi transition on the External Interrupt 1 signal will cause an interrupt.
0400	R/W	Internal Interrupt. When set, a Lo - Hi transition on the Internal Interrupt signal will cause an interrupt. This interrupt can be set to come from any one of the sixteen standard event signals.
0800	R/W	PLL Dropout. When set, a Lo - Hi transition on the Phase Locked Loop signal will cause an interrupt.
1000	R/W	1 Second. When set, a Lo - Hi transition on the 1 Second signal will cause an interrupt.
2000	R/W	Finished. When set, an interrupt will occur when the last event has been issued.
8000	R/W	Enable Interrupts. When set, all currently enabled interrupts will be enabled. When reset no interrupt will occur.

4.4 Interrupt Status register.

This register mimics the Interrupt Control register. Each allocated bit in the Interrupt Control register (excluding the Enable Interrupt bit) has the same bit allocated in this register. A Hi on any bit indicates that it is in interrupt state. Any set interrupt bits are reset on reading of the register. To get an accurate picture of which signal caused the interrupt, the contents of this register should be ANDed with the contents of the Interrupt Control register. It is recommended that this register be read using a 16 bit operation. If using 2 X 8 bit read operations to read this register, the least significant byte must be read first. The reading of the least significant byte loads the interrupt status into internal holding registers as well as clearing the interrupt line to the CPU.

INTERRUPT STATUS REGISTER (Offset 0x4)



Mask	Type	Description.
0001	R	Reference FIFO Empty interrupt.
0002	R	Reference FIFO Half Empty interrupt.
0004	R	Unused - Lo.
0008	R	Late Event interrupt.

0010	R	Checksum Error interrupt.
0020	R	False Sync interrupt.
0040	R	Missed Sync interrupt.
0080	R	Frame Loaded interrupt.
0100	R	DMA Done interrupt.
0200	R	External Interrupt.
0400	R	Internal Interrupt.
0800	R	PLL Dropout interrupt.
1000	R	1 Second interrupt.
2000	R	Finished interrupt.
4000	R	Unused - Lo.
8000	R	Unused - Lo.

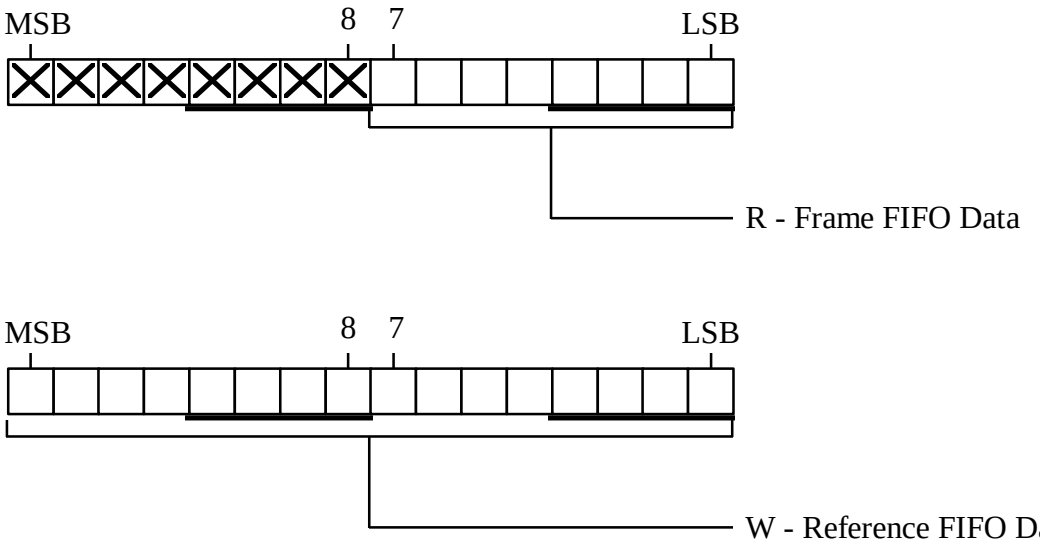
4.5 Reference FIFO / Frame FIFO.

These two FIFOs occupy the same address. The Reference FIFO is a 16 bit wide write only register, and the Frame FIFO an 8 bit wide read only register.

Time and Event data is written in the following order: LS TIME word, Middle Time word MS time word, Event word. If writing to the FIFOs with byte transfers, issue the MS byte of each word first, followed by the LS byte.

Frame reading may be done by byte transfers. In a normal frame there are 117 bytes to be read.

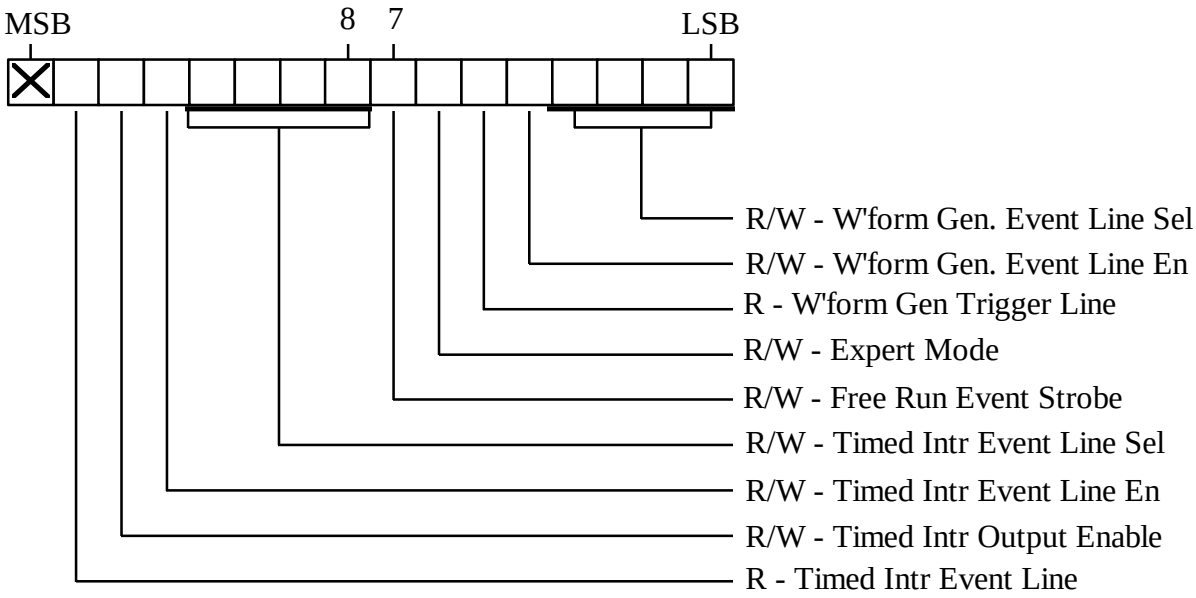
FIFO REGISTERS (Offset 0x6)



4.6 Event Output Control/Status Register.

This register allows setting of basic control points within then output gate array.

EVENT OUTPUT CONTROL/STATUS REGISTER (Offset 0x8)



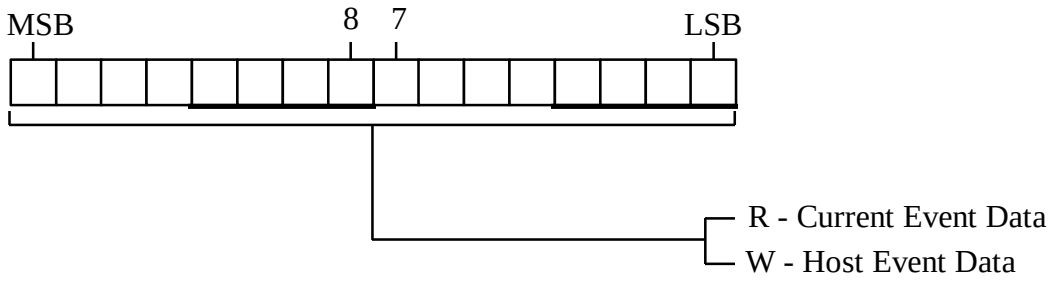
Bit	Type	Description.
000f	R/W	This contains the binary code for the event line to control the Waveform Generator. Any one of the 16 lines may be selected
0010	R/W	Waveform Generator enabled. When set the output to the Waveform Generator enable becomes active and the event selected by bits 0-3 will enable the Waveform Generator.
0020	R	Waveform Generator Trigger Line. This is a copy of the selected bit sent to trigger the Waveform Generator. Useful for testing.
0040	R/W	Expert Mode. When set allows the host to issue an event by writing to the Host Event Register. An event strobe will be issued after the least significant byte has been written. In a standard PC 16 bit I/O operation using 8 bit transfers the LS byte is written last. Thus when using 8 bit I/O operation the LS byte must be sent last. Expert Mode allows programmed events and their corresponding event strobe to still occur, but does not guard against concurrent events.
0080	R/W	Free Run Event Strobe. This bit when set causes the Event Strobe to be issued every microsecond. All timing with respect to TCLK is maintained.
0f00	R/W	This contains the binary code for the event line to control the Timed Interrupt. Any one of the 16 lines may be selected.

1000	R/W	Timed Interrupt enabled. When set the output to the Timed Interrupt becomes active and the event selected by bits 8-11 will be sent to the Input Controller gate array.
2000	R/W	Timed Interrupt Output Enable. When set enables the tri-state output pin to feed the interrupt circuit. This line is normally shared with the second external interrupt, so if a second external interrupt is used this line should be tri-stated.
4000	R	Timed interrupt Event Line. This is a copy of the selected bit sent to trigger the Timed interrupt pin. Useful for testing.
8000	R/W	Reserved

4.7 Host Event register.

This register allows the host to send an event and to read back the current event. When written, the Host Event register is loaded and the contents is then transferred to the event outputs at the completion of the write operation, finishing with an event strobe. When read the event currently on the output lines is channelled back to the host. The host can only write to this register when the Expert Mode bit is set in the Output Control register. On writing an event the timing of the output and the corresponding strobe is the same as for a timed event. If performing 8 bit transfers to this register, the least significant byte must be written last, as it is this transfer that initiates the corresponding event strobe.

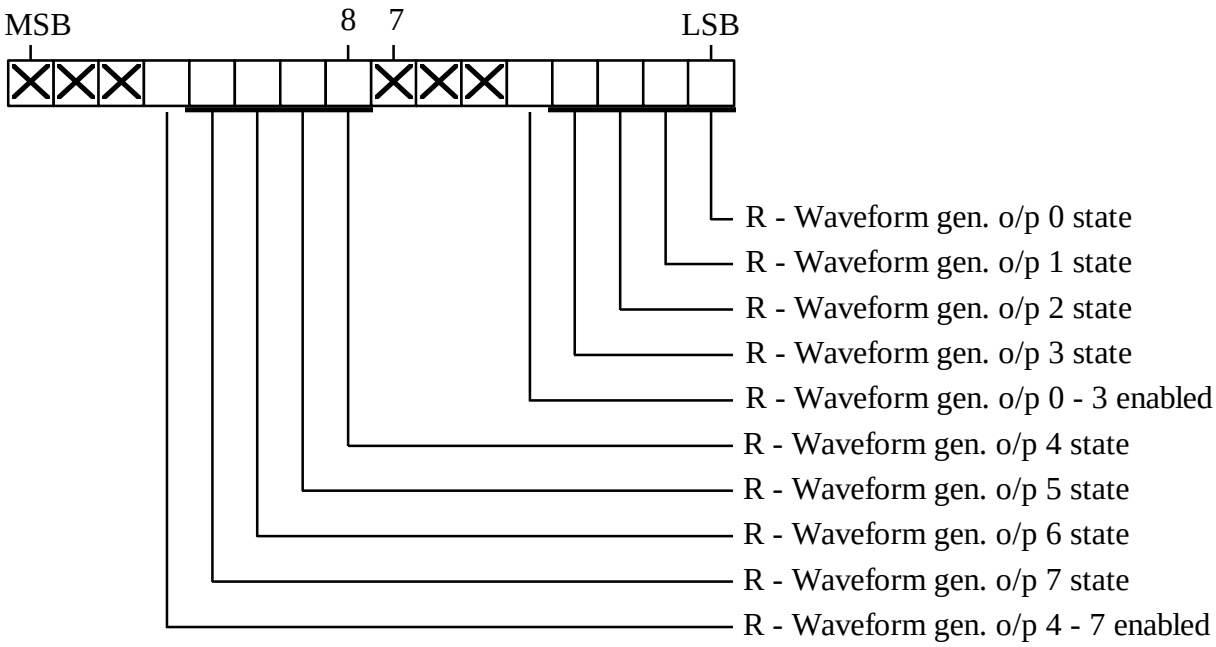
HOST EVENT REGISTER (Offset 0xA)



4.8 Status Register.

This register indicated the status of the Waveform generator gate array.

WAVEFORM STATUS REGISTER (Offset 0x10)

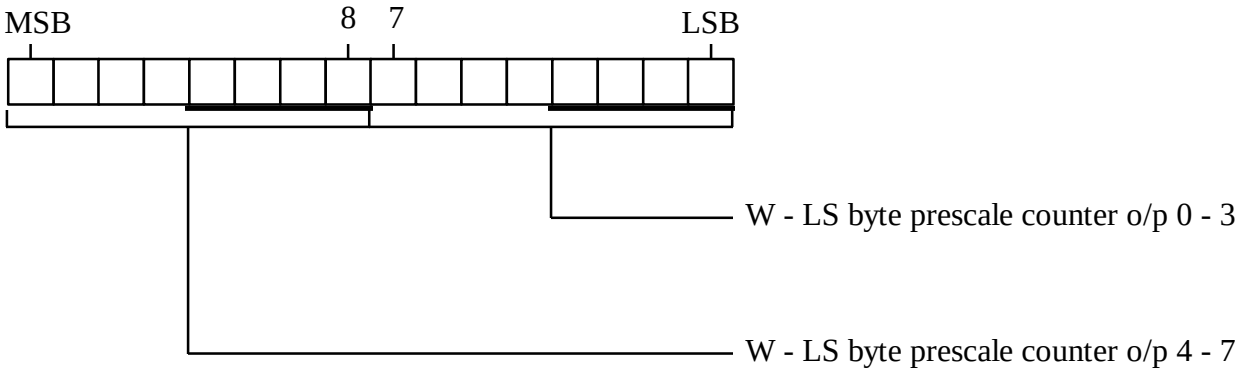


Mask	Type	Description.
0001	R	State of Waveform output 0.
0002	R	State of Waveform output 1.
0004	R	State of Waveform output 2.
0008	R	State of Waveform output 3.
0010	R	When Hi the Waveform generator is enabled for Waveform outputs 0 - 3.
0100	R	State of Waveform output 4.
0200	R	State of Waveform output 5.
0400	R	State of Waveform output 6.
0800	R	State of Waveform output 7.
1000	R	When Hi the Waveform generator is enabled for Waveform outputs 4 - 7.

4.9 Waveform LS Byte Prescale Register.

This register is the LS prescale value of the prescale counter.

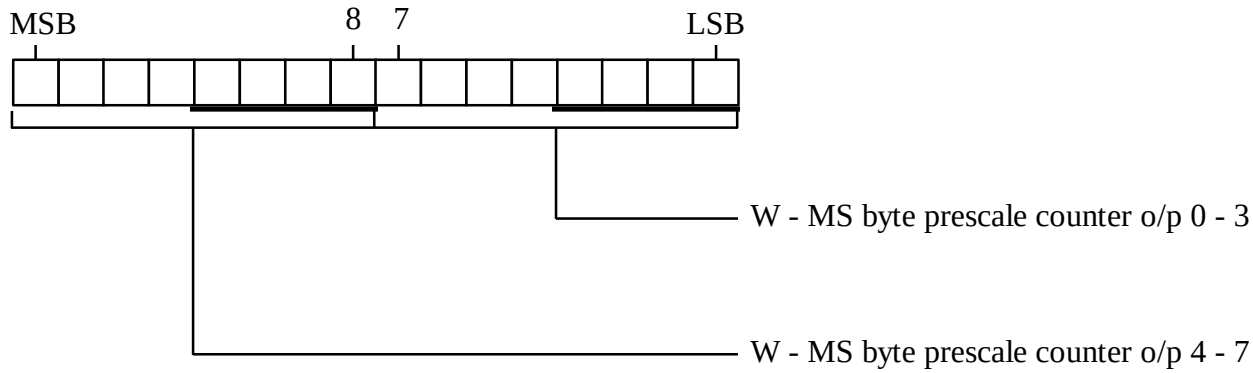
WAVEFORM LS BYTE PRESCALE REGISTER (Offset 0x12)



Mask	Type	Description.
00ff	R/W	LS value for prescale counter for Waveform bits 0 - 3.
ff00	R/W	LS value for prescale counter for Waveform bits 4 - 7.

4.10 MS Prescale Register.
This register is the MS prescale value of the prescale counter.

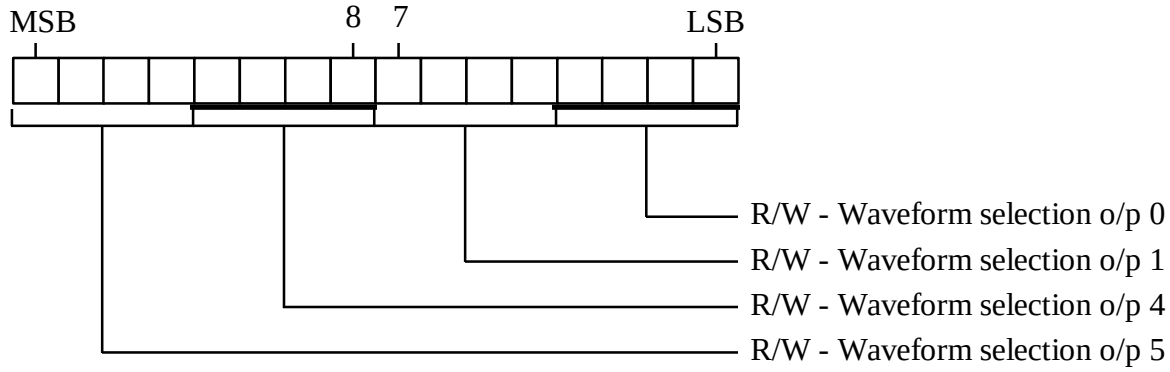
WAVEFORM MS BYTE PRESCALE REGISTER (Offset 0x14)



Mask	Type	Description.
00ff	R/W	MS value for prescale counter for Waveform bits 0 - 3.
ff00	R/W	MS value for prescale counter for Waveform bits 4 - 7.

4.11 Waveform selection Register 1.
This register selects the waveform for Waveform outputs 0, 1, 4, and 5.

WAVEFORM SELECTION REGISTER 1 (Offset 0x16)



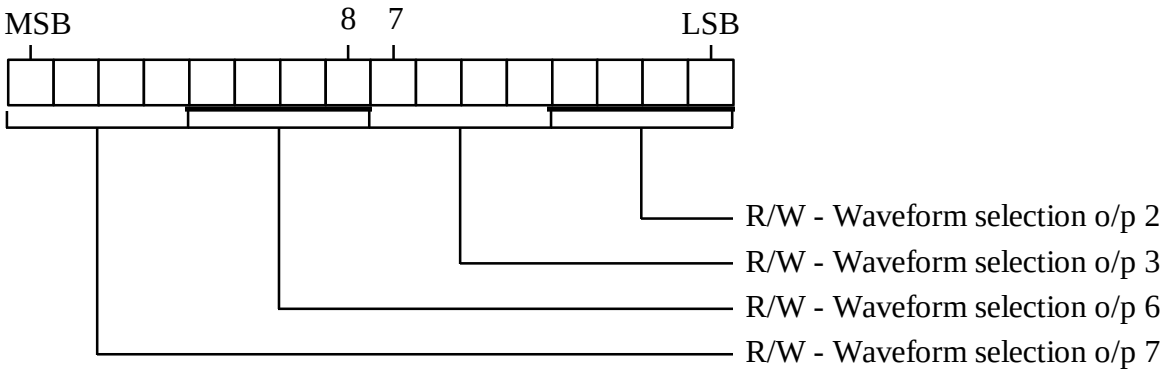
Mask	Type	Description.
000f	R/W	Binary code to select a waveform as given in table above for Waveform output 0.
00f0	R/W	Binary code to select a waveform as given in table above for Waveform output 1.

0f00	R/W	Binary code to select a waveform as given in table above for Waveform output 4.
f000	R/W	Binary code to select a waveform as given in table above for Waveform output 5.

4.12 Waveform selection Register 2.

This register selects the waveform for Waveform outputs 2, 3, 6, and 7.

WAVEFORM SELECTION REGISTER 2 (Offset 0x18)



Mask	Type	Description.
000f	R/W	Four bit binary code to select a waveform as given in table above for Waveform output 2.
00f0	R/W	Four bit binary code to select a waveform as given in table above for Waveform output 3.
0f00	R/W	Four bit binary code to select a waveform as given in table above for Waveform output 6.
f000	R/W	Four bit binary code to select a waveform as given in table above for Waveform output 7.

5.0 Programming.

There are 4 basic operations of the Event Generator:

i) Frame Grabber,
ii) Event Generator,
iii) Waveform Generator,
iv) Read serial number.

This section is intended to indicate methods to program each section to work. There is a range of C code routines within the ATNF for both Borland C (MS-DOS) and High C (pSOS) to perform the above operations.

5.1 Frame Grabber.

Grabbing a frame involves 3 operations:

i) Initiate the process,
ii) Await successful Frame Loaded. This may be done either by polling or by interrupt,
iii) Data Recovery.

5.1.1 Frame Grab Initiation.

To initiate the process, it is necassary to set the Grab Frame bit in the Master Control/Status register. Before this is done, and if polling will be used to await the loading of the frame, the state of the Frame Loaded bit should be checked to see if has been reset. If it is not reset, read at least 1 byte from the FIFO register.

To initiate a Grab Frame, write the 16 bit value 0x0008 to the CSR register. It should be noted that if a Grab Frame has been initiated, but not yet complete, then subsequent initiations are ignored. The process of grabbing a frame may take up to 2mS, with no errors, and longer if transmission errors are encounted.

To initiate Frame Grab using polling:

```
while(inpw(EG_CONTROL) & 0x0008) != 0) inp(EG_FRAME_FIFO);
outpw(EG_CONTROL, 0x0008);
```

To initiate Frame Grab using interrupts:

```
outpw(EG_CONTROL, 0x0008);
```

5.1.2 Await successful Frame Loaded.

5.1.2.1 By polling.

Once initiated the software must loop until the Frame has successfully been loaded. The best way for this is to monitor the Frame Loaded bit in the Interrupt Status register. Then loop intil the Frame Loaded bit goes HI:

```
while(inpw(EG_CONTROL) & 0x0008) == 0);
```

It may be wise to insert a time out in this loop, to avoid the process getting stuck in the event that no frames have successfully been recovered.

5.1.2.1 By interrupt.

To allow for interrupts it is necassary to enable the Event Generator interrupt mechanism. This is done by setting the Frame Loaded interrupt control line and the Enable Interrupt control line, by writing 0x8080 to the Interrupt Control register:

```
outpw((EG_INTERRUPT_CONTROL), 0x8080);
```

It is also necessary to set up the interrupt controller of the CPU but this is outside the scope of this note.

The interrupt will now occur once a frame has successfully been grabbed. Upon interrupt, the ISR should check that the interrupt was caused by the Frame Loaded bit, before recovering the data by:

```
InterruptState = inpw(EG_INTERRUPT_STATUS);
if(InterruptState & 0x80) == 0) return;
.....Recover data routine...
```

It should be noted the contents of the Interrupt Status register is cleared after each read, so a copy should be made, if the status is used elsewhere. It is recommended to read the Interrupt status register with a 16 bit operation. If 2 X 8 bit operations are used, the least significant byte must be read first.

5.1.3 Data Recovery.

Recovery of the data is a simple process of reading the Frame FIFO. This is done by repeatedly reading the Frame FIFO until the Frame FIFO flag in the Master Control/Status register goes hi, or the required number of bytes to a maximum of 117, have been read:

```
n = 0
while(n < 117) {
        if(inpw(EG_CONTROL) & 0x0100) != 0) {
                printf("\nFrame FIFO empty!");
                break;
                }
        data[n] = inp( EG_FRAME_FIFO);
        ++n;
        }
```

The exact format of the data, is dependent on the sending clock, however, the first 8 bytes are reseverd for the BAT, so that after recovery,

data[0] - will have the LS 8 bits of BAT,

and

data [7] will have the MS 8 bits of data.

Regardless of how many bytes of data are recovered from the Frame FIFO, the next Grab Frame initiation will clear the lot.

5.2 Event Generator.

Generating of events involves 2 operations:

i) Initialize the Event Generator,

ii) Load the scheduled time and Event.

5.2.1 Initializing the Event Generator.

To initialize the Event Generator, it is necessary to set both the Purge Event bit and the Reset Ref FIFO bit in the Master Control/Status register. This may be done by writing the 16 bit value 0x0014 to this register. Initializing the Event Generator should only be done at system startup or whenever a fresh start is required.
To initialize the Event Generator:

```
outpw(EG_CONTROL, 0x0014);
```

5.2.2 Load the scheduled time and Event.

To load the scheduled time and Event, is done by writing 4 X 16 bit words of data to the Reference FIFO. The order of the words is:

1. Least Significant word TIME
2. Middle word TIME
3. Most Significant word TIME
4. Event word.

TIME is the least significant 48 bits of the BAT, which is a 64 bit unsigned integer, and must be in the future of the current BAT, but by no more than 2**26 microseconds (~2.25 years).
To load the scheduled time and Event, using 32 bit integer math:

```
unsigned long int BAT[2];     /* int == 16 bit, long int == 32 bit */
unsigned int event;

        outpw(EG_REF_FIFO, (Bat[0] & 0xffff));
        outpw(EG_REF_FIFO, ((Bat[0] >> 16) & 0xffff));
        outpw(EG_REF_FIFO, (Bat[1] & 0xffff));
        outpw(EG_REF_FIFO, Event);
```

5.3 Waveform Generator.

There are 4 basic operations to run the waveforn generator:

i) Load prescale counters,
ii) Select output signals,
iii) Select Event Generator line to control Waveform Generator,
iv) Enable Waveform Generator.

5.3.1 Load Prescale Counters.

After calculating the desired period of the waveform generator outputs in conjunction with the possible range of rates available, it is necessary to load the prescale counters. See section 2.6. As the waveform generator is made up of two 4 bit units, so the prescale counters are 2 X 8 bit registers spread over two 16 bit words.
If prescale value for outputs 0 - 3 is 0x1234, and the prescale value for outputs 4 -7 is 0x5678, then to load the prescale counters:

```
        outpw(WFG_PRESCALE_0, 0x7834);
        outpw(WFG_PRESCALE_1, 0x5612);
```

5.3.2 Select Output Signals.
In conjunction with the calculation of the prescale values, it is necessary to load the output signal selection registers. To load the output signal selection registers with the same number as the output signal (ie output #0 set to signal #0, output #1 set to signal #1, etc) :

```
outpw(WFG_PRESCALE_0, 0x3210);
outpw(WFG_PRESCALE_1, 0x7654);
```

5.3.2 Select Event Generator line to control Waveform Generator.
The Waveform Generator is controlled (run/stop) from one of the Event generator lines. It is therefore necessary to choose one of these lines and setup the Event Output Control/Status Register. To load the Event Output Control/Status Register for Event Generator line #5:

```
unsigned int current;
    current = inpw(EG_OUTPUT_CONTROL);
    current = (current & 0xfff0) | 0x5; /* set only LS 4 bits */
    outpw(EG_OUTPUT_CONTROL, current);
```

5.3.2 Enable Waveform Generator.
To allow the choosen Event Generator line to control the Waveform Generator, it is necessary to set the Waveform Generator Event line bit in the Event Output Control/Status Register. If this bit is not set the Waveform generator cannot be controlled. Similarly, if this bit is set but should not be, activity in the chosen event line will cause the Waveform Generator to operate. To set the Waveform Generator Event line bit in the Event Output Control/Status Register:

```
unsigned int current;
    current = inpw(EG_OUTPUT_CONTROL);
    current = (current & 0xffef) | 0x10; /* set only that bit */
    outpw(EG_OUTPUT_CONTROL, current);
```

To reset the Waveform Generator Event line bit in the Event Output Control/Status Register:

```
unsigned int current;
    current = inpw(EG_OUTPUT_CONTROL);
    current = current & 0xffef); /* clear only that bit */
    outpw(EG_OUTPUT_CONTROL, current);
```

5.4 Read serial number.
The serial number is strored in a PROM as a C string. To access it requires 5 operations:
1. Insert additional wait states in the Event Generator ISA bus controller.
2. Reset the Serial Number PROM,
3. Read in a bit,
4. Increment Prom counter then repeat point 3 untill all bits recovered,

5. Remove wait states set in (1) above.
This operation may interfere with other operations of the Event Generator, so it should be performed before initialization, or with all other access inhibited.

5.4.1 Insert additional wait states in the Event generator ISA bus controller.
To insert the required extra wait states into the Event Generator ISA bus controller to allow for the slower PROMs, set the Insert 4 Wait States bit in the Master Control/Status register by writing 0x2 to this register:

```
outpw(EG_CONTROL, 0x2);
```

5.4.2 Reset the Serial Number PROM.
To reset the Serial Number PROM, set the Grab Frame/ Serial PROM Reset bit in the Master Control/Status register by writing 0x8 to this register. This operation may be performed at the same time as 5.4.1 above.

```
outpw(EG_CONTROL, 0x8);
```

5.4.3 Read in a bit.
The 1 bit contents is available for reading from the Most Significant bit of the Master Control/Status register:

```
unsigned int data;
char serial_num;
    data = inpw(EG_CONTROL);
    if((data & 0x8000) != 0) serial_num = 1;
    else serial_num = 0;
```

5.4.4 Increment Prom counter then repeat point 3 untill all bits recovered.
After each bit is read, it is necessary to increment the internal address counter in the Serial Number PROM. This is done by reading the Frame FIFO:

```
inpw(EG_FRAME_FIFO);
```

Then point 5.4.3 is repeated to build up the bits to make a C string. See section 2.5 for details of the bit structure in the OEM PROM.

5.4.5 Remove wait states.
After the contents of the PROM has been read, the Event Generator ISA bus Controller should be returned to standard wait state operation. Failure to do so will result in slower operation of the Event Generator ISA bus interface, and may also reduce the performance of the host PC. To reset the wait states in the Event Generator ISA bus controller:

```
outpw(EG_CONTROL, 0x0);
```

6.0 Programming Examples.

6.1 Example 1.

Read the serial number prom.
Enter routine with Base address of event generator, pointer to character string to where ROM data will be put, and maximum number of bytes to read.
Returns number of bytes read, negative value if failed.
Routine first resets the serial number prom then starts looking for a high bit. On receiving a high bit it clocks a further 8 bits then starts collecting the data. It continues until it gets 8 consecutive 0's (C string terminator).
Written for Borlandc.

Format:
int get_serial_num(base_addr, string, bufsize)

Arguements.
Name: base_addr
Type: unsigned integer
Access: read
Mechanism: by value
The base address of the event generator from which to read the serial number.

Name: string
Type: unsigned character array
Access: write
Mechanism: by reference
The character array where the data from the serial number prom will be placed.

Name: bufsize
Type: signed integer
Access: read
Mechanism: by value
The size of the character array where the read data will be placed.

Return value:
The number of bytes retrieved from the prom or a negative value if failed:
-1 - no start sequence found.
-2 - illegal buffer size or buffer size too small.

```
#define U_CONTROL 0x1
#define FIFO_REG 0x6
int get_serial_num(int base_addr, char *string, int bufsize)
{
int n;
int m;
int mask;
```

```
// check to see if enough room in buffer
        if(bufsize <= 0) return(-2);

// reset serial num prom by grabbing a frame
// also set "insert wait states" bit

        outp(base_addr, (inp(base_addr) | 0xa));

// wait a short while
        for(n = 0; n < 1000; ++n);

// now search for first hi bit. Return error if not found after 1000 bits
        n = 0;
        while(1) {
                if((inp(base_addr + U_CONTROL) & 0x80) == 0) break;
                inp(base_addr + FIFO_REG);/* clock serial PROM */
                ddelay(100);    /* short delay */
                ++n;
                if(n >= 1000) {
                        outp(base_addr, ((inp(base_addr) | 0x2) - 0x2));
                        return(-1);
                        }
                }

// clock next 8 bits
        for(n = 0; n < 8; ++n) inp(base_addr + FIFO_REG);

// now get data until C string terminator found
        n = 0;
        while(n < bufsize) {
                string[n] = 0;
                mask = 1;
                for(m = 0; m < 8; ++m) {
                        ddelay(1000);  /* short delay */
                        if((inp(base_addr + U_CONTROL) & 0x80) == 0x80)
                                string[n] = string[n] | mask;
                        inp(base_addr + FIFO_REG);
                        mask = mask * 2;
                        }
                if(string[n] == 0) {
                        outp(base_addr, ((inp(base_addr) | 0x2) - 0x2));
                        return(strlen(string));
```

```
                    }
                ++n;
                }

// here on buffer overflow so clear 4 wait states bit and exit error
        outp(base_addr, ((inp(base_addr) | 0x2) - 0x2));
        return(-2);
        }

void ddelay(int count)
{
        while(count > 0) --count;
        return;
}
```

6.2 Example 2

Write Event Time and Event to event FIFO. This routine extracts the three 16 bit words of BAT time and places them into the Reference FIFO, then places the given event into the Reference FIFO. Written for Borlandc.

Format:
int put_out(port, time, event)

Arguements.

Name:	port
Type:	unsigned integer
Access:	read
Mechanism:	by value

The base address of the event generator to where the data will be written too.

Name:	time
Type:	unsigned long integer (4 bytes) array
Access:	read/write
Mechanism:	by reference

The array containing the BAT thst will be loaded into the FIFO. BAT time is an array of 32 bit numbers with the least significant value in the zeroth element of the array.

Name:	event
Type:	unsigned integer
Access:	read
Mechanism:	by value

The 16 bit event that will be loaded into the FIFO.

Returned value:
0.

```
#define FIFO_REG 0x6

int put_out(unsigned int port, unsigned long int *time, unsigned int event)
{
unsigned int ioval;

        ioval = port + FIFO_REG;

        outpw(ioval, time[0]); /* LS time word */
        outpw(ioval, (time[0]/ 0x10000));      /* middle time word */
        outpw(ioval, time[1]); /* MS time word */
        outpw(ioval, event);    /* event word */

        return(0);
}
```

6.3 Example 3
48 bit addition for 32 bit PC machines based on a 16 bit integer compiler (Borlandc). Routine adds two arguements and places result back into first arguement. Written for Borlandc.

Format:
void add_time(time, delta)

Arguements.
Name: time
Type: unsigned long integer (4 bytes) array
Access: read/write
Mechanism: by reference
The array containing the BAT. Thisis an array of 32 bit numbers with the least significant value in the zeroth element of the array.

Name: delta
Type: unsigned long integer (4 bytes) array
Access: read/write
Mechanism: by reference
The array containing the time that will be added to the first arguement. This is an array of 32 bit numbers with the least significant value in the zeroth element of the array.

Returned value:

```
void.

void add_time(unsigned long int *time, unsigned long int *delta)
{
int n;

// Do first addition
        time[0] = time[0] + delta[0];

// Check for carry
        if(time[0] < delta[0]) {          /* carry */
                time[1] = (time[1] + delta[1] + 1) & 0xffff;
                }

        else time[1] = (time[1] + delta[1]) & 0xffff;   /* no carry */
        return;
}
```

7.0 Engineering Modifications.

The following is a list of the faults that have been found on the board, along with their suggested repair. When undertaking any of the fixes, do all the fixes to bring the board upto the top state.

7.1
15 November 1994.
Problem:
Event Generator does not reset when PC reset button pressed or when watch dog timer forces a reset.
Priority:
Low. Should be fixed as needed
Fix:
Remove R25 (10K ohm resistor). Insert a 1/8 W 1K ohm resistor across D3.
Cause:
The external reset line is ORed by a diode to the RC reset circuit. This reset signal through the diode does not go low enough for the receiving gate (pin3, U17).

7.2
March 1997.
Problem:
Event Generator interrupt fails with some brands of CPU card.
Priority:
Medium. Should be fixed at first convenience.

Fix:
Re-design of Input Controller Xilinx (U12). This re-design removes the shared interrupt feature of previous designs. It also removes the “Interrupt on FIFO full” feature as it was potentially slower than the loading process with faster CPU cards.
Cause:
Exact cause unknown, but believed to be from either interrupt line not held LOW long enough or not going LOW before an EOI command being sent to the interrupt controller.

7.3
April 1997.
Problem:
False indications on PLL lock error.
Priority:
Low. Fix only as needed.
Fix:
Change R12 from 10K ohm to 1K ohm.
Cause:
Leakage currents from either U10 (LM324) and/or from U16 are sufficient to cause a voltage drop of greater than 0.8V across 10K ohms (R12).
This need only be fixed if false PLL lock indications are of concern.

7.4
February 1998.
Problem:
Serial number PROM not being read properly, Serial Number PROM being damaged.
Priority:
High. Should be fixed ASAP.
Fix:
On Solder side of PCB, cut the two tracks connected to pin 22 of U14, adjacent this pin. Solder an insulated jumper wire from pin 23 of U14 to pin 1 of U15.
Cause:
The PAL, PAR2SER at location U14 uses pin 22 as an internal intermediate variable. This causes this pin to be an output which conflicts with the Serial Number PROM output. This may corrupt the Serial Number data when being read, and in some cases damage the Serial Number PROM.

7.5
February 1998.
Problem:
U22 Xilinx may not program reliably.
Priority:
Low. Change only if problem shows.
Fix:

Change resistor R24 to 3.9KΩ.
Cause:
The mode pins on the Xilinx determine how the Xilinx will program. One of these mode pins is used as an outout once the Xilinx is configured. To avoid conflict between the mode selection circuit and Xilinx output drivers, the signal is decoupled by resistor R24. If the value is too large to drain the current from pin 13 on U17, the mode circuit may not pull the mode pin low enough at configuration time.

7.6
July 2000.
Problem:
U14 and U16 not GAL20V8 compatable.
Priority:
Low. Change only if U14 and/or U16 need replacing, or on new production units.
Fix:
U14: On top side cut vertical track adjacent pin 13 between U14 and U13, just near "U13" silk screen text. On bottom side cut track just to the right of via located about half way between pin 1 and 24 of U14. On the bottom side run an insulated wire from pin 1 on U14 to pin 13 on U13. Ground pin 13 on U14 running an insulated wire from pin 12 to pin 13 on this IC.
U16: On bottom side cut track running to pin 13 on U16 adjacent this pin. Ground pin 13 on U16 by running an insulated wire from pin 12 to pin 13 on this IC.
Cause:
EP600 obsolete. Once modified, U14 and U16 no longer EP600 compatable.