EPICS

MRF Event System

(Modular Register Map)

Software Reference Manual

Eric Björklund
Los Alamos National Laboratory
Los Alamos Neutron Science Center (LANSCE)
Document Revision 1:
October 5, 2009

Table of Contents

1. Event System Features

1.1 General Features

The Micro Research Finland (MRF) event system is an extremely flexible event-driven timing system that uses fiber-optic Ethernet technology to transmit timing events, clock signals, and arbitrary data streams at speeds up to 125 megahertz. The event system can transmit up to 254 discreet event codes, up to 8 clock signals, and a data buffer of up to 2,048 bytes. The event stream can be simultaneously synchronized with a high-speed signal source (such as an accelerator's RF frequency) and a low-speed signal source (such as the AC mains frequency). These events, signals, and data buffers can generate hardware gates and clocks, software interrupts, time-stamps, and time-critical data streams.

1.1.1 *The Event Stream, The RF Clock, & The Event Clock.*

The event stream is a continuous flow of "Event Frames" which consist of two bytes, the event code and a distributed bus data byte, as shown below:

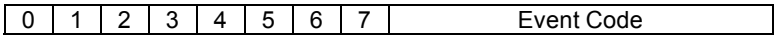

Figure 1-1
Event Frame Format

The event frames are transmitted using an 8b/10b protocol, so each frame is actually 20 bits long when it is transmitted. The bits of an event frame are transmitted at the "RF Clock" speed – so named because it is the clock derived from and synchronized with the high-speed signal source (the RF clock). The "RF Clock" speed determines the "Event Clock" speed. For example, if the RF Clock is 2.5 gigahertz, then the "Event Clock" speed will be 125 megahertz ($2.5 \times 10^9 / 20$) and the maximum clock frequency for signals in the distributed data bus byte will be 62.5 megahertz (125 / 2). The "RF Clock" speed is limited by the speed of the fiber optic transceivers.

The Event Clock is the primary time measurement unit in the MRF event system.

The distributed data bus byte may be time-shared between the eight "clock" clock signals and the bytes of an arbitrary data buffer of up to 2,048 bytes. When this happens, the event frames alternate between the clock signals and the data buffer bytes, reducing the maximum clock signal frequency to 31.25 megahertz with a 125 Mhz event clock. Similarly, the maximum transmission rate for the data buffer is 62.5 Mbytes/second with a 125 Mhz event clock.

1.1.2 *Special-Function Event Codes*

In addition to triggering gates or interrupts, events may also be assigned special functions such as generating time-stamps or synchronizing local counters. Most of the special functions only affect the event receivers and can be re-defined locally at the event receiver (see section 4.2). Generally, however, it is better to either stick to the system defaults or create your own system-wide assignments using the site configuration file (`MRF_CONFIG_SITE`).

Table 1-1 gives the default event codes for special-function events.

Event Code	User-Assignable	Function
0x00	No	Null event code. Not transmitted.
0x70	Yes	Shift a “0” into the event receiver’s time-stamp seconds shift register.
0x71	Yes	Shift a “1” into the event receiver’s time-stamp seconds shift register.
0x79	Yes	Stop event receiver’s event log.
0x7A	Yes	Heartbeat event code.
0x7B	Yes	Reset event receiver’s local counters.
0x7C	Yes	Increment event receiver’s time-stamp event counter register.
0x7D	Yes	Reset event receiver time-stamp.
0x7E	Yes	Latch time-stamp.
0x7F	No	End of sequence code. Stops event sequencer. Not transmitted.

Table 1-1
Default Values for Special-Function Events

1.1.3 *Time-stamping*

The MRF event system provides the ability to time-stamp events, data, or actions. It may also be used as a time provider to the generalTime package.

Each event receiver maintains a running time-stamp synchronized from the event link. The time-stamp consists of a 32-bit “Seconds Counter” and a 32-bit “Event Counter”.

The “Event Counter” can be configured to count scaled event clock ticks, “Event Counter” events (default code = 0x7C), or clock signal 4 from the distributed data bus.

The “Seconds Counter” does not actually count anything. Its value is loaded serially into a shift register by a series of “Time-stamp Shift Register” events.” The “Time-stamp Shift Register 0” event (default code = 0x70) will shift a 0 into the least significant bit of the shift register. The “Time-stamp Shift Register 1” event (default code = 0x71) will shift a 1 into the least significant bit of the shift register. 32 “Time-stamp Shift Register” events are required to load a new value into the time-stamp seconds shift register.

The “Reset Time-stamp” event (default code = 0x7D) will load the time-stamp seconds register with the contents of the time-stamp shift register and clear the time-stamp event counter register.

Alternatively, you may use the clock signals 4 through 7 from the distributed data bus instead of events to manage the event receiver time-stamps. This is called the “Diamond Method” because it originated at the Diamond Light Source. Some special hardware is required to use the “Diamond Method”, but it allows you to have a completely automatic time-stamp system, with no software intervention required. Under the "Diamond Method", clock signal 5 on the distributed data bus will generate a “Time-stamp Reset” event (0x7D) at the event generator. Clock signals 6 and 7 generate the “0” and “1'” “Time-stamp Shift Register” events (0x70 and 0x71 respectively) at the event generator. Clock signal 4 on the distributed data bus can be used to increment the “Event Counter” component of the time-stamp at the event receiver. This is not strictly required, but is a convenient method of distributing the high-frequency clock. Special hardware is usually required to generate these signals from an external time source (such as a GPS).

The active time-stamp may be “latched” it into a set of “Time-stamp Latch” registers where it can be read out by the software. Time-stamp latching may be triggered by a “Latch Time-stamp” event (default code = 0x7E), or by calling the `ErGetTimeStamp()` routine (provided it is configured to latch the time-stamp before reading it).

1.2 Event Generator Features

The Event Generator is responsible of creating and sending out timing events to an array of Event Receivers. In addition to events the Event Generator can also distribute eight simultaneous signals sampled with the event clock rate. These signals may be provided externally or generated on-board by programmable multiplexed counters.

1.2.1 *Event Sequences*

An "event sequence" is an ordered list of time-stamped event codes that an event generator can store and "play back". An event sequence may contain up to 2048 entries. Each entry consists of:

- An event code (8 bits)
- A time-stamp (double)
- An enable flag (boolean)
- A relative priority (32-bits)

The time-stamp determines when the event should occur relative to the start of the sequence. Time-stamps may be expressed in units of event clock ticks, nanoseconds, microseconds, milliseconds, seconds, minutes, or hours. The enable flag determines whether or not the event will be delivered when its time-stamp comes up. The relative priority determines which event will "win" if more than one event has the same time-stamp. Section 3.3 describes the records used to create an event sequence entry.

An event sequence is a software construct. An application may define any number of sequences. An event sequence becomes active when it is loaded into a "sequence RAM" and triggered.

1.2.2 *Sequence RAMs*

Each Event Generator contains two event sequence RAMs. Each event sequence RAM is capable of storing up to 2,048 entries. A sequence RAM entry consists of an 8-bit event code and a 32-bit time-stamp. Sequence RAMs may be triggered from the event generator's multiplexed counters, from the AC mains synchronization logic, and from software. Both sequence RAMs may be active simultaneously.

When the sequence RAM is triggered, an internal event clock counter starts counting. The counter value is compared to the time-stamp of the next event in the sequence RAM table. When the counter value matches the time-stamp in the sequence RAM table, the specified event code is transmitted. The internal time-stamp counter will roll over to 0 when its maximum value (0xffffffff) is reached. This means that the time offset between two consecutive events in the RAM may be anywhere between 1 to 2^{32} event clock ticks.

There are two special event codes which are not transmitted, the null event code 0x00 and "end sequence" code 0x7f. The null event code may be used if the time between two consecutive events should exceed 2^{32} event clock cycles. This feature allows you to specify large time-stamps, or time-stamps with large units (e.g. in "hours") that would exceed 32 bits when translated into event clock ticks. It is worth noting that specifying large time-stamps in a sequence will reduce the number of events the sequence can hold due to the necessity of inserting these "rollover" events into the sequence. The "end sequence" code stops the sequencer, resets the sequence RAM address and clears the time-stamp counter. When you create an event sequence, the software will automatically place an "end sequence" event as the last element in the sequence RAM.

An event sequence RAM has three “Repeat Modes”:

- **Normal Mode**: Once triggered, the events in the RAM are played back at their designated time-stamps. When the “end sequence” code is encountered the sequence RAM will halt and restart on the next trigger.
- **Continuous Mode**: After triggering, the events in the RAM are played back at their designated time-stamps. When the “end sequence” code is encountered the sequence is replayed immediately, without waiting for another trigger.
- **Single Cycle Mode**: After triggering, the events in the RAM are played back at their designated time-stamps. When the “end sequence” code is encountered, the sequencer is disabled and does not respond to further triggers until it is re-enabled.

The contents of the sequence RAMs may be updated at any time, however updating an active sequence RAM could have unpredictable results. One common method for updating the event sequence is to have both sequence RAMs set to the same operating mode. While one sequence RAM is active, updates are applied to the inactive sequence RAM. At the end of the active sequence, the active sequence RAM is disabled and the updated sequence RAM is enabled. The software allows for the switch to occur immediately after the end of the current sequence, or to wait for a software trigger so that multiple updates can be applied simultaneously.

Sections 3.3 and 3.4 describe how to set up an event sequence configure the sequence RAMs.

1.2.3 *Updating Event Sequences*

An “inactive” event sequence (one that has not been loaded into a sequence RAM) may be updated at any time using the records defined in section 3.3. An active sequence may also be updated at any time, however there is a risk of non-deterministic behavior if you attempt to update an active sequence while it is currently executing.

Updating an active sequence is simplified by using the OP_MODE record (see section 3.4.1). When the OP_MODE record is in “Single-Seq” mode, the software behaves as if there is only one sequencer. Only one event sequence will execute at a time. When updates are made to the active event sequence, they are written to the inactive sequence RAM. Depending on how the UPDATE_MODE record (section 3.4.2) is set, the updates will take effect immediately upon the next trigger after the current sequence ends (“Immediate” mode) or they will accumulate until a software switch is requested (“Accumulate” mode).

When the OP_MODE record is in “Multi-Seq” mode, control over the sequence RAMs is left to the application. Section 3.4 gives more details on the differences between “Single-Seq” and “Multi-Seq” modes.

1.3 Event Receiver Features

Event Receivers decode timing events and signals from an optical event stream transmitted by an Event Generator. Events and signals are received at predefined rate the event clock that is usually divided down from an accelerators main RF reference. The event receivers lock to the phase event clock of the Event Generator and are thus phase locked to the RF reference. Event Receivers convert event codes transmitted by an Event Generator to hardware outputs. They can also generate software interrupts and store the event codes with globally distributed time-stamps into FIFO memory to be read by the processor.

1.3.1 *Interface to generalTime*

2. Software Installation (*TBS*)

To be specified…

3. Event Generator Records

Event generator records will have the device type (DTYP) field set to “MRF EVG”. The input (INP) or output (OUT) link fields will use the VME_IO hardware specification regardless of the event generator’s actual bus type. The VME_IO specification provides a convenient format for specifying the logical card number, a parameter string to contain a description of the record’s function, and a signal number to indicate which signal, counter, port, or event to apply the function to.

The INP or OUT fields will contain a string of the form:

```
#Cn Sm @function
```

Where:

n	=	The logical card number of the event generator card
m	=	Which signal, counter, port, or event number to act on
function	=	Defines the record’s function.

3.1 Event Generator Control and Status Records

Introductory verbage ??????

An event generator card control or status record will have the device type (DTYP) field set to

"EVG"

The output link (OUT) or input link (INP) field will contain a string of the form:

`#Cn S0 @XXXXXX`

Where:

n = The logical card number of the event generator card, and
XXXXXX = Defines which function to perform.

Note that the signal number is not used for general control and status records.

Table 3-1summarizes the function values for event generator card control and status records.

Function Name	Record Type	Default Value	Description
ENABLE	bo	Enabled	Enable or disable the event generator card.
ENABLE	bi	N/A	Displays whether the event generator card is enabled or disabled.
SEND_EVENT	longout	N/A	Generate a software event.
ID_FORM_FACTOR	stringin	N/A	Displays the form factor for event generator card *n*.
ID_FW_REV	stringin	N/A	Displays the firmware revision level for event generator card *n*.
ID_SERIAL_NUM	stringin	N/A	Displays the serial number of event generator card *n*.
EVCLOCK_SOURCE	bo	Internal	Set the source for generating the event clock
EVCLOCK_RF_DIV	longout	1	Set the external RF source divider.
EVCLOCK_SPEED	ai/ao	System Default	Display/Set the event link clock speed in Megahertz.
UNITS	mbbo	*System Default*	Sets the default time units for scheduling events in the sequence RAMs.

Table 3-1
Event Generator Control and Status Functions

3.1.1 *Enable/Disable Event Generator*

Enable/Disable Event Generator (#C*n* S0 @ENABLE) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable event generator card *n*
1	Enabled	Enable event generator card *n*

Enables or disables the event generator card. If the database contains no ENABLE record for a given event generator card the value will default to "Enabled".

If multiple ENABLE records exist for the same event generator card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.1.2 *Generate Software Event*

Generate Software Event (#C*n* S0 @SEND_EVENT) (longout Record)

Processing this record will cause whatever event code is contained in the VAL field to be sent out on the event link. The VAL field must contain between 1 and 255.

3.1.3 *Event Generator ID Records*

Event generator ID records are string input (stringin) records that display static information about an event generator card. The information in an ID record does not change, so the record only needs to be processed once at system initialization time.

Display Form Factor (#C*n* S0 @ID_FORM_FACTOR) (stringin Record)

Displays the form factor of the event generator card (e.g. VME, PMC, cRIO) and in some cases the bus location.

Display Firmware Revision (#C*n* S0 @ID_FW_REV) (stringin Record)

Displays the firmware revision level of the event generator card.

Display Serial Number (#C*n* S0 @ID_SERIAL_NUM) (stringin Record)

When available, displays the event generator card's serial number. For VME cards, the serial number is the card's MAC address. For PMC and PCI cards the serial number can be set by the user.

3.1.4 *Event Clock Generation*

The event generator is responsible for generating the "Event Clock". The event clock determines the speed at which the events, distributed bus signals, and data buffer bytes are transmitted on the event link. The event clock can either be generated internally by the event generator, or it can be derived by dividing down an external RF source. Using the external RF source allows you to synchronize your event system with the accelerator's RF frequency.

Set Event Clock Source (#C*n* S0 @EVCLOCK_SOURCE) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Internal	Event clock is generated internally.
1	RF Input	Event clock is generated by dividing down an external RF signal.

Determines whether the event clock is generated internally or from the RF input signal.

If the database contains no EVCLOCK_SOURCE record for a given event generator card, the event clock source will default to "Internal".

If multiple EVCLOCK_SOURCE records exist for the same event generator card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set the RF Input Divider (#C*n* S0 @EVCLOCK_RF_DIV) (longout Record)

Sets the divider for the RF Input signal. Valid values are from 1 to 32 excluding 13. When the clock source (EVCLOCK_SOURCE record) is set to "RF Input", the frequency of the event clock is determined by the frequency of the RF Input signal divided by the value specified in this record. The RF Input frequency must be within the range 50 MHz to 1.6 GHz. The generated event clock frequency must be in the range 50 MHz to 125 MHz. Because of these constraints, higher divider numbers will result in higher required RF frequencies, narrower RF frequency ranges, and lower event clock frequencies.

Table 3-2 shows the relationship between the RF divider selection, the allowable RF Input frequency range, and the generated event clock and bit rate ranges.

Divider	RF Input Frequency	Event Clock Frequency	Bit Rate
1	50 MHz – 125 MHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
2	100 MHz – 250 MHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
3	150 MHz – 375 MHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
4	200 MHz – 500 MHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
5	250 MHz – 625 MHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
6	300 MHz – 750 MHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
7	350 MHz – 875 MHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
8	400 MHz – 1.0 GHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
9	450 MHz – 1.125 GHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
10	500 MHz – 1.25 GHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
11	550 MHz – 1.375 GHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
12	600 MHz – 1.5 GHz	50 MHz – 125 MHz	1.0 Gb/s – 2.5 Gb/s
13	OFF	OFF	OFF
14	700 MHz – 1.6 GHz	50 MHz – 114 MHz	1.0 Gb/s – 2.528 Gb/s
15	750 MHz – 1.6 GHz	50 MHz – 107 MHz	1.0 Gb/s – 2.133 Gb/s
16	800 MHz – 1.6 GHz	50 MHz – 100 MHz	1.0 Gb/s – 2.000 Gb/s
17	850 MHz – 1.6 GHz	50 MHz – 94 MHz	1.0 Gb/s – 1.882 Gb/s
18	900 MHz – 1.6 GHz	50 MHz –88 MHz	1.0 Gb/s – 1.777 Gb/s
19	950 MHz – 1.6 GHz	50 MHz –84 MHz	1.0 Gb/s – 1.684 Gb/s
20	1.0 GHz – 1.6 GHz	50 MHz –80 MHz	1.0 Gb/s – 1.600 Gb/s
21	1.05 GHz – 1.6 GHz	50 MHz – 76 MHz	1.0 Gb/s – 1.523 Gb/s
22	1.1 GHz – 1.6 GHz	50 MHz – 72 MHz	1.0 Gb/s – 1.454 Gb/s
23	1.15 GHz – 1.6 GHz	50 MHz – 69 MHz	1.0 Gb/s – 1.391 Gb/s
24	1.2 GHz – 1.6 GHz	50 MHz – 66 MHz	1.0 Gb/s – 1.333 Gb/s
25	1.25 GHz – 1.6 GHz	50 MHz – 64 MHz	1.0 Gb/s – 1.280 Gb/s
26	1.3 GHz – 1.6 GHz	50 MHz – 61 MHz	1.0 Gb/s – 1.230 Gb/s
27	1.35 GHz – 1.6 GHz	50 MHz – 59 MHz	1.0 Gb/s – 1.185 Gb/s
28	1.4 GHz – 1.6 GHz	50 MHz – 57 MHz	1.0 Gb/s – 1.142 Gb/s
29	1.45 GHz – 1.6 GHz	50 MHz – 55 MHz	1.0 Gb/s – 1.103 Gb/s
30	1.5 GHz – 1.6 GHz	50 MHz – 53 MHz	1.0 Gb/s – 1.066 Gb/s
31	1.55 GHz – 1.6 GHz	50 MHz – 51 MHz	1.0 Gb/s – 1.032 Gb/s
32	1.6 GHz	50 MHz	1.0 Gb/s

Table 3-2
RF Divider, RF Input Range, Event Clock Range, and Transmission Bit Rate Relationships

If the database contains no EVCLOCK_RF_DIV record for a given event generator card, the RF Divider value will default to 1.

If multiple EVCLOCK_RF_DIV records exist for the same event generator card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Event Clock Speed (#C*n* S0 @EVCLOCK_SPEED) (ao Record)

Sets the event clock speed in Megahertz.

If the event clock source (EVCLOCK_SOURCE record) is set to "Internal", then you must specify an event clock speed. If the database contains no EVCLOCK_SPEED record for a given event generator card, the event clock speed will default to the value specified by the `EgConfigEventClockSpeed()` routine (called in the IOC startup script). If there has been no call to the `EgConfigEventClockSpeed()` routine, the event clock speed will be default to the site-specific default (as specified in the `MRF_CONFIG_SITE` file). If no site-specific default value has been set (and none of the other methods have been used to set the event clock speed), the event generator card will be disabled and an error message will be displayed on the IOC console terminal and in the error log.

If the event clock source (EVCLOCK_SOURCE record) is set to "RF Input", it is still a good idea to specify an event clock speed. This will ensure that the event generator software knows what the event clock speed is supposed to be. This is particularly important if you are chaining event generators together. It is also useful for converting between normal time units (seconds, milliseconds, microseconds, etc.) and event clock ticks.

If multiple EVCLOCK_RF_DIV records exist for the same event generator card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.1.5 *Event Generator Status Records*

Display Enable State (#C*n* S0 @ENABLE) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event generator is disabled
1	Enabled	Event generator is disabled

Displays whether the event generator card is enabled or disabled.

Display Event Clock Speed (#C*n* S0 @EVCLOCK_SPEED) (ai Record)

Displays the event clock speed in Megahertz. The value displayed reflects the actual value in the event generator's fractional synthesizer. If your site requires multiple event links running at different clock speeds, then the CLOCK_SPEED record could be a useful check to make sure that the event generator card is generating the correct event clock rate for its event link.

Set Default Time Units for Event Generator Card *n* (#C*n* S0 @UNITS) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Ticks	The default time units for event generator card *n* will be "event clock ticks."
1	Nanoseconds	The default time units for event generator card *n* will be nanoseconds.
2	Microseconds	The default time units for event generator card *n* will be microseconds.
3	Milliseconds	The default time units for event generator card *n* will be milliseconds.
4	Seconds	The default time units for event generator card *n* will be seconds.
5	Minutes	The default time units for event generator card *n* will be minutes.
6	Hours	The default time units for event generator card *n* will be hours.

If the database contains no UNITS record for a given event generator card, the system default time unit, as specified in the `MRF_CONFIG_SITE` file, will apply.

If multiple UNITS records exist for the same event generator card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.2 Sequence Configuration Records

An "event sequence is a collection of up to 2,048 time-stamped "sequence events". An application may create as many event sequences as it wants. Each event sequence is uniquely identified by its sequence number. An event sequence becomes active when it is loaded into a sequence RAM and triggered. Event sequences are described in detail in Section 1.2.1.

An event sequence configuration record will have the device type (DTYP) field set to

"`EVG Sequence`"

The output link (OUT) or input link (INP) field will contain a string of the form:

`#C`*`n`* `S`*`m`* `@SEQ_`*`XXXXXX`*

Where:

n	=	The logical card number of the event generator card
m	=	The unique sequence number for this event sequence, and
`SEQ_`*`XXXXXX`*	=	Defines which function to perform.

Table 3-3 summarizes the function values for an event sequence configuration record.

Function Name	Record Type	Default Value	Description
SEQ_UNITS	mbbo	*System Default*	Sets the default time-stamp units for event sequence *m*.
SEQ_RAM	longin	N/A	Which sequence RAM (if any) contains this sequence

Table 3-3
Event Sequence Configuration Functions

3.2.1 *Set Time Units for Sequence*

Set Time Units For Event Sequence *m* (#C*n* S*m* @SEQ_UNITS) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Ticks	The time-stamps for sequence events (as specified in the EVENT_TIME records) will be specified in units of "event clock ticks."
1	Nanoseconds	The time-stamps for sequence events (as specified in the EVENT_TIME records) will be specified in units of nanoseconds.
2	Microseconds	The time-stamps for sequence events (as specified in the EVENT_TIME records) will be specified in units of microseconds.
3	Milliseconds	The time-stamps for sequence events (as specified in the EVENT_TIME records) will be specified in units of milliseconds.
4	Seconds	The time-stamps for sequence events (as specified in the EVENT_TIME records) will be specified in units of seconds.
5	Minutes	The time-stamps for sequence events (as specified in the EVENT_TIME records) will be specified in units of minutes.
6	Hours	The time-stamps for sequence events (as specified in the EVENT_TIME records) will be specified in units of hours.

Sets the time units for event sequence *m* on event generator card *n*. This record determines how the value in an EVENT_TIME record (see section 3.3) will be interpreted.

If the database contains no SEQ_UNITS record for a given sequence, the default units for the event generator card will be used. If the event generator card has not set a default time unit, the system default time unit, as specified in the `MRF_CONFIG_SITE` file, will be used.

If multiple SEQ_UNITS records exist for the same event sequence, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.2.2 *Display Sequence RAM Containing This Sequence*

Display Which Sequence RAM Contains Event Sequence *m* (#C*n* S*m* @SEQ_RAM) (longin Record)

Displays which sequence RAM (if any) currently contains event sequence *m*. If the event sequence has been loaded into multiple RAMs, the value will reflect the last RAM loaded. A value of "-1" indicates that the event sequence is not loaded into any RAM.

3.3 Sequence Event Configuration Records

Sequence event configuration records determine the order and timing of when events are generated by the event sequencer.

A sequence event configuration record will have the device type (DTYP) field set to

"EVG Sequence"

The output link (OUT) field will contain a string of the form:

#Cn Sm @EVENT_*XXXXXX Name*

Where:

n	=	The logical card number of the event generator card
m	=	The sequence number of the event being configured
EVENT_*XXXXXX*	=	Defines which function to perform, and
Name	=	The event name

A minimum of two records are required to define a sequence RAM event: A long output (longout) record to set the event code, and an analog output (ao) record to set the time in the sequence when the event should be delivered. Optional records may also be included to enable or disable the event or set the event's priority relative to other events that may want the same time-stamp. The event "name" is a user-defined string that is used to associate all the records that define a single event. Each event in an event sequence must have a unique event name. However, the same name may be re-used in a different event sequence. For example, the event named "Start-RF" can appear in event sequence 1 and in event sequence 2. However, if the "Start-RF" event appears twice in event sequence 1, it must be given different names such as "Start-RF-1" and "Start-RF-2".

Table 3-4 summarizes the function values for sequence event configuration records.

Function Name	Record Type	Default Value	Description
EVENT_CODE	longout	0	The event code to assign to sequence event number *m*.
EVENT_TIME	ao	0	Time-stamp for when sequence event *m* should be delivered.
EVENT_ENABLE	bo	Enabled	Enable/disable the generation of sequence event number *m*.
EVENT_PRIORITY	longout	0	Relative priority of sequence event number *m* if there is a time-stamp collision.

Table 3-4
Sequence Event Configuration Functions

3.3.1 *Set Sequence Event Code*

Set Sequence Event Code (#C*n* S*m* @EVENT_CODE *Name*) (longout Record)

Sets the event code for the sequence event named "*Name*" in event sequence *m*. The event code must be a number between 0 and 255. The value "0" is the "Null Event" and disables event "*Name*" from being generated.

If the database contains no EVENT_CODE records for event "*Name*", then event "*Name*" will not be enabled.

If multiple EVENT_CODE records exist for the same event number, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.3.2 Set Sequence Event Time (#C*n* S*m* @EVENT_TIME *Name*) (ao Record)

Set Sequence Event Time

Sets the time-stamp for when to deliver sequence event "*Name*" in event sequence *m*. The time-stamp is relative to the time the sequence gets triggered. The time value is a double-precision value that can be interpreted in units as small as "event clock ticks" or as large as "hours". By default, the time units (typically microseconds) will come from the system-wide time unit default specified in the `MRF_CONFIG_SITE` file. The system-wide default can be overridden by a card-specific default by including an UNITS record in the database. The card-specific default can be overridden for a particular sequence by including a SEQ_UNITS record for the sequence.

If the database contains no EVENT_TIME records for event "*Name*", then event "*Name*" will not be enabled.

If multiple EVENT_CODE records exist for the same event number, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.3.3 Enable/Disable Sequence Event (#C*n* S*m* @EVENT_ENABLE *Name*) (bo Record)

Enable/Disable Sequence Event

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable sequence event *m* on event generator card *n*
1	Enabled	Enable sequence event *m* on event generator card *n*

Enables or disables sequence event "*Name*" in event sequence *m*.

If the database contains no EVENT_ENABLE record for a given event name, but it does contain EVENT_CODE and EVENT_TIME records for that event name, the value will default to "Enabled".

If multiple EVENT_ENABLE records exist for the same event name, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.3.4 Set Sequence Event Priority (#C*n* S*m* @EVENT_PRIORITY *Name*) (longout Record)

Set Sequence Event Priority

Sets the relative priority sequence event "*Name*" in event sequence *m*. The relative priority can be any integer value. The assumption is that 0 is the "median" priority, that positive numbers have higher priority and negative numbers have lower priority.

A sequence event's priority is only important when two events try to occupy the same relative time in the sequence. This can either be a temporary condition, as when an event time is being "knobbed" (or "slidered") and it collides with a stationary event, or it can be "permanent", as when the time unit conversion ends up assigning two events to the same time-stamp. When two sequence events are assigned to the same time slot, the event with the lower priority will be relocated to the next closest event slot (the technical term for this is "jostling").

If the database contains no EVENT_PRIORITY record for a given sequence event name, the value will default to "0".

If multiple EVENT_ENABLE records exist for the same event name, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.4 Sequence RAM Configuration and Control Records

The event sequence RAMs are described in detail in section 1.2.1. This section describes the functions that apply to individual sequence RAMs.

An sequence RAM configuration/control record will have the device type (DTYP) field set to

"EVG RAM"

The output link (OUT) or input link (INP) field will contain a string of the form:

#Cn Sm @XXXXXX

Where:

n	=	The logical card number of the event generator card
m	=	Which sequence RAM (1 or 2) to act on, and
XXXXXX	=	Defines which function to perform.

The OP_MODE, UPDATE_MODE, and SWITCH records apply to both sequence RAMs. The value of the signal specifier ("S") in the output link for these records is not used and is typically set to 0.

The ENABLE, TRIGGER, REPEAT_MODE, LOAD, and START records apply to a single sequence RAM. The value of the signal specifier ("S") in the output link for these records should be specified as either "S1" or "S2".

Table 3-5 summarizes the function values for sequence RAM configuration and control records.

Function Name	Record Type	Default Value	Description
OP_MODE	bo	Single-Seq	Set the sequencer operating mode for EVG card *n*.
UPDATE_MODE	bo	Immediate	Set the sequence RAM update mode when EVG card *n* is in "Single-Seq" mode.
SWITCH	bo	N/A	Switch active sequence RAM on EVG card *n*.
ENABLE	bo	Disabled	Enable/Disable sequence RAM *m* on EVG card *n*.
TRIGGER	mbbo	None	Trigger mechanism for sequence RAM *m* on EVG card *n*.
REPEAT_MODE	mbbo	Normal	Set the repeat mode for sequence RAM *m* on EVG card *n*.
LOAD	longout	N/A	Load the specified event sequence number into sequence RAM *m* on EVG card *n*.
START	bo	Stop	Software start or stop sequence RAM *m* on EVG card *n*.

Table 3-5
Sequence RAM Configuration/Control Functions

3.4.1 *Set Sequencer Operating Mode*

Set Sequencer Operating Mode (#C*n* S0 @OP_MODE) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Single-Seq	Behave as if there were only one sequencer.
1	Multi-Seq	Utilize multiple sequencers.

The OP_MODE record determines how the software utilizes the event generator sequence RAMs.

In "Single-Seq" mode, the software hides the fact that there are two sequence RAMs and behaves as if there were only one sequencer. In "Single-Seq" mode, the software only responds to records for sequence RAM 1. When the active sequence is updated, the changes are loaded into the inactive sequence, where they will take effect either on the next trigger or after a manual "Switch" command – depending on the state of the UPDATE_MODE record. If a new sequence is "loaded" (via a "LOAD" record), the new sequence is treated as if it were an update (i.e. it is loaded into the inactive RAM and takes effect according to the value of the UPDATE_MODE record).

In "Multi-Seq" mode, both sequence RAMs are available and completely under the control of the application via the LOAD, ENABLE, START, REPEAT, and TRIGGER records.

If the database contains no OP_MODE record for a given event generator card, the operating mode will default to "Single-Seq".

If multiple OP_MODE records exist for the same event generator card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.4.2 *Set Sequencer Update Mode*

Set Sequencer Update Mode (#C*n* S0 @UPDATE_MODE) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Immediate	Sequence RAM updates are made to the inactive sequence RAM. When the current sequence ends, the updated sequence RAM becomes the new active RAM.
1	Accumulate	Sequence RAM updates are made to the inactive sequence RAM. The currently active sequence RAM remains active until a software trigger occurs (the SWITCH record, for example).

The UPDATE_MODE record is only relevant if the sequence RAM operating mode is "Single-Seq". In "Single-Seq" mode, the UPDATE_MODE controls whether a change to any of the sequence event records takes effect immediately ("Immediate") or if several changes are allowed to accumulate and all take effect at once when a SWITCH record is processed ("Accumulate")

If the UPDATE_MODE is "Immediate", then any change to active sequence will be recorded in the inactive sequence RAM. When the current sequence ends, the active sequence RAM will become inactive and the recently updated sequence RAM will be become active.

If the UPDATE_MODE is “Accumulate”, any changes to the active sequence will be stored in memory, but will not take effect until a software trigger like the SWITCH record is processed. At that time, the updated sequence information is written to the inactive sequence RAM. . When the current sequence ends, the active sequence RAM will become inactive and the recently updated sequence RAM will be become active.

If the database contains no UPDATE_MODE record for a given event generator card, the update mode will default to “Immediate”.

If multiple UPDATE_MODE records exist for the same event generator card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.4.3 Switch Active Sequence RAM (#C*n* S0 @SWITCH) (bo Record)

Switch Active Sequence RAM

Raw Value	Engineering Value (in template files)	Description
0	*	No switch in progress
1	Switch	Switch the active sequence RAM on event generator card *n*.

This is a momentary binary output record. When a “1” (“Switch”) is written to the record, the currently active sequence RAM will be allowed to complete its sequence, at which point it will become inactive and the other sequence RAM will become active. This function is only relevant when the sequence RAM OP_MODE is “Single-Seq” and the UPDATE_MODE is “Accumulate”.

Multiple SWITCH records may exist for a single event generator card.

3.4.4 Enable/Disable Sequence RAM (#C*n* S*m* @ENABLE) (bo Record)

Enable/Disable Sequence RAM

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable sequence RAM *m* on EVG card *n*.
1	Enabled	Disable sequence RAM *m* on EVG card *n*.

Enables or disables sequence RAM *m* on event generator card *n*. Once a sequence RAM has been enabled, it will start sequencing on the next trigger (as defined by the TRIGGER record).

If the database contains no ENABLE record for a given sequence RAM, the value will default to “Disabled”.

If multiple ENABLE records exist for the same sequence RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

If the sequence RAM OP_MODE is set to “Single-Seq”, only the ENABLE record for sequence RAM 1 will have any effect.

3.4.5 *Set Sequence RAM Trigger Mechanism*

Set Sequence RAM Triggering Mechanism (#C*n* S*m* @TRIGGER) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	None	Sequence RAM *m* will never be triggered.
1	Manual	Sequence RAM *m* is triggered manually (e.g. using the START record)
2	Manual Both	Sequence RAM *m* is triggered by the same START record that triggers the other sequence RAM.
3	AC	Sequence RAM *m* is triggered by the output of the AC zero crossing synchronization logic.
4	Mux Ctr 0	Sequence RAM *m* is triggered by the output of Multiplexed Counter 0.
5	Mux Ctr 1	Sequence RAM *m* is triggered by the output of Multiplexed Counter 1.
6	Mux Ctr 2	Sequence RAM *m* is triggered by the output of Multiplexed Counter 2.
7	Mux Ctr 3	Sequence RAM *m* is triggered by the output of Multiplexed Counter 3.
8	Mux Ctr 4	Sequence RAM *m* is triggered by the output of Multiplexed Counter 4.
9	Mux Ctr 5	Sequence RAM *m* is triggered by the output of Multiplexed Counter 5.
10	Mux Ctr 6	Sequence RAM *m* is triggered by the output of Multiplexed Counter 6.
11	Mux Ctr 7	Sequence RAM *m* is triggered by the output of Multiplexed Counter 7.

This record determines the mechanism by which sequence RAM *m* is triggered. Sequence RAMs may be triggered from the AC zero-crossing synchronization logic, from the multiplexed counters, or manually.

If the database contains no TRIGGER record for a given sequence RAM, the value will default to “None”.

If multiple TRIGGER records exist for the same sequence RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

If the sequence RAM OP_MODE is set to “Single-Seq”, only the TRIGGER record for sequence RAM 1 will have any effect.

3.4.6 *Set Sequence RAM Repeat Mode*

Set Sequence RAM Repeat Mode (#C*n* S*m* @REPEAT_MODE) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Normal	After trigger, sequence RAM *m* will run to completion, then stop and wait for the next trigger.
1	Continuous	After trigger, sequence RAM *m* will repeat the sequence continuously without requiring any additional triggers
2	Single	After trigger, sequence RAM *m* will run to completion, then stop and disable the sequencer. The sequence RAM must be re-enabled before it will run again.

If the database contains no REPEAT_MODE record for a given sequence RAM, the value will default to "Normal".

If multiple REPEAT_MODE records exist for the same sequence RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

If the sequence RAM OP_MODE is set to "Single-Seq", only the REPEAT_MODE record for sequence RAM 1 will have any effect.

3.4.7 *Load an Event Sequence into a Sequence RAM*

Load Sequence RAM (#C*n* S*m* @LOAD) (longout Record)

Loads the event sequence specified in the VAL field into sequence RAM *m*. If the OP_MODE is "Single-Seq", loading a new sequence is treated the same as updating the active sequence. If the UPDATE_MODE is "Immediate", the newly loaded sequence will become active on the first trigger after the current sequence ends. If the UPDATE_MODE is "Accumulate", the newly loaded sequence does not become active until the SWITCH record is processed.

If the sequence RAM OP_MODE is set to "Single-Seq", only the LOAD record for sequence RAM 1 will have any effect.

If multiple REPEAT_MODE records exist for the same sequence RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

3.4.8 *Start/Stop Sequence RAM*

Start/Stop Sequence RAM (#C*n* S*m* @START) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Stop	Stop sequence RAM *m* on EVG card *n*.
1	Start	Start sequence RAM *m* (if enabled) on EVG card *n*.

This record allows you to manually control the sequence RAMs. Writing a "0" ("Stop") to the record will stop sequence RAM *m* from executing, but will leave it enabled. Writing a "1" ("Start") to the record will cause sequence RAM *m* to start (assuming it is enabled). This record is most useful when the triggering mode (TRIGGER record) is set to "Manual" or "Manual Both".

If the database contains no START record for a given sequence RAM, the value will default to "Stop".

If multiple START records exist for the same sequence RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

If the sequence RAM OP_MODE is set to "Single-Seq", only the START record for sequence RAM 1 will have any effect.

4. Event Receiver Records

EPICS records that support the Event Receiver module are divided into six categories:

- General Event Receiver Control and Status Records
- Event Definition Records
- Pulse Generator Configuration Records
- Clock Configuration Records
- Output Gate Configuration Records
- Input Gate Configuration Records

A set of standard EPICS analog input (ai), analog output (ao), binary input (bi), binary output (bo), long input (longin), long output (longout), multi-bit binary input (mbbi), multi-bit binary output (mbbo) string input (stringin), and event records are organized into template files for each of the categories above.

Event receiver records will have the device type (DTYP) field set to “MRF EVR”. The input (INP) or output (OUT) link fields will use the VME_IO hardware specification regardless of the event receiver’s actual bus type. The VME_IO specification provides a convenient format for specifying the logical card number, a parameter string to contain a description of the record’s function, and a signal number to indicate which signal, pulser, port, or event to apply the function to.

The INP or OUT fields will contain a string of the form:

```
#Cn Sm @function
```

Where:

n	=	The logical card number of the event receiver card
m	=	Which signal, pulser, port, or event number to act on
function	=	Defines the record’s function.

4.1 Event Receiver Control and Status Records

General status and control records can be subdivided into the following categories:

- **General control records** – Enable and disable the event receiver card, select which event mapping RAM to use, control event forwarding, and display the event clock speed.
- **Event receiver ID records** – Display static information about the event receiver card. This information can include the card's form factor, bus address, firmware revision number and serial number.
- **Event receiver status records** – Display error indicators such as receiver error, heartbeat timeout, FIFO full and clock synchronization as well as error counters.
- **Event logging records** – Enables, starts ,and stops event logging.

An event receiver card control or status record will have the device type (DTYP) field set to "MRF EVR" and the output link (OUT) or input link (INP) field will contain a string of the form:

`#C`*n* `S0 @EVR_`*XXXXXX*

Where:

n	=	The logical card number of the event receiver card, and
`EVR_`*XXXXXX*	=	Defines which function to perform.

Note that the signal number is not used for general control and status records.

Table 4-1summarizes the function values for event receiver card control and status records.

Function Name	Record Type	Default Value	Description
EVR_ENABLE	bo	Enabled	Enable or disable the event receiver card.
EVR_ENABLE	bi	N/A	Displays whether the event receiver card is enabled or disabled.
EVR_FORWARD_ENABLE	mbbo	Disabled	Controls forwarding of events on the event receiver's transmit link.
EVR_MAP_RAM	mbbo	1	Select which event mapping RAM to use.
EVR_PULSER_UNITS	mbbo	*System Default*	Sets the default time units for pulser delay and width records (see section 4.3.2).
EVR_ID_FORM_FACTOR	stringin	N/A	Displays the form factor for event receiver card *n*.
EVR_ID_FW_REV	stringin	N/A	Displays the firmware revision level for event receiver card *n*.
EVR_ID_SERIAL_NUM	stringin	N/A	Displays the serial number of event receiver card *n*.
EVR_EVCLOCK_SPEED	ai	N/A	Displays the event link clock speed in Megahertz.
EVR_ERROR_CLOCK	bi	N/A	Displays whether or not the event receiver card is synchronized with the event clock.
EVR_ERROR_RX	bi	N/A	Displays whether there is a receive error on the event link.
EVR_ERROR_HB	bi	N/A	Displays whether there is a heartbeat timeout on event receiver card *n*.
EVR_ERROR_FF	bi	N/A	Displays whether there is a FIFO Full error on event receiver card *n*.
EVR_ERROR_COUNT_RX	longin	N/A	Displays whether there is a receive error on the event link.
EVR_ERROR_COUNT_HB	longin	N/A	Displays whether there is a heartbeat timeout on event receiver card *n*.
EVR_ERROR_COUNT_FF	longin	N/A	Displays whether there is a FIFO Full error on event receiver card *n*.
EVR_ERROR_RESET	bo	N/A	Reset the error event receiver error counters
EVR_LOG_ENABLE	mbbo	Disabled	Controls logging of received events.
EVR_LOG_START	bo	Stop	Starts or Stops event logging (if enabled) on event receiver card *n*.
EVR_LOG_RESET	bo	N/A	Reset the event log on event receiver card *n*.
EVR_LOG_RUNNING	bi	N/A	Displays whether the event log is running on event receiver card *n*.

Table 4-1
Event Receiver Control and Status Functions

4.1.1 *Enable/Disable Event Receiver*

Enable/Disable Event Receiver (#C*n* S0 @EVR_ENABLE) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable event receiver card *n*
1	Enabled	Enable event receiver card *n*

Enables or disables the event receiver card. If the database contains no EVR_ENABLE record for a given event receiver card the value will default to "Enabled".

If multiple EVR_ENABLE records exist for the same event receiver card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.1.2 *Event Forwarding Control*

Enable Event Forwarding (#C*n* S0 @EVR_FORWARD_ENABLE) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disables event forwarding for event receiver card *n*
1	None	Enables event forwarding on event receiver card *n*, but only those events specifically enabled by EVENT_FORWARD records (see section 4.2.2) will be forwarded.
2	All	Enables event forwarding on event receiver card *n*, All events will be forwarded except those events specifically disabled by EVENT_FORWARD records (see section 4.2.2).

Controls whether events from the event receiver's receive link get forwarded on the event receiver's transmit link. If event forwarding is enabled, you have the choice of only forwarding incoming events that have been explicitly enabled for forwarding by an EVENT_FORWARD record, or forwarding all incoming events except those explicitly disabled by an EVENT_FORWARD record (see section 4.2.2 for information about the EVENT_FORWARD record).

If the database contains no EVR_FORWARD_ENABLE records, the default action will be to disable event forwarding.

If multiple EVR_FORWARD_ENABLE records exist for the same event receiver card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.1.3 *Select Event Mapping RAM*

Select Event Mapping RAM (#C*n* S0 @EVR_MAP_RAM) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable event mapping for event receiver card *n*
1	1	Enable event mapping RAM 1 on event receiver card *n*
2	2	Enable event mapping RAM 2 on event receiver card *n*

Selects which event mapping RAM to use for decoding received events, or disable event mapping. If event mapping is disabled, the event receiver will ignore all incoming events. It will still be able to process distributed data bus signals and data buffer information.

If the database contains no EVR_MAP_RAM records, the default action will be to enable event mapping RAM 1.

If multiple EVR_MAP_RAM records exist for the same event receiver card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.1.4 *Set Pulser Default Time Units*

Set Default Time Units for Pulsers (#C*n* S0 @EVR_PULSER_UNITS) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Ticks	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as “event clock ticks”.
1	Nanoseconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as nanoseconds.
2	Microseconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as microseconds.
3	Milliseconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as milliseconds.
4	Seconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as seconds.
5	Minutes	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as minutes.
6	Hours	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as hours.
7	Days	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as days.
8	Weeks	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as weeks.
9	Fortnights	The time values in the PULSER_DELAY and PULSER_WIDTH records for EVR card *n* will be interpreted as fortnights.

Sets the default time unit for the event receiver’s pulsers. This record determines how the values in the PULSER_DELAY and PULSER_WIDTH records (see section 4.3.2) are interpreted.

If the database contains no EVR_PULSER_UNITS record for a given event receiver card, the system default time unit, as specified in the `MRF_CONFIG_SITE` file, will apply.

If multiple EVR_PULSER_UNITS records exist for the same event receiver card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.1.5 *Event Receiver ID Records*

Event receiver ID records are string input (stringin) records that display static information about an event receiver card. The information in an ID record does not change, so the record only needs to be processed once at system initialization time.

Display Form Factor (#C*n* S0 @EVR_ID_FORM_FACTOR) (stringin Record)

Displays the form factor of the event receiver card (e.g. VME, PMC, cRIO) and in some cases the bus location.

Display Firmware Revision (#C*n* S0 @EVR_ID_FW_REV) (stringin Record)

Displays the firmware revision level of the event receiver card.

Display Serial Number (#C*n* S0 @EVR_ID_SERIAL_NUM) (stringin Record)

When available, displays the event receiver card's serial number. For VME cards, the serial number is the card's MAC address. For PMC and PCI cards the serial number can be set by the user.

4.1.6 *Event Receiver Status Records*

Display Event Link Clock Speed (#C*n* S0 @EVR_EVCLOCK_SPEED) (ai Record)

Displays the event clock speed in Megahertz. The value displayed is the value read from the fractional synthesizer. If your site requires multiple event links running at different clock speeds, then the EVR_EVCLOCK_SPEED record could be a useful check to make sure that the event receiver card is connected to the correct event link.

Display Enable State (#C*n* S0 @EVR_ENABLE) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event receiver is disabled
1	Enabled	Event receiver is disabled

Displays whether the event receiver card is enabled or disabled.

Display Event Clock Error State (#C*n* S0 @EVR_ERROR_CLOCK) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	OK	The event receiver card is synchronized with the event clock.
1	Fault	The event receiver card is not synchronized with the event clock.

Displays whether or not the event receiver card is synchronized with the event clock. If this record indicates “Fault”, there is either a problem with the event link or else the event receiver’s clock speed is not set correctly.

Display Event Link Receiver Error State (#C*n* S0 @EVR_ERROR_RX) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	OK	The event link connection for event receiver card *n* is OK
1	Fault	There is currently a fault condition on the event link connection for event receiver card *n*.

Displays whether or not there is a receiver error on event receiver card *n*. Receiver errors can be caused by a bit error or a loss of signal.

Display Heartbeat Error State (#C*n* S0 @EVR_ERROR_HB) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	OK	The event link heartbeat for event receiver card *n* is OK
1	Fault	The event link heartbeat counter for event receiver card *n* has expired.

Displays whether or not there is a heartbeat error on event receiver card *n*. In order to keep the heartbeat counter from expiring, the event generator needs to periodically send a "Heartbeat Event" (see section 4.2.4) out on the event link. The heartbeat event should be broadcast at least once per second. The heartbeat counter will expire after approximately 1.6 seconds.

Display FIFO Full Error State (#C*n* S0 @EVR_ERROR_FF) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	OK	The event FIFO for event receiver card *n* is not full.
1	Fault	The event FIFO for event receiver card *n* is full.

Displays whether or not the event FIFO on event receiver card *n* is full. The event FIFO is used for time-stamping, posting EPICS events, and other specialized functions that require an interrupt to be generated on the receipt of an event. If the event FIFO fills up, it is possible to lose event interrupts.

Display Event Link Receiver Error Count (#C*n* S0 @EVR_ERROR_COUNT_RX) (longin Record)

Displays the number of event link receiver errors that have occurred since the last time the error counters were reset.

Display Heartbeat Error Count (#C*n* S0 @EVR_ERROR_COUNT_HB) (longin Record)

Displays the number of heartbeat timeouts that have occurred since the last time the error counters were reset.

Display FIFO Full Error Count (#C*n* S0 @EVR_ERROR_COUNT_FF) (longin Record)

Displays the number of event FIFO full errors that have occurred since the last time the error counters were reset.

Reset Error Counters (#C*n* S0 @EVR_ERROR_RESET) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	*	Reset not in progress
1	Reset	Reset the error counters to zero on event receiver card *n*.

This is a momentary binary output record. When a 1 ("Reset") is written to the record, the error counters on event receiver card *n* will be reset to 0. The record will then go back to its 0 state.

4.1.7 *Event Logging Control Records*

Enable Event Logging (#C*n* S0 @EVR_LOG_ENABLE) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disables event logging for event receiver card *n*
1	None	Enables event logging on event receiver card *n*, but only those events specifically enabled by EVENT_LOG records (see section 4.2.3) will be forwarded.
2	All	Enables event logging on event receiver card *n*, All events will be logged except those events specifically disabled by EVENT_LOG records (see section 4.2.3).

Controls whether events from the event receiver's receive link get logged to the event receiver's 511 event circular buffer. If event logging is enabled, you have the choice of only logging incoming events that have been explicitly enabled for logging by an EVENT_LOG record, or logging all incoming events except those explicitly disabled by an EVENT_LOG record (see section 4.2.3 for information about the EVENT_LOG record).

If the database contains no EVR_LOG_ENABLE records, the default action will be to disable event logging.

If multiple EVR_LOG_ENABLE records exist for the same event receiver card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Start/Stop Event Logging (#C*n* S0 @EVR_LOG_START) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Stop	Stops event logging for event receiver card *n*.
1	Start	Starts event logging on event receiver card *n* (assuming event logging has been enabled by the EVR_LOG_ENABLE record).

Starts or stops the event log on event receiver card *n*. Event logging can also be stopped by a "Stop Log" event (see section 4.2.3).

If the database contains no EVR_LOG_START records, the default action will be to stop event logging.

If multiple EVR_LOG_START records exist for the same event receiver card, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Reset Event Logging (#C*n* S0 @EVR_LOG_RESET) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	*	Reset not in progress
1	Reset	Reset the event log on event receiver card *n*.

This is a momentary binary output record. When a 1 ("Reset") is written to the record, the event log will be stopped (if it is running) and the contents of the log will be cleared. The record will then go back to its 0 state. The EVR_LOG_RESET record will work, even if the event log is disabled.

Display Event Logging State (#C*n* S0 @EVR_LOG_RUNNING) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	Stopped	Event logging is not running event receiver card *n*.
1	Running	Event logging is running event receiver card *n*.

Displays whether or not event logging is running on event receiver card *n*.

4.2 Event Configuration Records

At the event receiver end, the term "Event Configuration" refers to what the receiver should do with an event once it has been received. Timing system events can be configured to:

- Turn pulse generators on, off, or trigger a delay/width output sequence.
- Transmit and latch time-stamp information.
- Reset the local clock prescalers.
- Provide heartbeat information to ensure that the event link is up and the event generator is functioning.
- Flash an LED on the front panel of the event receiver.
- Be forwarded on the event receiver's transmit link.
- Post an EPICS event.

The items in the first bullet above are all pulse generator control functions and are covered in the section on "Pulse Generator Configuration Records" (section 4.3). The rest of the items above are referred to as "Internal Functions" and are covered in this section. Many of the internal functions are globally defined at compile time through the site configuration file. Your database would only need to include event configuration records from this section if you wanted to augment or override the default definitions, or if you want to post EPICS events.

Events are configured in the event receiver through an "Event Mapping RAM". Each event receiver has two event mapping RAMs. Only one RAM may be active at a time, but either RAM may be modified at any time. This allows the event receiver to rapidly switch between two "personalities" (see section 4.1.3 for selecting the event mapping RAM).

Most of the event internal functions are configured through a set of binary output (bo), records. An event record is used to post EPICS events. An event configuration record will have the device type (DTYP) field set to "MRF EVR" and the output link (OUT) field will contain a string of the form:

`#C`*n* `S`*m* `@EVENT_`*XXXXXX* `[`*r*`]`

Where:

n	=	The logical card number of the event receiver card
m	=	The event code being configured
`EVENT_`*XXXXXX*	=	Defines which function to perform, and (optional) which event mapping RAM to apply the function to.

The event mapping RAM specification (*r*) is optional, but if included it can have the following values:

Value	Description
1	Use event mapping RAM 1
2	Use event mapping RAM 2
A	Use the currently active event mapping RAM
I	Use the currently inactive event mapping RAM
B	Map the event using both event mapping RAMs

Table 4-2
Event Mapping RAM Selection Values

If the event mapping RAM specification is omitted, both mapping RAMS will be mapped ("Option B").

Table 4-3 summarizes the function values for event configuration records.

Function Name	Record Type	Default Value	Description
[*r*]	event	0	Post an EPICS event
EVENT_FORWARD [r]	bo	Disabled	Forward this event to the event receiver's transmit link.
EVENT_LOG [*r*]	bo	Disabled	Log this event in the event receiver's event log.
EVENT_LOG_STOP [*r*]	bo	Disabled	Stop event logging.
EVENT_HEARTBEAT [*r*]	bo	Disabled	Heartbeat event (useful indicator of event link continuity)
EVENT_LED [*r*]	bo	Disabled	Flashes red on the "EVT" LED on the event receiver's front panel
EVENT_PRESCALER_RESET [*r*]	bo	Disabled	Reset all the event receiver's clock prescalers.
EVENT_TS_RESET [*r*]	bo	Disabled	Reset the low-order 32-bit word of the event receiver's time-stamp counter.
EVENT_TS_LATCH [*r*]	bo	Disabled	Latch the current value of the event receiver's time-stamp counters
EVENT_TS_CLOCK [*r*]	bo	Disabled	Increment the low-order 32-bit word of the event receiver's time-stamp counter.
EVENT_TS_SECONDS_0 [*r*]	bo	Disabled	Shift a 0 into the high-order 32-bit word (seconds counter) of the event receiver's time-stamp.
EVENT_TS_SECONDS_1 [*r*]	bo	Disabled	Shift a 1 into the high-order 32-bit word (seconds counter) of the event receiver's time-stamp.

Table 4-3
Event Configuration Functions

4.2.1 *EPICS Event Posting*

An EPICS event is a software construct. Posting an EPICS event will cause record processing for any record that has its SCAN field set to "Event" and its EVNT field set to the same value as the posted EPICS event. EPICS event codes can range from 1 to 255.

Post EPICS Event (#C*n* S*m* @[*r*]) (event Record)

When timing system event *m* is seen on the event receiver card *n*, the event record will post the EPICS event specified in its VAL field. The EPICS event in the VAL field does not need to be the same as the timing system event specified in the INP field. If the VAL field is 0, no EPICS event will be posted.

A standard EPICS "event" record is used to post EPICS events. Since this is the only function the event record performs, the only parameter needed in the INP field is the optional event mapping RAM selector. . The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

If multiple event records exist for the same timing system event, each event record will be processed when the timing system event occurs.

4.2.2 *Event Forwarding*

Enable Event Forwarding for Event (#C*n* S*m* @EVENT_FORWARD [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be forwarded on the event receiver's transmit link.
1	Enabled	Event *m* will be forwarded on the event receiver's transmit link.

Enables (or disables) forwarding for event *m*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

Event forwarding is controlled by the EVR_FORWARD_ENABLE record (see section 4.1.2). If the EVR_FORWARD_ENABLE record is set to "Disabled", event forwarding is disabled and EVENT_FORWARD records have no effect. If the EVR_FORWARD_ENABLE record is set to "None", only those events enabled by an EVENT_FORWARD record will be forwarded. If the EVR_FORWARD_ENABLE record is set to "All", all events will be forwarded except those disabled by and EVENT_FORWARD record.

If multiple EVENT_FORWARD records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.2.3 *Event Logging Records*

Enable/Disable Event Logging for Event (#C*n* S*m* @EVENT_LOG [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be logged in the event receiver's circular log buffer.
1	Enabled	Event *m* will be logged in the event receiver's circular log.

Enables (or disables) logging for event *m*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

Event logging is controlled by the EVR_LOG_ENABLE record (see section 4.1.7). If the EVR_LOG_ENABLE record is set to "Disabled", event logging is disabled and EVENT_LOG records have no effect. If the EVR_LOG_ENABLE record is set to "None", only those events enabled by an EVENT_LOG record will be logged. If the EVR_LOG_ENABLE record is set to "All", all events will be logged except those disabled by and EVENT_LOG record.

If multiple EVENT_LOG records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Log Stop Event (#C*n* S*m* @EVENT_LOG_STOP [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not stop event logging on EVR card *n*..
1	Enabled	Enable event *m* to stop event logging on EVR card *n*.

Enables (or disables) event *m* to stop event logging on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

If the database contains no EVENT_LOG_STOP records for the specified event receiver, it will use the default "Log Stop" event (0x79) unless a different default is defined in the site configuration file. Note that enabling an event to be a "Log Stop" event does not automatically disable the default "Log Stop" event or any other events that may also be enabled as "LogStop" events.

If multiple EVENT_LOG_STOP records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.2.4 *Diagnostic Events*

Set Heartbeat Event (#C*n* S*m* @EVENT_HEARTBEAT [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be a Heartbeat event on EVR card *n*..
1	Enabled	Enable event *m* to be a Heartbeat event on EVR card *n*.

Enables (or disables) event *m* to be a heartbeat event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

Event receivers maintain a heartbeat counter which is reset by the receipt of a "Heartbeat" event. If the heartbeat counter times out (in approximately 1.6 seconds) an error flag will be set and, if enabled, a heartbeat interrupt will be generated.

If the database contains no EVENT_HEARTBEAT records for the specified event receiver, it will use the default heartbeat event (0x7A) unless a different default is defined in the site configuration file. Note that enabling an event to be a heartbeat event does not automatically disable the default heartbeat event or any other events that may also be enabled as heartbeat events.

If multiple EVENT_HEARTBEAT records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set LED Event (#C*n* S*m* @EVENT_LED [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be an LED event on EVR card *n*..
1	Enabled	Enable event *m* to be an LED event on EVR card *n*.

Enables (or disables) event *m* to be an LED event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

When an LED event is received by the event receiver, an LED light will flash on the event receiver's front panel.

If multiple EVENT_LED records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Prescaler Reset Event (#C*n* S*m* @EVENT_PRESCALER_RESET [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be a prescaler reset event on EVR card *n*..
1	Enabled	Enable event *m* to be a prescaler reset event on EVR card *n*.

Enables (or disables) event *m* to be a prescaler reset event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

When a prescaler reset event is received by the event receiver, all of the event receiver's local counters will be reset to their starting values. This allows all of the counter outputs to have the same phase across all event receivers. See section 4.4 for how to configure the local clocks.

If multiple EVENT_PRESCALER_RESET records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.2.5 *Time-Stamp Events*

The event receiver card maintains a running time-stamp that can be applied to events when they are stored in the event FIFO or logged to the event log. The time-stamp can also be read out to get the current time or latched to record a significant time (e.g. the start of a machine cycle).

The time-stamp consists of two 32-bit words. The high-order word is the "seconds counter" which typically represents the number of seconds from some epoch. Common examples include the Unix epoch (midnight, January 1, 1970) and the EPICS epoch (midnight, January 1, 1990). The seconds counter is not technically a counter. It is set by shifting bits from the event link into the register. The bits can either come from specially designated "times-stamp seconds" events, or from bits 6 and 7 of the distributed data bus. The seconds counter must be reloaded every second. The low-order word is the "high frequency" counter. This is an actual counter and can count either event clock ticks, specially designated "time-stamp clock" events, or bit 4 of the distributed data bus.

The time-stamping mechanism is usually set up at build time using the site configuration files. You can, however, use the following records to override the site defaults.

Set Time-Stamp Reset Event (#C*n* S*m* @EVENT_TS_RESET [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be a time-stamp reset event on EVR card *n*
1	Enabled	Enable event *m* to be a time-stamp reset event on EVR card *n*

Enables (or disables) event *m* to be a time-stamp reset event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

When a time-stamp reset event is seen by the event receiver, it will clear the high frequency counter (low-order 32 bits), and reload the seconds counter using the value previously shifted in to the seconds shift register.

If the database contains no EVENT_TS_RESET records for the specified event receiver, it will use the default time-stamp reset event (0x7B) unless a different default is defined in the site configuration file. Note that enabling an event to be a time-stamp reset event does not automatically disable the default time-stamp reset event or any other events that may also be enabled as time-stamp reset events.

If multiple EVENT_TS_RESET records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Time-Stamp Latch Event (#C*n* S*m* @EVENT_TS_LATCH [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be a time-stamp latch event on EVR card *n*
1	Enabled	Enable event *m* to be a time-stamp latch event on EVR card *n*

Enables (or disables) event *m* to be a time-stamp latch event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

When a time-stamp latch event is seen by the event receiver, it will "latch" the current time-stamp into a set of registers where it can safely be read by other software.

If the database contains no EVENT_TS_LATCH records for the specified event receiver, it will use the default time-stamp latch event (if specified) from the site configuration file. Note that enabling an event to be a time-stamp latch event does not automatically disable any other events that may also be enabled as time-stamp latch events.

If multiple EVENT_TS_LATCH records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Time-Stamp Clock Event (#C*n* S*m* @EVENT_TS_CLOCK [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be a time-stamp clock event on EVR card *n*
1	Enabled	Enable event *m* to be a time-stamp clock event on EVR card *n*

Enables (or disables) event *m* to be a time-stamp clock event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

When a time-stamp clock event is seen by the event receiver, and the event receiver is configured to use time-stamp clock events, it will increment the high frequency counter (low-order 32 bits) of the time-stamp.

If the database contains no EVENT_TS_CLOCK records for the specified event receiver, it will use the default time-stamp clock event (0x7C) unless a different default is defined in the site configuration file. Note that enabling an event to be a time-stamp clock event does not automatically disable the default time-stamp clock event or any other events that may also be enabled as time-stamp clock events.

If multiple EVENT_TS_CLOCK records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Time-Stamp Seconds 0 Event (#C*n* S*m* @EVENT_TS_SECONDS_0 [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be a time-stamp seconds 0 event on EVR card *n*
1	Enabled	Enable event *m* to be a time-stamp seconds 0 event on EVR card *n*

Enables (or disables) event *m* to be a "time-stamp seconds 0" event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

If the event receiver is configured to accept seconds shifting events, a "time-stamp seconds 0" event will shift a "0" value into the time-stamp seconds shift register. A sequence of "time-stamp seconds 0" and "time-stamp seconds 1" events are used to shift a new value (most significant bit first) into the time-stamp seconds shift register. The contents of the time-stamp seconds shift register will become the new "seconds counter" (high order 32-bits) of the time-stamp when the next time-stamp reset event occurs.

If the database contains no EVENT_TS_SECONDS_0 records for the specified event receiver, it will use the default "time-stamp seconds 0" event (0x70) unless a different default is defined in the site configuration file. Note that enabling an event to be a "time-stamp seconds 0" event does not automatically disable the default "time-stamp seconds 0" event or any other events that may also be enabled as "time-stamp seconds 0" events.

If multiple EVENT_TS_SECONDS_0 records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Time-Stamp Seconds 1 Event (#C*n* S*m* @EVENT_TS_SECONDS_1 [*r*]) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Event *m* will not be a time-stamp seconds 1 event on EVR card *n*
1	Enabled	Enable event *m* to be a time-stamp seconds 1 event on EVR card *n*

Enables (or disables) event *m* to be a "time-stamp seconds 1" event on event receiver card *n*. The optional parameter (*r*) can be used to indicate which event mapping RAM to use. Valid values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs.

If the event receiver is configured to accept seconds shifting events, a "time-stamp seconds 1" event will shift a "1" value into the time-stamp seconds shift register. A sequence of "time-stamp seconds 0" and "time-stamp seconds 1" events are used to shift a new value (most significant bit first) into the time-stamp seconds shift register. The contents of the time-stamp seconds shift register will become the new "seconds counter" (high order 32-bits) of the time-stamp when the next time-stamp reset event occurs.

If the database contains no EVENT_TS_SECONDS_1 records for the specified event receiver, it will use the default “time-stamp seconds 1” event (0x71) unless a different default is defined in the site configuration file. Note that enabling an event to be a “time-stamp seconds 1” event does not automatically disable the default “time-stamp seconds 1” event or any other events that may also be enabled as “time-stamp seconds 1” events.

If multiple EVENT_TS_SECONDS_1 records exist for the same event and the same event receiver card using the same mapping RAM, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.3 Pulse Generator Configuration Records

An Event Receiver pulse generator is defined by a set of ao, bi, bo, mbbo, and longout records. The output records (ao, bo, mbbo, and longout) set the pulse generator parameters and the input record (bi) reads the current state of the pulse generator output. A pulse generator configuration record will have the device type (DTYP) field set to "MRF EVR" and the output link (OUT) field will contain a string of the form:

`#C`*n* `S`*m* `@PULSER_`*XXXXXX* `[`*r*`]`

Where:

n	= The logical card number of the Event Receiver card
m	= The pulse generator number,
r	= (optional) the event mapping RAM to use when setting trigger, set, and reset events (see sections 4.1.3 and 4.2), and
`PULSER_`*XXXXXX*	= Defines which function to perform.

Table 4-4 summarizes the function values for a pulse generator.

Function Name	Record Type	Default Value	Description
PULSER_ENABLE	bo	Enabled	Enable/Disable pulse generator
PULSER_PRESCALER	longout	1	Set pulse generator prescaler register
PULSER_DELAY	ao	0	Set pulse generator delay register
PULSER_WIDTH	ao	1	Set pulse generator width record
PULSER_POLARITY	bo	Normal	Set pulse generator output polarity
PULSER_UNITS	mbbo	Ticks	Set time units for PULSER_DELAY and PULSER_WIDTH records.
PULSER_TRIGGER_MODE	mbbo	Trigger	Set pulse generator trigger mode.
PULSER_TRIGGER_EVENT [*r*]	longout	0	Set event to trigger the pulse generator's delay and width counters.
PULSER_SET_EVENT [*r*]	longout	0	Set event to set the pulse generator to the "Set" state
PULSER_RESET_EVENT [*r*]	longout	0	Set event to set the pulse generator to the "Reset" state.
PULSER_STATE	bi	N/A	Readback of pulse generator state
PULSER_SET	bo	N/A	Software set/reset pulse generator state

Table 4-4
Pulse Generator Functions

4.3.1 *Enable/Disable Pulse Generator*

Enable/Disable Pulse Generator (#C*n* S*m* @PULSER_ENABLE) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable pulse generator *m* on EVR card *n*
1	Enabled	Enable pulse generator *m* on EVR card *n*

If the database contains no PULSER_ENABLE record for a given pulse generator, but it does contain other pulse generator configuration records for that pulse generator, the value will

default to “Enabled”. If the database contains no pulse generator configuration records for a particular pulse generator, then the value will default to “Disabled”.

If multiple PULSER_ENABLE records exist for the same pulse generator, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.3.2 *Set Pulse Parameters*

Analog output (ao) records are used to set the pulse delay and width for triggered pulses. A long output record (longout) is used to set the prescaler value. A binary output (bo) record is used to set the pulse polarity, and a multi-bit binary output (mbbo) record is used to determine the time scale for the PULSER_DELAY and PULSER_WIDTH records.

Pulse Generator Prescaler Value (#C*n* S*m* @PULSER_PRESCALER) (longout Record)

Specifies the prescaler value for pulse generator *m* on EVR card *n*. The value must be between 1 and the maximum prescaler value (which may be different for different event receiver card types). The prescaler value divides the event clock feeding the delay and width counters. This allows the pulse generator to produce gates with longer delays and widths at reduced resolution.

If the database contains no PULSER_PRESCALER record for a given pulse generator the value will default to 1 (no prescaling).

If multiple PULSER_PRESCALER records exist for the same pulse generator, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Changing the value of the PULSER_PRESCALER record will not, as a rule, change the observed delay or width of a triggered pulse unless the PULSER_UNITS record is set to “Ticks”. When the PULSER_UNITS record is set to “Ticks” the pulse generator will count scaled event clock ticks. When the PULSER_UNITS record is set to any other value, the device support code will adjust the contents of the pulse generator’s delay and width registers to account for the new prescaler value. In this case, the only time the pulse delay or width will change is if the new prescaled values are outside the range of the counter registers.

Pulse Generator Delay (#C*n* S*m* @PULSER_DELAY) (ao Record)

Specifies the delay from the triggering event to the start of pulse for pulse generator *m* on EVR card *n*. The value must be between 0 and the maximum delay value (which may be different for different event receiver card types).

How the value in the PULSER_DELAY record is interpreted (time units) depends on the units set in the PULSER_UNITS record for pulser *m*. If there is on PULSER_UNITS record for pulser *m*, the value depends on the units set in the EVR_PULSER_UNITS record for

event receiver card *n*. If there is no EVR_PULSER_UNITS record for event receiver card *n*, then the value depends on the system default time unit, as specified in the `MRF_CONFIG_SITE` file. If the time unit (as derived above) is "Ticks", then the value in the PULSER_DELAY record is interpreted as "pre-scaled event clock ticks". In this case the integer part of the PULSER_DELAY record value is written directly to the pulse generator's delay register. If the more conventional time unit is specified (e.g., nanoseconds, microseconds, etc.), then the value written to the delay register will be computed from the PULSER_DELAY record value, the PULSER_PRESCALER value, and the event clock speed.

If multiple PULSER_DELAY records exist for the same pulse generator, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Pulse Generator Width (#C*n* S*m* @PULSER_WIDTH) (ao Record)

Specifies the pulse width for pulse generator *m* on EVR card *n*. The value must be between 0 and the maximum width value (which may be different for different event receiver card types).

How the value in the PULSER_WIDTH record is interpreted (time units) depends on the units set in the PULSER_UNITS record for pulser *m*. If there is on PULSER_UNITS record for pulser *m*, the value depends on the units set in the EVR_PULSER_UNITS record for event receiver card *n*. If there is no EVR_PULSER_UNITS record for event receiver card *n*, then the value depends on the system default time unit, as specified in the `MRF_CONFIG_SITE` file If the time unit (as derived above) is "Ticks", then the value in the PULSER_WIDTH record is interpreted as "pre-scaled event clock ticks". In this case the integer part of the PULSER_WIDTH record value is written directly to the pulse generator's width register. If the more conventional time unit is specified (e.g., nanoseconds, microseconds, etc.), then the value written to the width register will be computed from the PULSER_WIDTH record value, the PULSER_PRESCALER value, and the event clock speed.

If the database contains no PULSER_WIDTH record for a given pulse generator the value will default to 1.

If multiple PULSER_WIDTH records exist for the same pulse generator, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Pulse Generator Polarity (#C*n* S*m* @PULSER_POLARITY) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Normal	Pulse generator *m* on EVR card *n* will have normal polarity (set state is high, reset state is low)
1	Inverted	Pulse generator *m* on EVR card *n* will have inverted polarity (set state is low, reset state is high)

Sets the polarity of the output pulse to either “Normal” (high true) or “Inverted” (low true).

If the database contains no PULSER_POLARITY record for a given pulse generator the value will default to “Normal”.

If multiple PULSER_POLARITY records exist for the same pulse generator, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Pulse Generator Time Units (#C*n* S*m* @PULSER_UNITS) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Ticks	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as “event clock ticks”.
1	Nanoseconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as nanoseconds.
2	Microseconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as microseconds.
3	Milliseconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as milliseconds .
4	Seconds	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as seconds.
5	Minutes	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as minutes.
6	Hours	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted in as hours.
7	Days	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as days.
8	Weeks	The values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted in as weeks.
9	Fortnights	The time values in the PULSER_DELAY and PULSER_WIDTH records for pulse generator *m* on EVR card *n* will be interpreted as fortnights.

If the database contains no PULSER_UNITS record for a given pulse generator the value will default to the value specified in the EVR_PULSER_UNITS record (see section 4.1.4). If no EVR_PULSER_UNITS record exists for the event receiver card, the system default, as specified in the `MRF_CONFIG_SITE` file, will apply.

If multiple PULSER_UNITS records exist for the same pulse generator, only the first record initialized will be used. The other records will be disabled and error messages will be

displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

EPICS device support takes into account the event clock rate as well as the values of PULSER_UNITS and PULSER_PRESCALER record in computing the raw values for the PULSER_DELAY and PULSER_WIDTH records.

4.3.3 *Set Pulse Generator Trigger Events*

Long output (longout) records are used to set the trigger, set, and reset events for a pulse generator. A multi-bit binary output (mbbo) is used to determine how the pulse generator is triggered.

Pulse Generator Trigger Mechanism (#C*n* S*m* @PULSER_TRIGGER_MODE) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Trigger	Pulse generator *m* on EVR card *n* is triggered by the event specified in a PULSER_TRIGGER_EVENT record.
1	Latch	Pulse generator *m* on EVR card *n* is latched "on" by the event specified in a PULSER_SET_EVENT record and is latched "off" by the event specified in a PULSER_RESET_EVENT record. The PULSER_WIDTH, PULSER_DELAY, and PULSER_PRESCALER records, if present, are not used.
2	Multiple	Allows a pulse generator to have multiple TRIGGER, SET, and/or RESET events. This allows the most flexibility for controlling a pulse generator and carries the most potential for unexpected results.

If the database contains no PULSER_TRIGGER_MODE record for a given pulse generator the value will default to "Trigger"

If multiple PULSER_TRIGGER_MODE records exist for the same pulse generator, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

When "Trigger" is selected, only one PULSER_TRIGGER_EVENT record is permitted. PULSER_SET_EVENT and PULSER_RESET_EVENT records will be ignored. The values in the PULSER_WIDTH and PULSER_DELAY records determine the width of the pulse and its delay with respect to the triggering event. The PULSER_PRESCALER record will be used to determine the clock rate of the delay and width parameters.

When "Latch" is selected, only one PULSER_SET_EVENT record and one PULSER_ RESET_EVENT record are permitted. PULSER_TRIGGER_EVENT, PULSER_DELAY, PULSER_WIDTH, and PULSER_PRESCALER records will be ignored.

When "Multiple" is selected, you may have as many PULSER_TRIGGER_EVENT, PULSER_SET_EVENT, and PULSER_RESET_EVENT records as you want. They will all be active and will all try to control the pulse generator whenever their specified events occur. You are still limited, however, to only one PULSER_DELAY record, one PULSER_WIDTH record, and one PULSER_PRESCALER record.

Pulse Generator Trigger Event (#C*n* S*m* @PULSER_TRIGGER_EVENT [*r*]) (longout Record)

Specifies the trigger event for pulse generator *m* on EVR card *n*. The value must be between 0 and 255. You may optionally specify which event mapping RAM to use for this assignment. The legal values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs. See sections 4.1.3 and 4.2 for more information.

This record is only used if the PULSER_TRIGGER_MODE record is set to "Trigger" or "Multiple". When an event triggers a pulse generator the contents of the PULSER_DELAY, PULSER_WIDTH, PULSER_PRESCALER, and PULSER_UNITS records determine how long the pulse generator will wait after the event occurs before entering its set state and then how long it remains set before going to the reset state.

If the database contains no PULSER_TRIGGER_EVENT record for a given pulse generator, and the PULSER_TRIGGER_MODE record specifies "Trigger", the value will default to event 0 which will effectively disable the pulse generator as event 0 never occurs.

If multiple PULSER_TRIGGER_EVENT records exist for the same pulse generator, and the PULSER_TRIGGER_MODE record is set to "Trigger", only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Pulse Generator Set Event (#C*n* S*m* @PULSER_SET_EVENT [*r*]) (longout Record)

Specifies the "Set" event for pulse generator *m* on EVR card *n*. The value must be between 0 and 255. You may optionally specify which event mapping RAM to use for this assignment. The legal values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs. See sections 4.1.3 and 4.2 for more information.

This record is only used if the PULSER_TRIGGER_MODE record is set to "Latch" or "Multiple". When the specified event occurs, the pulse generator enters its "Set" state and remains there until a "Reset" event occurs or until the pulse generator is manually reset.

If multiple PULSER_SET_EVENT records exist for the same pulse generator using the same mapping RAM, and the PULSER_TRIGGER_MODE record is set to "Latch", only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Pulse Generator Reset Event (#C*n* S*m* @PULSER_RESET_EVENT [*r*]) (longout Record)

Specifies the "Reset" event for pulse generator *m* on EVR card *n*. The value must be between 0 and 255. You may optionally specify which event mapping RAM to use for this assignment. The legal values for *r* are given in Table 4-2 on page 38. If *r* is omitted, the event will be mapped into both event mapping RAMs. See sections 4.1.3 and 4.2 for more information.

This record is only used if the PULSER_TRIGGER_MODE record is set to “Latch” or “Multiple”. When the specified event occurs, the pulse generator enters its “Reset” state and remains there until a “Set” event occurs or until the pulse generator is manually set.

If multiple PULSER_RESET_EVENT records exist for the same pulse generator using the same mapping RAM, and the PULSER_TRIGGER_MODE record is set to “Latch”, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.3.4 *Read and Write the Pulse Generator State*

The output state of a pulse generator can also be read and controlled by software. A binary output (bo) record is provided to set or reset the output state of the pulse generator. A binary input (bi) record is provided to read the current state of the pulse generator.

Display Pulse Generator State (#C*n* S*m* @PULSER_STATE) (bi Record)

Raw Value	Engineering Value (in template files)	Description
0	Off	Pulse generator *m* on EVR card *n* is in its “Off” or “Reset” state.
1	On	Pulse generator *m* on EVR card *n* is in its “On” or “Set” state.

Reads the current state of a pulse generator.

Set Pulse Generator State (#C*n* S*m* @PULSER_SET) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Off	Forces pulse generator *m* on EVR card *n* to its “Off” or “Reset” state.
1	On	Forces pulse generator *m* on EVR card *n* to its “On” or “Set” state.

Sets the state of a pulse generator.

Multiple PULSER_SET records are allowed for the same pulse generator. The pulse generator output will reflect the last value set by the last PULSER_SET record to be processed.

4.4 Local Clock Configuration Records

Event Receivers are capable of generating local clock signals which are derived from and are multiples of the event clock frequency. Only one record, the CLOCK_PRESCALER record, is required to define an event receiver clock. A special "Clock Prescaler Reset" event (default 0x7B) will cause all event receiver clock prescalers to be synchronously reset so that the frequency outputs can have the same phase across all event receivers.

An event receiver clock configuration record will have the device type (DTYP) field set to "MRF EVR" and the output link (OUT) field will contain a string of the form:

`#C`*n* `S`*m* `@CLOCK_`*XXXXXX*

Where:

n	=	The logical card number of the Event Receiver card
m	=	The event receiver clock number, and
`CLOCK_`*XXXXXX*	=	Defines which function to perform.

Table 4-5 summarizes the function values for event receiver clocks.

Function Name	Record Type	Default Value	Description
CLOCK_PRESCALER	longout	1	Set clock prescaler value

Table 4-5
Event Receiver Clock Functions

Only one long output (longout) record is required for an event receiver clock.

Clock Prescaler (#C*n* S*m* @CLOCK_PRESCALER) (longout Record)

Specifies the divider for clock *m* on EVR card *n*. The value must be between 0 and the maximum clock prescaler value (which may be different for different event receiver card types).

If the database contains no CLOCK_PRESCALER record for a given clock, the value will default to 1.

If multiple CLOCK_PRESCALER records exist for the same clock, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.5 Output Port Configuration Records

Depending on the type of card, an event receiver can generate output signals at hardwired front-panel ports, Universal I/O front-panel ports, or transition board (rear panel) ports. Note that all transition board ports contain Universal I/O outputs. Consequently, when the term "Universal I/O port" is used in this section, it refers to the Universal I/O ports on the event receiver module's front panel and the term "Transition Board Port" refers to the Universal I/O ports on the transition board.

Output ports can be assigned to pulse generators (section 4.3), local clocks (section 4.4), or distributed data bus signals.

An event receiver output port configuration record will have the device type (DTYP) field set to "MRF EVR" and the output link (OUT) field will contain a string of the form:

`#C`*n* `S`*m* `@OUTPUT_`*XXXXXX*

Where:

n	=	The logical card number of the Event Receiver card
m	=	The port number of the Front Panel port, Transition Board port, or Universal I/O port, and
`OUTPUT_`*XXXXXX*	=	Defines which function to perform.

An mbbo record and a longout record pair are used to configure most output ports. The mbbo record determines what type of signal the output port is connected to, and the longout record determines which specific pulser, clock, or data bus bit is to be connected. If no configuration records exist for a particular output port, that port's output will be held low

In addition to signal type and signal number records, some output ports can take additional configuration records. The CML front panel outputs can be configured with bit patterns for low, high, turn on, and turn off states. Also, some Universal I/O LVPECL outputs can take local picosecond delay values.

Table 4-6 summarizes the function values for event receiver output ports.

Function Name	Record Type	Default Value	Description
OUTPUT_FP_TYPE	mbbo	Low	Defines what type of signal (pulser, local clock, data bus bit, etc.) the specified front panel output port is connected to.
OUTPUT_FP_CHAN	longout	0	Defines which signal (of the type specified by the matching OUTPUT_FP_TYPE record) will be connected to the front panel output port.
OUTPUT_TB_TYPE	mbbo	Low	Defines what type of signal (pulser, local clock, data bus bit, etc.) the specified transition board output port is connected to.
OUTPUT_TB_CHAN	longout	0	Defines which signal (of the type specified by the matching OUTPUT_TB_TYPE record) will be connected to the transition board output port.
OUTPUT_UNIV_TYPE	mbbo	Low	Defines what type of signal (pulser, local clock, data bus bit, etc.) the specified front panel Universal I/O output port is connected to.
OUTPUT_UNIV_CHAN	longout	0	Defines which signal (of the type specified by the matching OUTPUT_UNIV_TYPE record) will be connected to the front panel Universal I/O output port.
OUTPUT_FP_PATTERN_ENA	bo	Disabled	Enable front panel CML pattern output for an output port
OUTPUT_FP_PATTERN_LOW	longout	0x00000	Set pattern to output when the signal connected to the output port is low.
OUTPUT_FP_PATTERN_RISE	longout	0xFFFFF	Set pattern to output when the signal connected to the output port transitions from low to high.
OUTPUT_FP_PATTERN_HIGH	longout	0xFFFFF	Set pattern to output when the signal connected to the output port is high.
OUTPUT_FP_PATTERN_FALL	longout	0x00000	Set pattern to output when the signal connected to the output port transitions from high to low.
OUTPUT_UNIV_DELAY_ENA	bo	Disabled	Enable fine delay on a front panel Universal I/O LVPECL with delay output port.
OUTPUT_UNIV_DELAY	longout	0	Set the fine delay on a front panel Universal I/O LVPECL with delay output port.

Table 4-6
Event Receiver Output Port Configuration Functions

4.5.1 *Assign Signals to Output Ports*

A multi-bit binary output (mbbo) record is used to determine which type of signal an output port is assigned to. A long output (longout) record is used to determine which specific input signal to assign to the output port. The type of output port to be assigned is determined by the function field (OUTPUT_*xxx*_YYYYY), where "*xxx*" can either be:

FP Fixed front panel port
TB Transition board (rear panel) port
UNIV Front panel Universal I/O port

Output Port Signal Type (#C*n* S*m* @OUTPUT_*xxx*_TYPE) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Low	Output port is held low
1	High	Output port is held high
2	Pulser	Output port is assigned to a pulse generator.
3	DBus	Output port is assigned to a distributed data bus signal.
4	Clock	Output port is assigned to a local clock.

Defines which type of signal will be connected to the output port number *m* of type *xxx* on event receiver card *n*.

If the database contains no OUTPUT_*xxx*_TYPE record for a given pulse output port the value will default to "Low"

If multiple OUTPUT_*xxx*_TYPE records exist for the same output port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Output Port Signal Channel (#C*n* S*m* @OUTPUT_*xxx*_CHAN) (longout Record)

Defines which channel (of the signal type defined by the OUTPUT_*xxx*_TYPE record) will be connected to the output port number *m* of type *xxx* on event receiver card *n*.

If the database contains no OUTPUT_*xxx*_CHAN record for a given output port, the value will default to 0. If the value of the longout record exceeds the number of signals supported by the event receiver card for the specified output type (specified in the OUTUPT_*xxx*_TYPE record), an error will be displayed on the IOC console terminal and in the error log and the port's output will be held low.

If multiple OUTPUT_*xxx*_CHAN records exist for the same output port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.5.2 *CML Pattern Definition*

Long output (longout) records are used to define the 20-bit patterns that can be output when the signal connected to an output port is low, high, or transitioning between states. A binary output record is used to enable and disable CML pattern output.

Enable/Disable CML Pattern Output (#C*n* S*m* @OUTPUT_FP_PATTERN_ENA) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable CML pattern output for logical output port *m* on EVR card *n*.
1	Enabled	Enable CML pattern output for logical output port *m* on EVR card *n*.

The fixed front panel output port number (*m*) must correspond to one of the CML front panel outputs.

If the database contains no OUTPUT_FP_PATTERN_ENA record for a given CML output port, the value will default to “Disabled”.

If multiple OUTPUT_FP_PATTERN_ENA records exist for the same CML output port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

CML Pattern Output When Signal is Low (#C*n* S*m* @OUTPUT_FP_PATTERN_LOW) (longout Record)

Sets the pattern to be output when the signal connected to the output port is low. The value is a 20-bit unsigned integer between 0x00000 and 0xFFFFF (1,048,575). The high-order bit of the value is sent first. The low-order bit of the value is sent last.

The fixed front panel output port number (*m*) must correspond to one of the CML front panel outputs.

If the database contains no OUTPUT_FP_PATTERN_LOW record for a given CML output port, the value will default to 0x00000.

If multiple OUTPUT_FP_PATTERN_LOW records exist for the same CML output port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

CML Pattern Output When Signal is Rising (#C*n* S*m* @OUTPUT_FP_PATTERN_RISE) (longout Record)

Sets the pattern to be output when the signal connected to the output port transitions from low to high. The value is a 20-bit unsigned integer between 0x00000 and 0xFFFFF (1,048,575). The high-order bit of the value is sent first. The low-order bit of the value is sent last.

The fixed front panel output port number (*m*) must correspond to one of the CML front panel outputs.

If the database contains no OUTPUT_FP_PATTERN_RISE record for a given CML output port, the value will default to 0xFFFFF.

If multiple OUTPUT_FP_PATTERN_RISE records exist for the same CML output port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

CML Pattern Output When Signal is High (#C*n* S*m* @OUTPUT_FP_PATTERN_HIGH) (longout Record)

Sets the pattern to be output when the signal connected to the output port is high. The value is a 20-bit unsigned integer between 0x00000 and 0xFFFFF (1,048,575). The high-order bit of the value is sent first. The low-order bit of the value is sent last.

The fixed front panel output port number (*m*) must correspond to one of the CML front panel outputs.

If the database contains no OUTPUT_FP_PATTERN_HIGH record for a given CML output port, the value will default to 0xFFFFF.

If multiple OUTPUT_FP_PATTERN_HIGH records exist for the same CML output port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

CML Pattern Output When Signal is Falling (#C*n* S*m* @OUTPUT_FP_PATTERN_FALL) (longout Record)

Sets the pattern to be output when the signal connected to the output port transitions from high to low. The value is a 20-bit unsigned integer between 0x00000 and 0xFFFFF (1,048,575). The high-order bit of the value is sent first. The low-order bit of the value is sent last.

The fixed front panel output port number (*m*) must correspond to one of the CML front panel outputs.

If the database contains no OUTPUT_FP_PATTERN_FALL record for a given CML output port, the value will default to 0x00000.

If multiple OUTPUT_FP_PATTERN_FALL records exist for the same CML output port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.5.3 *Fine Delay for LVPECL With Delay Tuning Output Ports*

The LVPECL With Delay Tuning Universal I/O Module is capable of generating a fine delay of up to 10 nanoseconds (approximately) in 10 picosecond (approximately) steps. A long output (longout) record sets the value of the fine delay and a binary output (bo) record enables or disables the fine delay.

Enable/Disable LVPECL Fine Delay (#C*n* S*m* @OUTPUT_UNIV_DELAY_ENA) (bo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable fine delay generation for logical output port *m* on EVR card *n*.
1	Enabled	Enable fine delay generation for logical output port *m* on EVR card *n*.

In order for a delay to be generated, the front panel Universal I/O port number (*m*) must contain an “LVPECL with Delay Tuning” module.

If the database contains no OUTPUT_UNIV_DELAY_ENA record for a given “LVPECL with Delay Tuning” port, the value will default to “Disabled”.

If multiple OUTPUT_UNIV_DELAY_ENA records exist for the same port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set LVPECL Fine Delay (#C*n* S*m* @OUTPUT_UNIV_DELAY) (longout Record)

Sets the fine delay in a front panel Universal I/O LVPECL with Delay Tuning output port. The value must be between 0 and 1023. The delay resolution is between 9 and 10 picoseconds per step.

In order for a delay to be generated, the front panel Universal I/O port number (*m*) must contain an “LVPECL with Delay Tuning” module.

If the database contains no OUTPUT_UNIV_DELAY record for a given port, the value will default to 0.

If multiple OUTPUT_UNIV_DELAY records exist for the same port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.6 Input Port Configuration Records

An event receiver card may also have one or more active front panel input ports. These ports take a TTL input signal and can be programmed to:

- Output the signal to one or more distributed data bus lines on the event receiver's transmit link.
- Generate an internal event (edge triggered) that behaves the same as if it were received on the event link.
- Generate an external event (edge triggered) that is sent out on the event receiver's transmit link.
- Generate a stream of internal events (level triggered) at 1 microsecond intervals that behave the same as if they were received on the event link.
- Generate a stream of external events (level triggered) at 1 microsecond intervals that are sent out on the event receiver's transmit link.

An event receiver input port configuration record will have the device type (DTYP) field set to "MRF EVR" and the output link (OUT) field will contain a string of the form:

```
#Cn Sm @INPUT_XXXXXX
```

Where:

n	=	The logical card number of the Event Receiver card
m	=	The port number of the front panel input port, and
INPUT_*XXXXXX*	=	Defines which function to perform.

Table 4-7 summarizes the function values for event receiver input ports.

Function Name	Record Type	Default Value	Description
INPUT_INT_EVENT	longout	0	The event code to be internally generated in response to an input signal.
INPUT_EXT_EVENT	longout	0	The event code to send out on the event receiver's transmit link in response to an input signal.
INPUT_INT_TRIGGER	mbbo	Disabled	Defines the trigger mechanism for the internal event code.
INPUT_EXT_TRIGGER	mbbo	Disabled	Defines the trigger mechanism for the external event code.
INPUT_EXT_DBUS_MASK	longout	0	8-bit mask of which bits on the distributed data bus will echo the value of the input signal.

Table 4-7
Event Receiver Input Port Configuration Functions

4.6.1 *Input Port Event Generation*

The signal on the input port can trigger both internal and external events. "Internal" events are treated by the event receiver as if they came from the event link. The event is applied to the active event map and behaves as defined by the relevant event configuration records (see sections 4.2 and 4.3.3). "External" events are sent out on the event receiver's transmit link, and are not processed by the event receiver itself.

Both internal and external events can be either "edge triggered" or "level triggered". "Edge triggered" events are triggered only once – on either the rising or falling edge of the input signal. "Level triggered" events are triggered once every microsecond – either when the input signal is high, or when the input signal is low.

A long output (longout) record is used to determine the event code generated by the input port. A multi-bit binary output (mbbo) record is used to determine the trigger mechanism. The type of the generated event (internal or external) is determined by the function field (INPUT_*xxx*_YYYYYY), where "*xxx*" can either be:

INT Internal event (simulates event from the receive link)
EXT External event (sent out on the transmit link)

Set Internal or External Event Code (#C*n* S*m* @INPUT_*xxx*_EVENT) (longout Record)

Sets the event code to be generated in response to the signal from input port *m*. The value should be between 0 and 255. If the value is 0 or out of range, event generation in response to the input port signal is disabled.

If the database contains no INPUT_*xxx*_EVENT record for an input port, the value will default to "0", which effectively disables event generation for that input port.

If multiple INPUT_*xxx*_EVENT records exist for the same input port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Set Internal or External Event Triggering Mechanism (#C*n* S*m* @INPUT_*xxx*_TRIGGER) (mbbo Record)

Raw Value	Engineering Value (in template files)	Description
0	Disabled	Disable event generation from input port *m* on EVR card *n*.
1	Rising Edge	Generate the event code (internal or external) defined by the INPUT_*xxx*_EVENT record on the rising edge of a TTL signal connected to input port *m* on EVR card *n*.
2	Falling Edge	Generate the event code (internal or external) defined by the INPUT_*xxx*_EVENT record on the falling edge of a TTL signal connected to input port *m* on EVR card *n*.
3	Level High	Generate the event code defined by the INPUT_*xxx*_EVENT record once every microsecond for as long as the TTL signal connected to input port *m* on EVR card *n* is "high".
4	Level Low	Generate the event code defined by the INPUT_*xxx*_EVENT record once every microsecond for as long as the TTL signal connected to input port *m* on EVR card *n* is "low".

Sets the triggering mechanism for the event code associated with the internal or external event for input port *m*. If the database contains no INPUT_*xxx*_TRIGGER record for input port *m*, event code generation will be disabled for that input port.

If multiple INPUT_*xxx*_TRIGGER records exist for the same input port, only the first record initialized will be used. The other records will be disabled and error messages will be

displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

4.6.2 *Input Port Data Bus Signal Generation*

Set External Data Bus Signal Generation Mask (#C*n* S*m* @INPUT_EXT_DBUS_MASK) (longout Record)

Determines which bits on the distributed data bus will be affected by the TTL signal on input port *m*. The value is an eight-bit mask (from 0 to 255). If a bit is set in the bit mask, that line of the distributed data bus will reflect the value of the input signal on the event receiver's transmit link. The low-order bit of the mask (hex 01) controls bit 0 on the distributed data bus. The high-order bit of the mask (hex 80) controls bit 7 on the distributed data bus.

If the database contains no INPUT_EXT_DBUS_MASK record for an input port, the value will default to "0" and the input signal will not be sent out on the distributed data bus.

If multiple INPUT_EXT_DBUS_MASK records exist for the same input port, only the first record initialized will be used. The other records will be disabled and error messages will be displayed on the IOC console terminal and in the error log. Note that the order in which records are initialized is not guaranteed.

Document Revision History

	Date	Name	Comment
Document Revision 1:	8 Sep 2009	Eric Björklund	Preliminary Draft