Datasheet

VMESCmodule

VME System Controller Module

Version 1.3.2

INICORE INC.
5600 Mowry School Road
Suite 180
Newark, CA 94560
t: 510 445 1529 *f:* 510 656 0995 *e:* info@inicore.com
www.inicore.com

Table of Contents

Figure Index

Document History

The following table gives an overview of the document history and can help in the determination if the latest version of this document has been used.

Version	Date	Comment
1.0.0	10/8/10	Initial release
1.1.0	10/19/10	- Rearranged the interrupt vector to have the different interrupt flags in prioritized order - Added automatic interrupt status handler - Added DMA error interrupt - Modified interrupter implementation to support software and user side hardware interrupts
1.1.1	11/24/10	- Corrected lbus_slv_addr size - Corrected description of IS_VBERR generation - Added time-out definition to VMEbus arbiter - Refined description of ACFAIL_EBL register - Replaced VBB with VMESCmodule
1.2.0	12/6/10	- Added RMW lock interface on user side - Added master direct write feature (coupled transfer from local bus to VME bus) supporting D08, D16, and D32 mode for read, write, and read-modify-write - Corrected mnemonic for D08 data type - Corrected lbus_slv address width
1.2.1	4/8/11	- Figure 4: user side memory address is 31..0 and not 32..0 - Corrected DEV_CTRL VME bus address. - Modified byte endian decoding table on page 22 - Corrected bit mapping for D16 cycle of VME_IRQn_STAT register
1.2.2	2/15/12	- Corrected A32 code in SLVW_AM_AS table - Corrected slave window size and offset in example - SYSCLK driver is only an output. Updated driver diagram to show this - Changed polarity of SYSCON_DIR description - Fixed slave window naming inconsistencies - Clarified description for IS_IRQn and IE_IRQn register bits
1.2.3	4/6/12	- Updated endian selection table - Fixed description for IE_IRQx interrupt enables
1.2.4	4/8/13	- Corrected description for DMA handler DMA_LADDR and DMA_VADDR registers

Version	Date	Comment
1.3.0	7/2/14	- Changed location of local User_CSR registers to avoid location conflict with VME64x reserved memory space - Added CRAM_OWNER functionality - Added user-defined bit set/clear register - Added module-enable flag in bit-set/clear register - Added CR_ADER register and updated VSLW_xx registers to support VME64x compliant ADER operation - Added top-level generics to support backwards compatibility modes for the address map switch and the new ADER operation. This feature will be removed in a future release! - Added device version register DEV_VER and top-level generic G_USER_VERSION - Updated SLVW_OFFSET to 16-bit to match implementation - Added advanced interrupter to support D08(O), D16 and D32 interrupt vectors. This feature can be selected using the G_INTERRUPTER generic. - Increased the number of slave windows to 8 - Added new slave window decode options
1.3.1	04/16/15	- Introduced auto-dtack backwards compatibility mode - Changed location of auto-dtack enable bit in slave access decoder - Refined bus_slv_byte_valid description
1.3.2	10/26/14	- Local bus address for DMA transfers is 32-bit and not 24-bit.

1. Overview

The VMESCmodule is a VME System Controller core designed for FPGA and ASIC integrations. The core contains VME Slave and Master functions as well as System Controller features such as bus timer, arbiter, IACK daisy-chain driver, system clock driver, and provisioning for CR/CSR.

The core contains all functionality needed for a VME system controller design. It can as well be used in situations where only VME Master or VME Slave functions are needed.

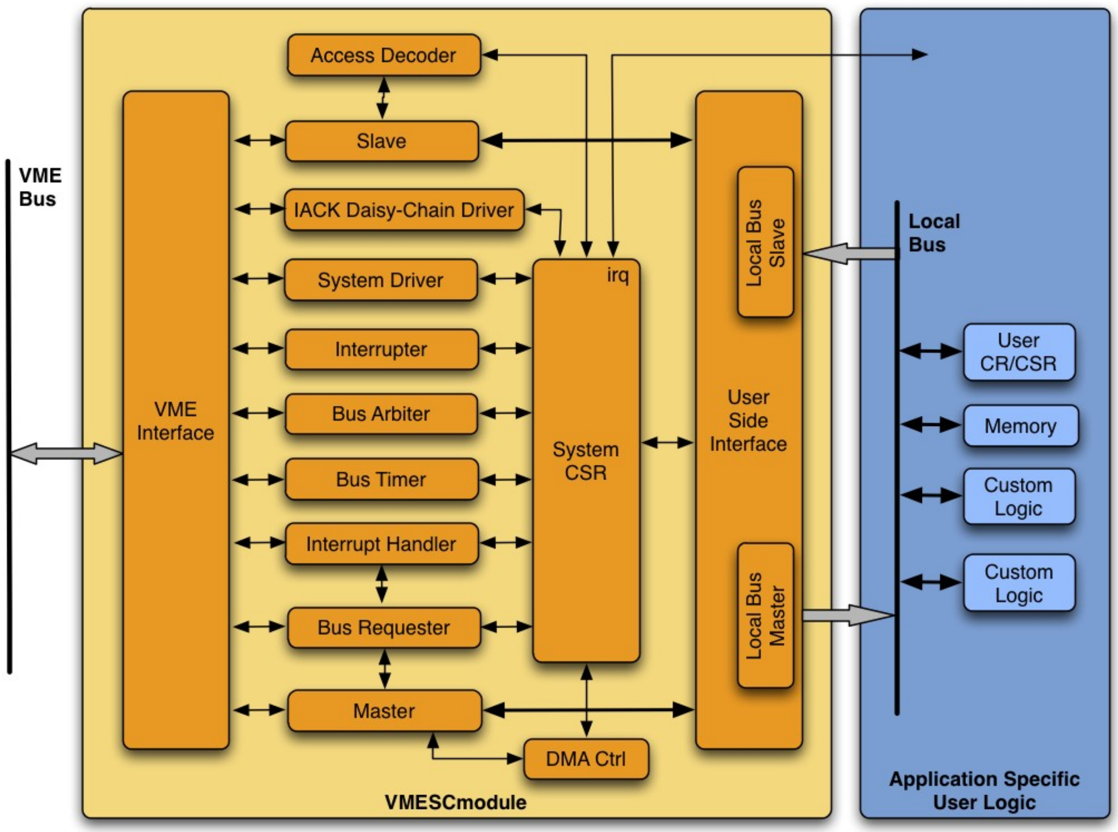

Figure 1.1: VME System Controller Block Diagram

1.1. Features

System Controller

- Bus Arbiter
  - Fixed priority
  - Round robin
- Bus Timer
  - Programmable 1-255 μs timeout
- SYSCLOCK* driver
- SYSFAIL* driver
- First Slot Detector
- IACK daisy-chain driver

Master Interface

- Coupled transfers for single data cycles
- Addressing modes: A16, A24, A32
- Data types: D08(EO), D16, D32
- Access modes: Read, write, read-modify-write
- Auto-dtack backwards compatibility mode

DMA Engine

- Used to transfer data blocks
- Addressing modes: A16, A24, A32
- Data modes: D08(EO), D16, D32, D32-BLT, D64-MBLT
- Supports data read-ahead and posted write to increase throughout
- Selectable constant local bus address for DMA transfers to/from FIFOs
- Address translation

Slave Interface

- Addressing modes: A16, A24, A32
- Data types: D08(EO), D16, D32, D32-BLT, D64-MBLT
- Access modes: Read, write, read-modify-write
- Selectable rescinding DTACK
- Provides big-endian to little-endian conversion option
- Supports up to 8 slave windows
- Auto-dtack backwards compatibility mode

Interrupt Handler

- Automatically fetches STATUS/ID vector from pending VME interrupt requests
- Supports D08(O), D16, and D32

Interrupter

- D08(O), D16 and D32
- Software interrupt request (ROAK)
- User interrupt request (RORA)
- Programmable interrupt level and type

Bus Requester

- RWD (release when done) and ROR (release on request) arbitration schemes
- FAIR requester
- Supports early withdrawal of bus request

Local Bus Interface

- Fully synchronous bus interface for user logic
- User selectable wait-states
- Optional big-endian to little-endian conversion

CR/CSR

- Contains address decoding for CR/CSR space
- Local CSR configuration registers

1.2. Deliverables

- RTL code
- Self-verifying system-level testbench
- Simulation and synthesis scripts
- Synthesis information
- User guide

1.3. Functional Description

1.3.1. VME Master

Single cycle VME transfers originating from the local bus can directly generate single VMEbus data transfer cycles. These are coupled transactions.

The VME master supports following type of data transfer modes:

- Address mode: A16, A24, A32
- Data types: D08(EO), D16, D32
- Data modes: Read, write, read-modify-write

1.3.2. DMA Handler

The DMA handler is used when more than just a few data cycles are needed to transfer data between the local bus and the VME bus.

Control registers reside in the internal CSR. They can be either programmed from the local bus or from the VME bus. Following options are available:

- VME start address
- Local bus start address
- Address mode: A16, A24, A32
- Data types: D08(EO), D16, D32, D32-BLT, D64-MBLT
- Transfer size in beats
- Selectable read-ahead and posted-write
- Transfer direction (VME read or VME write)
- Selectable constant local bus address for DMA transfers to/from local FIFOs
- Address translation

VME Write Operation

1) CPU configures the DMA transfer using the DMA_LADDR, DMA_VADDR, and DMA_CMD registers.
   - DMA_LADDR defines the source address of the local bus
   - DMA_VADDR defines the destination address of the VME bus
   - DMA_CMD configures the data transfer such as data type, length of transfer, address modifier code
2) VME Master requests VME bus access. Once granted, the Master transfers data from the local bus to external VME slave.
3) DMA done interrupt flag IS_DMADONE is asserted upon successful completion of data transfer, IS_DMAERR is asserted if an error is detected.

VME Read Operation

1) CPU configures the DMA transfer using the DMA_LADDR, DMA_VADDR, and DMA_CMD registers.
   - DMA_VADDR defines the source address of the VME bus.
   - DMA_LADDR defines the destination address of the local bus.
   - DMA_CMD configures the data transfer such as data type, length of transfer, address modifier code.
2) VME Master requests VME bus access. Once granted, the Master transfers data from the external VME slave to the local bus.
3) The DMA done interrupt flag IS_DMADONE is asserted upon successful completion of data transfer, IS_DMAERR is asserted if an error is detected.

Example:

To read a data block from VME address 0x40000100 ~ 0x400001FF and save it in the local memory at location 0x1000-0x101FF, the DMA registers are set as follows:

```
DMA_VADDR   = 0x40000100
DMA_LADDR   = 0x1000
```

Once the DMA address register is set, the transfer is started by setting the DMA command register:

```
DMAC_RAE    = 0       // disable read ahead
DMAC_PW     = 0       // disable posted write
DMAC_WIDTH  = 4       // this is D32-BLT access
DMAC_SIZE   = 0x3F    // need 64 beats/cycles to transfer 256 bytes
DMAC_AM     = 0x0F    // use A32 supervisory block
DMAC_RWN    = 0       // perform VME read operation
DMAC_ABORT  = 0       // no abort
DMAC_REQ    = 1       // start transfer
```

1.3.3. VME Slave

Using the VME slave, other VME masters can access the local configuration registers as well as devices and memories connected to the user-side interface. To access the user-side interface, four separate memory windows are available that map a section of the local user-side memory into the VME address space.

The VME slave supports following type of data transfer modes:

- Addressing modes: A16, A24, A32
- Data types: D08(EO), D16, D32, D32-BLT, D64-MBLT
- Access modes: Read, write, read-modify-write

VME Slave Window

The VMESCmodule maps eight different VME memory windows into the local user-side memory space:

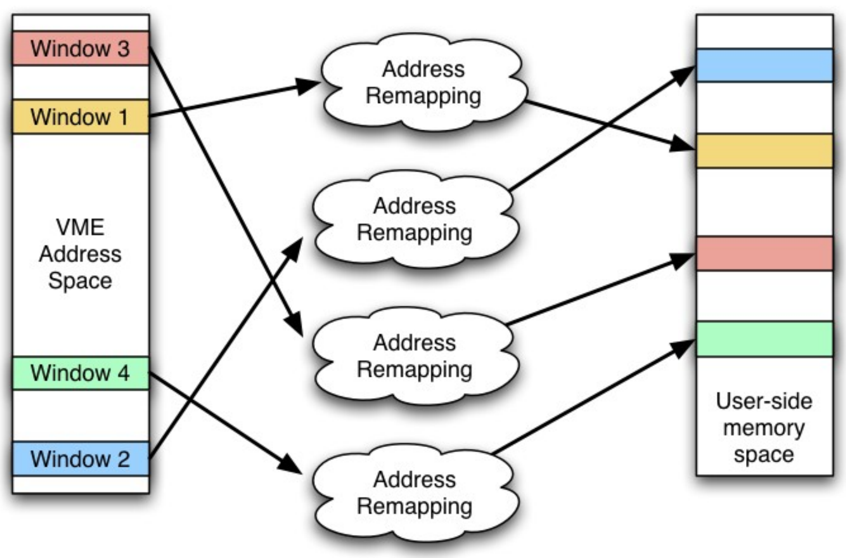

Figure 1.2: VME Slave Windows

To detect if a VME slave access matches one of these windows, following tasks are performed:

1) The VME address is masked with the VME Address Mask Register (SLVWn_ADEM) and then compared with the expected CSR ADER compare bits (CSR_ADERn.C)
2) The VME address modifier is compared with the preprogrammed value (CSR_ADERn.AM)
3) The slave address window needs to be enabled (SLVWn_EBL) and the module_enable bit set in the CSR BIT-SET/CLEAR register.

For gate-count optimizations, the 8 slave windows can be configured using top-level generics:

Generic Name	Description
G_VME_SLVWn_AV n=1..8	Slave window available For gate-count optimization, each slave access window can individually disabled. 0: Slave window is not available 1: Slave window is available
G_VME_SLVWn_SIZE n=1..8	Slave window size The window size is defined as 256 x $2^{G\_VME\_SLVWn\_SIZE}$: 0: 256 bytes 1: 512 bytes 2: 1k bytes … 15: 8M bytes Others: not valid

This decoding procedure is shown in following figure:

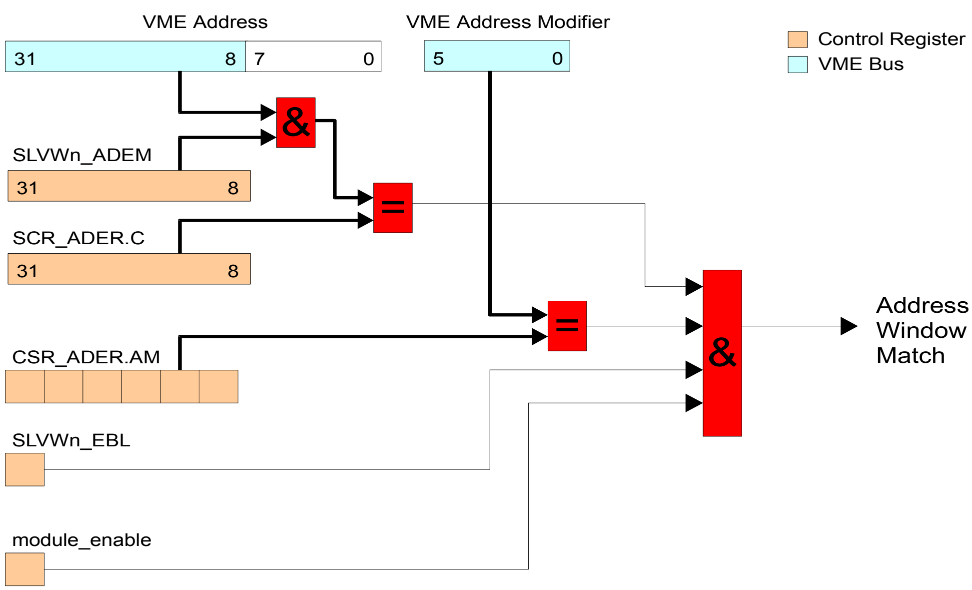

Figure 1.3: VME Slave Address Space Decoding

Once a success full match is determined, the local user side memory address is calculated based on the VME address and the address offset (SLVWn_OFFSET) as shown in following figure:

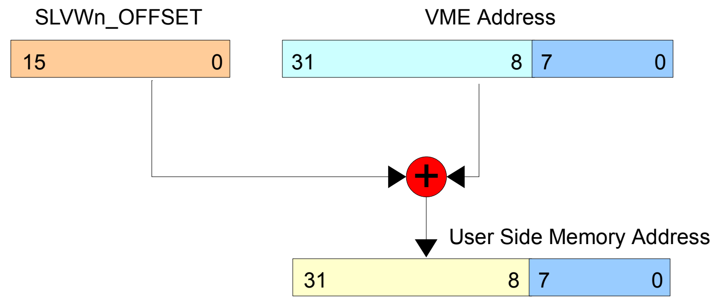

Figure 1.4: VME Slave Address Calculation

The lower 8 address bits are left as they are. This leads to a minimal window size of 256 bytes. Depending on the address mask register, the window size may be bigger, but it is always a multiple of 256 bytes and it is always aligned on a 256 byte boundary.

The slave window decoder becomes active once the module-enable flag in the bit-set register is set.

Example

This example shows how the VME address range 0x10001000-0x100013FF is mapped onto the local memory locations 0x1400-0x17FF using a window size of 1k byte. Only single-cycle A32 data access is supported.

The local memory address is calculated as follows:

```
user_addr[7:0]  = vme_addr[7:0]
user_addr[23:8] = vme_addr[23:8] + VSLVM_OFFSET[15:0]
```

This is the configuration to properly detect access to the slave window 1 and perform the necessary address translation::

```
SLVW1_OFFSET  = 0x04     // address offset

SLVW1_ADM     = 0xFFFFFC // mask register: bits 1:0 are don't care
CSR_ADER1.C   = 0x100010 // address decoder compare register
CSR_ADER1.AM  = 0x09     // am register for A32 non-privileged data
SLVW1_EBL     = 1        // enable slave window
```

Enhanced Address Window Decoding

The Address Decoder compaRe (ADER) register defined in the VME64x standard allows one Address Modifier (AM) code per window. If a memory window has to support single-cycle and block transfer modes, two slave windows are necessary.

The VMESCmodule provides additional decoder flags that allow a more flexible use of the slave windows while still supporting the VME64X ADER features.

With the slave access decoding register (SLV_ACC_DECn), each slave window supports following additional access modes:

- Data access overwrite[1]
- Program access overwrite[1]
- Non-privileged access overwrite
- Supervisory access overwrite
- BLT access overwrite
- MBLT access overwrite

Example:

Assuming that the slave window 2 is configured for A24 non-privileged program access (AM code 0x3A) and the BLT access overwrite flag is set, then this slave window will support A24 non-priv - ileged block transfer (AM code 0x3B) too.

1.3.4. VME Bus Requester

The Bus Requester module is used by the VME Master block and the Interrupt Handler to request bus access. It can be configured using the internal Configuration and Status Register (CSR).

- Supports RWD (release when done) and ROR (release on request) arbitration schemes
- FAIR requester
- Supports early withdrawal of bus request
- Configurable bus request level

1.3.5. VME System Controller

The VME core can become the system controller when it is located in slot 01 of a VME system.

Bus Arbiter

The bus arbiter can be configured to either support fixed-priority or round-robin arbitration:

- Fixed priority arbiter
  In this mode, bus requests are served from level 3 through 0. The highest priority

1 These flags enable single-cycle transfers.

request is served first.
If a bus request with a higher priority is detected, the bus arbiter tries to clear the bus by asserting BCLR*.

- Round robin arbiter
  In this mode, all levels are served in a round robin mode. Scanning from levels 3 to 0. Only one grant is issue per level.

If the requester doesn't assert BBSY* within 16 us, the arbiter withdraws the grant and asserts the IS_VARBITER interrupt status bit.

Interrupt Daisy-chain Driver

As part of the system controller, the VMESCmodule contains an interrupt daisy-chain driver.

1.3.6. Utility Functions

Bus Timer

A programmable timer measures the time (1 μs – 255 μs) between DS* assertion and the DTACK* generation. The timer is started when DS* is asserted and cleared with DTACK* or BERR*. If the timer exceeds the configured time BERRTIMER, BERR* is asserted to abort the currently pending transaction.

System Clock Driver

The system clock driver generates a continuous 16 MHz signal SYSCLOCK. This signal is always available, even during reset. The system clock driver is disabled of the VMESCmodule is not a system controller.

System Reset Driver

The system reset driver is active when the VMESCmodule is a system controller.

SYSRESET* is driven

- under software control using the SYS_CTRL.SRESET register
- when the hardware reset RESET_N is active
- when ACFAIL* is asserted as shown in the figure 1.5. This feature can be enabled by using the SYS_CTRL.ACFAIL_EBL configuration register.

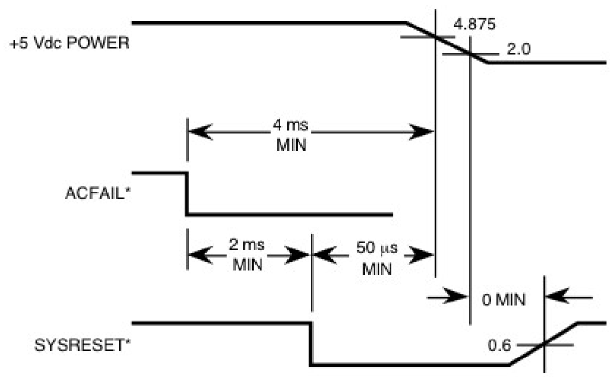

Figure 1.5: Power Monitor Power Failure Timing

See paragraph 1.3.12 Reset Logic (page 25) for more information on reset generation.

System Failure Diagnostics

In VME systems SYSFAIL* is used as indicator for ongoing system failure analysis or as an indicator of a system failure.

SYSFAIL* can be set and cleared under software control using the SDES bit of the BIT_SET and BIT_CLEAR registers. The user side signal user_sysfail_n is provided for diagnostics. Usually, it drives a status LED to help a visual inspection to determine which board has failed.

A top-level generic is available to set the SYSFAIL* behavior at power-up or system reset:

Generic Name	Description
G_SYSFAIL_MODE	SYSFAIL* Mode Selection Upon hardware reset or a system reset, the core can assert SYSFAIL* 0: Do not assert SYSFAIL* upon reset 1: Assert SYSFAIL* upon reset

If SYSFAIL* is driven at power-up, it has to be cleared by using the BIT_CLEAR.SDES bit. SYSFAIL* is driven independently of the system controller mode.

First Slot Detector

The first slot detector logic determines if the board is located in slot 01 of the VME system. This functionality is enabled by setting the external pin VAUTOCFG to 1. The first slot detection works as follows:

- 40ms after reset, the first slot detector evaluates the level of VBGI_N[3] as defined in VME Auto System Controller operation. If a low value is sampled, the board becomes a system controller.
- Using the SYSCTRL_SET command register, the host processor can force a board to become System Controller.

If the automatic first slot detector logic is disabled (VAUTOCFG = 0), the VME core becomes system controller when VCFG_SYSCON is 1 or when set using the SYSCTRL_SET register.

1.3.7. Interrupt Handling

VME Bus Interrupter

The VME Bus Interrupter returns the local STATUS/ID vector VINT_STAT during a VME IACK cycle when the interrupt level matches a pending request. There are two possible interrupt request sources:

- Software interrupt
  A software interrupt request is created by setting the VINT_SWREQ bit. This will cause a VME interrupt on the level defined by VINT_SWIRQ.
  The software interrupt is automatically acknowledged during a VME IACK cycle (ROAK).
- User interrupt
  By asserting the input user_vint_req, a VME interrupt is generated. The level is defined by VINT_UIRQ.
  To acknowledge the user interrupt, the interrupt service routine needs first to acknowledge the external interrupt source before acknowledging the VIS_UIRQ flag (RORA).

The interrupter can either operate as a simple or an advanced interrupter. The mode can be selected using the top-level generic G_INTERRUPTER.

- Simple Interrupter: G_INTERRUPTER = 0
  This is a D08(O) interrupter where bit 0 of the interrupt vector identifies the interrupt source (0: software interrupt, 1: user interrupt). The other bits of the interrupt vector are set according to the VINT_STAT register.
- Advanced Interrupter: G_INTERRUPTER = 1
  The advanced interrupter can be configured to generate D08(O), D16 or D32 interrupt vectors using the VINT_TYPE register. Each interrupt source has its own 32-bit STATUS/ID register (user interrupts: VINT_STAT, software interrupts: VINT_STAT_SW).

The VME interrupt request is only generated when the respective interrupt source is enabled by setting its VINT_EBL flag to one.

During the VME interrupt service routine, the VME interrupt handler performing the IACK cycle accesses the VME interrupt status register VINT_STATUS to determine the interrupter source and acknowledge it.

If both a software interrupt and a user interrupt are pending on the same level, then the software interrupt will be acknowledged first.

VME Bus Interrupt Handler

The VME Interrupt Handler can autonomously fetch the STATUS/ID vector on all seven VME interrupt levels. Only interrupt levels are served that have their respective IE_IRQn flag set. The interrupt request with the highest priority is served first.

If a VME IRQ[n]* is pending, the local interrupt request enable IE_IRQn is set, and and the local interrupt status bit IS_IRQn is not set, then the Interrupt Handler will fetch the STATUS/ID vector by issuing an IACK cycle using the requested priority level.

In order for the Interrupt Handler to access the VME bus, he first has to request bus ownership. He does so using the bus master request level set in the VMSTREQ register. The Interrupt Handler has a higher priority to access the VME bus than the DMA engine. If a DMA operation is already in progress, the Interrupt Handler has to wait until the end of this operation. If the Bus Requester is programmed for *release when done* operation, then the Interrupt Handler has to request bus ownership again, otherwise with *release when done*, the handler can directly execute the IACK cycle. The STATUS/ID vector fetched will be stored in the respective VME_IRQn_STAT register. Upon completion of this cycle, the IS_IRQn bit is set.

Once the user application has read the STATUS/ID from VME_IRQn_STAT, it has to acknowledge this interrupt by writing a one to the IS_IRQn flag. If a new VME IRQ[7..1]* interrupt request happens while the local interrupt status bit is still set, the new request will not be served. This guarantees that the user application will not miss a STATUS/ID vector. Once the IRQn interrupt is cleared by setting the IS_IRQn bit, the Interrupt Handler can serve an other VME interrupt at the same level.

If a bus error BERR* is detected during a VME cycle, VME_IRQn_ERR and IS_IRQn bits are set. This means that VME_IRQn_STAT is only valid if the respective VME_IRQn_ERR bit is not set.

Following flowchart shows the process of serving an VME interrupt request and how the resulting local interrupt request is processed by the user application.

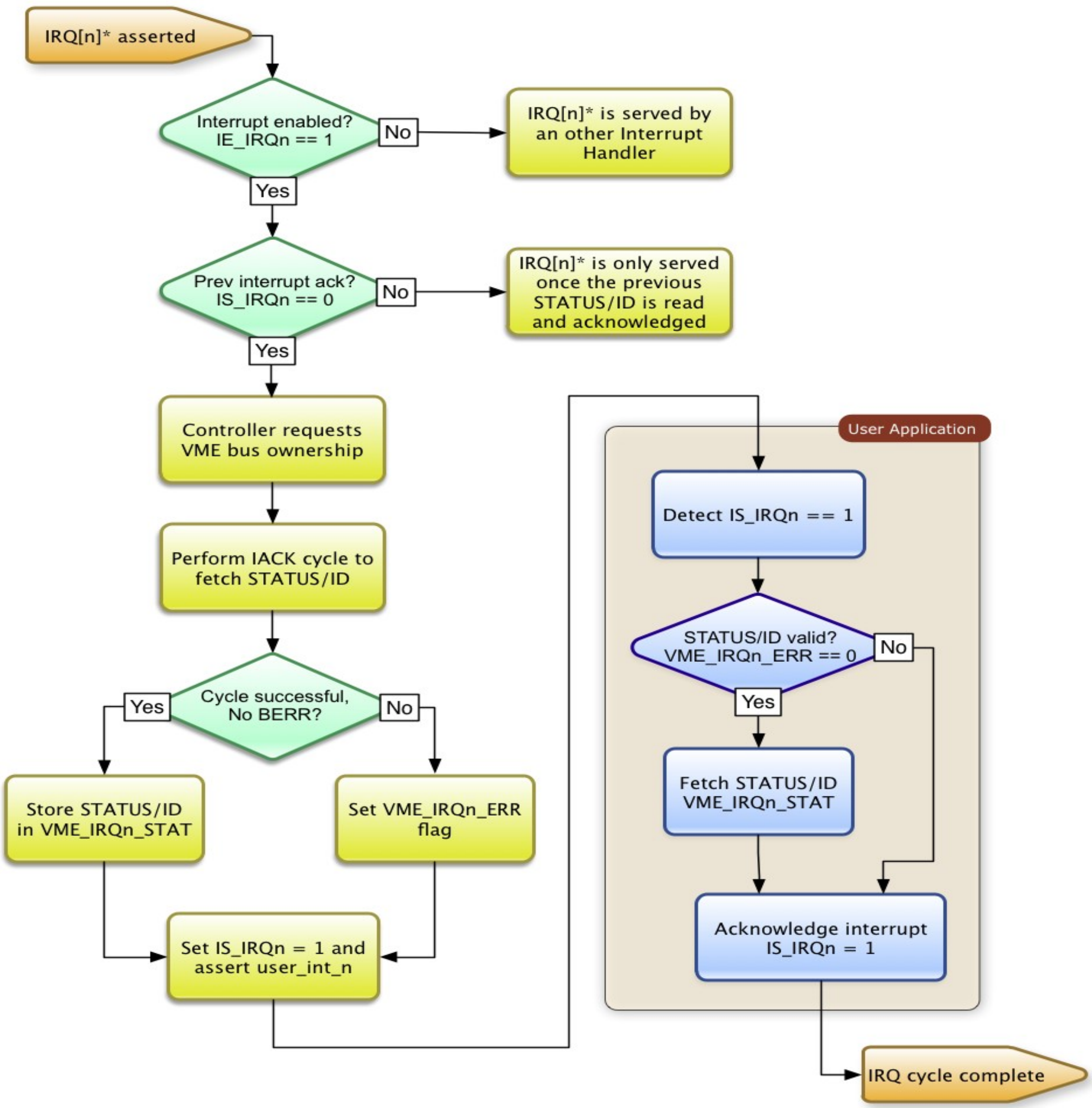

Figure 1.6: Interrupt Handler Flow Chart

Interrupt Controller

Several interrupts are generated based upon different local interrupt events. Each interrupt source can be individually enabled.

Interrupt sources:

- Mailbox 0-3 write access
- DMA cycle done
- DMA error
- VME bus error
- VME bus timer expired
- AC Fail
- System Fail
- VME software interrupt
- VME IRQ[7:1] interrupt

1.3.8. Control And Status Registers

All control and status registers can be accessed either by an external VME master or by the local CPU.

- Access via VME
  All control and status registers are mapped into the CR/CSR space as defined the the VME and VME64x specification.
- Access via local bus
  All control and status registers are directly accessible using the User CSR Port.

The VMESCmodule contains an internal arbiter to protect against data corruption due to concurrent CSR access.

1.3.9. Mailbox Registers

Mailbox registers are used as a communication channel between the VME bus and the local CPU. When an external VME master writes to the user-side memory, he can set a specific code in the mailbox. A write operation will generate an interrupt to the CPU to indicate that new data is available. Mailboxes can be used as flow control mechanism.

4 separate mailbox registers are available to provide a communication path between the VME bus and the local bus, or vice-versa.

- Read and write access is provided from VME bus and local bus
- Writing to the mailbox register will set the respective irq_mbox[n] interrupt source. If the respective interrupt is enabled, a local bus interrupt is generated.

1.3.10. Semaphores

The System Controller has 4 semaphore registers. They can be used as access control to common resources such as VME slave memory window.

Each semaphore is 8-bits. Bit 7 is the semaphore bit and the other 7 bits are used as a tag. In order to have semaphores working properly, the system setup needs to guarantee that all tags are unique (eg, not two masters use the same tag!).

Operation:

1) Write new semaphore tag to semaphore register and set bit 7
2) Read back semaphore. If read value matches the requested semaphore tag, the semaphore is granted. If semaphore is not granted, restart at 1)
3) Normal operation
4) Once the semaphore is not needed anymore, write to semaphore with bit 7 cleared.

The semaphore bit (bit 7) is used as access control. When set, the semaphore is protected from updates. To clear a semaphore, write to it with bit 7 set to zero. Only the port that requested the semaphore my clear it! E.g., if the VME port set the semaphore, only the VME port can clear the semaphore, if the local CPU set the semaphore, only the local CPU may clear it.

1.3.11. Registers

All System Controller's internal control and status registers are accessible from the VME bus and the user-side bus. The unused 510kB of CSR space is available on the user-side as User CR/CSR.

Memory Mapping

The memory mapping is shown in following figure:

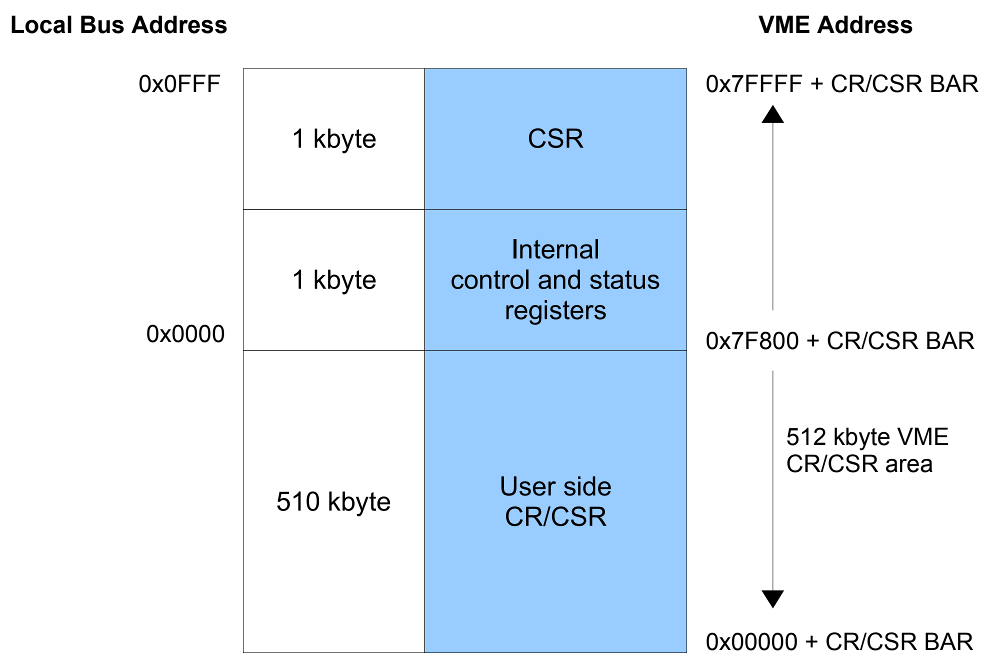

Figure 1.7: CR/CSR Memory Mapping

The initial CR/CSR BAR address is assigned upon power-up according to the geographic address board location. This can be later changed by the system software.

Access to the CSR space from the VME bus uses the A24 CR/CSR AM code.

Endian Selection

All internal registers are 32-bit wide and represent data in little endian format.

– The VMESCmodule automatically changes the data format between the VME big-endian and the registers little-endian format.

– Using the LENDIAN configuration bit, the user can select the endian format of the user side bus interface.

The behavior of the LENDIAN configuration is shown below:

– LENDIAN = 0: Local bus is big endian

– LENDIAN = 1: Local bus is little endian

Type	Address	VME				LENDIAN	VME Bus				User Side Bus			
		DS1*	DS0*	A01	LWORD*		[31:24]	[23:16]	[15:8]	[7:0]	[31:24]	[23:16]	[15:8]	[7:0]
Word	0	0	0	0	0	0	B(0)	B(1)	B(2)	B(3)	B(0)	B(1)	B(2)	B(3)
Half-word	2	0	0	1	1				B(2)	B(3)			B(2)	B(3)
	0	0	0	0	1				B(0)	B(1)	B(0)	B(1)		
Byte	3	1	0	1	1					B(3)				B(3)
	2	0	1	1	1				B(2)				B(2)	
	1	1	0	0	1					B(1)		B(1)		
	0	0	1	0	1				B(0)		B(0)			
Word	0	0	0	0	0	1	B(0)	B(1)	B(2)	B(3)	B(3)	B(2)	B(1)	B(0)
Half-word	2	0	0	1	1				B(2)	B(3)	B(3)	B(2)		
	0	0	0	0	1				B(0)	B(1)			B(1)	B(0)
Byte	3	1	0	1	1					B(3)	B(3)			
	2	0	1	1	1				B(2)			B(2)		
	1	1	0	0	1					B(1)			B(1)	
	0	0	1	0	1				B(0)					B(0)

Note:

– *B(0)* indicates VME Byte(0), *B(1)* is Byte(1), etc

– *Address* is a byte address (31:0).

1.3.12. Reset Logic

Several different sources can reset the VME System Controller core:

- RESET_N: Local hardware reset
- VSYSRSETI_N: VME system reset input
- VACFAILI_N: AC failure detection input
- SYS_CTRL.SRESET register: VME system reset register
- SYS_CTRL.LRESET register: User side reset
- BIT_SET/CLR.LRSTS register: Local board reset

Depending on which reset source is used, different parts of system are affected:

Reset Source	VMESCmodule	user_reset_n	VSYSRESETO_N[2]
RESET_N	performed	asserted	asserted
VSYSRESETI_N	performed	asserted	–
VACFAILI_N	performed	asserted	asserted, when enabled with SYS_CTRL.ACFAIL_EBL
SRESET	–	–	asserted for 200ms
LRESET	–	asserted for 200ms	–
LRSTS	–	asserted/released	–

1.3.13. System Clock

The VMESCmodule is clocked with the 64 MHz system clock CLK_SYS. No other clocks are needed.

2 SYSRESETO_N is only driven when the VMESCmodule is a system controller.

2. Signal Description

This chapter describes all interface signals of the VMESCmodule.

2.1. Global Signals

Pin Name	Direction	Description
CLK_SYS	in	System clock, 64 MHz
RESET_N	in	Reset input, asynchronous active low

2.2. VME Bus Signals

Pin Name	Direction	Description
VA_IN[31:1]	in	VME address bus, input
VA_OUT[31:1]	out	VME address bus, output
VA_INT_DRV_N	out	Internal VME address bus drive enable 0: Core drives output 1: Core doesn't drive output
VA_DIR	out	VME address bus transceiver direction 0: VME bus is driving signals (from VME bus) 1: VMESCmodule is driving signals (to VME bus)
VACFAILI_N	in	VME ACFAIL* indicator input
VAM_IN[5:0]	in	VME address modifier code, input
VAM_OUT[5:0]	out	VME address modifier code, output
VAM_INT_DRV_N	out	Internal VME address modifier code drive enable 0: Core drives output 1: Core doesn't drive output
VAM_DIR	out	VME address modifies code transceiver direction 0: VME bus is driving signals (from VME bus) 1: VMESCmodule is driving signals (to VME bus)
VAS_N_IN	in	VME address strobe, input
VAS_N_OUT	out	VME address strobe, output
VAS_N_INT_DRV_N	out	Internal VME address strobe drive enable 0: Core drives output 1: Core doesn't drive output

Pin Name	Direction	Description
VAS_DIR	out	VME address strobe transceiver direction 0: VME bus is driving signals (from VME bus) 1: VMESCmodule is driving signals (to VME bus)
VBBSYI_N	in	VME bus busy input
VBBSYO	out	VME bus busy output
VBCLR_N_IN	in	VME bus clear, input
VBCLR_N_OUT	out	VME bus clear, output
VBCLR_N_INT_DRV_N	out	Internal VME bus clear drive enable 0: Core drives output 1: Core doesn't drive output
VBERRI_N	in	VME bus error input
VBERRO	out	VME bus error output
VBGI_N[3:0]	in	VME bus grant input
VBGO_N[3:0]	out	VME bus grant output
VBRI_N[3:0]	in	VME bus request input
VBRO[3:0]	out	VME bus request output
VD_IN[31:0]	in	VME data bus, input
VD_OUT[31:0]	out	VME data bus, output
VD_INT_DRV_N	out	Internal VME data bus drive enable 0: Core drives output 1: Core doesn't drive output
VD_DIR	out	VME data bus transceiver direction 0: VME bus is driving signals (from VME bus) 1: VMESCmodule is driving signals (to VME bus)
VDRV_N	out	Global VME transceiver drive enable 0: Transceiver driver enabled 1: Transceiver driver disabled
VDS_N_IN[1:0]	in	VME data strobe, input
VDS_N_OUT[1:0]	out	VME data strobe, output
VDS_N_INT_DRV_N	out	Internal VME data strobe drive enable 0: Core drives output 1: Core doesn't drive output
VDS_DIR	out	VME data strobe transceiver direction 0: VME bus is driving signals (from VME bus) 1: VMESCmodule is driving signals (to VME bus)
VDTACK_N_IN	in	VME data transfer acknowledge, input
VDTACK_N_OUT	out	VME data transfer acknowledge, output
VDTACK_N_INT_DRV_N	out	Internal VME data transfer acknowledge, drive enable 0: Core drives output 1: Core doesn't drive output

Pin Name	Direction	Description
VDTACK_DIR	out	VME DTACK* transceiver direction 0: VME bus is driving VDTACK_N (from VME bus) 1: VMESCmodule is driving DTACK* (to VME bus)
VDTACK_DRV_N	out	VME DTACK* transceiver drive enable 0: Transceiver drive enable 1: Transceiver drive disable
VGA_N[4:0]	in	VME geographical addressing
VGAP_N	in	VME geographical addressing parity
VIACK_N_IN	in	VME interrupt acknowledge, input
VIACK_N_OUT	out	VME interrupt acknowledge, output
VIACK_N_INT_DRV_N	out	Internal VME interrupt acknowledge, drive enable 0: Core drives output 1: Core doesn't drive output
VIACKI_N	in	VME IACK* daisy-chain input
VIACKO_N	out	VME IACK* daisy-chain output
VIRQI_N[7:1]	in	VME IRQ* input
VIRQO[7:1]	out	VME IRQ* output
VLWORD_N_IN	in	VME longword data transfer size indicator, input
VLWORD_N_OUT	out	VME longword data transfer size indicator, output
VLWORD_N_INT_DRV_N	out	Internal VME longword data transfer size indicator, drive enable 0: Core drives output 1: Core doesn't drive output
VSYSFAILI_N	in	VME SYSFAIL* input
VSYSFAILO_N	out	VME SYSFAIL* output
VSYSCLK	out	VME SYSCLK* output
VCFG_SYSCON	in	VME system controller configuration This setting becomes active if VAUTOCFG=0. 0: This board is not a system controller 1: This board is a system controller
VAUTOCFG	in	VME system controller automatic configuration enable 0: System configuration based on VCFG_SYSCON input 1: Enable automatic system configuration
VSYSCON_DIR	out	VME system controller 1: VME bus bridge chip is system controller 0: VME bus bridge chip is not system controller
VSYSRESETO	out	VME system reset output
VSYSRESETI_N	in	VME system reset input
VWRITE_N_IN	in	VME write indicator, input
VWRITE_N_OUT	out	VME write indicator, output

Pin Name	Direction	Description
VWRITE_N_INT_DRV_N	out	VME write indicator, drive enable 0: Core drives output 1: Core doesn't drive output

Note:

– For better noise immunity, VIACKI_N and VBGI_N[3:0] should use schmitt-trigger type inputs

2.3. VMEbus Signals External Buffering Example

The following figures shows how external buffers can be connected to the VMESCmodule.

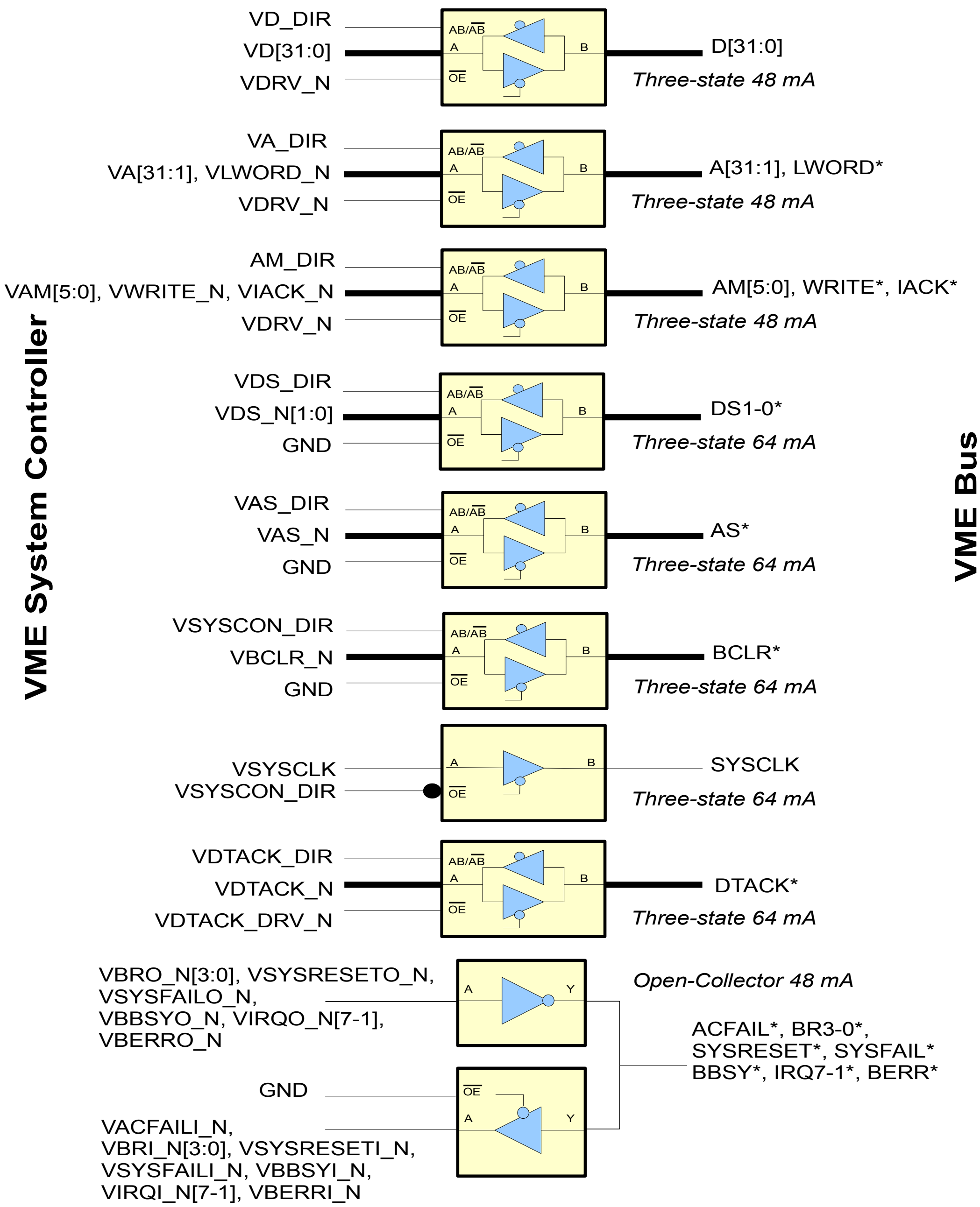

Figure 2.1: External VME transceiver connectivity

2.4. User Side Interfaces

The VMESCmodule contains two different user-side interfaces, the local bus master port and the local bus slave port. The slave port is used to access the core's internal CSR space whereas the master port provides a way for the core to access memories and registers located on the user-side.

2.4.1. Special Purpose User Side Signals

Pin Name	Type	Description
user_int_n	out	Interrupt request This signal is used to report interrupt requests based on the interrupt status and enable register (INT_STATUS/INT_EBL) to the user side. 0: Interrupt request is pending 1: No interrupt request pending
user_reset_n	out	Local system reset output. This pin is asserted using the LRESET control bit. 0: Reset is active 1: Normal operation
user_sysfail_n	out	System Failure Diagnostics This signal is used for diagnostics to assist a visual inspection to determine which board has failed. 0: The boards SYSFAIL* control register is asserted 1: Normal operation
user_int_status[17:0]	out	Masked interrupt status register This vector represents the INT_STATUS register masked with the INT_EBL register. [17]: Mailbox 3 Interrupt Status [16]: Mailbox 2 Interrupt Status [15]: Mailbox 1 Interrupt Status [14]: Mailbox 0 Interrupt Status [13]: VME Arbiter Timer Error Status [12]: VME Bus Error Interrupt Status [11]: DMA Error Interrupt Status [10]: DMA Done Interrupt Status [9]: VME Software Interrupt Acknowledge Status

Pin Name	Type	Description
user_int_status[17:0] *continued*	out	Masked interrupt status register, *continued* [8]: VME Interrupt Request 1 Status [7]: VME Interrupt Request 2 Status [6]: VME Interrupt Request 3 Status [5]: VME Interrupt Request 4 Status [4]: VME Interrupt Request 5 Status [3]: VME Interrupt Request 6 Status [2]: VME Interrupt Request 7 Status [1]: VME SYSFAIL Interrupt Status [0]: VME ACFAIL Interrupt Status
user_vint_req	in	User VME interrupt request This input is used to create a VME interrupt request. The VME interrupt request will remain asserted as long as user_vint_req is asserted (RORA). 0: No user side VME interrupt request pending 1: User side VME interrupt request pending
user_bit_set_event[7:0]	out	User-bit set event
user_bit_clear_event[7:0]	out	User-bit clear event
user_bit_status[7:0]	in	User-bit status

2.4.2. Local Bus Master Port

The local bus master port is 32-bit wide. VME cycles such as D08(EO) or D16 are mapped to the respective byte position in the 32-bit word. A D64-MBLT cycle is translated into two consecutive 32-bit local bus cycles.

Pin Name	Type	Description
lbus_mstr_acc_req	out	Data access request Active high until lbus_mstr_acc_ack acknowledges the request (or VME bus error occurs).
lbus_mstr_acc_ack	in	User-side acknowledgment signal User side access is finished by asserting lbus_mstr_acc_ack for one clock cycle.
lbus_mstr_addr[31:2]	out	Registered VME address bus
lbus_mstr_am[5:0]	out	Registered VME address bus modifier
lbus_mstr_data_wr[31:0]	out	Local data bus that contains the data written to the user side. During a write operation, lbus_mstr_data_wr is valid when usr_acc_req is asserted.
lbus_mstr_data_rd[31:0]	in	Local data bus that contains the data read from the user side. During a read operation lbus_mstr_data_rd must be valid when lbus_mstr_acc_ack is asserted.

Pin Name	Type	Description
lbus_mstr_rwn	out	Data read/write indicator 0: Write 1: Read
lbus_mstr_lock	out	Data cycle lock indicator used for read-modify-write cycles 0: No action 1: Lock target resource until consecutive write cycle finished
lbus_mstr_byte_valid[3:0]	out	User data byte valid indicator Indicates which byte of the lbus_mstr_data_wr/ lbus_mstr_data_rd bus is valid or requested. [0]: lbus_mstr_data_rd[7:0] is valid [1]: lbus_mstr_data_rd[15:8] is valid [2]: lbus_mstr_data_rd[23:16] is valid [3]: lbus_mstr_data_rd[31:24] is valid
lbus_mstr_sel_crcsr	out	User memory select – CRCSR 0: No access 1: The user-side CR/CSR memory is selected
lbus_mstr_sel_slvw[7:0]	out	User memory select – Slave Window [0]: Slave window 1 access 0: No access 1: Slave access to memory window 1 [1]: Slave window 2 access 0: No access 1: Slave access to memory window 2 [2]: Slave window 3 access 0: No access 1: Slave access to memory window 3 [3]: Slave window 4 access 0: No access 1: Slave access to memory window 4 [4]: Slave window 5 access 0: No access 1: Slave access to memory window 5 [5]: Slave window 6 access 0: No access 1: Slave access to memory window 6 [6]: Slave window 7 access 0: No access 1: Slave access to memory window 7 [7]: Slave window 8 access 0: No access 1: Slave access to memory window 8
lbus_mstr_sel_mstr	out	User memory select – Master 0: No access 1: Master access to user-side memory

Local Bus Master Port Write Cycle

Following figure shows two local bus write cycles with different wait-states. While an access is in process, all signals coming from the VME core are stable. The end of the access is indicated by the user-side logic by asserting lbus_mstr_acc_ack.

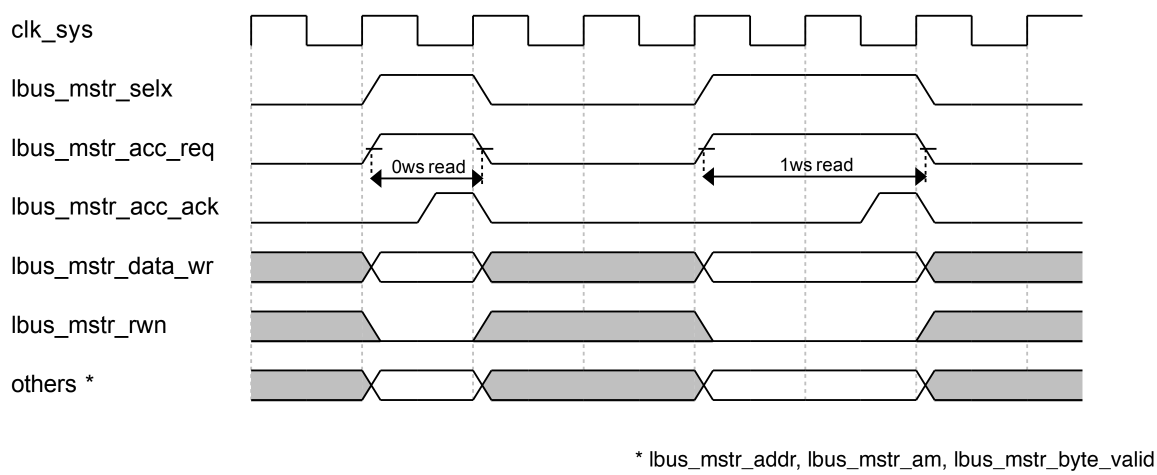

Figure 2.2: User write cycle with different wait-states

Local Bus Master Port Read Cycle

The read cycle access is similar to the write cycle. While a read is performed, lbus_mstr_data_rd must be valid at the rising edge of the clock when lbus_mstr_acc_ack is asserted.

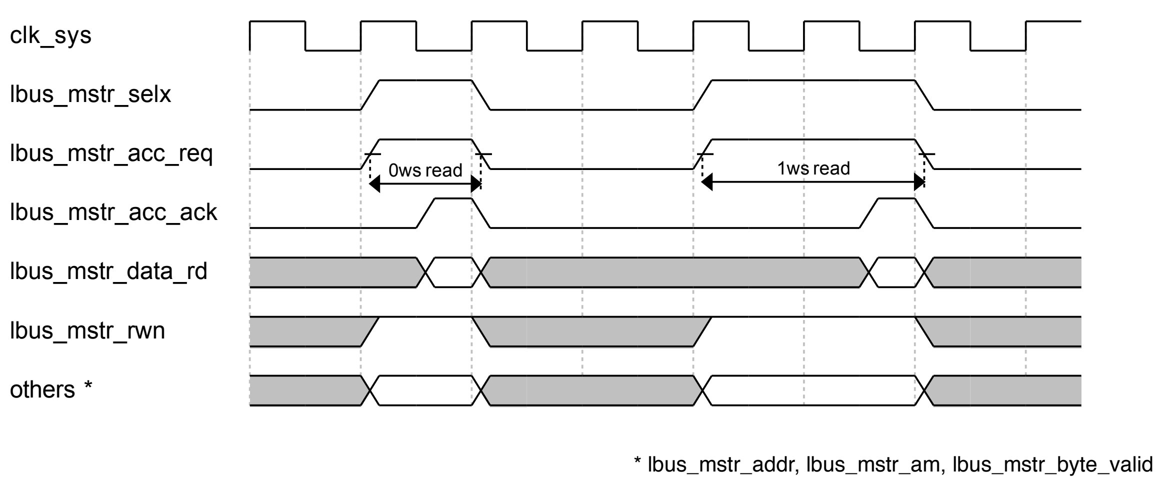

Figure 2.3: User read cycle with different wait-states

Local Bus Master Port Read-Modify-Write Cycle

Whenever a D08(EO), D16 or D32 read cycle is detected, lbus_mstr_lock is asserted. It will remain asserted until either vas_n is released after the read cycle or until the following write cycle is completed.

Following figure shows a regular read-modify-write cycle where lbus_mstr_lock stays asserted during the read and write cycles.

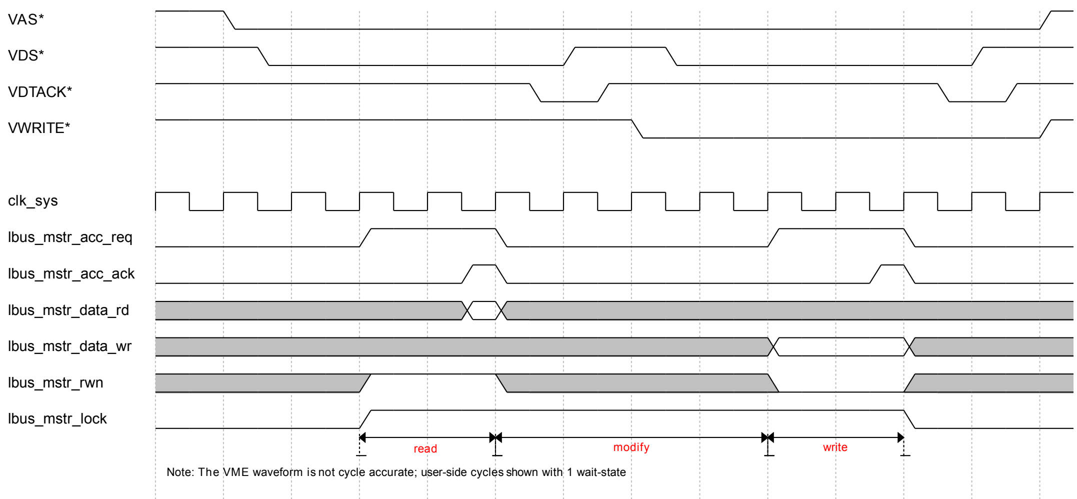

Figure 2.4: User read-modify-write cycle

Following figures shows how lbus_mstr_lock stays asserted while the local cycle is already completed. Using lock to guarantee atomic read-modify-write execution on a memory resource will slow down the local data path.

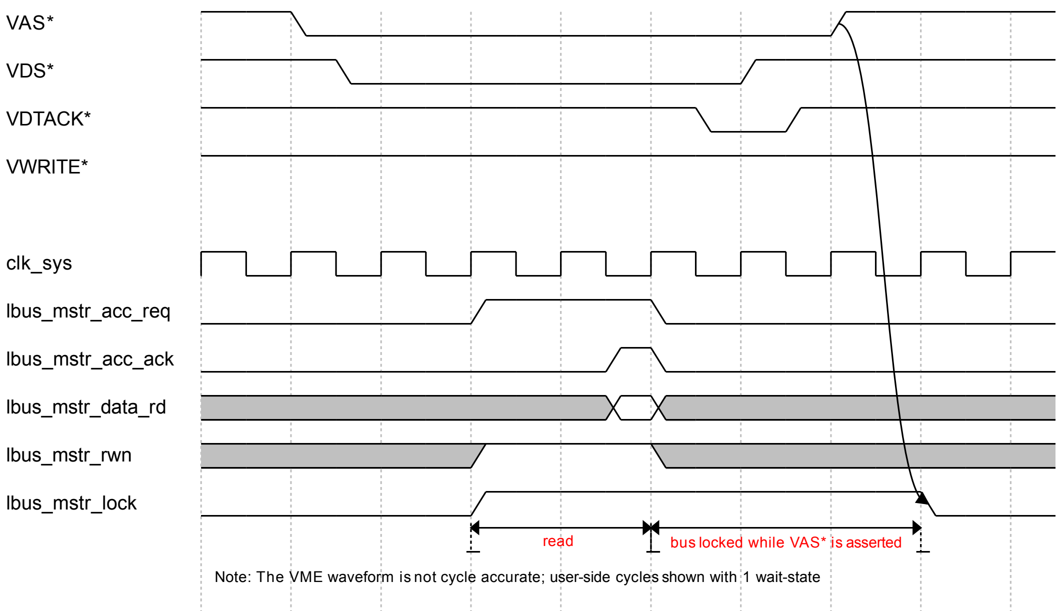

Figure 2.5: User read-only cycle

2.4.3. Local Bus Slave Port

The local bus slave port allows the user-side logic to access the System Controller's internal configuration and status registers or to create a coupled VME data transfer.

When performing a VME read or write operation, lbus_slv_byte_valid[3:0] is used to determine the VME cycle type (D08(EO), D16 or D32).

Pin Name	Type	Description
lbus_slv_acc_req	in	Data access request Active high until lbus_slv_acc_ack acknowledges the request (or VME bus error occurs).
lbus_slv_acc_ack	out	Data access acknowledge 0: Normal operation 1: The user side access cycle successfully finished This signal is asserted for one clock cycle.
lbus_slv_acc_nack	out	Data access not acknowledged 0: Normal operation 1: The user side access cycle is aborted due to an error This signal is asserted for one clock cycle.
lbus_slv_sel_csr	in	Slave select, CR/CSR memory space

Pin Name	Type	Description
lbus_slv_sel_vme	in	Slave select, VME address space
lbus_slv_addr[31:2]	in	Access address For CR/CSR memory space access, only the bit range [9:2] is taken into account.
lbus_slv_am[5:0]	in	VME address modifier
lbus_slv_data_wr[31:0]	in	Data write bus
lbus_slv_data_rd[31:0]	out	Data read bus
lbus_slv_byte_valid[3:0]	in	Data byte valid indicator Indicates which bytes of the lbus_slv_data_wr or lbus_slv_data_rd are valid. [0]: data[7:0] is valid [1]: data[15:8] is valid [2]: data[23:16] is valid [3]: data[31:24] is valid
lbus_slv_rwn	in	Data read/write indicator 0: Write 1: Read
lbus_slv_lock	in	Data cycle lock indicator used for read-modify-write cycles 0: No action 1: Lock target resource until consecutive cycle finished

Local Bus Slave Port CSR Write Cycle

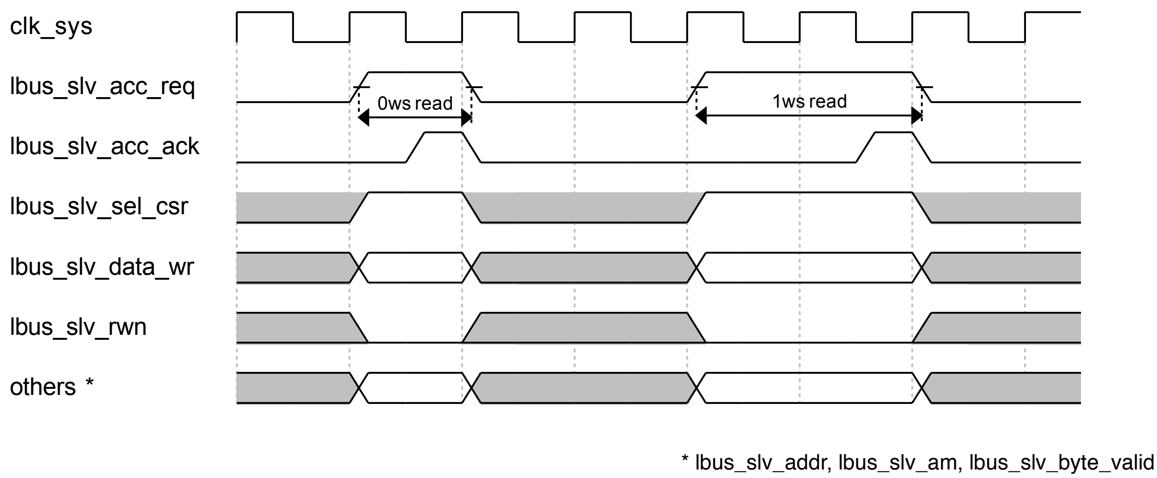

Figure 2.6: Local bus slave port CSR write timing

Local Bus Slave Port CSR Read Cycle

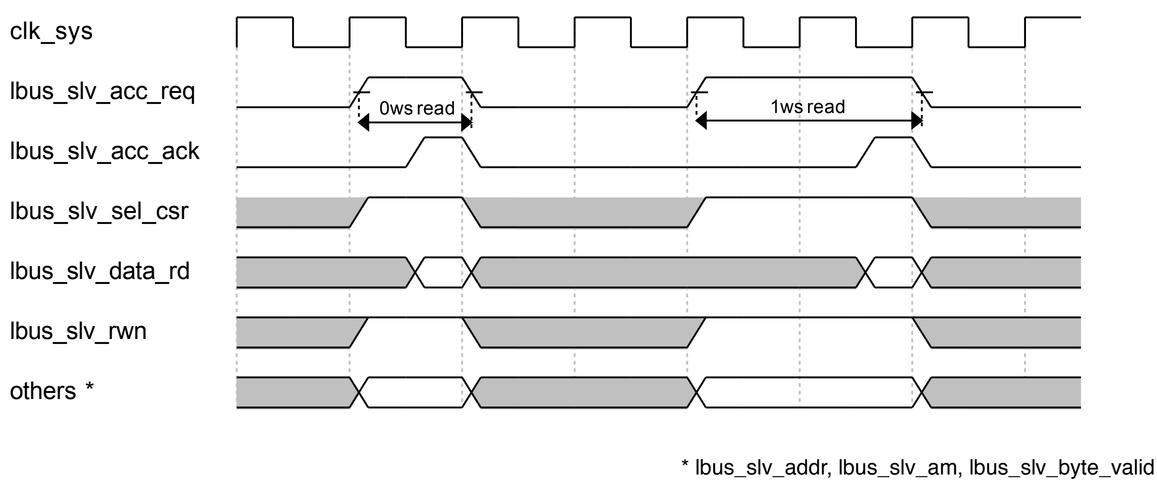

Figure 2.7: Local bus slave port CSR read timing

2.4.4. VME Access Cycles

When lbus_slv_sel_vme is asserted, user side cycles are directly translated into the respective D08(EO), D16, or D32 VME cycles. In case of a VME bus error, the transaction is aborted by driving lbus_slv_acc_nack high for one clock cycle. Read-modify-write cycles are generated by asserting lbus_slv_lock.

Local Bus Slave Port VME Read Cycle

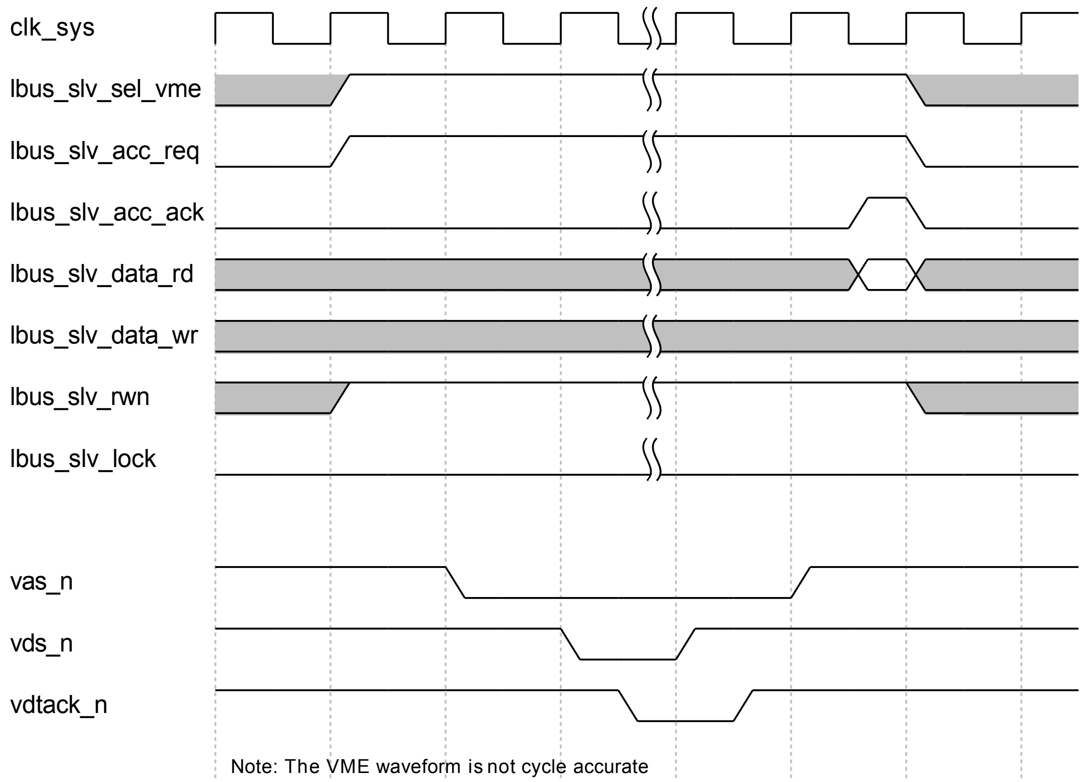

Figure 2.8: Coupled local bus VME read cycle

Local Bus Slave Port VME Read Cycle With Bus Error

If a bus error happens during a VME cycle, the local bus access is terminated by asserting lbus_slv_nack for one clock cycle.

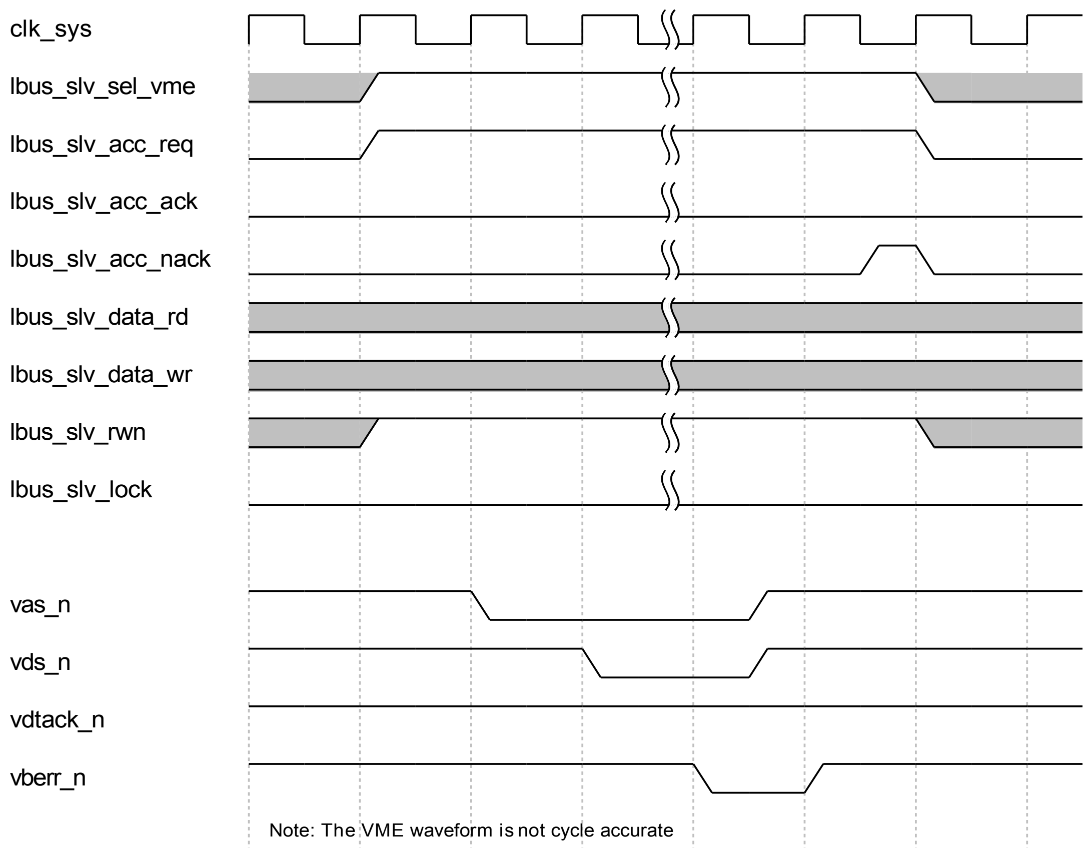

Figure 2.9: Coupled local bus VME read cycle with bus error

Local Bus Slave Port VME Read-Modify-Write Cycle

To create a VME read-modify-write cycle, the user has to assert lbus_slv_lock during the read cycle. This will cause the VME master to keep the VME address strobe vas_n asserted between the read and write cycle.

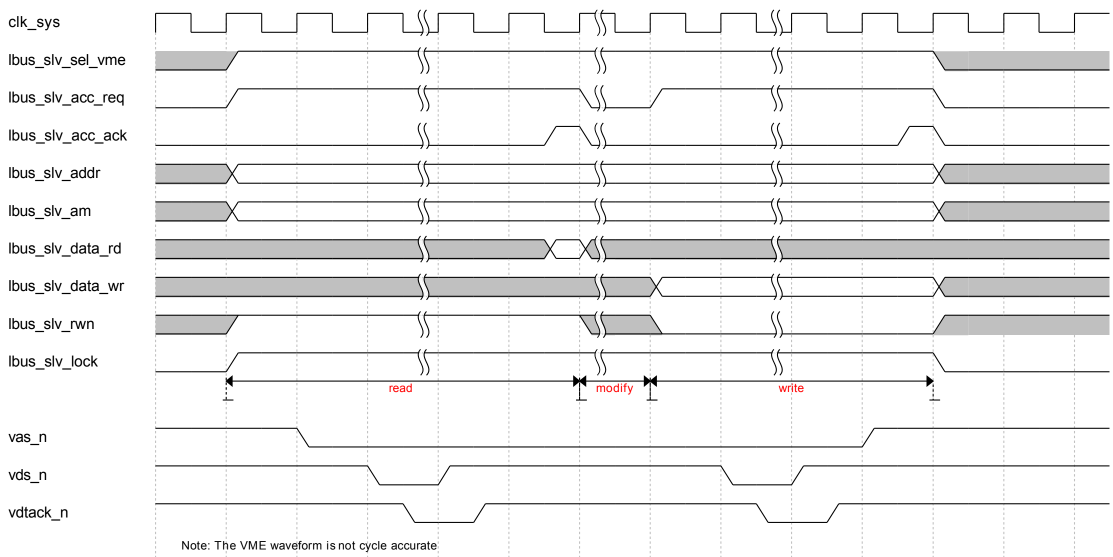

Figure 2.10: Coupled local bus VME read-modify-write cycle

Auto-DTACK

Auto-DTACK is a backwards compatibility mode that implements a *non* VME compliant behavior that is used by some now obsolete VME master and slave controllers. It allows the master to send a broadcast message to several slaves without having the slaves assert DTACK. The master will assert DTACK based on the configuration settings ADTACK_T1 and ADTACK_T2.

This feature is selected by asserting lbus_slv_auto_dtack when requesting a VME access from the local slave port:

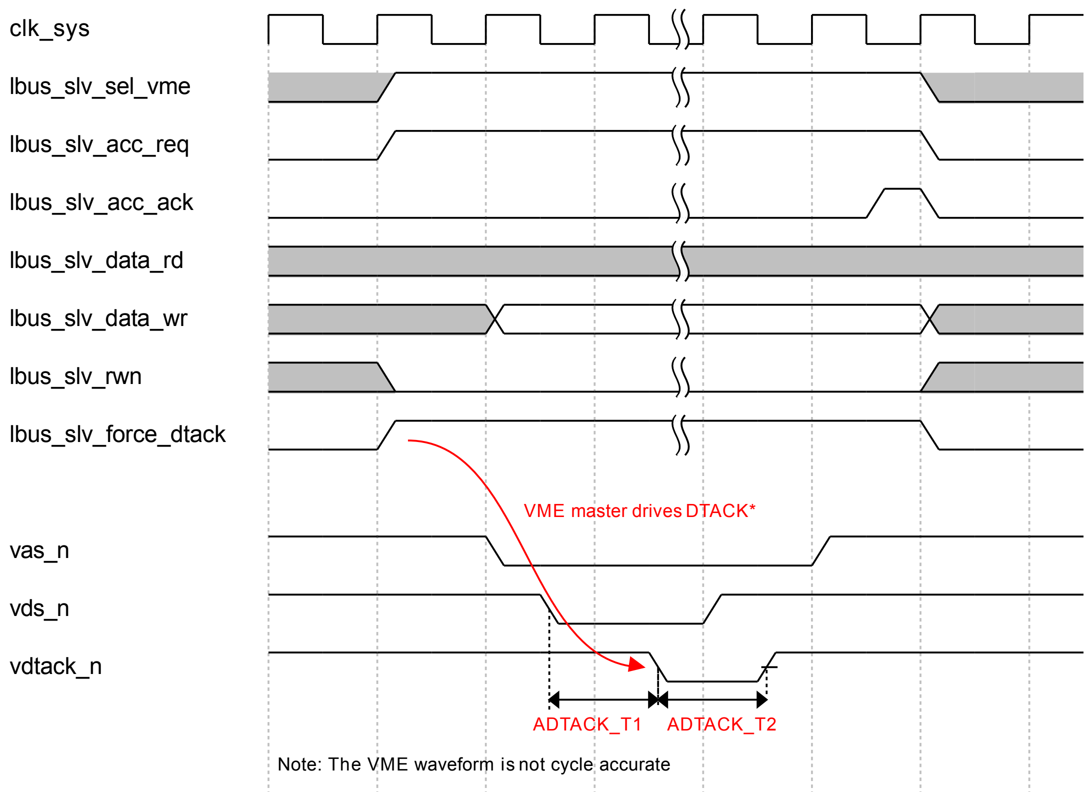

Figure 2.11: Auto-DTACK compatibility mode

Auto-DTACK should only be used to interface with old hardware and systems that require this feature and never should be used for new designs.

3. Core Configuration

The core behavior can be set by using following top-level generics:

Generic Name	Description
G_VME_SLVWn_AV n=1..8	Slave window available For gate-count optimization, each slave access window can individually disabled. 0: Slave window is not available 1: Slave window is available
G_VME_SLVWn_SIZE n=1..8	Slave window size The window size is defined as 256 x $2^{G\_VME\_SLVWx\_SIZE}$: 0: 256 bytes 1: 512 bytes 2: 1k bytes … 15: 8M bytes Others: not valid
G_SYSFAIL_MODE	SYSFAIL* Mode Selection Upon hardware reset or a system reset, the core asserts SYSFAIL* 0: Do not assert SYSFAIL* upon reset 1: Assert SYSFAIL* upon reset Once the core has completed its internal failure analysis routine, SYSFAIL* has to be released by the application by using the BIT_CLEAR.SDES register.
G_VME_SLV_DTACK	Rescinding DTACK enable The VME slave block can use rescinding dtack to accelerate data transmission. 0: Disabled 1: Enabled
G_COMP_UCSR_MEM	User CSR memory mapping backwards compatibility Starting with core version 1.3, the base address of the user CSR changed from 0x7FC00 to 0x7F800. To provide backwards compatibility with existing implementations, this generic enables to revert back to the original memory mapping. 0: User CSR starts at 0x7F800 (default) 1: User CSR starts at 0x7FC00 (backwards compatibility setting) ***Backwards compatibility mode should not be used for new designs and will be discontinued in the future!***

Generic Name	Description
G_COMP_SLVW_DEC	Slave window decoder backwards compatibility Starting with core version 1.3, the slave access decoder operation changed. Now the address decoding works as defined in the VME64x standard. To provide backwards compatibility with existing implementation, this generic enables to revert back to the original slave access decoder implementation. 0: ADER based implementation 1: SLVW_xx based implementation (backwards compatibility setting) ***Backwards compatibility mode should not be used for new designs and will be discontinued in the future!***
G_USER_VERSION	User Version Number Using this generic, the user can set a FPGA implementation specific revision number. The integer generic is interpreted as a 16-bit number and mapped to bits 31..16 of the DEV_VER version register.
G_INTERRUPTER	Interrupter selection 0: Simple D08(O) interrupter with one STATUS/ID vector 1: Advanced D08(O), D16 and D32 interrupter with separate STATUS/ID vectors for software and user interrupts.

3.1. Rescinding DTACK

The VME64 specification allows DTACK to be operated as a rescinding signal instead of an open-collector class signal. This results in an accelerated bus cycle. This feature can be selected by setting the top level generic G_VME_SLV_DTACK = 1.

4. Programmers Guide

4.1. Internal CSR Memory Space

Following table provides an overview of all registers that are part of the VMESCmodule.

Address Offset		Name	Description
VME	Local Bus		
Control/Status Registers			
0x7FFFC	0x7FC	CRBAR	CR/CSR Base Address Register (BAR)
0x7FFF8	0x7F8	BIT_SET	Bit set register
0x7FFF4	0x7F4	BIT_CLEAR	Bit clear register
0x7FFF0	0x7F0	CRAM_OWNER	Configuration RAM (CRAM) Owner Register
0x7FFEC	0x7EC	UBIT_SET	User bit-set register
0x7FFE8	0x7E8	UBIT_CLEAR	User bit-clear register
0x7FFD0 - 0x7FFDC	0x7D0 - 0x7DC	CSR_ADER7	Function 7 ADER
0x7FFC0 - 0x7FFCC	0x7C0 - 0x7CC	CSR_ADER6	Function 6 ADER
0x7FFB0 - 0x7FFBC	0x7B0 - 0x7BC	CSR_ADER5	Function 5 ADER
0x7FFA0 - 0x7FFAC	0x7A0 - 0x7AC	CSR_ADER4	Function 4 ADER
0x7FF90 - 0x7FF9C	0x790 - 0x79C	CSR_ADER3	Function 3 ADER
0x7FF80 - 0x7FF8C	0x780 - 0x78C	CSR_ADER2	Function 2 ADER
0x7FF70 - 0x7FF7C	0x770 - 0x77C	CSR_ADER1	Function 1 ADER
0x7FF60 - 0x7FF6C	0x760 - 0x76C	CSR_ADER0	Function 0 ADER
User Control Status Registers			
0x7F938	0x138	SLV_ACC_DEC5	Slave Access Decoder 5
0x7F934	0x134	SLV_ACC_CMP5	Slave Access Address Decoder Compare Register 5
0x7F930	0x130	SLV_ACC_MSK5	Slave Access Address Decoder Mask Register 5
0x7F928	0x128	SLV_ACC_DEC6	Slave Access Decoder 6
0x7F924	0x124	SLV_ACC_CMP6	Slave Access Address Decoder Compare Register 6
0x7F920	0x120	SLV_ACC_MSK6	Slave Access Address Decoder Mask Register 6
0x7F918	0x118	SLV_ACC_DEC7	Slave Access Decoder 7
0x7F914	0x114	SLV_ACC_CMP7	Slave Access Address Decoder Compare Register 7
0x7F910	0x110	SLV_ACC_MSK7	Slave Access Address Decoder Mask Register 7
0x7F908	0x108	SLV_ACC_DEC8	Slave Access Decoder 8
0x7F904	0x104	SLV_ACC_CMP8	Slave Access Address Decoder Compare Register 8
0x7F900	0x100	SLV_ACC_MSK8	Slave Access Address Decoder Mask Register 8

Address Offset		Name	Description
VME	Local Bus		
0x7F8F0	0x0F0	DEV_CTRL	Device Control Register
0x7F8FC	0x0FC	DEV_VER	Device Version
0x7F8D0	0x0D0	SYS_CTRL	System Controller
0x7F8B0	0x0B0	VME_MSTR	VME Master Controller
0x7F8A8	0x0A8	SLV_ACC_DEC1	Slave Access Decoder 1
0x7F8A4	0x0A4	SLV_ACC_CMP1	Slave Access Address Decoder Compare Register 1
0x7F8A0	0x0A0	SLV_ACC_MSK1	Slave Access Address Decoder Mask Register 1
0x7F898	0x098	SLV_ACC_DEC2	Slave Access Decoder 2
0x7F894	0x094	SLV_ACC_CMP2	Slave Access Address Decoder Compare Register 2
0x7F890	0x090	SLV_ACC_MSK2	Slave Access Address Decoder Mask Register 2
0x7F888	0x088	SLV_ACC_DEC3	Slave Access Decoder 3
0x7F884	0x084	SLV_ACC_CMP3	Slave Access Address Decoder Compare Register 3
0x7F880	0x080	SLV_ACC_MSK3	Slave Access Address Decoder Mask Register 3
0x7F878	0x078	SLV_ACC_DEC4	Slave Access Decoder 4
0x7F874	0x074	SLV_ACC_CMP4	Slave Access Address Decoder Compare Register 4
0x7F870	0x070	SLV_ACC_MSK4	Slave Access Address Decoder Mask Register 4
0x7F85C	0x06C	DMA_STAT	DMA Status Register
0x7F858	0x068	DMA_CMD	DMA Command Register
0x7F854	0x064	DMA_LADDR	DMA Local Address Register
0x7F850	0x060	DMA_VADDR	DMA VME Address Register
0x7F85C	0x05C	MAILBOX1	Mailbox Register 1
0x7F858	0x058	MAILBOX2	Mailbox Register 2
0x7F854	0x054	MAILBOX3	Mailbox Register 3
0x7F850	0x050	MAILBOX4	Mailbox Register 4
0x7F840	0x040	SEMAPHORE	Semaphore Register
0x7F83C	0x03C	VME_INT_STAT_SW	VME Interrupter Software STATUS/ID
0x7F838	0x038	VME_INT_MAP	VME Interrupter Map
0x7F834	0x034	VME_INT_STAT	VME Interrupter STATUS/ID
0x7F830	0x030	VME_INT	VME Interrupter
0x7F82C	0x02C	VME_IRQ1_STAT	VME IRQ1 STATUS/ID
0x7F828	0x028	VME_IRQ2_STAT	VME IRQ2 STATUS/ID
0x7F824	0x024	VME_IRQ3_STAT	VME IRQ3 STATUS/ID
0x7F820	0x020	VME_IRQ4_STAT	VME IRQ4 STATUS/ID
0x7F81C	0x01C	VME_IRQ5_STAT	VME IRQ5 STATUS/ID
0x7F818	0x018	VME_IRQ6_STAT	VME IRQ6 STATUS/ID
0x7F814	0x014	VME_IRQ7_STAT	VME IRQ7 STATUS/ID
0x7F810	0x010	VME_IRQH_CMD	VME Interrupt Handler Command
0x7F80C	0x00C	VINT_STATUS	VME Interrupt Status Register
0x7F808	0x008	VINT_EBL	VME Interrupt Enable Register
0x7F804	0x004	INT_STATUS	Interrupt Status Register
0x7F800	0x000	INT_EBL	Interrupt Enable Register

Note:

- Undefined register locations will read as 0x00.
- All registers support byte, half-word, and word access cycles.

4.2. Description Of Registers

4.2.1. Device Control Register: DEV_CTRL

VME Address Offset	Local Bus Address
0x7F8F0	0x0F0

Bits	Function	Description	R/W	Reset value
31:24	RESERVED	N/A	R	0x0
0	LENDIAN	Local bus endian selection 0: Local bus is big endian 1: Local bus is little endian	R/W	0x0

4.2.2. Device Version: DEV_VER

The device version uses following format *X.Y.Z*-rc*N*

VME Address Offset	Local Bus Address
0x7F8FC	0x0FC

Bits	Function	Description	R/W	Reset value
31:15	VERUSR	User version Using the top-level generic G_USER_VERSION, the user can set an FPGA specific revision number.	R	n/a
15:12	VERX	Core major version	R	0x1
11:8	VERY	Core minor version	R	0x3
7:4	VERZ	Core patch version	R	n/a
3:0	VERN	Core release candidate 0: Official release 1..15: Release candidate	R	n/a

4.2.3. System Controller: SYS_CTRL

VME Address Offset	Local Bus Address
0x7F8D0	0x0D0

Bits	Function	Description	R/W	Reset value
31:24	RESERVED	N/A	R	0x0

Bits	Function	Description	R/W	Reset value
23:16	BERRTIMER	BERR Timer 0: BERR timer is not active 1: Timeout is 1us .. 255: Timeout is 255us	R/W	0x0
15:13	RESERVED	N/A	R	0x0
12	ACFAIL_EBL	ACFAIL* Detect Enable 1: SYSRESET* is generated upon detection of ACFAIL* as show in figure 1.5. 0: No action	R/W	0x0
11	LRESET	Local Reset 1: The USER_RESET_N output is asserted for 200ms. This is used to re-initialize the local system. 0: No action Reading this register bit will return the state of USER_RESET_N	R/W	0x0
10	SRESET	VME system reset: SYSRESET 1: The VSYSRESETO_N output is asserted for 200ms. 0: No action Reading back this register bit will return VSYSRESTI_N.	R/W	0x0
9	BUS_ARB	Bus arbiter 1: Fixed priority In this mode, bus requests are served from level 3 through 0. The highest request is served first. If a bus request with a higher priority is detected, the bus arbiter tries to clear the bus by asserting BCLR*. 0: Round robin priority In this mode, all levels are served in a round robin mode. Scanning from levels 3 to 0. Only one grant is issue per level.	R/W	0x0
8	SYSCTRL_SET	Activate System Controller 1: This board is the system controller 0: No action	W	0x0
7	RESERVED	N/A	R	0x0
6	SYSCTRL	System Controller Status 1: This board is system controller 0: This board is not system controller The board automatically becomes system controller upon power-up when the geographic address indicates that the board is located in slot 1. Firmware can decide to overwrite this setting by using the Activate/De-Activate System Controller command registers.	R	-
5	GAP	Geographic Address Parity This bit represents the inverted input state of the VGAP_N input.	R	-
4:0	GA	Geographic Address These bits represent the inverted input state of the VGA_N[4:0] inputs.	R	-

4.2.4. VME Master Controller: VME_MSTR

VME Address Offset	Local Bus Address
0x7F8B0	0x0B0

Bits	Function	Description	R/W	Reset value
31:24	ADTACK_T1	Auto-DTACK T1 (setup time) This parameter defines the DTACK setup time as a multiple of system clock periods. It is the delay between when the master asserts DS0/DS1 and DTACK. The effective number is the programmed number plus 1. *Auto-DTACK is a backwards compatibility mode to interface with older, non-VME compliant hardware that requires this feature.*	R/W	0x0
23:16	ADTACK_T2	Auto-DTACK T2 (hold time) This parameter defines how long the master asserts DTACK as a multiple of the system clock periods. The effective number is the programmed number plus 1. *Auto-DTACK is a backwards compatibility mode to interface with older, non-VME compliant hardware that requires this feature.*	R/W	0x0
15:4	RESERVED	N/A	R	0x0
3	VMSTREL	VME Master Release Mode 1: Release on request (ROR) The bus is only released when an other master requests bus ownership. 0: Release when done (RWD) The bus is release upon completion of the current transfer.	R/W	0x0
2	VMSTFAIR	VME Master Fair Mode The bus requester observes the FAIR request type. A bus request is only asserted if no other bus request is pending with the same priority. 1: Use FAIR requesting scheme 0: Use direct request	R/W	0x0
1:0	VMSTREQ	VME Master Request Level	R/W	0x0

4.2.5. Slave Access Decoding (1-8): SVL_ACC_DECn

	VME Address Offset	Local Bus Address
Window 1	0x7F8A8	0x0A8
Window 2	0x7F898	0x098
Window 3	0x7F888	0x088

	VME Address Offset	Local Bus Address
Window 4	0x7F878	0x078
Window 5	0x7F938	0x138
Window 6	0x7F928	0x128
Window 7	0x7F918	0x118
Window 8	0x7F908	0x108

Bits	Function	Description	R/W	Reset value
31	SLVW_EBL	Slave Window Enable 0: Windows is disabled 1: Window is enabled	R/W	0x0
30	SLVW_ADTACK	Auto-DTACK 1: Auto-DTACK support enabled 0: Normal operation For proper operation, the VME slave requires an auto_dtack setup time ADTACK_T1 of 5 or more. *Auto-DTACK is a backwards compatibility mode to interface with older, non-VME compliant hardware that requires this feature.*	R/W	0x0
29		MBLT Access Overwrite 1: Slave responds to MBLT access cycles 0: As defined in ADER.AM	R/W	0x0
28		BLT Access Overwrite 1: Slave responds to BLT access cycles 0: As defined in ADER.AM	R/W	0x0
27		Supervisory Access Overwrite 1: Supervisory access to this window is enabled 0: As defined in ADER.AM	R/W	0x0
26		Non-Privileged Access Overwrite 1: Non-privileged access to this window is enabled 0: As defined in ADER.AM	R/W	0x0
25		Program Access Overwrite (SCT) 1: Program access to this window is enabled 0: As defined in ADER.AM	R/W	0x0
24		Data Access Overwrite (SCT) 1: Data access to this window is enabled 0: As defined in ADER.AM	R/W	0x0
23:8	SLVW_OFFSET	Address Offset	R/W	0x0
7	RESERVED	N/A	R	0x0

Bits	Function	Description	R/W	Reset value
6:4	SLVW_SEL	Slave Window Selector By default, there is a direct mapping between the slave address window and the user memory select signal: slave window n uses lbus_mstr_sel_slvw[n] signal. This mapping can be changed so several windows can use the same user memory select signal. 0: Slave window n uses lbus_mstr_sel_slvw[0] 1: Slave window n uses lbus_mstr_sel_slvw[1] 2: Slave window n uses lbus_mstr_sel_slvw[2] 3: Slave window n uses lbus_mstr_sel_slvw[3] 4: Slave window n uses lbus_mstr_sel_slvw[4] 5: Slave window n uses lbus_mstr_sel_slvw[5] 6: Slave window n uses lbus_mstr_sel_slvw[6] 7: Slave window n uses lbus_mstr_sel_slvw[7] After reset, this register is configured as follows: – Slave window 0 uses lbus_mstr_sel_slvw[0] – Slave window 1 uses lbus_mstr_sel_slvw[1] – Slave window 2 uses lbus_mstr_sel_slvw[2] – Slave window 3 uses lbus_mstr_sel_slvw[3] – Slave window 4 uses lbus_mstr_sel_slvw[4] – Slave window 5 uses lbus_mstr_sel_slvw[5] – Slave window 6 uses lbus_mstr_sel_slvw[6] – Slave window 7 uses lbus_mstr_sel_slvw[7]	R/W	0x0
3:2	RESERVED	n/a	R/W	0x0
1	SLVW_DFS	Dynamic Function Sizing This bit controls if dynamic function sizing is supported. If supported, the SLVW_ADEM register can be read using the ADER register. 1: Dynamic function sizing is supported 0: Dynamic function sizing is not supported	R/W	0x0
0	SLVW_FAF	Fixed-Address Function This bit controls whether the ADER register of this slave window is programmable. If the register is not programmable, it needs to be configured before this bit is set! 1: The ADER register is not programmable 0: The ADER register is programmable	R/W	0x0

When backwards compatibility mode G_COMP_SLVW_DEC is used, bits 7..0 have an alternative function. Backwards compatibility mode should not be used for new designs and will be discontinued in the future!

Bits	Function	Description	R/W	Reset value
7	SLVW_AM_MBLT	MBLT Access 1: Slave responds to MBLT access cycles 0: Slave does not respond	R/W	0x0
6	SLVW_AM_BLT	BLT Access 1: Slave responds to BLT access cycles 0: Slave does not respond	R/W	0x0
5:4	SLVW_AM_AS	Address Space Defines which in which VME address space this window is located. 0: A16 1: A24 2: A32 OTHERS: RESERVED	R/W	0x0
3	SLVW_AM_SA	Supervisory Access 1: Supervisory access to this window Is enabled 0: Supervisory access to this window Is not enabled	R/W	0x0
2	SLVW_AM_NPA	Non-Privileged Access 1: Non-privileged access to this window Is enabled 0: Non-privileged access to this window Is not enabled	R/W	0x0
1	SLVW_AM_PA	Program Access 1: Program access to this window Is enabled 0: Program access to this window Is not enabled	R/W	0x0
0	SLVW_AM_DA	Data Access 1: Data access to this window Is enabled 0: Data access to this window Is not enabled	R/W	0x0

4.2.6. Slave Access Address Decoder Compare Register (1-8): SLV_ACC_CMPn

This register is only available when the G_COMP_SLVW_DEC backwards compatibility mode is used! Backwards compatibility mode should not be used for new designs and will be discontinued in the future!

	VME Address Offset	Local Bus Address
Window 1	0x7F8A4	0x0A4
Window 2	0x7F894	0x094
Window 2	0x7F884	0x084
Window 3	0x7F874	0x074
Window 5	0x7F934	0x134
Window 6	0x7F924	0x124
Window 7	0x7F914	0x114
Window 8	0x7F904	0x104

Bits	Function	Description	R/W	Reset value
31:24	SLVW_ADER	Address Decoder Compare Register, bits 31:24	R/W	0x0
23:16		Address Decoder Compare Register, bits 23:16	R/W	0x0
15:8		Address Decoder Compare Register, bits 15:8	R/W	0x0
7:0	RESERVED	N/A	R	0x0

4.2.7. Slave Access Address Decoder Mask Register (1-8): SLV_ACC_MSKn

	VME Address Offset	Local Bus Address
Window 1	0x7F8A0	0x0A0
Window 2	0x7F890	0x090
Window 3	0x7F880	0x080
Window 4	0x7F870	0x070
Window 5	0x7F930	0x130
Window 6	0x7F920	0x120
Window 7	0x7F910	0x110
Window 8	0x7F900	0x100

Bits	Function	Description	R/W	Reset value
31:24	SLVW_ADEM	Address Decoder Mask Register, bits 31:24	R/W	0x0
23:16		Address Decoder Mask Register, bits 23:16	R/W	0x0
15:8		Address Decoder Mask Register, bits 15:8	R/W	0x0
7:0	RESERVED	N/A	R	0x0

4.2.8. DMA Status Register: DMA_STAT

VME Address Offset	Local Bus Address
0x7F86C	0x06C

Bits	Function	Description	R/W	Reset value
31:18	RESERVED	N/A	R	0x0
17:8	DMAS_BTCNT	DMA Beat Count This counter indicates which beat is being transmitted. Using this field, the user logic can determine where an error happened and resume data transmission from there.	R	0x0
7:4	RESERVED	N/A	R	0x0

Bits	Function	Description	R/W	Reset value
3:2	DMAS_ERRC	DMA Error Code 0: VME Bus Error Detected 1: VME Retry Detected 2: User Abort 3: Reserved	R	0x0
1	DMAS_ERR	DMA Error 1: DMA operation completed with error 0: Normal operation	R	0x0
0	DMAS_REQ	DMA Request Pending 1: A DMA request is pending 0: DMA operation is complete	R	0x0

4.2.9. DMA Command Register: DMA_CMD

VME Address Offset	Local Bus Address
0x7F868	0x068

Bits	Function	Description	R/W	Reset value
31:23	RESERVED	N/A	R	0x0
24	DMAC_FA	DMA Fixed Local Bus Address When data is transferred to or from a local FIFO, the local address shall remain constant between consecutive BLT or MBLT data transfers. 1: Local address remains constant 0: Local address is incremented with each successive transfer	R/W	0x0
23	DMAC_RAE	DMA Command Read Ahead Enable To accelerate data transfer from the local bus to the VME bus, local data can be pre-fetched 1: Enabled 0: Disabled	R/W	0x0
22	DMAC_PWE	DMA Command Posted Write Enable To accelerate data transfer from the VME bus to the local bus, a VME transaction can be terminated before the data write on the local bus side was completed. 1: Enabled 0: Not enabled	R/W	0x0

Bits	Function	Description	R/W	Reset value
21:19	DMAC_WIDTH	DMA Command WIDTH Defines the data width 0: D08(EO) 1: D16 2: Reserved 3: D32 4: D32-BLT 5: Reserved 6: D64-MBLT 7: Reserved	R/W	0x0
18:9	DMAC_SIZE	DMA Command SIZE This defines the size of a data transfer in beats. 0: 1 beat of data 1: 2 beats of data ... 1023: 1024 beats of data	R/W	0x0
8:3	DMAC_AM	DMA Command AM This is the VME address modifier used for this transfer.	R/W	0x0
2	DMAC_RWN	DMA Command RWN 1: This is VME read operation 0: This is a VME write operation	R/W	0x0
1	DMAC_ABORT	DMA Abort Request 1: Request a DMA abort 0: No action	R/W	0x0
0	DMAC_REQ	DMA Transfer Request 1: Start a DMA transfer 0: No action	R/W	0x0

4.2.10. DMA Local Address Register: DMA_LADDR

VME Address Offset	Local Bus Address
0x7F864	0x064

Bits	Function	Description	R/W	Reset value
31:0	DMA_LADDR	DMA Local Address [31:0] This register contains the local address. The core does not handle miss-aligned cycles or UAT cycles. Following limitations apply: – D16 transfers: DMA_LADDR[0] = 0 – D32 transfers: DMA_LADDR[1:0] = 0 – D64 transfers: DMA_LADDR[2:0] = 0	R/W	0x0

4.2.11. DMA VME Address Register: DMA_VADDR

VME Address Offset	Local Bus Address
0x7F860	0x060

Bits	Function	Description	R/W	Reset value
31:0	DMA_VADDR	DMA VME Address [31:0] This register contains the VME address. The core does not handle miss-aligned cycles or UAT cycles. Following limitations apply: – D16 transfers: DMA_VADDR[0] = 0 – D32 transfers: DMA_VADDR[1:0] = 0 – D64 transfers: DMA_VADDR[2:0] = 0	R/W	0x0

4.2.12. Mailbox Registers (1-4): MAILBOXn

	VME Address Offset	Local Bus Address
MBOX1:	0x7F85C	0x05C
MBOX2:	0x7F858	0x058
MBOX3:	0x7F854	0x054
MBOX4:	0x7F850	0x050

Bits	Function	Description	R/W	Reset value
31:0	MBOXn	Mailbox register n	R/W	0x0

– Writing to the mailbox register will set the respective irq_mbox[n] interrupt source. If the respective interrupt is enabled, a local bus interrupt is generated.

4.2.13. Semaphore Registers (0-3): SEMAPHORE

VME Address Offset	Local Bus Address
0x7F840	0x040

Bits	Function	Description	R/W	Reset value
31:24	SEMA3	Semaphore register 3 [31]: Semaphore bit 3 [30:24]: Semaphore tag 3	R/W	0x0

Bits	Function	Description	R/W	Reset value
23:16	SEMA2	Semaphore register 2 [23]: Semaphore bit 2 [22:16]: Semaphore tag 2	R/W	0x0
15:8	SEMA1	Semaphore register 1 [15]: Semaphore bit 1 [14:8]: Semaphore tag 1	R/W	0x0
7:0	SEMA0	Semaphore register 0 [7]: Semaphore bit 0 [6:0]: Semaphore tag 0	R/W	0x0

- A semaphore can be set when the semaphore bit is zero
- Writing to the semaphore register with the semaphore bit being set has not effect
- A semaphore can be cleared by writing zero to the semaphore bit

4.2.14. VME Interrupter Map: VME_INT_MAP

VME Address Offset	Local Bus Address
0x7F838	0x038

Bits	Function	Description	R/W	Reset value
Others	RESERVED	N/A	R	0x0
9:8	VINT_TYPE	Interrupter Type 0: D08(O) interrupter 1: D16 interrupter 2: D32 interrupter 3: N/A Interrupter types 1 and 2 are only available when using the advanced interrupter mode.	R/W	0x0
6:4	VINT_UIRQ	User Interrupt Map	R/W	0x0
2:0	VINT_SWIRQ	Software Interrupt Map	R/W	0x0

The interrupt map register is used to define which VME interrupt level and interrupter type is used for a particular interrupt request. To generate a VME IRQ4* interrupt, set the map register to 0x4. Setting the map register to 0x0 disables the particular interrupt source. If two interrupt requests on the same level exists, the software interrupt request will be served first.

A D08(O) interrupter responds do 8, 16 and 32 bit interrupt acknowledge cycles. A D16 interrupters responds to 16 and 32 bit acknowledge cycles. A D32 interrupter only responds to 32 bit interrupt acknowledge cycles.

4.2.15. VME Interrupter STATUS/ID: VME_INT_STAT

The functionality of this register depends on the interrupter setting G_INTERRUPTER. The simple interrupter is selected with G_INTERRUPTER = 0; the advanced interrupter with G_INTERUPTER = 1.

VME Address Offset	Local Bus Address
0x7F834	0x034

Bits	Function	Description	R/W	Reset value
Simple Interrupter				
31:8	RESERVED	N/A	R	0x0
7:1	VINT_STAT	VME Interrupt STATUS/ID Register Bit 7..1 of the STATUS/ID vector used during an IACK cycle.	R/W	0x0
0		VME Interrupt STATUS/ID Register, bit 0 Bit 0 of the STATUS/ID vector used during an IACK cycle. 0: Software IACK cycle 1: Hardware IACK cycle[3]	R	0x0
Advanced Interrupter				
31:0	VINT_STAT	VME Interrupt User STATUS/ID Register This is the user interrupt STATUS/ID vector returned during an IACK cycle. A D08(O) interrupter only drives bits 7:0, a D16 interrupt drives bits 15:0 while a D32 drives all bits.	R/W	0x0

4.2.16. VME Interrupter STATUS/ID: VME_INT_STAT_SW

This register is only available when using the advanced interrupter mode.

VME Address Offset	Local Bus Address
0x7F83C	0x03C

Bits	Function	Description	R/W	Reset value
31:0	VINT_STAT_SW	VME Interrupt Software STATUS/ID Register This is the software interrupt STATUS/ID vector returned during an IACK cycle. A D08(O) interrupter only drives bits 7:0, a D16 interrupt drives bits 15:0 while a D32 drives all bits.	R/W	0x0

3 The current implementation only supports one hardware iack, the user interrupt request.

4.2.17. VME Interrupter: VME_INT

VME Address Offset	Local Bus Address
0x7F830	0x030

Bits	Function	Description	R/W	Reset value
31:1	RESERVED	N/A	R	0x0
0	VINT_SWIRQ	VME Software Interrupt Request This register is used to create a software interrupt request. Write: 0: No action 1: Create software interrupt request Read: 0: No software interrupt is pending 1: Software interrupt is pending	R/W	0x0

A software interrupt request is automatically acknowledged during an VME IACK cycle or by writing 1 to the VIS_SWIRQ flag.

4.2.18. VME IRQn Status/ID: VME_IRQn_STAT

	VME Address Offset	Local Bus Address
VME_IRQ7_STAT	0x7F814	0x014
VME_IRQ6_STAT	0x7F818	0x018
VME_IRQ5_STAT	0x7F81C	0x01C
VME_IRQ4_STAT	0x7F820	0x020
VME_IRQ3_STAT	0x7F824	0x024
VME_IRQ2_STAT	0x7F828	0x028
VME_IRQ1_STAT	0x7F82C	0x02C

Bits	Function	Description	R/W	Reset value
		D08(O) cycle:		
31:9	RESERVED	N/A	R	0x0
8	VINTHn_ERR	Asserted when IACK cycle caused bus error	R	0x0
7:0	VINTHn_STAT	STATUS/ID fetched during the IACK cycle for level n	R	0x0
		D16 cycle:		
31:17	RESERVED	N/A	R	0x0
16	VINTHn_ERR	Asserted when IACK cycle caused bus error	R	0x0
15:0	VINTHn_STAT	STATUS/ID fetched during the IACK cycle for level n	R	0x0
		D32 cycle:		
31:0	VINTHn_STAT	STATUS/ID fetched during the IACK cycle for level n	R	0x0

The VINTHn_ERR flag is visible in the VME_INT_CMD register too.

4.2.19. VME Interrupt Handler Command: VME_INT_CMD

VME Address Offset	Local Bus Address
0x7F810	0x010

Bits	Function	Description	R/W	Reset value
31:10	RESERVED	N/A	R	0x0
9:8	VINTH_TYPE	Interrupt vector type This defines what type of interrupt status/ID is fetched during an IACK cycle. 0: D08(O) 1: D16 2: D32 3: Not valid	R/W	0x0
7	RESERVED	N/A	R	0x0
6	VINTH1_ERR	Error status for interrupt service level 1 0: IACK cycle was successful 1: IACK cycle was terminated with a bus error	R	0x0
5	VINTH2_ERR	Error status for interrupt service level 2 0: IACK cycle was successful 1: IACK cycle was terminated with a bus error	R	0x0
4	VINTH3_ERR	Error status for interrupt service level 3 0: IACK cycle was successful 1: IACK cycle was terminated with a bus error	R	0x0

Bits	Function	Description	R/W	Reset value
3	VINTH4_ERR	Error status for interrupt service level 4 0: IACK cycle was successful 1: IACK cycle was terminated with a bus error	R	0x0
2	VINTH5_ERR	Error status for interrupt service level 5 0: IACK cycle was successful 1: IACK cycle was terminated with a bus error	R	0x0
1	VINTH6_ERR	Error status for interrupt service level 6 0: IACK cycle was successful 1: IACK cycle was terminated with a bus error	R	0x0
0	VINTH7_ERR	Error status for interrupt service level 7 0: IACK cycle was successful 1: IACK cycle was terminated with a bus error	R	0x0

4.2.20. VME Interrupt Status Register: VINT_STATUS

VME Address Offset	Local Bus Address
0x7F80C	0x00C

Bits	Function	Description	R/W	Reset value
1	VIS_UIRQ	VME User Interrupt Request Status 1: User interrupt request is pending 0: User interrupt request is not pending	R/W	0x0
0	VIS_SWIRQ	VME Software Interrupt Request Status 1: Software interrupt request is pending 0: Software interrupt request is not pending	R/W	0x0

The VME interrupt status registers are accessed from the VME side to clear a pending interrupt request. VIS_SWIRQ is automatically cleared during a VME IACK cycle (ROAK) while VIS_UIRQ has to be cleared as part of the interrupt service routine (RORA). Since VIS_UIRQ is the result of an external interrupt request, the external interrupt source register needs to be cleared prior to clearing the VIS_UIRQ bit.

- If an interrupt source is pending, the respective interrupt status flag reads one.
- To clear an interrupt status flag, write a one to the respective bit to be cleared.
- The interrupt status bits are independent of the interrupt enable bits.

4.2.21. VME Interrupt Enable Register: VINT_EBL

VME Address Offset	Local Bus Address
0x7F808	0x008

Bits	Function	Description	R/W	Reset value
1	VIE_UIRQ	VME User Interrupt Request Enable 1: User interrupt request is enabled 0: User interrupt request is not enabled	R/W	0x0
0	VIE_SWIRQ	VME Software Interrupt Request Enable 1: Software interrupt request is enabled 0: Software interrupt request is not enabled	R/W	0x0

– The interrupt enable bits are used to generate the VME interrupt request

4.2.22. Interrupt Status Register: INT_STATUS

VME Address Offset	Local Bus Address
0x7F804	0x004

Bits	Function	Description	R/W	Reset value
17	IS_MBOX3	Mailbox 3 Interrupt Status A VME write to mailbox 3 was detected.	R/W	0x0
16	IS_MBOX2	Mailbox 2 Interrupt Status A VME write to mailbox 2 was detected.	R/W	0x0
15	IS_MBOX1	Mailbox 1 Interrupt Status A VME write to mailbox 1 was detected.	R/W	0x0
14	IS_MBOX0	Mailbox 0 Interrupt Status A VME write to mailbox 0 was detected.	R/W	0x0
13	IS_VTIMER	VME Arbiter Timer Error Status A VME arbiter timeout error happened.	R/W	0x0
12	IS_VBERR	VME Bus Error Interrupt Status The bus timer expired or the VME BERR was asserted to terminate the current cycle.	R/W	0x0
11	IS_DMAERR	DMA Error Interrupt Status When set, a DMA operation was aborted due to an error.	R/W	0x0
10	IS_DMADONE	DMA Done Interrupt Status The requested DMA operation completed successfully.	R/W	0x0

Bits	Function	Description	R/W	Reset value
9	IS_SWIACK	VME Software Interrupt Acknowledge Status This bit is set when the software IRQ from the VME interrupter has been served.	R/W	0x0
8	IS_IRQ1	VME Interrupt Request 1 Status This bit is set when the VME interrupt level 1 was served.	R/W	0x0
7	IS_IRQ2	VME Interrupt Request 2 Status This bit is set when the VME interrupt level 2 was served.	R/W	0x0
6	IS_IRQ3	VME Interrupt Request 3 Status This bit is set when the VME interrupt level 3 was served.	R/W	0x0
5	IS_IRQ4	VME Interrupt Request 4 Status This bit is set when the VME interrupt level 4 was served.	R/W	0x0
4	IS_IRQ5	VME Interrupt Request 5 Status This bit is set when the VME interrupt level 5 was served.	R/W	0x0
3	IS_IRQ6	VME Interrupt Request 6 Status This bit is set when the VME interrupt level 6 was served.	R/W	0x0
2	IS_IRQ7	VME Interrupt Request 7 Status This bit is set when the VME interrupt level 7 was served.	R/W	0x0
1	IS_SYSFAIL	VME SYSFAIL Interrupt Status This bit is set if the VME SYSFAIL input is asserted.	R/W	0x0
0	IS_ACFAIL	VME ACFAIL Interrupt Status This bit is set if the VME ACFAIL input is asserted.	R/W	0x0

- If an interrupt source is pending, the respective interrupt status flag reads one.
- To clear an interrupt status flag, write a one to the respective bit to be cleared.
- The interrupt status bits are independent of the interrupt enable bits except for the IS_IRQn bits. IS_IRQn is only updated when the respective IE_IRQn register is set.

4.2.23. Interrupt Enable Register: INT_EBL

VME Address Offset	Local Bus Address
0x7F800	0x000

Bits	Function	Description	R/W	Reset value
17	IE_MBOX3	Mailbox 3 Interrupt Enable	R/W	0x0
16	IE_MBOX2	Mailbox 2 Interrupt Enable	R/W	0x0
15	IE_MBOX1	Mailbox 1 Interrupt Enable	R/W	0x0
14	IE_MBOX0	Mailbox 0 Interrupt Enable	R/W	0x0
13	IE_VTIMER	VME Arbiter Timer Error Enable	R/W	0x0

Bits	Function	Description	R/W	Reset value
12	IE_VBERR	VME Bus Error Interrupt Enable	R/W	0x0
11	IE_DMAERR	DMA Error Interrupt Enable	R/W	0x0
10	IE_DMADONE	DMA Done Interrupt Enable	R/W	0x0
9	IE_SWIACK	VME Software Interrupt Acknowledge Enable	R/W	0x0
8	IE_IRQ1	VME Interrupt Request 1 Enable 1: VME interrupt level 1 is served 0: VME interrupt level 1 is not served	R/W	0x0
7	IE_IRQ2	VME Interrupt Request 2 Enable 1: VME interrupt level 2 is served 0: VME interrupt level 2 is not served	R/W	0x0
6	IE_IRQ3	VME Interrupt Request 3 Enable 1: VME interrupt level 3 is served 0: VME interrupt level 3 is not served	R/W	0x0
5	IE_IRQ4	VME Interrupt Request 4 Enable 1: VME interrupt level 4 is served 0: VME interrupt level 4 is not served	R/W	0x0
4	IE_IRQ5	VME Interrupt Request 5 Enable 1: VME interrupt level 5 is served 0: VME interrupt level 5 is not served	R/W	0x0
3	IE_IRQ6	VME Interrupt Request 6 Enable 1: VME interrupt level 6 is served 0: VME interrupt level 6 is not served	R/W	0x0
2	IE_IRQ7	VME Interrupt Request 7 Enable 1: VME interrupt level 7 is served 0: VME interrupt level 7 is not served	R/W	0x0
1	IE_SYSFAIL	VME SYSFAIL Interrupt Enable	R/W	0x0
0	IE_ACFAIL	VME ACFAIL Interrupt Enable	R/W	0x0

- The interrupt enable bits are used to generate the external interrupt request to the CPU.
- To enable an interrupt, set its respective enable bit to one, set it to zero to disable it.
- When set, the IE_IRQn bits enable the interrupt handler for the respective interrupt level.

4.2.24. Function N Address Decoder Compare (ADER) Register

VME Address	Local Bus Address	Bits	Name	Description
0x7FFDC	0x79C	31:24	CSR_ADER8[7:0]	Function 8 Address Decoder Compare Register
0x7FFD8	0x798	31:24	CSR_ADER8[15:8]	
0x7FFD4	0x794	31:24	CSR_ADER8[23:16]	
0x7FFD0	0x793	31:24	CSR_ADER8[31:24]	
0x7FFCC	0x79C	31:24	CSR_ADER7[7:0]	Function 7 Address Decoder Compare Register

VME Address	Local Bus Address	Bits	Name	Description
0x7FFC8	0x798	31:24	CSR_ADER7[15:8]	
0x7FFC4	0x794	31:24	CSR_ADER7[23:16]	
0x7FFC0	0x793	31:24	CSR_ADER7[31:24]	
0x7FFBC	0x79C	31:24	CSR_ADER6[7:0]	Function 6 Address Decoder Compare Register
0x7FFB8	0x798	31:24	CSR_ADER6[15:8]	
0x7FFB4	0x794	31:24	CSR_ADER6[23:16]	
0x7FFB0	0x793	31:24	CSR_ADER6[31:24]	
0x7FFAC	0x79C	31:24	CSR_ADER5[7:0]	Function 5 Address Decoder Compare Register
0x7FFA8	0x798	31:24	CSR_ADER5[15:8]	
0x7FFA4	0x794	31:24	CSR_ADER5[23:16]	
0x7FFA0	0x793	31:24	CSR_ADER5[31:24]	
0x7FF9C	0x79C	31:24	CSR_ADER4[7:0]	Function 4 Address Decoder Compare Register
0x7FF98	0x798	31:24	CSR_ADER4[15:8]	
0x7FF94	0x794	31:24	CSR_ADER4[23:16]	
0x7FF90	0x793	31:24	CSR_ADER4[31:24]	
0x7FF8C	0x78C	31:24	CSR_ADER3[7:0]	Function 3 Address Decoder Compare Register
0x7FF88	0x788	31:24	CSR_ADER3[15:8]	
0x7FF84	0x784	31:24	CSR_ADER3[23:16]	
0x7FF80	0x783	31:24	CSR_ADER3[31:24]	
0x7FF7C	0x77C	31:24	CSR_ADER2[7:0]	Function 2 Address Decoder Compare Register
0x7FF78	0x778	31:24	CSR_ADER2[15:8]	
0x7FF74	0x774	31:24	CSR_ADER2[23:16]	
0x7FF70	0x773	31:24	CSR_ADER2[31:24]	
0x7FF6C	0x76C	31:24	CSR_ADER1[7:0]	Function 1 Address Decoder Compare Register
0x7FF68	0x768	31:24	CSR_ADER1[15:8]	
0x7FF64	0x764	31:24	CSR_ADER1[23:16]	
0x7FF60	0x763	31:24	CSR_ADER1[31:24]	
0x7FF60 - 0x7FF9C	0x763 - 0x79C	23:0	RESERVED	Reading back from these reserved bits will return zero.

Following table shows the functionality of the different CSR_ADER bits:

Bits	Function	Description	R/W	Reset value
31:8	C[31:8]	Address bus compare bits	R/W	–
7:2	AM[5:0]	Address modifier compare bits	R	0x0

Bits	Function	Description	R/W	Reset value
1	DFSR	Dynamic Function Size Read If dynamic function sizing is supported (SLVW_DFS = 1), this bit is used to read the address decoder mask (VSLW_ADEM) register using the CSR_ADER[31:8] bit locations. Write 1: CSR_ADER[31:8] latches VSLW_ADEM[31:8][4] 0: No action Read: 1: CSR_ADER[31:8] 0: CSR_ADER[31:8] If dynamic function sizing is not supported (SLVW_DFS = 0) writing to this register has no effect. Once VSLW_ADEM is latched into CSR_ADER and read, the register needs to restored to its original content.	R/W	0x0
0	XAM	XAM mode The VMESCmodule doesn't support A64 addressing mode. Setting this bit has no effect.	R	0x0

- The C and AM bits are read-only when the VMESCmodule is configured for fixed-address function (SLVW_FAF = 1). In this case CR_ADER must be configured before setting the SLVW_FAF bit!

4 The content of VSLW_ADEM[31:8] is stored in a shadow register that is displayed when reading form CSR_ADER while its DFSR bit is set. The content of CSR_ADER[31:8] is not modified.

4.2.25. User-Defined Bit Set Register: UDBIT_SET

VME Address Offset	Local Bus Address
CRBAR + 0x7FFFE8	0x7E8

Bits	Function	Description	R/W	Reset value
31	UBITSET.7	User bit set register 7 Write: 1: Assert user_bit_set_event[7] output for one clock cycle 0: No action Read: 1: user_bit_status[7] input is high 0: user_bit_status[7] input is low	R/W	–
30	UBITSET.6	User bit set register 6	R/W	–
29	UBITSET.5	User bit set register 5	R/W	–
28	UBITSET.4	User bit set register 4	R/W	–
27	UBITSET.3	User bit set register 3	R/W	–
26	UBITSET.2	User bit set register 2	R/W	–
25	UBITSET.1	User bit set register 1	R/W	–
24	UBITSET.0	User bit set register 0	R/W	–
23:0	RESERVED	N/A	R	0x0

4.2.26. User-Defined Bit Clear Register: UBIT_CLEAR

VME Address Offset	Local Bus Address
CRBAR + 0x7FFFEC	0x7EC

Bits	Function	Description	R/W	Reset value
31	UBITCLR.7	User bit clear register 7 Write: 1: Assert user_bit_clear_event[7] output for one clock cycle 0: No action Read: 1: user_bit_status[7] input is high 0: user_bit_status[7] input is low	R/W	–
30	UBITCLR.6	User bit clear register 6	R/W	–
29	UBITCLR.5	User bit clear register 5	R/W	–
28	UBITCLR.4	User bit clear register 4	R/W	–

Bits	Function	Description	R/W	Reset value
27	UBITCLR.3	User bit clear register 3	R/W	–
26	UBITCLR.2	User bit clear register 2	R/W	–
25	UBITCLR.1	User bit clear register 1	R/W	–
24	UBITCLR.0	User bit clear register 0	R/W	–
23:0	RESERVED	N/A	R	0x0

4.2.27. CRAM Owner Register:CRAM_OWNER

VME Address Offset	Local Bus Address
CRBAR + 0x7FFFF0	0x7F0

Bits	Function	Description	R/W	Reset value
31:24	CRAM_OWN	CRAM Owner Register This register implements a simple semaphore that prevents a user-write while it contains a non-zero value.	R/W	0x0
23:0	RESERVED	N/A	R	0x0

- The CRAM_OWNER register can only be set when it is zero
- Writing to this register while it contains a non-zero value has no effect
- This register is cleared by setting the BIT_CLEAR.CRAMOC flag

4.2.28. CSR Bit Clear Register: BIT_CLEAR

VME Address Offset	Local Bus Address
CRBAR + 0x7FFFF4	0x7F4

Bits	Function	Description	R/W	Reset value
31	LRSTC	Local Reset Clear[5] Write: 1: Releases user_reset_n 0: No action Read: 1: user_reset_n is asserted 0: user_reset_n is not asserted	R/W	0x0

5 Minimum assertion time for user_reset_n is 200ms.

Bits	Function	Description	R/W	Reset value
30	SDEC	SYSFAIL Driver Enable Clear Write: 1: Releases VSYSFAIL_O_N 0: No action Read: 1: VSYSFAIL_O_N is asserted 0: VSYSFAIL_O_N is not asserted	R/W	0x0
29	RESERVED	N/A	R	0x0
28	MODEBLC	Module Enable Set Write: 1: Disable slave window access 0: No action Read: 1: Module enabled 0: Module not enabled	R/W	0x0
27	BERRSC	Bus Error Status Clear 1: Clears the bus error status bit (user_reset_n=1) 0: No action	R/W	0x0
26	CRAMOC	CRAM Owner Clear Write: 1: Clear CRAM owned flag 0: No action Read: 1: CRAM owned 0: CRAM available When the CRAM_OWNER register contains a non-zero value, the CRAM is 'owned' and reading this register returns 1. Writing 1 to this register clears the CRAM_OWNER register.	R/W	0x0
25:0	RESERVED	N/A	R	0x0

Note:

- Reading back will return the current state and not the last written value!
- BERRSC: This bit is set when the VMESCmodule drives the VME BERR*. Writing a one clears the Bus Error Status Bit.

4.2.29. CSR Bit Set Register: BIT_SET

VME Address Offset	Local Bus Address
CRBAR + 0x7FFF8	0x7F8

Bits	Function	Description	R/W	Reset value
31	LRSTS	Local Reset Set[2] Write: 1: Asserts user_reset_n=0 0: No action Read: 1: user_reset_n is asserted 0: user_reset_n is not asserted	R/W	0x0
30	SDES	SYSFAIL Driver Enable Set Write: 1: Asserts VSYSFAIL_O_N 0: No action Read: 1: VSYSFAIL_O_N is asserted 0: VSYSFAIL_O_N is not asserted	R/W	0x0
29	RESERVED	N/A	R	0x0
28	MODEBLS	Module Enable Set Write: 1: Enable slave window access 0: No action Read: 1: Module enabled 0: Module not enabled	R/W	0x0
27	BERRSS	Bus Error Status Set 1: Sets the bus error status bit 0: No action	R/W	0x0
26	CRAMOS	CRAM Owner Set Read: 1: CRAM owned 0: CRAM available When the CRAM_OWNER register contains a non-zero value, the CRAM is 'owned' and reading this register returns 1. This register is read-only.	R	0x0
25:0	RESERVED	N/A	R	0x0

Note:

- Reading back will return the current state and not the last written value!
- BERRSS: This bit is set when the VMESCmodule drives the VME BERR*. Writing a one

sets the Bus Error Status Bit.

4.2.30. CSR Base Register: CRBAR

VME Address Offset	Local Bus Address
CRBAR + 0x7FFFC	0x7FC

Bits	Function	Description	R/W	Reset value
31:27	CRBAR	CR/CSR Base Address	R/W	-
26:0	RESERVED	N/A	R	0x0

Note:

- The CRBAR register is initialized based upon the value read from the geographical address pins (VGA_N). This value can be changed by software.
- CRBAR is compared with VA[23:19] to determine if the board is selected by the current VME transfer.

About Inicore

- FPGA and ASIC Design
- Easy-to-use IP Cores
- System-on-Chip Solutions
- Consulting Services
- ASIC to FPGA Migration
- Obsolete ASIC Replacements

Inicore is an experienced system design house providing FPGA / ASIC and SoC design services. The company's expertise in architecture, intellectual property, methodology and tool handling provides a complete design environment that helps customers shorten their design cycle and speed time to market. Our offering covers feasibility study, concept analysis, architecture definition, code generation and implementation. When ready, we deliver you a FPGA or take your design to an ASIC provider, whatever is more suitable for your unique solution.

Customer Advantages

We offer one-stop shopping for everything from the specifications to the chip or module solution. Our experience and fast turnaround time reduces your development costs and increases your returns from the market. Your system is not limited by the level of expertise and standard chip solutions you happen to have in-house. Achieve market success by differentiating and optimizing your product. Reusability builds the basis for further developments in the ever-decreasing product life cycle.

Visit us @ www.inicore.com