Micro-Research Finland Oy
Välitalontie 83 C, FI-00660 Helsinki, Finland

VME Event Receiver (VME-EVR-230)

and

VME Event Receiver with CML Outputs

(VME-EVR-230RF)

Technical Reference

Firmware Version D507

Contents

Introduction

Event Receivers (EVR) recover the clock signal from the event stream transmitted by an Event Generator and generate an event clock that is phase locked to the Event Generator event clock and thus 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 timestamps into FIFO memory to be read by a CPU module from the VME bus.

Functional Description

After recovering the event clock the Event Receiver demultiplexes the event stream to 8-bit distributed bus data and 8-bit event codes. Introduced in EVR firmware version D307, the distributed bus may be configured to share its bandwidth with data transmission.

Event Decoding

The Event Receiver provides two mapping RAMs of 8 × 16 bits. Only one of the RAMs can be active at a time, the other one may be modified from VME. The event code is applied to the address lines of the active mapping RAM. The 16-bit data programmed into a specific memory location pointed to by the event code determines what actions will be taken. In addition to the mapping RAMs each of bit 0 to 6 of the event code may generate a trigger event which is a pulse with the length of a single event cycle. There are also a few special event codes to reset prescaler outputs, control the timestamp event counter and reset the heartbeat timeout counter.

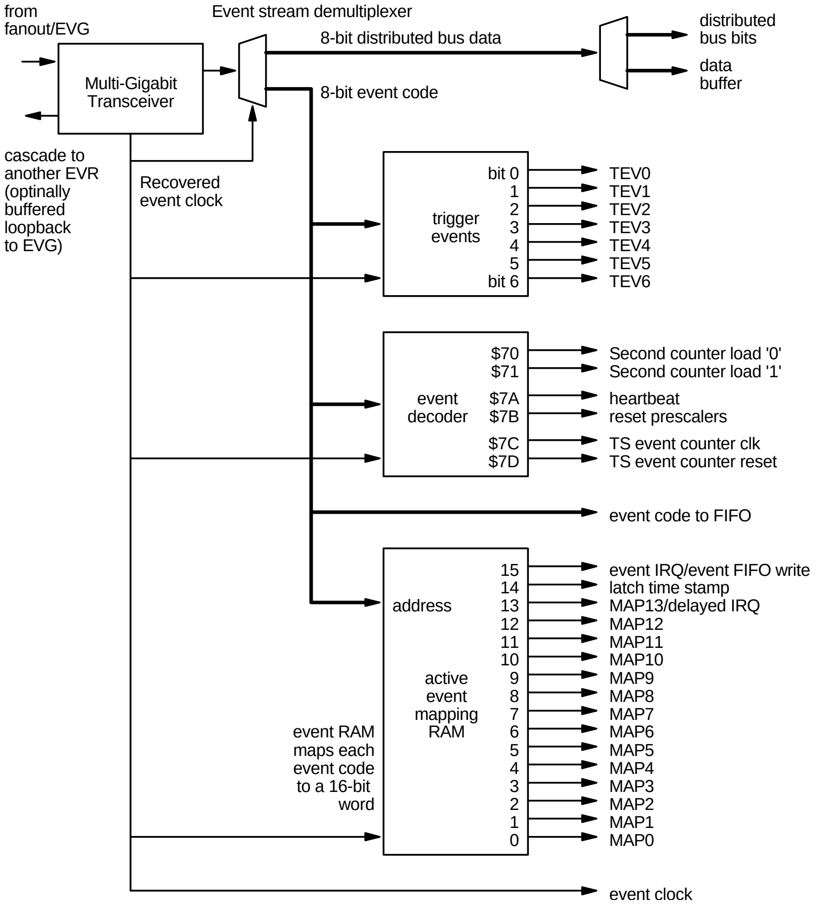

Figure 1: Event Stream Decoding

A heartbeat monitor is provided to receive heartbeat events (event code $7A). The heartbeat counter is reset upon receiving the heartbeat event code. If no heartbeat is received the counter times out (approx. 1.6 s) and a heartbeat flag is set. The Event Receiver may be programmed to generate a heartbeat interrupt.

Event FIFO and Timestamp Events

The Event System provides a global timebase to attach timestamps to collected data and performed actions. The time stamping system consists of a 32-bit timestamp event counter and a 32-bit seconds counter. The timestamp event counter either counts received timestamp counter clock events or runs freely with a clock derived from the event clock. Starting from firmware version D308 the event counter is also able to run on a clock provided on the distributed bus bit 4. The event counter clock source is determined by the prescaler value and distributed bus enable register. When the prescaler value is greater than 0 the prescaler output is used. Otherwise the clock source is defined by the distributed bus enable register. The timestamp event counter is cleared at the next event counter rising clock edge after receiving a timestamp event counter reset event. The seconds counter is updated serially by loading zeros (event code $70) and ones (event code $71) into a shift register MSB first. The seconds register is updated from the shift register at the same time the timestamp event counter is cleared.

The timestamp event counter and seconds counter contents may be latched into a timestamp latch. Latching is determined by the active event map RAM and may be enabled for any event code.

An event FIFO memory is implemented to store selected event codes with attached timing information. The 80-bit wide FIFO can hold up to 511 events. The recorded event is stored along with 32-bit seconds counter contents and 32-bit timestamp event counter contents at the time of reception. The event FIFO as well as the timestamp counter and latch are accessible from VME.

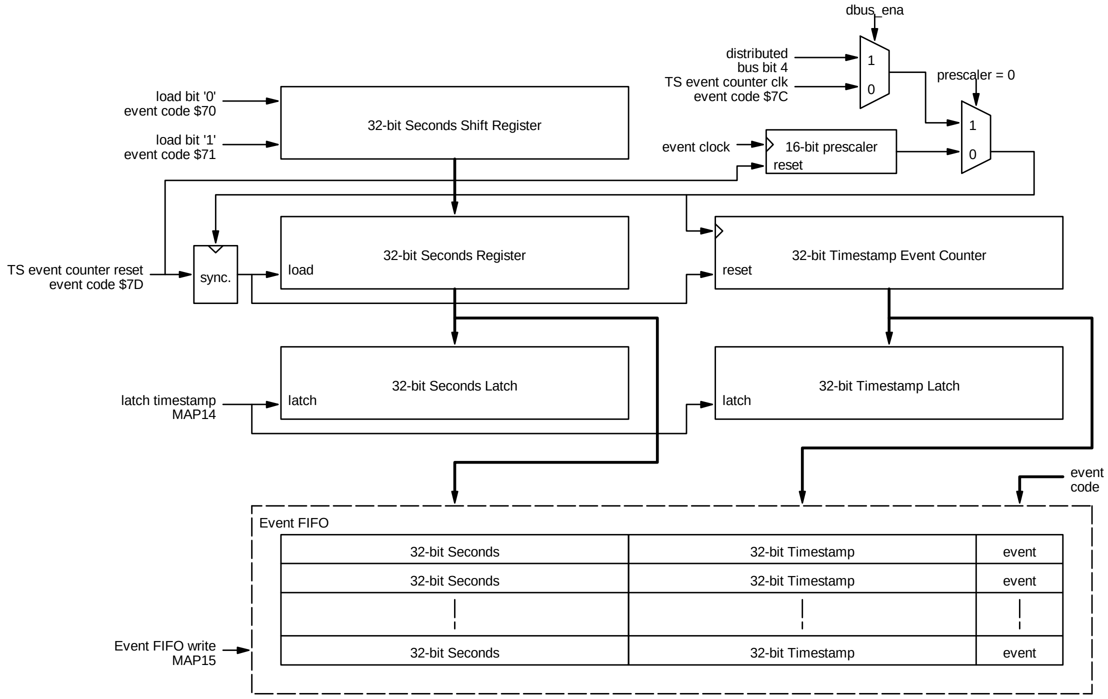

Figure 2: Event FIFO and Timestamping

Distributed Bus and Data Transmission

The distributed bus is able to carry eight simultaneous signals sampled with the event clock rate over the fibre optic transmission media. The distributed bus signals may be output on shared OTP/DBUS signals via the transition board or programmable front panel outputs.

In latest firmware versions the distributed bus bandwidth may be shared by transmission of a configurable size data buffer to up to 2 kbytes. When data transmission is enabled the distributed bus bandwidth is halved. The remaining bandwidth is reserved for transmitting data with a speed up to 62.5 Mbytes/s (event clock rate divide by two).

Hardware Outputs

The Event Receiver can generate up to 32 simultaneous outputs on the VME P2 connector. Transition modules provide TTL and optical outputs. The outputs may be selected from multiple sources. There are also a few outputs available in the front panel including two Universal I/O slots for up to four outputs.

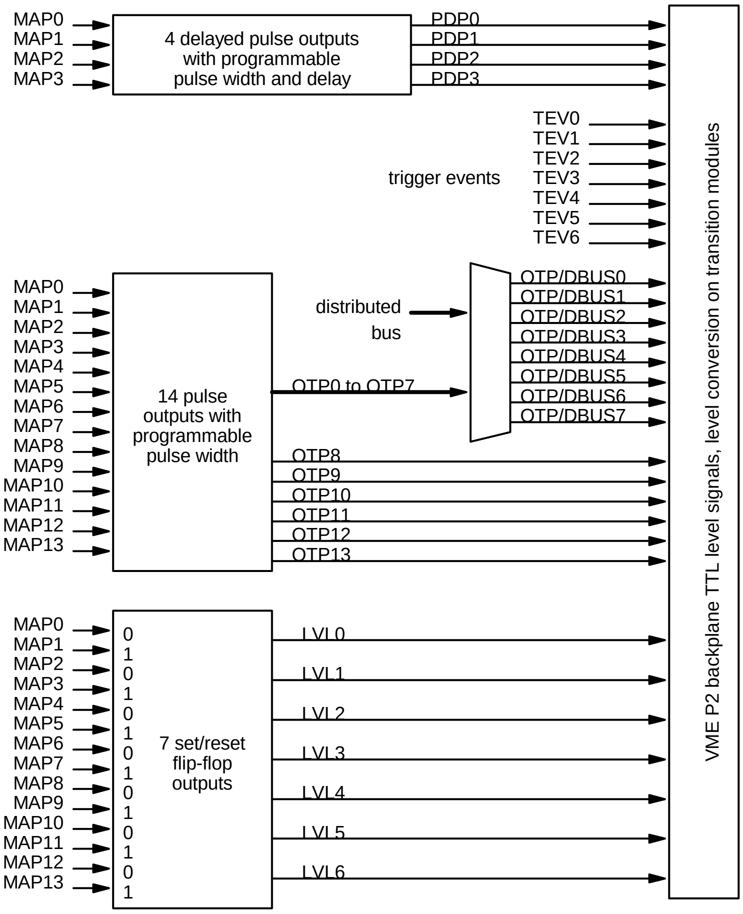

Figure 3: Event Receiver Hardware Outputs

There are fourteen pulse outputs (called OTP for historical reasons) of programmable delay, width and polarity. For each channel the pulse delay may be adjusted from 0 to 32^2-1 event clock cycles (up to 34.3 s with event clock of 125 MHz) and the pulse width may be adjusted from 1 to 65535 event clock cycles (8 ns to 524 µs with event clock of 125 MHz). Eight pulse outputs share the output pin with the distributed bus signals. The mapping (pulse/distributed bus pin) for each of the shared pins may be selected independently.

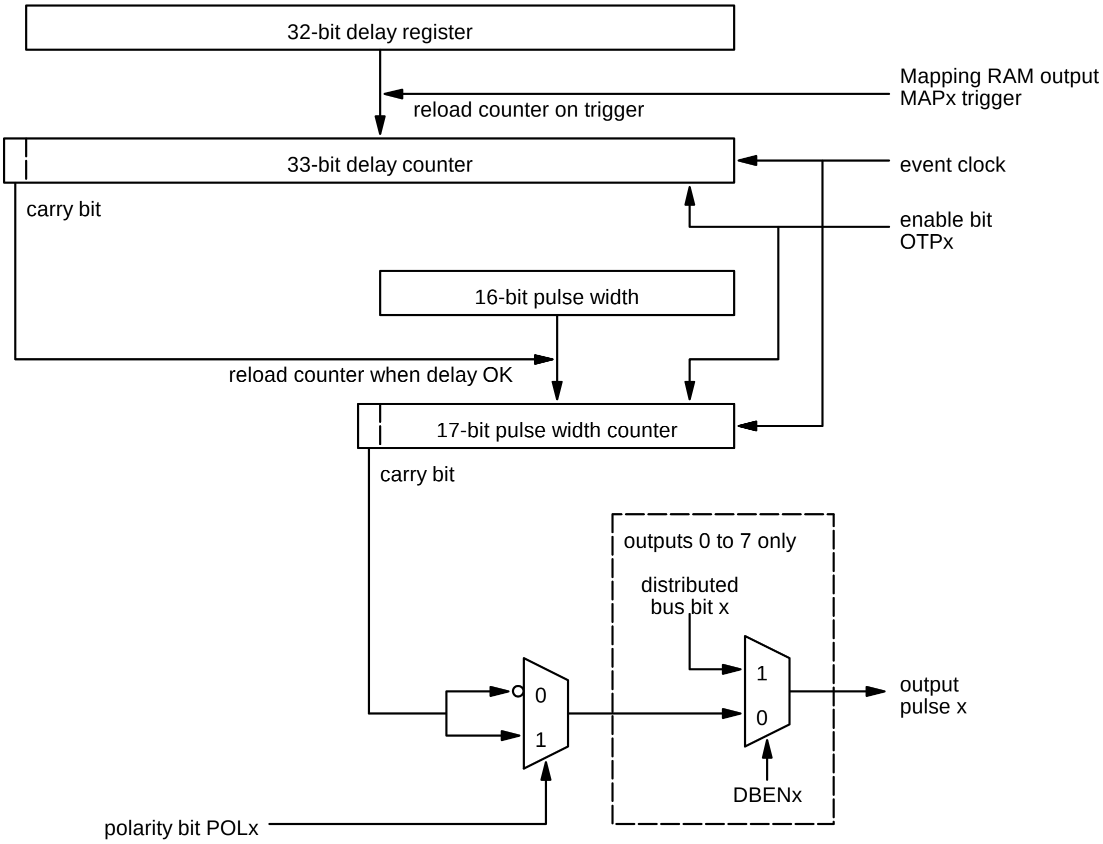

Figure 4: Programmable Width Pulse Outputs

Flip flop outputs may be programmed to change their state on desired event codes.

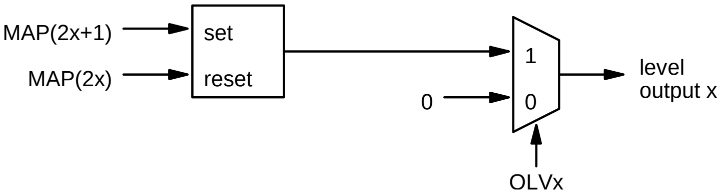

Figure 5: Level Outputs

Four extended delayed pulse outputs (called DGP for historical reasons) provide programmable delay, width and polarity like the pulse outputs. The delay and width counters share a 16-bit programmable prescaler which generates counting frequencies from event clock/65536 to the event clock rate. The delay and width counters both are 32-bits wide and thus allow maximum delays and pulse widths up to 625 h at event clock rate of 125 MHz.

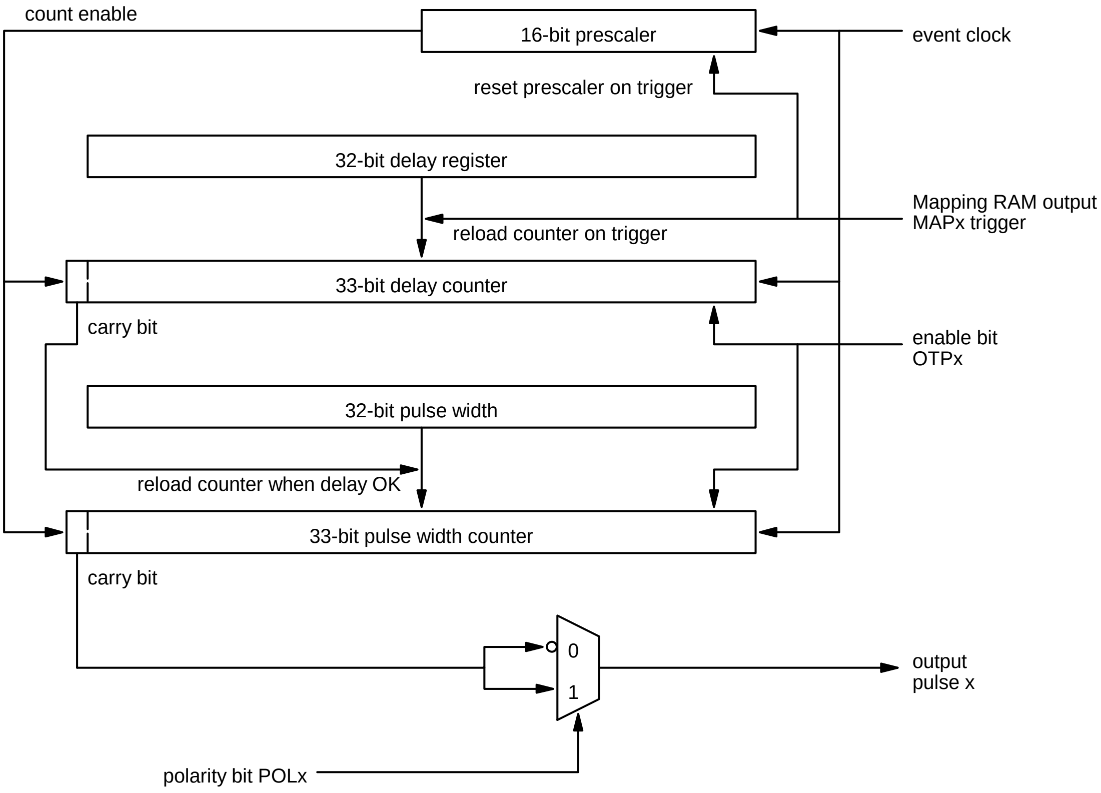

Figure 6: Programmable Delayed Pulse Outputs

Bits 0 to 6 of the received event code may be used to generate trigger event outputs. The width of a trigger event is one event clock cycle.

Prescaler Outputs

The Event Receiver provides three programmable prescaler outputs which may be mapped to front panel outputs. The frequencies are derived from the event clock. A special event code reset prescalers $7B causes the prescalers to be synchronously reset, so the frequency outputs will be in same phase across all event receivers.

Programmable Front Panel Connections

Front Panel TTL Outputs

The VME-EVR-230 provides eight programmable TTL outputs in the front panel TTL0 to TTL7 whereas the number of TTL level outputs in the VME-EVR-230RF is limited to four (TTL0 to TTL3). These outputs are capable of driving a TTL level signal into a 50 ohm ground terminated coaxial cable. The source for these signals are determined by mapping registers which allow selecting different types of pulse outputs, prescalers and distributed bus signals.
The number of TTL level outputs in the VME-EVR-230RF is limited to four (TTL0 to TTL3).

Front Panel CML Outputs (VME-EVR-230RF only)

Front Panel CML Outputs provide low jitter differential signals with special outputs. The outputs can work in different configurations: pulse mode and frequency mode.

CML Pulse Mode

The source for these outputs is selected in a similar way than the TTL outputs using mapping registers, however, the output logic monitors the state of this signal and distinguishes between state low (00), rising edge (01), high state (11) and falling edge (10). Based on the state a 20 bit pattern is sent out with a bit rate of 20 times the event clock rate.

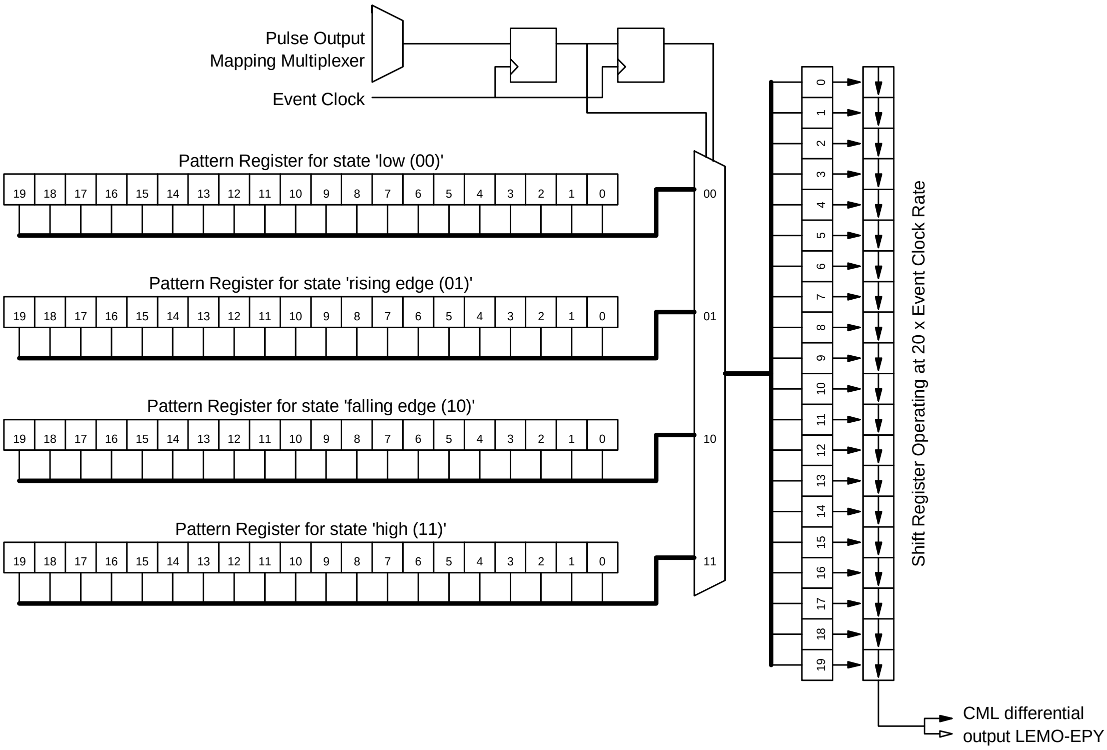
Pulse Output
Mapping Multiplexer
Event Clock
Pattern Register for state 'low (00)'
Pattern Register for state 'rising edge (01)'
Pattern Register for state 'falling edge (10)'
Pattern Register for state 'high (11)'
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
00
01
10
11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Shift Register Operating at 20 x Event Clock Rate
CML differential
output LEMO-EPY

Figure 7: Block Diagram of Programmable CML Outputs

- When the source for a CML output is low and was low one event clock cycle earlier (state low), the CML output repeats the 20 bit pattern stored in pattern_00 register.
- When the source for a CML output is high and was low one event clock cycle earlier (state rising), the CML output sends out the 20 bit pattern stored in pattern_01 register.
- When the source for a CML output is high and was high one event clock cycle earlier (state high), the CML output repeats the 20 bit pattern stored in pattern_11 register.
- When the source for a CML output is low and was high one event clock cycle earlier (state falling), the CML output sends out the 20 bit pattern stored in pattern_10 register.

For an event clock of 125 MHz the duration of one single CML output bit is 400 ps. These outputs allow for producing fine grained adjustable output pulses and clock frequencies.

CML Frequency Mode

In frequency mode one can generate clocks where the clock period can be defined in steps of 1/20th part of the event clock cycle i.e. 400 ps step with an event clock of 125 MHz. There are some limitations, however:

- Clock high time and clock low time must be ≥ 20/20th event clock period steps
- Clock high time and clock low time must be < 65536/20th event clock period steps

The clock output can be synchronized by one of the pulse generators, distributed bus signal etc. When a rising edge of the mapped output signal is detected the frequency generator takes its output value from the trigger level bit and the counter value from the trigger position register. Thus one can adjust the phase of the synchronized clock in 1/20th steps of the event clock period.

Usage example: Australian synchrotron booster clock. We have following:

- Event clock of 499.654 MHz/4
- Storage ring 360 RF buckets
- Booster 217 RF buckets
- Booster and storage ring coincidence clock on DBUS7

The CML outputs are running at a rate of 20 times the event clock or 499.654 MHz * 5, thus the booster revolution period is 217 * 5 CML bit periods. In CML frequency mode we can now set the output period (pulse high time + pulse low time) to 217 * 5 = 1085 bits. For approximately 50% duty cycle we set the pulse high time to 542 (0x21e) and the pulse low time to 543 (0x21f).

The actual register settings required are:
Write 0x00000011 to CML Control register (CMLxENA)
Write 0x021e to CML High Period Count register (CMLxHP)
Write 0x021f to CML Low Period Count register (CMLxLP)

We also need to set the trigger from DBUS7 by setting up register FPOutMapx.

To change the generated clock phase in respect to the trigger we can select the trigger polarity by bit CMLTL in the CML Control register and the trigger position also in the CML Control register.

Front Panel Universal I/O Slots

Universal I/O slots provide different types of output with exchangeable Universal I/O modules. Each module provides two outputs e.g. two TTL output, two NIM output or two optical outputs. The source for these outputs is selected with mapping registers.

Interlock Input

An interlock input is provided using a UNIV-TTLIN-IL Universal I/O module mounted in slot UNIVIO0/1. The interlock signal is applied to input UNIVIO0. The functioning of the interlock is following: when the circuit is open, the event receiver outputs for which interlock is enabled are disabled (output low). When the input circuit is closed (center pin of interlock input is grounded/shorted to connector ground) all outputs are enabled. Each transition board output and all front panel outputs may be enabled independently.

Starting with firmware version D505 an interlock latch mode is introduced that allows the input to be configured such that the “open” interlock input state is latched and the state has to be reset by software access. The state of interlock can also be shown on the previously unused front panel "FAIL" led.

Configurable Size Data Buffer

Starting from EVR firmware version D307 the reception of data over the event system link is possible. The buffer size is configured in the Event Generator to up to 2 kbytes. The Event Receiver is able to receive buffers of any size from 4 bytes to 2 kbytes in four byte (long word) increments.

Data reception is enabled by changing the distributed bus mode for data transmission (*mode* = 1 in Data Buffer Control Register). This halves the distributed bus update rate. Before a data buffer can be received the data buffer receiver has to be enabled (write *enable* = 1 in control register). This clears the checksum error flag and sets the rx_enable flag. When a data buffer has been received the rx_enable flag is cleared and rx_complete flag is set. If the received and computed checksums do not match the checksum error flag is set.

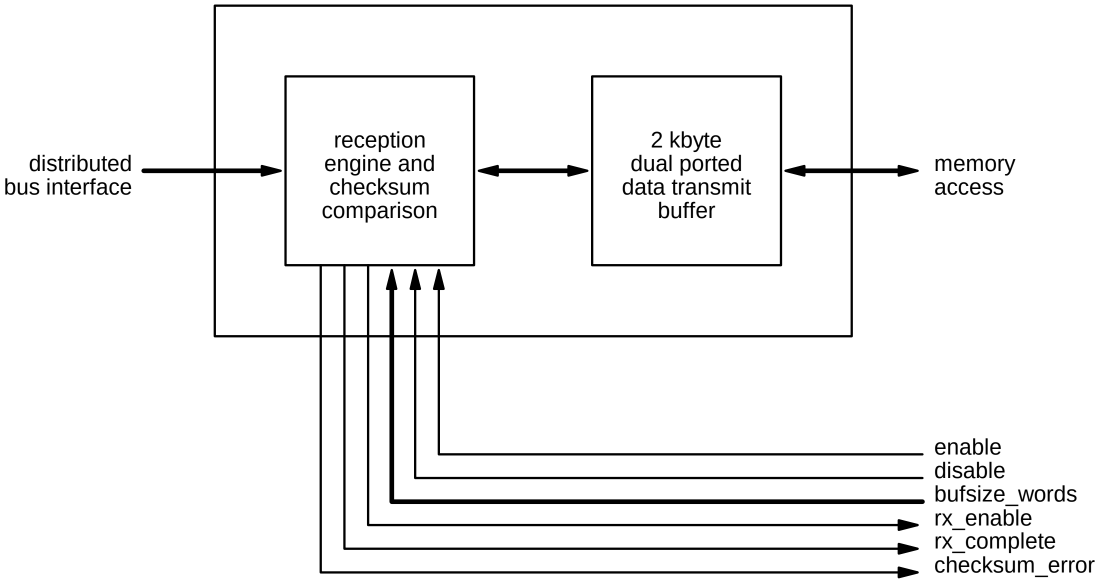

Figure 8: Data Receive Buffer

The size of the data buffer transfer can be read from the control register. An interrupt may be generated after reception of a data buffer.

Interrupt Generation

The Event Receiver has multiple interrupt sources which all have their own enable and flag bits. The following events may be programmed to generate an interrupt:

- Receiver violation: bit error or the loss of signal.
- Lost heartbeat: heartbeat monitor timeout.
- Write operation of an event to the event FIFO.
- Event FIFO is full.
- Data Buffer receive complete.

In addition to the events listed above a delayed interrupt is provided. The delayed interrupt is triggered by event map RAM bit 13. A 16-bit prescaler running with the event clock frequency and a 16-bit delay counter determine the interrupt delay.

External Event Input

An external hardware input is provided to be able to take an external pulse to generate an internal event. This event will be handled as any other received event.

Programmable Reference Clock

The event receiver requires a reference clock to be able to synchronise on the incoming event stream sent by the event generator. For flexibility a programmable reference clock is provided to allow the use of the equipment in various applications with varying frequency requirements.

Fractional Synthesiser

The clock reference for the event receiver is generated on-board the event receiver using a fractional synthesiser. A Micrel (http://www.micrel.com) SY87739L Protocol Transparent Fractional-N Synthesiser with a reference clock of 24 MHz is used. The following table lists programming bit patterns for a few frequencies.

Event Rate	Configuration Bit Pattern	Reference Output	Precision (theoretical)
499.8 MHz/4 = 124.95 MHz	0x00FE816D	124.95 MHz	0
499.654 MHz/4 = 124.9135 MHz	0x0C928166	124.907 MHz	-52 ppm
476 MHz/4 = 119 MHz	0x018741AD	119 MHz	0
106.25 MHz (fibre channel)	0x049E81AD	106.25 MHz	0
499.8 MHz/5 = 99.96 MHz	0x025B41ED	99.956 MHz	-40 ppm
50 MHz	0x009743AD	50.0 MHz	0
499.8 MHz/10 = 49.98 MHz	0x025B43AD	49.978 MHz	-40 ppm

The event receiver reference clock is required to be in ±100 ppm range of the event generator event clock.

Non-Volatile Storage for Frequency Configuration

The reference clock setting and a delay line initialisation value for event clock resynchronisation are stored in non-volatile memory inside the IP2022 microcontroller.

Connections

Front Panel Connections

The front panel of the VME-EVR-230 Event Receiver is shown in Figure 9 and VME-EVR-230RF in Figure 10: VME-EVR-230RF Event Receiver Front PanelFigure 10 respectively.

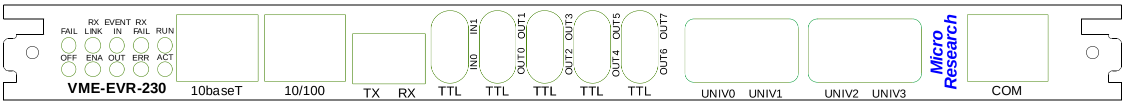

Figure 9: VME-EVR-230 Event Receiver Front Panel

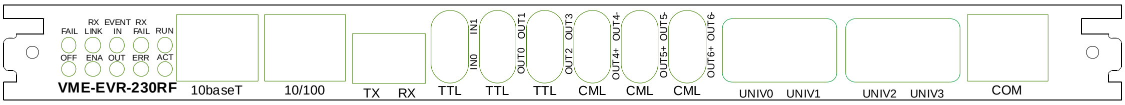

Figure 10: VME-EVR-230RF Event Receiver Front Panel

The front panel of the Event Receiver includes the following connections and status leds:

Connector / Led	Style	Level	Description
FAIL	Red Led		Module Failure/Interlock active
OFF	Blue Led		Module not Configured/Powered Down
RX LINK	Green Led		Receiver Link Signal OK
ENA	Green Led		Event Receiver Enabled
EVENT IN	Yellow Led		Incoming Event (RX)
EVENT OUT	Yellow Led		Active HW output
RX FAIL	Red Led		Receiver Violation
ERR	Red Led		SY87739L reference not locked
RUN	Green Led		Ubicom IP2022 software running
ACT	Yellow Led		Ubicom IP2022 telnet connection active
10baseT with LEDs	RJ45 green Led amber Led	10baseT	10baseT Ethernet Connection link established link activity
10/100	RJ45		(reserved)
TX	LC	optical	Transmit Optical Output (TX)
RX	LC	optical	Receiver Optical Input (RX)
TTL IN0	LEMO-EPY	TTL	External Event Input
TTL IN1	LEMO-EPY	TTL	(reserved)
TTL OUT0	LEMO-EPY	TTL	Programmable TTL Output 0
TTL OUT1	LEMO-EPY	TTL	Programmable TTL Output 1
TTL OUT2	LEMO-EPY	TTL	Programmable TTL Output 2
TTL OUT3	LEMO-EPY	TTL	Programmable TTL Output 3
TTL OUT4	LEMO-EPY	TTL	Programmable TTL Output 4[1]
TTL OUT5	LEMO-EPY	TTL	Programmable TTL Output 5
TTL OUT6	LEMO-EPY	TTL	Programmable TTL Output 6
TTL OUT7	LEMO-EPY	TTL	Programmable TTL Output 7
CML OUT4	LEMO-EPY	CML	Programmable CML Output 4[2]
CML OUT5	LEMO-EPY	CML	Programmable CML Output 5
CML OUT6	LEMO-EPY	CML	Programmable CML Output 6

[1] TTL outputs TTL4-TTL7 available on VME-EVR-230 only

[2] CML outputs available on VME-EVR-230RF only

UNIV0/1	Universal slot		Universal Output 0/1
UNIV2/3	Universal slot		Universal Output 2/3
COM	RJ45	RS232	(reserved)

VME P2 User I/O Pin Configuration

The following table lists the connections to the VME P2 User I/O Pins.

Pin	**Signal**
A1	Transition board ID0
A2	Transition board ID1
A3-A10	Ground
A11	Transition board ID2
A12	Transition board ID3
A13-A15	Ground
A16	Transition board handle switch
A17-A26	Ground
A27-A31	+5V
A32	Power control for transition board
C1	delayed pulse output 0
C2	delayed pulse output 1
C3	delayed pulse output 2
C4	delayed pulse output 3
C5	trigger event output 0
C6	trigger event output 1
C7	trigger event output 2
C8	trigger event output 3
C9	trigger event output 4
C10	trigger event output 5
C11	trigger event output 6
C12	programmable width pulse / distributed bus output 0
C13	programmable width pulse / distributed bus output 1
C14	programmable width pulse / distributed bus output 2
C15	programmable width pulse / distributed bus output 3
C16	programmable width pulse / distributed bus output 4
C17	programmable width pulse / distributed bus output 5
C18	programmable width pulse / distributed bus output 6
C19	programmable width pulse / distributed bus output 7
C20	programmable width pulse output 8
C21	programmable width pulse output 9
C22	programmable width pulse output 10
C23	programmable width pulse output 11
C24	programmable width pulse output 12
C25	programmable width pulse output 13
C26	level output 0

C27	level output 1
C28	level output 2
C29	level output 3
C30	level output 4
C31	level output 5
C32	level output 6

Programming Details

CR/CSR Support

The Event Receiver module provides CR/CSR Support as specified in the VME64x specification. The CR/CSR Base Address Register is determined after reset by the inverted state of VME64x P1 connector signal pins GA4*-GA0*. In case the parity signal GAP* does not match the GAx* pins the CR/CSR Base Address Register is loaded with the value 0xf8 which corresponds to slot number 31.

Note: the boards can be used in standard VME crates where geographical pins do not exist, in this case the user may either insert jumpers to set the geographical address or use the default setting when the board's CR/CSR base address will be set to 0xf8.

After power up or reset the board responds only to CR/CSR accesses with its geographical address. Prior to accessing Event Receiver functions the board has to be configured by accessing the boards CSR space.

The Configuration ROM (CR) contains information about manufacturer, board ID etc. to identify boards plugged in different VME slots. The following table lists the required field to locate an Event Receiver module.

CR address	Register	VME-EVR-230RF
0x27, 0x2B, 0x2F	Manufacturer's ID (IEEE OUI)	0x000EB2
0x33, 0x37, 0x3B, 0x3F	Board ID	0x455246E6

For convenience functions are provided to locate VME64x capable boards in the VME crate.

```
STATUS vmeCRFindBoard(int slot, UINT32 ieee_oui, UINT32 board_id,
                      int *p_slot);
```

To locate the first Event Receiver in the crate starting from slot 1, the function has to be called following:

```
#include "vme64x_cr.h"
int slot = 1;
int slot_evr;
vmeCRFindBoard(slot, MRF_IEEE_OUI, MRF_EVR200RF_BID, &slot_evr);
```

If this function returns `OK`, an Event Receiver board was found in slot `slot_evr`.

Event Receiver Function 0/1 Registers

The Event Receiver specific register are accessed via Function 0 and Function 1 as specified in the VME64x specification. The access size for Function 0 has been limited to 2 kbytes (0x0800) so not all EVR registers are accessible through this Function. The access size for Function 1 is 64 kbytes, so this function should not be used for A16 access because it would occupy the whole A16 range. Contrary to the VME64x specification the address/address modifier compare logic does not distinguish between privileged and non-privileged accesses and accepts both.

To enable a Function, the address decoder compare register for the Function in CSR space has to be programmed. For convenience a function to perform this is provided, too:

```
STATUS vmeCSRWriteADER(int slot, int func, UINT32 ader);
```

To configure Function 0 of a Event Receiver board in slot 3 to respond to A16 accesses at the address range 0x1800-0x1FFF the function has to be called with following values:

```
vmeCSRWriteADER(3, 0, 0x18A4);
```

ADER contents are composed of the address mask and address modifier, the above is the same as:

```
vmeCSRWriteADER(3, 0, (slot << 11) | (VME_AM_SUP_SHORT_IO << 2));
```

To get the memory mapped pointer to the configured Function 0 registers on the Event Receiver board the following VxWorks function has to be called:

```
MrfEvrStruct *pEvr;
sysBusToLocalAdrs(VME_AM_SUP_SHORT_IO, (char *) (slot << 11),
                  (void *) pEvr);
```

Note: using the data transmission capability requires more than 4 kbytes, so using function 1 with addressing mode A24 is suggested, following:

```
vmeCSRWriteADER(3, 1, (slot << 19) | (VME_AM_STD_USR_DATA << 2));
MrfEvrStruct *pEvr;
sysBusToLocalAdrs(VME_AM_STD_USR_DATA, (char *) (slot << 19),
                  (void *) pEvr);
```

Register Map

Address Offset	Register	Type	Description
0x000	Control	UINT16	Control/Status Register
0x002	MapAddr	UINT16	Mapping RAM Address Register (8 bit)
0x004	MapData	UINT16	Mapping RAM Data Register
0x006	PulseEnable	UINT16	Output Pulse Enable Register
0x008	LevelEnable	UNIT16	Level Output Enable Register
0x00A	TriggerEnable	UINT16	Trigger Pulse Enable Register
0x00C	EventCounter	UNIT16	Timestamp Event Counter (LSW)

0x00E	EventCounter	UINT16	Timestamp Event Counter (MSW)
0x010	TSLatch	UINT16	Timestamp Latch (LSW)
0x012	TSLatch	UINT16	Timestamp Latch (LSW)
0x014	EventFIFO	UINT16	Series 100 Compatible Event FIFO (LSW) bits 15 - 8 – LSB of Timestamp Counter bits 7 - 0 – Event Code
0x016	EventFIFO	UINT16	Series 100 Compatible Event FIFO (MSW) bits 23 - 8 of Timestamp Counter
0x018	PDPEnable	UINT16	Delayed Pulse Enable Register
0x01A	PDPSelect	UINT16	Delayed Pulse Select Register
0x01C	PDPDelay	UINT16	Series 100 Compatible Multiplexed Delay Register (16 LSB only)
0x01E	PDPWidth	UINT16	Series 100 Compatible Multiplexed Width Register (16 LSB only)
0x020	IrqVector	UINT16	VME Interrupt Vector Register
0x022	IrqEnable	UINT16	Interrupt Enable Register
0x024	DBusEnable	UINT16	Distributed Bus Enable Register
0x026	DBusData	UINT16	Distributed Bus Data Register
0x028	PDPPrescaler	UINT16	Multiplexed Prescaler Register
0x02A	EventPrescaler	UINT16	Event Counter Prescaler Register
0x02C	(Reserved)	UINT16	(Reserved)
0x02E	FirmwareVersion	UINT16	Event Receiver Firmware Version Register
0x030	(Reserved)	UINT32	(Reserved)
0x034	(Reserved)	UINT32	(Reserved)
0x038	(Reserved)	UINT32	(Reserved)
0x03C	InterlockCtrl	UINT16	Interlock input control register
0x03E	FPMap7	UINT16	Front Panel TTL Output 7 Map Register (VME-EVR-230 only)
0x040	FPMap0	UINT16	Front Panel TTL Output 0 Map Register
0x042	FPMap1	UINT16	Front Panel TTL Output 1 Map Register
0x044	FPMap2	UINT16	Front Panel TTL Output 2 Map Register
0x046	FPMap3	UINT16	Front Panel TTL Output 3 Map Register
0x048	FPMap4	UINT16	Front Panel TTL/CML Output 4 Map Register
0x04A	FPMap5	UINT16	Front Panel TTL/CML Output 5 Map Register
0x04C	FPMap6	UINT16	Front Panel TTL/CML Output 6 Map Register
0x04E	UsecDivider	UINT16	Divider to get from Event Clock to 1 MHz
0x050	ExtEvent	UINT16	External Event Code Register

0x052	ClockControl	UINT16	Event Clock Control Register
0x054	SecondsSR	UINT32	Seconds Shift Register
0x058	TSSec	UINT32	Timestamp Latch Seconds Register
0x05C	TBIlock	UINT32	Transition Board Interlock Enable Register
0x060	EvFIFOSec	UINT32	Event FIFO Seconds Register
0x064	EvFIFOEvCnt	UINT32	Event FIFO Event Counter Register
0x068	OutputPolarity	UINT32	Output Polarity Register (for pulse outputs)
0x06C	ExtDelay	UINT32	Multiplexed Delay Register
0x070	ExtWidth	UINT32	Multiplexed Width Register
0x074	Prescaler_0	UINT16	Prescaler 0 divider
0x076	Prescaler_1	UINT16	Prescaler 1 divider
0x078	Prescaler_2	UINT16	Prescaler 2 divider
0x07A	DataBufCtrl	UINT16	Data Buffer Control and Status Register
0x080	FracDiv	UINT32	SY87739L Fractional Divider Configuration Word
0x084	(Reserved)	UINT32	Reserved
0x088	InitPS	UINT32	Initial delay line phase shift
0x08A	(Reserved)	UINT32	Reserved
0x090	UnivMap0	UINT16	Front Panel Univ Output 0 Map Register
0x092	UnivMap1	UINT16	Front Panel Univ Output 1 Map Register
0x094	UnivMap2	UINT16	Front Panel Univ Output 2 Map Register
0x096	UnivMap3	UINT16	Front Panel Univ Output 3 Map Register
0x098	UnivGPIO	UINT32	Front Panel Universal Slots GPIO Register
0x0A0	CML4Pat00	UINT32	20 bit output pattern for state low
0x0A4	CML4Pat01	UINT32	20 bit output pattern for state rising edge
0x0A8	CML4Pat10	UINT32	20 bit output pattern for state falling edge
0x0AC	CML4Pat11	UINT32	20 bit output pattern for state high
0x0B0	CML4Ena	UINT32	CML 4 Output Control Register
0x0C0	CML5Pat00	UINT32	20 bit output pattern for state low
0x0C4	CML5Pat01	UINT32	20 bit output pattern for state rising edge
0x0C8	CML5Pat10	UINT32	20 bit output pattern for state falling edge
0x0CC	CML5Pat11	UINT32	20 bit output pattern for state high
0x0D0	CML5Ena	UINT32	CML 5 Output Control Register
0x0E0	CML6Pat00	UINT32	20 bit output pattern for state low
0x0E4	CML6Pat01	UINT32	20 bit output pattern for state rising edge
0x0E8	CML6Pat10	UINT32	20 bit output pattern for state falling edge
0x0EC	CML6Pat11	UINT32	20 bit output pattern for state high
0x0F0	CML6Ena	UINT32	CML 6 Output Control Register
0x0F4 – 0x7FF	(Reserved)	UINT32	Reserved
0x800 – 0xFFF	DataBuf		Data Buffer Receive Memory

Control and Status Register

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x000	EVREN	IRQEN	RSTS	HRTBT	IRQFL	LTS	MAPEN	MAPRS

Bit	Function
EVREN	Event Receiver Master enable.
IRQEN	VME irq enable. When 0 all interrupts are disabled.
RSTS	Write 1 to reset timestamp event counter and timestamp latch.
HRTBT	Lost heartbeat flag. Write 1 to reset.
IRQFL	Event FIFO interrupt flag. Write 1 to reset.
LTS	Write 1 to latch timestamp from timestamp event counter to timestamp latch.
MAPEN	Event mapping RAM enable.
MAPRS	Mapping RAM select bit for event decoding. 0 - mapping RAM 1, 1 - mapping RAM 2.

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x001	NFRAM	VMERS	AUTOI	RSADR DIRQ	RSFIFO	FF	FNE RSDIRQ	RXVIO

Bit	Function
NFRAM	Write 1 to null fill (clear) mapping RAM selected by VMERS. This bit changed to 0 when the selected RAM has been cleared.
VMERS	Mapping RAM select bit for VME access. 0 - mapping RAM 1, 1 - mapping RAM 2.
AUTOI	Enable mapping RAM auto increment mode. When set the address is automatically incremented by one upon every access from VME.
RSADR	Write 1 to reset mapping RAM address register (write only).
DIRQ	Delayed interrupt flag.
RSFIFO	Write 1 to clear event FIFO.
FF	Event FIFO full flag. Write 1 to reset flag.
FNE	FIFO not empty flag. Indicated whether there are event in event FIFO.

Mapping RAM address register

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x003	Mapping RAM common address register							

Mapping RAM data register

address	bit 15	bit 0
0x004	Mapping RAM common data register	

Output pulse enable register

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x006			OTP13	OTP12	OTP11	OTP10	OTP9	OTP8

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x007	OTP7	OTP6	OTP5	OTP4	OTP3	OTP2	OTP1	OTP0

Bit	Function
OTPx	Enable programmable width output pulse x.

Output level enable register

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x009		OLV6	OLV5	OLV4	OLV3	OLV2	OLV1	OLV0

Bit	Function
OLVx	Enable level output pulse x.

Trigger event enable register

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x00B		TEV6	TEV5	TEV4	TEV3	TEV2	TEV1	TEV0

Bit	Function
TEVx	Enable trigger event output pulse x.

Timestamp event counter register

address	bit 15 … bit 0
0x00C	Timestamp event counter (LSW, read only)

address	bit 15 … bit 0
0x00E	Timestamp event counter (MSW, read only)

Timestamp event latch register

address	bit 15 … bit 0
0x010	Timestamp event latch (LSW, read only)

address	bit 15 … bit 0
0x012	Timestamp event latch (MSW, read only)

Event FIFO data register

address	bit 15 … bit 8
0x014	Event FIFO data register, bits 7 – 0 of timestamp event counter

address	bit 7 … bit 0
0x015	Event FIFO data register, event code

address	bit 15 … bit 0
0x016	Event FIFO data register, bits 23 – 8 of timestamp event counter

Programmable delayed pulse output enable register

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x019	POL3	POL2	POL1	POL0	PDP3	PDP2	PDP1	PDP0

Bit	**Function**
POLx	Delayed pulse output x polarity: 0 - active high, 1 - active low.
PDPx	Delayed pulse output x enable.

Programmable pulse / delay select register

address	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
0x01b				DSEL4	DSEL3	DSEL2	DSEL1	DSEL0

DSEL	**Register selected**
00000	Programmable delayed pulse 0
00001	Programmable delayed pulse 1
00010	Programmable delayed pulse 2
00011	Programmable delayed pulse 3
00100	Delayed interrupt
10000	Programmable width pulse 0
10001	Programmable width pulse 1
10010	Programmable width pulse 2
10011	Programmable width pulse 3
10100	Programmable width pulse 4
10101	Programmable width pulse 5
10110	Programmable width pulse 6
10111	Programmable width pulse 7
11000	Programmable width pulse 8
11001	Programmable width pulse 9
11010	Programmable width pulse 10
11011	Programmable width pulse 11
11100	Programmable width pulse 12
11101	Programmable width pulse 13

Programmable Delayed Pulse / Delayed Interrupt Delay Register

address	**bit 15**	**bit 0**
0x01C	Programmable Delayed Pulse / Delayed Interrupt Delay Register	

Programmable Width Pulse / Delayed Pulse Width Register

address	**bit 15**	**bit 0**
0x01E	Programmable Width Pulse / Delayed Pulse Width Register	

VME Interrupt Vector Register

address	**bit 7**	**bit 0**
0x021	VME interrupt vector register	

Interrupt configuration register

address	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
0x023			IEDBUF	IEDIRQ	IEEVT	IEHRT	IEFF	IEVIO

Bit	**Function**
IEDBUF	Data Buffer interrupt enable.

IEDIRQ	Delayed interrupt enable.
IEEVT	Event interrupt enable.
IEHRT	Lost heartbeat interrupt enable.
IEFF	Event FIFO full interrupt enable.
IEVIO	Receiver violation interrupt enable.

Distributed bus enable register

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x024				DBEVC				

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x025	DBEN7	DBEN6	DBEN5	DBEN4	DBEN3	DBEN2	DBEN1	DBEN0

Bit	Function
DBEVC	Event counter clock source when prescaler = 0 0 – Event counter is counting increment timestamp event codes 0x7c 1 – Event counter is running on signal distributed on DBUS bit 4
DBENx	OTPx output select: 0 - programmable width pulse x, 1 - distributed bus bit x.

Distributed bus data register

address	bit 7 … bit 0
0x027	Distributed bus data register (read only)

Programmable Delayed Pulse / Delayed Interrupt Prescaler Register

address	bit 15 … bit 0
0x028	Programmable Delayed Pulse / Delayed Interrupt Prescaler Register

Timestamp Event Counter Clock Prescaler Register

address	bit 15 … bit 0
0x02A	Timestamp Event Counter Clock Prescaler Register

Front Panel Output Multiplexer Registers

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x03E		FPIL7	FP7SEL5	FP7SEL4	FP7SEL3	FP7SEL2	FP7SEL1	FP7SEL0
0x040		FPIL0	FP0SEL5	FP0SEL4	FP0SEL3	FP0SEL2	FP0SEL1	FP0SEL0
0x042		FPIL1	FP1SEL5	FP1SEL4	FP1SEL3	FP1SEL2	FP1SEL1	FP1SEL0
0x044		FPIL2	FP2SEL5	FP2SEL4	FP2SEL3	FP2SEL2	FP2SEL1	FP2SEL0
0x046		FPIL3	FP3SEL5	FP3SEL4	FP3SEL3	FP3SEL2	FP3SEL1	FP3SEL0
0x048		FPIL4	FP4SEL5	FP4SEL4	FP4SEL3	FP4SEL2	FP4SEL1	FP4SEL0
0x04A		FPIL5	FP5SEL5	FP5SEL4	FP5SEL3	FP5SEL2	FP5SEL1	FP5SEL0
0x04C		FPIL6	FP6SEL5	FP6SEL4	FP6SEL3	FP6SEL2	FP6SEL1	FP6SEL0
0x090		UNIL0	UN0SEL5	UN0SEL4	UN0SEL3	UN0SEL2	UN0SEL1	UN0SEL0
0x092		UNIL1	UN1SEL5	UN1SEL4	UN1SEL3	UN1SEL2	UN1SEL1	UN1SEL0

0x094		UNIL2	UN2SEL5	UN2SEL4	UN2SEL3	UN2SEL2	UN2SEL1	UN2SEL0
0x096		UNIL3	UN3SEL5	UN3SEL4	UN3SEL3	UN3SEL2	UN3SEL1	UN3SEL0

FPxSEL, UNxSEL *)	**Signal selected**
000000	Programmable Delayed Pulse Output 0
000001	Programmable Delayed Pulse Output 1
000010	Programmable Delayed Pulse Output 2
000011	Programmable Delayed Pulse Output 3
000100	Trigger Event Output 0
000101	Trigger Event Output 1
000110	Trigger Event Output 2
000111	Trigger Event Output 3
001000	Trigger Event Output 4
001001	Trigger Event Output 5
001010	Trigger Event Output 6
001011	Programmable Width Pulse Output 0
001100	Programmable Width Pulse Output 1
001101	Programmable Width Pulse Output 2
001110	Programmable Width Pulse Output 3
001111	Programmable Width Pulse Output 4
010000	Programmable Width Pulse Output 5
010001	Programmable Width Pulse Output 6
010010	Programmable Width Pulse Output 7
010011	Programmable Width Pulse Output 8
010100	Programmable Width Pulse Output 9
010101	Programmable Width Pulse Output 10
010110	Programmable Width Pulse Output 11
010111	Programmable Width Pulse Output 12
011000	Programmable Width Pulse Output 13
011001	Level Output 0
011010	Level Output 1
011011	Level Output 2
011100	Level Output 3
011101	Level Output 4
011110	Level Output 5
011111	Level Output 6
100000	Distributed Bus Data 0
100001	Distributed Bus Data 1
100010	Distributed Bus Data 2
100011	Distributed Bus Data 3
100100	Distributed Bus Data 4
100101	Distributed Bus Data 5
100110	Distributed Bus Data 6
100111	Distributed Bus Data 7
101000	Prescaler 0
101001	Prescaler 1
101010	Prescaler 2
111110	Output tied high

111111	Output tied low

*) FP0 to FP7 are front panel TTL outputs 0 to 7, FP4 to FP6 are front panel CML outputs 4 to 6 on VME-EVR-230RF, UN0 to UN3 are Universal Outputs 0 to 3

Bit	**Function**
FPILx	Front Panel/Universal I/O Interlock enable for pin x
UNILx	0 – normal operation, interlocking disabled
	1 – interlocking enabled, output x pulled low when interlock input is open (input pin high)

External Event Code Register

address	**bit 7** … **bit 0**
0x051	External Event Code Register

Clock Control Register

address	**bit 15**	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
0x052	RECDCM RUN	RECDCM INITDONE	RECDCM PSDONE	EVDCM STOPPED	EVDCM LOCKED	EVDCM PSDONE	CGLOCK	RECDCM PSDEC

address	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
0x053	RECDCM PSINC	RECDCM RES	EVDCM PSDEC	EVDCM PSINC	EVDCM SRUN	EVDCM SRES	EVDCM RES	EVCLKSEL

Bit	**Function**
CGLOCK	Micrel SY87739L locked (read-only)
EVCLKSEL	Event clock source select
	0 – use externally resynchronised clock (default)
	1 – use internal recovered clock

Seconds Shift Register

address	**bit 31** … **bit 0**
0x054	Seconds Shift Register (read-only)

Timestamp Latch Seconds Register

address	**bit 31** … **bit 0**
0x058	Timestamp Latch Seconds Register (read-only)

Transition Board Interlock Register

address	**bit 31**	**bit 30**	**bit 29**	**bit 28**	**bit 27**	**bit 26**	**bit 25**	**bit 24**
0x05C	TBIL31	TBIL30	TBIL29	TBIL28	TBIL27	TBIL26	TBIL25	TBIL24

address	**bit 23**	**bit 22**	**bit 21**	**bit 20**	**bit 19**	**bit 18**	**bit 17**	**bit 16**
0x05D	TBIL23	TBIL22	TBIL21	TBIL20	TBIL19	TBIL18	TBIL17	TBIL16

address	**bit 15**	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
0x05E	TBIL15	TBIL14	TBIL13	TBIL12	TBIL11	TBIL10	TBIL9	TBIL8

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x05F	TBIL7	TBIL6	TBIL5	TBIL4	TBIL3	TBIL2	TBIL1	TBIL0

Bit	Function
TBILx	Transition Board Interlock enable for pin x 0 – normal operation, interlocking disabled 1 – interlocking enabled, transition board output x pulled low when interlock input is open (input pin high)

Event FIFO Extended Timestamp Registers

address	bit 31 … bit 0
0x060	Event FIFO Seconds Register (read-only)
0x064	Event FIFO Event Counter Register (read-only)

Polarity Register

address	bit 31	bit 30	bit 29	bit 28	bit 27	bit 26	bit 25	bit 24
0x068								POL24

address	bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16
0x069	POL23	POL22	POL21	POL20	POL19	POL18	POL17	POL16

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x06A	POL15	POL14	POL13	POL12	POL11			

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x06B					POL3	POL2	POL1	POL0

Bit	Function
POL0-3	Programmable Delayed Pulse Output Polarity 0 – normal polarity (pulse active high) 1 – inverted polarity
POL11	Output Pulse 0 (OTP0) Polarity 0 – normal polarity (pulse active high) 1 – inverted polarity
…	…
POL24	Output Pulse 13 (OTP13) Polarity 0 – normal polarity (pulse active high) 1 – inverted polarity

Data Buffer Control and Status Register

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x07A	DBRX/ DBENA	DBRDY/ DBDIS	DBCS	DBEN	RXSIZE(11:8)			

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

0x07B	RXSIZE(7:0)

Bit	Function
DBRX	Data Buffer Receiving (read-only)
DBENA	Set-up for Single Reception (write ‘1’ to set-up)
DBRDY	Data Buffer Transmit Complete / Interrupt Flag
DBDIS	Stop Reception (write ‘1’ to stop/disable)
DBCS	Data Buffer Checksum Error (read-only) Flag is cleared by writing ‘1’ to DBRX or DBRDY or disabling data buffer
DBEN	Data Buffer Enable Data Buffer Mode ‘0’ – Distributed bus not shared with data transmission, full speed distributed bus ‘1’ – Distributed bus shared with data transmission, half speed distributed bus
RXSIZE	Data Buffer Received Buffer Size (read-only)

SY87739L Fractional Divider Configuration Word

address	bit 31 … bit 0
0x080	SY87739L Fractional Divider Configuration Word

Configuration Word	Frequency with 24 MHz reference oscillator
0x0C928166	124.907 MHz
0x009743AD	50 MHz
0xC25B43AD	49.978 MHz

Front Panel Universal I/O GPIO Register

address	bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16
0x099	GPDIR7	GPDIR6	GPDIR5	GPDIR4	GPDIR3	GPDIR2	GPDIR1	GPDIR0

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x09A	GPOUT7	GPOUT6	GPOUT5	GPOUT4	GPOUT3	GPOUT2	GPOUT1	GPOUT0

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x09B	GPIN7	GPIN6	GPIN5	GPIN4	GPIN3	GPIN2	GPIN1	GPIN0

Bit	Function
GPDIRx	Front Panel Universal I/O GPIOx signal direction: 0 – input (from Universal I/O module) 1 – output (to Universal I/O module)
GPOUTx	Front Panel Universal I/O GPIOx signal output state (when output enabled): 0 – output low 1 – output high
GPINx	Front Panel Universal I/O GPIOx signal state: 0 – pin low 1 – pin high

GPIO0	Universal I/O slot 0/1 header pin 2 (Module pin 20)
GPIO1	Universal I/O slot 0/1 header pin 4 (Module pin 18)
GPIO2	Universal I/O slot 0/1 header pin 18 (Module pin 4)
GPIO3	Universal I/O slot 0/1 header pin 20 (Module pin 2)
GPIO4	Universal I/O slot 2/3 header pin 2 (Module pin 20)
GPIO5	Universal I/O slot 2/3 header pin 4 (Module pin 18)
GPIO6	Universal I/O slot 2/3 header pin 18 (Module pin 4)
GPIO7	Universal I/O slot 2/3 header pin 20 (Module pin 2)

CML Output Pattern Registers (CMLxPatxx)

bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16
				19 MSB	18	17	16

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
15	14	13	12	11	10	9	8

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
7	6	5	4	3	2	1	0 LSB

Bit 19 MSB is sent out first, LSB last

CML Output Control Register

Address	bit 31	bit 16
	Frequency mode trigger position	

Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
		CMLTL	CMLMD			CMLRES	CMLPWD	CMLENA

CMLTL CML Frequency mode trigger level
CMLMD CML Mode Select:
00 = classic mode
01 = frequency mode
10, 11 = undefined
CMLRES CML Reset
1 = reset CML output (default on EVR power up)
0 = normal operation
CMLPWD CML Power Down
1 = CML outputs powered down (default on EVR power up)
0 = normal operation
CMLENA CML Enable
0 = CML output disabled (default on EVR power up)
1 = CML output enabled

Network Interface

A 10baseT network interface is provided to upgrade the FPGA firmware and set up boot options. It is also possible to control the module over the network interface.

Assigning an IP Address to the Module

By default the modules uses DHCP (dynamic host configuration protocol) to acquire an IP address. In case a lease cannot be acquired the IP address set randomly in the 169.254.x.x subnet. The board can be programmed to use a static address instead if DHCP is not available.

The module can be located looking at the lease log of the DHCP server or using a Windows tool called Locator.exe.

Using Telnet to Configure Module

To connect to the configuration utility of the module issue the following command:

```
telnet 192.168.1.32 23
```

The latter parameter is the telnet port number and is required in Linux to prevent negotiation of telnet parameters which the telnet server of the module is not capable of.

The telnet server responds to the following commands:

Command	Description
b	Show/change boot parameters, IP address etc.
d	Dump 16 bytes of memory
h / ?	Show Help
m <address> [<data>]	Read/Write FPGA CR/CSR, Function 0
r	Reset Board
s	Save boot configuration & dynamic configuration values into non-volatile memory
t	Tune delay line for event clock recovery
+	Manually increase delay line delay $^{*)}$
-	Manually decrease delay line delay $^{*)}$
u	Update IP2022 software
q	Quit Telnet

*) This option has been added with IP2022 software version 060309 for VME-EVR-230RF (displayed in output from help command)

Boot Configuration (command b)

Command b displays the current boot configuration parameters of the module. The parameter may be changed by giving a new parameter value. The following parameters are displayed:

Parameter	Description
Use DHCP	0 = use static IP address, 1 = use DHCP to acquire address, net mask etc.
IP address	IP address of module
Subnet mask	Subnet mask of module

Default GW	Default gateway
FPGA mode	FPGA configuration mode 0 – FPGA is not configured after power up 1 – FPGA configured from internal Flash memory 2 – FPGA is configured from FTP server
FTP server	FTP server IP address where configuration bit file resides
Username	FTP server username
Password	FTP server password
FTP Filename	FTP server configuration file name
Flash Filename	Configuration file name on internal flash
µs divider	Integer divider to get from event clock to 1MHz, e.g. 125 for 124.9135 MHz
Fractional divider configuration word	Micrel SY87739UMI fractional divider configuration word to set refenrence for event clock

Note that after changing parameters the parameters have to be saved to internal flash by issuing the Save boot configuration (s) command. The changes are applied only after resetting the module using the reset command or hardware reset/power sequencing.

Memory dump (command d)

This command dumps 16 bytes of memory starting at the given address, if the address is omitted the previous address value is increased by 16 bytes.

The most significant byte of the address determines the function of the access:

Address	**Function**
0x78000000	CR/CSR space access
0x7a000000	EVR registers access

To dump the start of the EVR register map issue the ‘d’ command from the telnet prompt:

```
VME-EVR-230RF -> d 7a000000 ↵
Addr 7a000000:  1005 0001 0000 0000 0000 0000 0000 0000
VME-EVR-230RF -> d ↵
Addr 7a000010:  0000 0000 0000 0000 0000 0000 0000 0000
VME-EVR-230RF ->
```

Memory modify (commands d and m)

The access size is always a short word i.e. two bytes.

To check the status register from the telnet prompt:

```
VME-EVR-230RF -> m 7a000000 ↵
Addr 7a000000 data 1005
VME-EVR-230RF ->
```

To clear the violation flag issue:

```
VME-EVR-230RF -> m 7a000000 1005 ↵
Addr 7a000000 data 0000
VME-EVR-230RF ->
```

Tuning Delay Line (command t)

The VME Event Receiver VME-EVR-230RF has to be configured for proper event clock rate and the on-board delay line has to be tuned for the operating conditions. Before setting up the board make sure you have an Event Generator with the correct event clock connected to the Event Receiver. Also, let the EVR reach operating temperature (power on for 10 minutes in crate). See previous section for setting up the event clock rate.

To start tuning issue command ‘t’ from the telnet prompt:

```
VME-EVR-230RF -> t ↵
Starting tuning...
Adjusted sampling phase to 75
Initial DCM phase -85
Fine tuned sampling phase to 78
Final DCM phase -73.
VME-EVR-230RF ->
```

After tuning the tuned values have to be stored in non-volatile memory:

```
VME-EVR-230RF -> s ↵
Confirm save (yes/no) ? yes ↵
Configuration saved.
VME-EVR-230RF ->
```

Upgrading IP2022 Microprocessor Software (command u)

To upgrade the Ubicom IP2022 microprocessor software download the upgrade image containing the upgrade to the module using TFTP:

Linux

In Linux use e.g. interactive tftp:

```
$ tftp 192.168.1.32
tftp> bin
tftp> put upgrade.bin /fw
tftp> quit
```

Windows

In Windows command prompt issue the following command:

```
C:\> tftp –i 192.168.1.32 PUT upgrade.bin /fw
```

When the upgrade image has been downloaded and verified, enter at the telnet prompt following:

```
VME-EVR-230 -> u ↵
Really update firmware (yes/no) ? yes ↵
Self programming triggered.
```

The Event Receiver starts programming the new software and restarts.

Upgrading FPGA Configuration File

When the FPGA configuration file resides in internal flash memory a new file system image has to be downloaded to the module. This is done using TFTP protocol:

Linux

In Linux use e.g. interactive tftp:

```
$ tftp 192.168.1.32
tftp> bin
tftp> put filesystem.bin /
tftp> quit
```

Windows

In Windows command prompt issue the following command:

```
C:\> tftp –i 192.168.1.32 PUT filesystem.bin /
```

Now the FPGA configuration file has been upgraded and the new configuration is loaded after next reset/power sequencing.

Note! Due to the UDP protocol it is recommended to verify (read back and compare) the filesystem image before restarting the module. This is done following:

Linux

In Linux use e.g. interactive tftp:

```
$ tftp 192.168.1.32
tftp> bin
tftp> get / verify.bin
tftp> quit
$ diff filesystem.bin verify.bin
$
```

If files differ you should get following message:

```
Binary files filesystem.bin and verify.bin differ
```

Windows

In Windows command prompt issue the following command:

```
C:\> tftp –i 192.168.1.32 GET / verify.bin
C:\> fc /b filesystem.bin verify.bin
Comparing files filesystem.bin and verify.bin
FC: no differences encountered
```

UDP Remote Programming Protocol

The VME-EVR can be remotely programmed using the 10baseT Ethernet interface with a protocol over UDP (User Datagram Protocol) which runs on top of IP (Internet Protocol). The default port for remote programming is UDP port 2000. The UDP commands are built upon the following structure:

<table>
<tr><td>access_type (1 byte)</td><td>status (1 byte)</td><td>data (2 bytes)</td></tr>
<tr><td colspan="3">address (4 bytes)</td></tr>
<tr><td colspan="3">ref (4 bytes)</td></tr>
</table>

The first field defines the access type:

access_type	Description
0x01	Read Register from module
0x02	Write and Read back Register from module

The second field tells the status of the access:

Status	Description
0	Command OK
-1	Bus ERROR (Invalid read/write address)
-2	Timeout (FPGA did not respond)
-3	Invalid command

The access size is always a short word i.e. two bytes. The most significant byte of the address determines the function of the access:

Address	**Function**
0x78000000	CR/CSR space access
0x7a000000	EVR registers access

Read Access (Type 0x01)

The host sends a UDP packet to port 2000 of the VME-EVR with the following contents:

<table>
<tr><td>access_type (1 byte)
0x01</td><td>status (1 byte)
0x00</td><td>data (2 bytes)
0x0000</td></tr>
<tr><td colspan="3">address (4 bytes)
0x7a000000 (Control and Status register Function 0 address)</td></tr>
<tr><td colspan="3">ref (4 bytes)
0x00000000</td></tr>
</table>

If the read access is successful the VME-EVR replies to the same host and port the message came from with the following packet:

<table>
<tr><td>access_type (1 byte)
0x01</td><td>status (1 byte)
0x00</td><td>data (2 bytes)
0x0032</td></tr>
<tr><td colspan="3">address (4 bytes)
0x7a000000 (Control and Status register Function 0 address)</td></tr>
<tr><td colspan="3">ref (4 bytes)
0x00000000</td></tr>
</table>

Write Access (Type 0x02)

The host sends a UDP packet to port 2000 of the VME-EVR with the following contents:

<table>
<tr><td>access_type (1 byte)
0x02</td><td>status (1 byte)
0x00</td><td>data (2 bytes)
0x0001</td></tr>
<tr><td colspan="3">address (4 bytes)
0x7a000002 (Mapping RAM Address register Function 0 address)</td></tr>
<tr><td colspan="3">ref (4 bytes)
0x00000000</td></tr>
</table>

If the write access is successful the VME-EVR replies to the same host and port the message came from with the following packet:

<table>
<tr><td>access_type (1 byte)
0x02</td><td>status (1 byte)
0x00</td><td>data (2 bytes)
0x0001</td></tr>
<tr><td colspan="3">address (4 bytes)
0x80000000 (Mapping RAM Address register Function 0 address)</td></tr>
<tr><td colspan="3">ref (4 bytes)
0x00000000</td></tr>
</table>

Notice that in the reply message the data returned really is the data read from the address specified in the address field so one can verify that the data really was written ok.