DWMVA VME Adapter Technical Manual

Order Number: EK–DWMVA–TM–001

This manual is for software developers who write drivers and application programs for the DWMVA adapter in VAX 6000 systems.

Digital Equipment Corporation

First Printing, August 1991

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	PDP	VAXcluster
DEC LANcontroller	ULTRIX	VAXELN
DECnet	UNIBUS	VMS
DECUS	VAX	XMI
DWMVA	VAXBI	digital™

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

FIGURES

TABLES

Preface

This manual presents a detailed technical description of the DWMVA I/O adapter that connects a VMEbus to the XMI bus of a VAX 6000 computer system. It provides complete discussions of the hardware operations of the component modules and bit-level functional descriptions of all adapter registers.

Audience

This manual is for software developers who write driver and application programs for the DWMVA I/O adapter.

Document Structure

The manual consists of eight chapters and two appendixes.

- **Chapter 1, Overview**, describes the major components of the DWMVA adapter, provides a summary of the buses, and introduces the types of DWMVA transactions.
- **Chapter 2, Address Mapping**, discusses mapping of VME addresses to XMI address space. The chapter explains how an XMI address is translated to a VME address in a CPU transaction, and how a VME address is translated to an XMI address in a DMA transaction.
- **Chapter 3, VME System Control**, describes the components and functions of the VME system controller, including the VME arbitration subsystem.
- **Chapter 4, Transactions**, discusses the two types of transactions, CPU and DMA, processed by the DWMVA. CPU transactions are initiated by the CPU and perform reads and writes on VME devices, while DMA transactions are initiated by a VME device and perform reads and writes to XMI memory. This chapter also discusses the translation of commands over the XMI to the VMEbus and from the VMEbus to the XMI data paths.
- **Chapter 5, VMEbus Interface**, discusses the substructures of the VMEbus and describes the VMEbus signals.
- **Chapter 6, Interrupts**, discusses the VME-to-XMI interrupt protocol, the interrupt request levels, and interrupt handler selection.
- **Chapter 7, Registers**, provides bit-level descriptions of functions performed by the register sets on the two modules of the DWMVA adapter.
- **Chapter 8, Initialization**, discusses the various methods used to initialize the DWMVA adapter.
- **Appendix A, VME Interface Signal List**, gives the pin assignments on the two connectors of the VME interface.

- **Appendix B, VME-to-XMI Byte Swapping**, discusses how the DWMVA implements byte swapping to allow VME data to appear correctly on the XMI bus and XMI data to appear correctly on the VMEbus.

A **Glossary** provides additional reference support.

Associated Documents

Other documents related to the DWMVA adapter include:

- *DWMVA VME Adapter Installation Guide*, EK–DWMVA–IN

  Describes the installation of the DWMVA I/O adapter in a VAX 6000 computer system.

- *BA62 VME Enclosure*, EK–VME01–IN

  Describes the BA62 enclosure that is used as an expansion cabinet to provide a VMEbus backplane and to house the C3200 module.

- *IEEE Standard for a Versatile Backplane Bus: VMEbus*, IEEE Std 1014, 1987.

  Provides complete specifications for the VMEbus.

- *VMS Version 5.4-3 Release Notes*, AA–PHUFA–TE

  Includes a chapter (Open Bus Driver Support Features) that discusses VMS support for VMEbus devices.

- *VMS Device Support Manual*, AA-PBPWA-TE

  Describes the components of a VMS device driver and the basic rules that device drivers must observe.

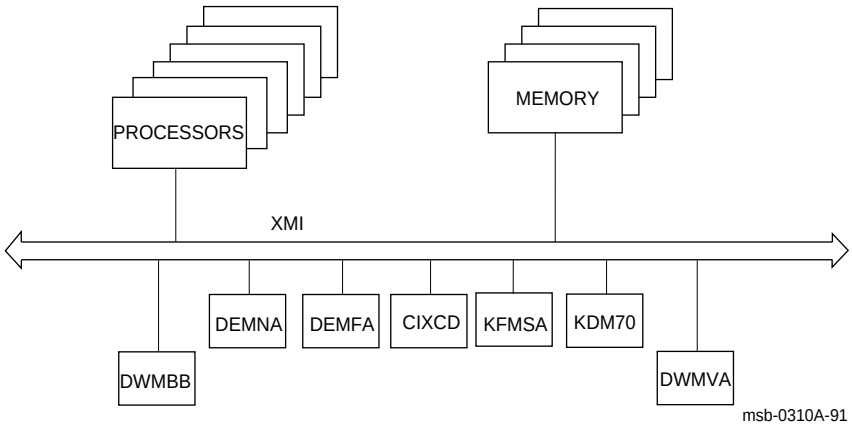
- *VMS Device Support Reference Manual*, AA-PBPXA-TE

  Describes driver data structures, routines, and entry points.

1 Overview

The DWMVA adapter connects to the I/O segment of the VAX 6000 XMI bus and interfaces the synchronous XMI bus to the VMEbus, an asynchronous industry-standard bus. The DWMVA implements the handshaking protocol and acts as a channel for data flow between the two buses. Figure 1–1 is a block diagram showing the DWMVA adapter on the XMI bus.

Figure 1–1 DWMVA Adapter on the XMI Bus



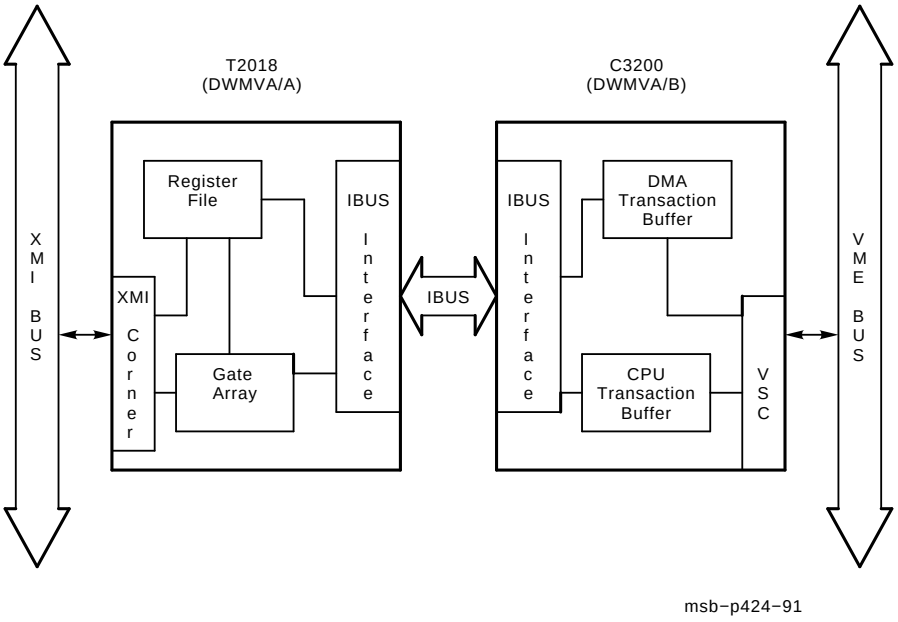
1.1 Major Components

The DWMVA subsystem consists of two modules:

- T2018 (DWMVA/A)
- C3200 (DWMVA/B)

The T2018 module is on the XMI bus and the C3200 module is on the VMEbus. The two modules are connected through the IBUS, which is a physical path between the system XMI bus and the VMEbus. Figure 1–2 shows a block diagram of the DWMVA adapter.

Figure 1–2 DWMVA Block Diagram



The T2018 module contains the XMI and IBUS interfaces, registers, a data buffer, and state machines for transmitting and receiving data. The XMI Corner is a circuit area on all XMI nodes that provides the interface to the XMI bus. This corner handles the distribution of the XMI clock, control, and data lines to the T2018 module. The T2018 allows access to XMI addresses from VME through on-board 64K page map registers.

The C3200 module contains the following functional blocks: VME and IBUS interfaces, control logic, registers, data buffers, the VME interrupt handler, and the VME system controller (VSC).

1.2 Major Buses

Three major buses are used to exchange information between the host computer system and an I/O device connected to the VMEbus:

- XMI bus
- VMEbus
- IBUS

The T2018 uses the XMI bus to communicate with the processor. The C3200 communicates with a VME device through the VMEbus. The IBUS provides communication between the T2018 module and the C3200 module.

1.2.1 XMI Bus

The XMI is a 64-bit wide, pended, synchronous bus that can process multiple read requests at any given time. It has a cycle time of 64 ns, allowing an effective bandwidth on the bus of 100 Mbytes/second. The XMI protocol supports quadword, octaword, and hexword reads and writes to XMI memory space. The DWMVA, however, allows only quadword and octaword transactions to XMI memory. The DWMVA accepts only longword transactions to its address space.

1.2.2 VMEbus

The VMEbus is an asynchronous, interlocked bus that processes one transaction at a time. The VME protocol, defined by IEEE 1014, consists of four subbuses: the data transfer bus, the arbitration bus, the priority interrupt bus, and the utility bus. The VME supports 1-, 2-, 3-, and 4-byte transfers as well as block transactions consisting of multiple 1-, 2-, or 4-byte transfers over the data transfer bus, a nonmultiplexed data/address path. The VME has an effective bandwidth of 40 Mbytes/second.

Chapter 5 provides an overview of the VMEbus. For a more complete treatment of the VMEbus, refer to the IEEE VMEbus specification. (See, for example, *VMEbus*, A standard specification for a versatile backplane bus, IEEE Computer Society Publication P1014, March 1987.)

1.2.3 IBUS

The IBUS is the communications path between the two modules of the DWMVA. The IBUS data path consists of a 4-bit function field, IB I<3:0>, and a 32-bit, multiplexed address/data field, IB D<31:0>. The IBUS can transfer address or data every 200 ns, yielding an effective bandwidth of 16 Mbytes/second.

In addition to the bidirectional lines (address and data), the IBUS includes many lines to carry control signals. The signals driven from the T2018 to the C3200 are used to indicate the status of the T2018's buffers, while the signals from the C3200 to the T2018 are used to control the operation of the IBUS.

1.3 Transactions

The DWMVA conducts two types of transactions:

- CPU transactions
- DMA transactions

A CPU transaction is initiated by a processor on the XMI bus. The processor is the commander. The DWMVA becomes the responder. A DMA transaction begins on the VMEbus and targets XMI memory through the DWMVA adapter. Transactions are discussed in Chapter 4.

1.4 Interrupts

The DWMVA accepts longword-aligned VME interrupts and generates XMI INTR transactions in response to them. The DWMVA (or any other XMI device) does not issue interrupts to the VMEbus. Interrupts are discussed in Chapter 6.

2 Address Mapping

This chapter discusses the XMI I/O address space and explains how the VME address space is mapped to the XMI I/O adapter space.

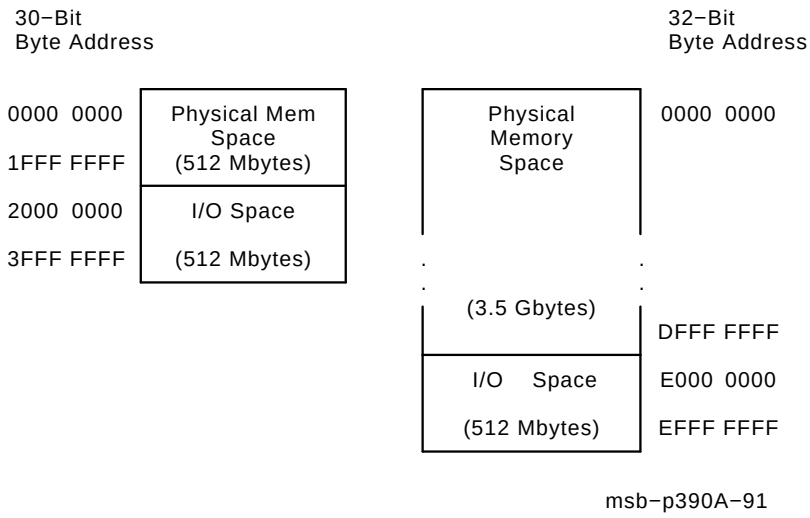
The XMI supports one terabyte (2^{40}) of address space, accessible with 40-bit addresses. Since a VAX 6000 series system supports 30- or 32-bit addresses, the maximum space available to a single system on the XMI is (2^{32}) bytes, which is 4 gigabytes.

The VAX 6000 series systems use one of three addressing modes depending on the model and the environment.

- 30-bit addressing (Models 200 through 500)
- 32-bit addressing (Model 500 and above)
- 30-bit addressing in a 32-bit environment (Model 500 and above)

Figure 2–1 shows how memory and I/O space are divided in the 30-bit and the 32-bit addressing modes.

Figure 2–1 XMI Memory and I/O Address Space



2.1 XMI Memory Space

Memory address space is the lower part of the address space no matter which address mode, 30-bit or 32-bit, is used. A VAX 6000 system using 30-bit addressing cannot access the 3 Gbytes of memory space between address 2000 0000 (hex) and DFFF FFFF.

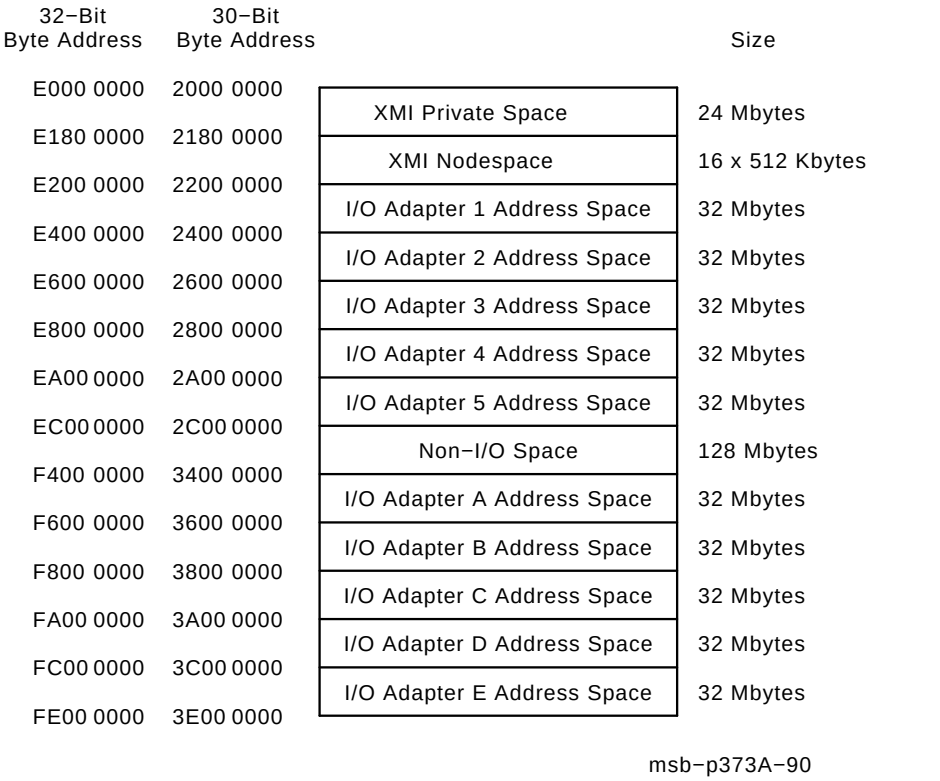
2.2 XMI I/O Space

The maximum amount of I/O space available for a VAX 6000 is 512 Mbytes regardless of the addressing mode. The I/O space is divided into three sections:

- Private space
- Nodespace
- I/O adapter space

The I/O space is allocated as shown in Figure 2–2.

Figure 2–2 XMI I/O Space Address Allocation



2.2.1 Private Space

The XMI private space is a 24-Mbyte address region located from E000 0000 to E17F FFFF (32-bit address) or from 2000 0000 to 217F FFFF (30-bit address). References to XMI private space are serviced by resources local to a node, such as local device CSRs and boot ROM. The references are not broadcast on the XMI.

2.2.2 Nodespace

The VAX 6000 platform XMI nodespace is a collection of sixteen 512-Kbyte regions located from E180 0000 to E1FF FFFF (32-bit address) or from 2180 0000 to 21FF FFFF (30-bit address). Each XMI node is allocated one of the fourteen 512-Kbyte regions for its control and status registers (nodes 0 and F are not implemented). The starting address of the 512-Kbyte region associated with a given node (BB) is computed as follows:

BB = E180 0000 + Node ID * 8 0000 (32-bit address)

BB = 2180 0000 + Node ID * 8 0000 (30-bit address)

Table 2–1 gives the address ranges of the 14 XMI nodespace regions implemented on the VAX 6000 series.

Table 2–1 XMI Nodespace Addresses

Slot	Node	Nodespace	I/O Window Space (DWMVA)
1	1	E188 0000 – E18F FFFF[1]	E200 0000 – E3FF FFFF
2	2	E190 0000 – E197 FFFF	E400 0000 – E5FF FFFF
3	3	E198 0000 – E19F FFFF	E600 0000 – E7FF FFFF
4	4	E1A0 0000 – E1A7 FFFF	E800 0000 – E9FF FFFF
5[2]	5	E1A8 0000 – E1AF FFFF	EA00 0000 – EBFF FFFF
6	6	E1B0 0000 – E1B7 FFFF	N/A[3]
7	7	E1B8 0000 – E1BF FFFF	N/A[3]
8	8	E1C0 0000 – E1C7 FFFF	N/A[3]
9	9	E1C8 0000 – E1CF FFFF	N/A[3]
10[2]	A	E1D0 0000 – E1D7 FFFF	F400 0000 – F5FF FFFF
11	B	E1D8 0000 – E1DF FFFF	F600 0000 – F7FF FFFF
12	C	E1E0 0000 – E1E7 FFFF	F800 0000 – F9FF FFFF
13	D	E1E8 0000 – E1EF FFFF	FA00 0000 – FBFF FFFF
14	E	E1F0 0000 – E1F7 FFFF	FC00 0000 – FDFF FFFF

[1]32-bit addresses are converted to 30-bit addresses by changing the most significant byte from E to 2 and from F to 3.

[2]These slots cannot be used on VAX 6000 Models 200, 300, and 400 due to processor restrictions.

[3]Slots in the center of the XMI card cage have no I/O connectors because of the daughter card's presence.

2.2.3 I/O Adapter Address Space

The XMI I/O adapter address space consists of ten 32-Mbyte address regions (windows) used to access I/O devices. The I/O adapter address space accessed by the DWMVA is determined by the XMI slot in which it is installed. Table 2–1 also shows the I/O window space for each XMI adapter. All 4 gigabytes of the VME address space are accessible from a 32-Mbyte I/O window space.

The DWMVA accepts only longword-length references to its XMI adapter address space. These references are then translated to their corresponding VME transactions or, internally, as DWMVA register transactions.

2.3 VME Address Space

The VMEbus supports 4 gigabytes (2^{32}) of address space. Unlike the XMI, the VME address space is not divided into memory and I/O spaces. To address a byte in this space, VME data transfer bus lines DS0*, DS1*, and LWORD* are used in conjunction with the VME address lines A01–A31.

The VMEbus allows devices of different address widths to coexist on the bus at any given time. The address width can be 16 bits, 24 bits, or 32 bits. The master indicates the nature of the current address by asserting an appropriate value on the VME address modifier lines (see Section 5.1.1). Table 2–2 shows the address space accessible with each address mode.

Table 2–2 VME Address Modes

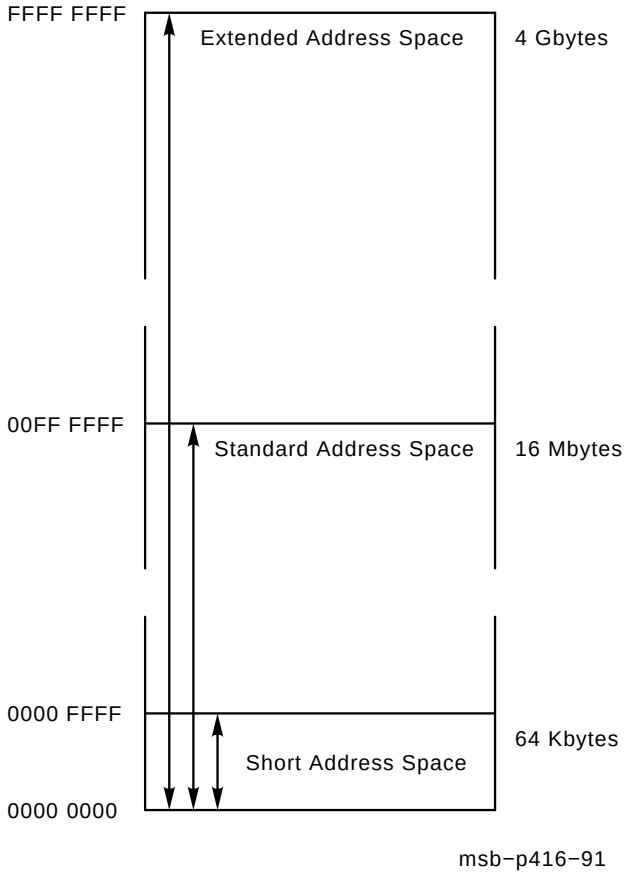
Address Mode	Address Width	Accessible VME Space
Extended	32 bits	4 Gbytes
Standard	24 bits	16 Mbytes
Short	16 bits	64 Kbytes

Figure 2–3 shows the VME address map.

NOTE: In VME-initiated (DMA) transactions, the DWMVA makes a distinction between addresses it will accept and addresses it will not by using the VME Address Range Enable Register (see VESR in Chapter 7). If enabled, the DWMVA accepts any extended VME address with VME address bits A29–A31 = 000. The DWMVA can also be configured to accept standard VME addresses with VME address bit A23 = 0. The DWMVA does not support short address DMA transactions.

Since the VMEbus and the XMI use different addressing schemes, address translation is required to move data from one bus to the other. The DWMVA translates XMI addresses to VME addresses in CPU transactions, when data is moved from the XMI bus to the VMEbus. Conversely, the DWMVA translates VME addresses to XMI addresses in DMA transactions, when data is moved from the VMEbus to the XMI.

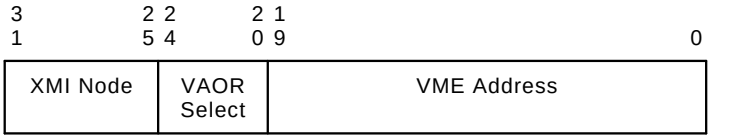
Figure 2–3 VME Address Map



2.4 Address Translation in CPU Transactions

In a CPU transaction, the 4-Gbyte VME address space is mapped to the 32-Mbyte XMI adapter space by decoding the 32-bit CPU transaction command as shown in Figure 2–4 and Figure 2–5.

Figure 2–4 CPU Transaction Command Format



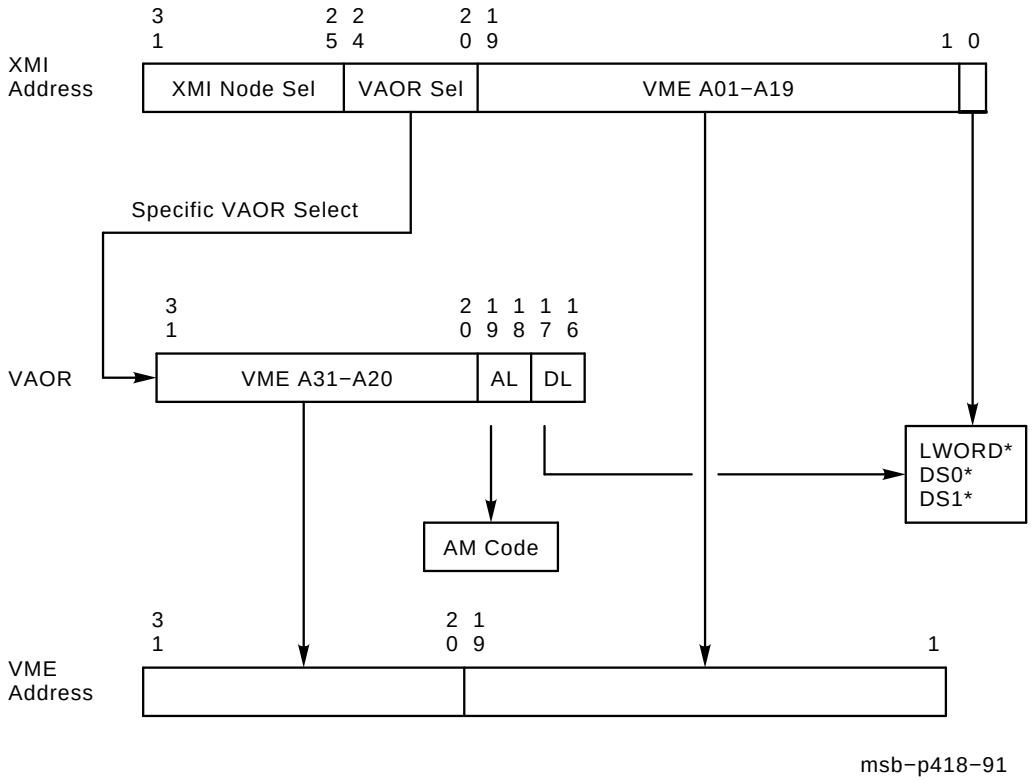
The fields of the CPU transaction command are described in Table 2–3.

Table 2–3 CPU Transaction Command

Bit Field	Description
<31:25>	XMI Node—This field is used to access a 32-Mbyte region of DWMVA adapter space on the XMI. Each XMI node responds to a unique value in this field.
<24:20>	VAOR Select—The VAOR Select field selects one of the 32 CPU Address Offset Registers that is used to supply the upper 12 address bits, address length, and data length information for the CPU transaction. This field selects the offset value that is appended to the VME address field (bits <19:0>) of the CPU transaction address to generate the corresponding VME address. See Chapter 7 for information on the CPU Address Offset Registers.
<19:0>	VME Address—This field contains the lower 20 bits of the VME address for the CPU transaction.

Figure 2–5 shows how VME addresses are generated from XMI addresses. The XMI address (CPU transaction address, shown at the top) provides the lower 20 bits of the VME address. The other bits of the VME address as well as the address length and data length information for the transaction are provided by the appropriate fields of the CPU Transaction Address Offset Register (see Chapter 7) determined by the VAOR Select field. This address generation scheme allows access to any 32-Mbyte VME address region through the 32-Mbyte XMI window. Each VME address region consists of 32 1-Mbyte sections and is selected by one of the 32 values provided by the VAOR Select field.

Figure 2–5 Building VME Addresses



2.5 Address Translation in DMA Transactions

This section discusses translation of VME addresses into XMI physical (VAX) addresses in DMA read/write transactions. The DWMVA implements five modes of VAX address translation:

- No address translation
- 34-bit VAX address translation
- 40-bit VAX address translation
- 40-bit VAX address translation using 4-Kbyte page size
- 40-bit VAX address translation using 8-Kbyte page size

The DWMVA defaults to no address translation mode at power-up or node reset. The address translation mode is selected at system initialization by loading the Mapping Register Mode Enable field (bits<19:17> of the T2018 Utility Register) with the appropriate configuration.

NOTE: Normally, the VMS operating system uses the 34-bit address translation mode.

2.5.1 No Address Translation

In no address translation mode the XMI physical address is identical to the VME address. The upper address bits of the extended XMI address format, XMI A<39:29>, are forced to zero. The steps used to generate a VAX address from a VME address using no translation mode are as follows:

1 Check Upper Address Bit

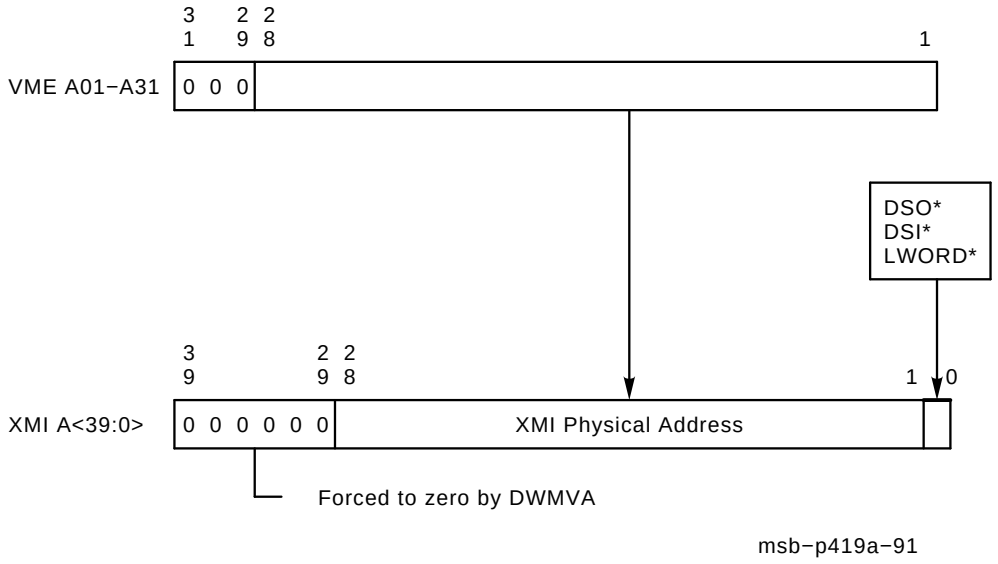
VME A29–A31 must be zero.

2 Generate XMI Address

Load zeros into XMI A<39:29>.
Load VME A0–A28 into XMI A<28:0>.

Figure 2–6 shows the 29-bit VAX address generation in no translation mode.

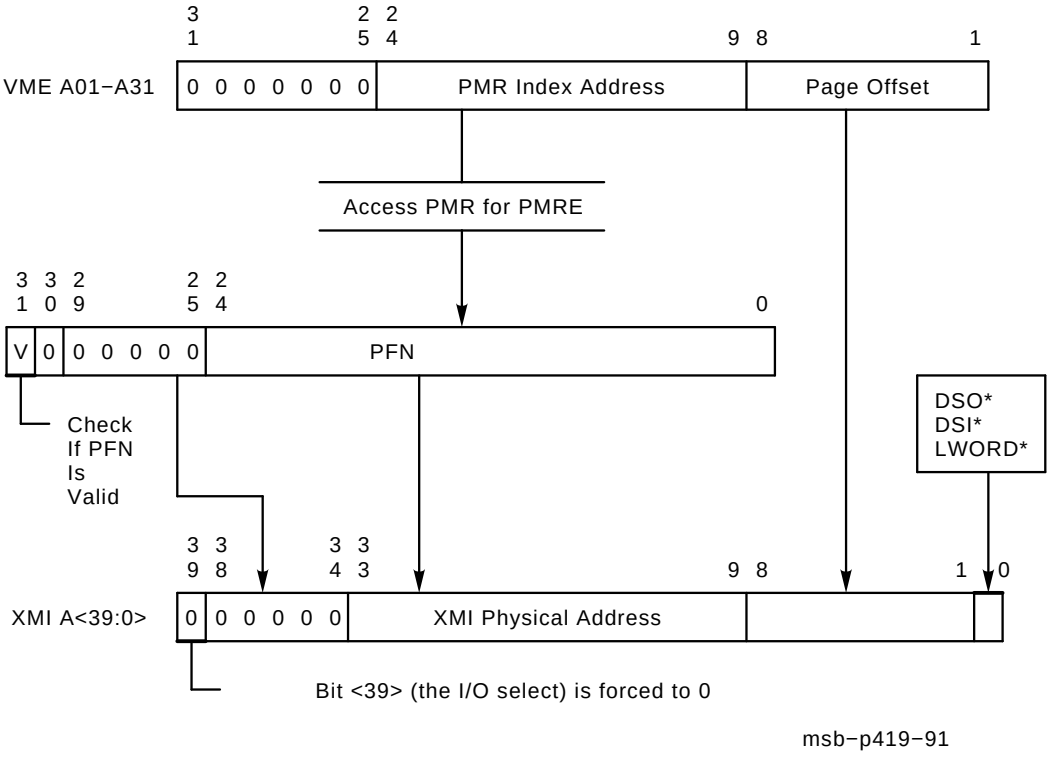
Figure 2–6 No Translation Mode VAX Address



2.5.2 34-Bit VAX Address Translation

In 34-bit VAX address translation mode (see Figure 2–7), the DWMVA can map only the first 32 Mbytes of VME memory address space to XMI memory address space. Since the page size is 512 bytes, this is the maximum range that can be mapped with 64K page map register (PMR) entries.

Figure 2–7 34-Bit VAX Address Translation



The translation of a VME DMA address to a 34-bit XMI address uses VME address bits VME A09–A24 as an index into the PMRs. These bits select the page map register entry (PMRE) that contains the required VAX page frame number (PFN). Because in this mode the DWMVA only maps the first 32 Mbytes of VME memory address space, the upper bits of the VME address, VME A25–A31, must be zero. The validity of the PFN is checked and if good the PFN is used to complete the DMA address translation. The 34-bit physical address is obtained by combining the PFN field of the PMRE (PMRE<24:0>) with VME address bits VME A0–A08. The unused upper address bits (XMI A<39:34>) are forced to zero. The steps used for 34-bit address translation are as follows:

1 Check Upper Address Bits

VME A25–A31 must all be zero.

2 Access PMR for PMRE

VME address VME A09–A24 used as an index into the PMR to fetch the PMRE.

3 Check PMRE Valid Bit

If PMRE<31> = 1, then PFN is valid.
If PMRE<31> = 0, then PFN is invalid and transaction is aborted.

4 ECC Check

If no error or correctable error, then PFN is good.
If uncorrectable error, then PFN is bad and the transaction is aborted.

5 Generate XMI Address

Load zeros into XMI A<39:34>.
Load PMRE<24:0> into XMI A<33:9>.
Load VME A0–A08 into XMI A<8:0>.

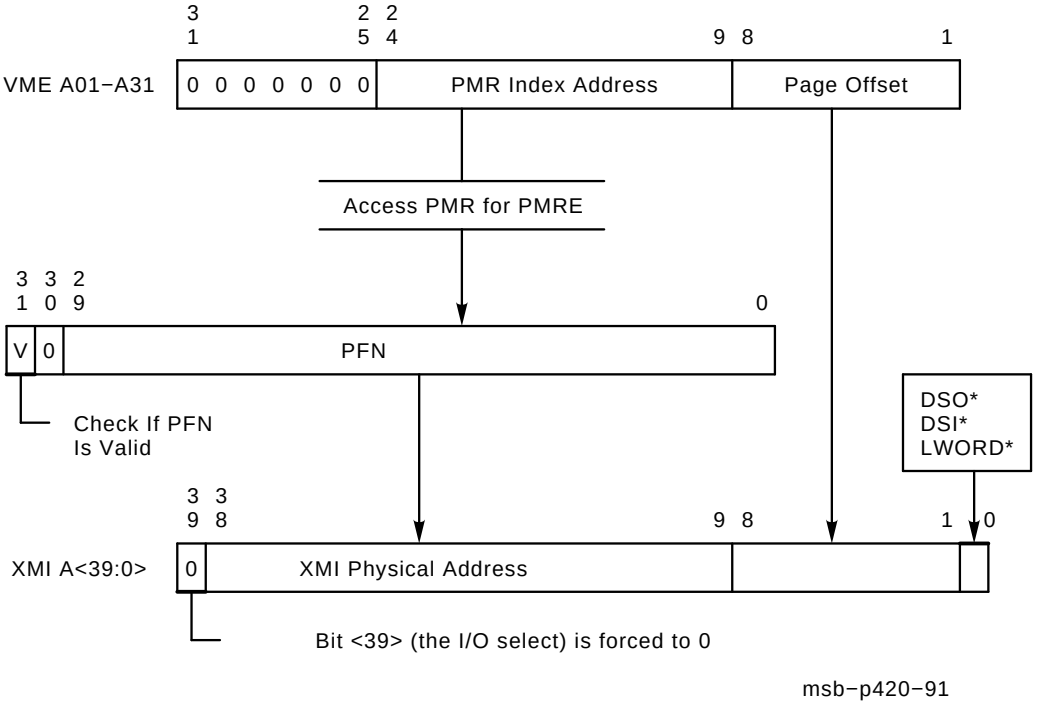
2.5.3 40-Bit VAX Address Translation

The 40-bit VAX address translation mode uses three different page sizes: 512 bytes, 4 Kbytes, and 8 Kbytes. The mapped address range depends on the selected page size.

2.5.3.1 512-Byte Page Size

When using a 512-byte page size in the 40-bit VAX address translation mode, the DWMVA maps only the first 32 Mbytes of VME memory address space to XMI memory address space (see Figure 2–8). This is the maximum range that can be mapped with 64K PMR entries.

Figure 2–8 40-Bit VAX Address Translation Using 512-Byte Page Size



The translation of a VME DMA address to a 40-bit XMI address uses VME address bits VME A09–A24 as an index into the PMRs. These bits select the PMRE that contains the required PFN. Because in this mode the DWMVA only maps the first 32 Mbytes of VME memory address space, the upper address bits of the VME address, VME A25–A31, must be zero. The validity of the PFN is checked and if good the PFN is used to complete the DMA address translation. The 40-bit physical address is obtained by combining the PFN field of the PMRE (PMRE<29:0>) with VME address bits VME A0–A08. The steps used for 40-bit address translation are as follows:

1 Check Upper Address Bits

VME A25–A31 must all be zero.

2 Access PMR for PMRE

VME address VME A09–A24 used as an index into the PMR to fetch the PMRE.

3 Check PMRE Valid Bit

If PMRE<31> = 1, then PFN is valid.
If PMRE<31> = 0, then PFN is invalid and transaction is aborted.

4 ECC Check

If no error or correctable error, then PFN is good.
If uncorrectable error, then PFN is bad and the transaction is aborted.

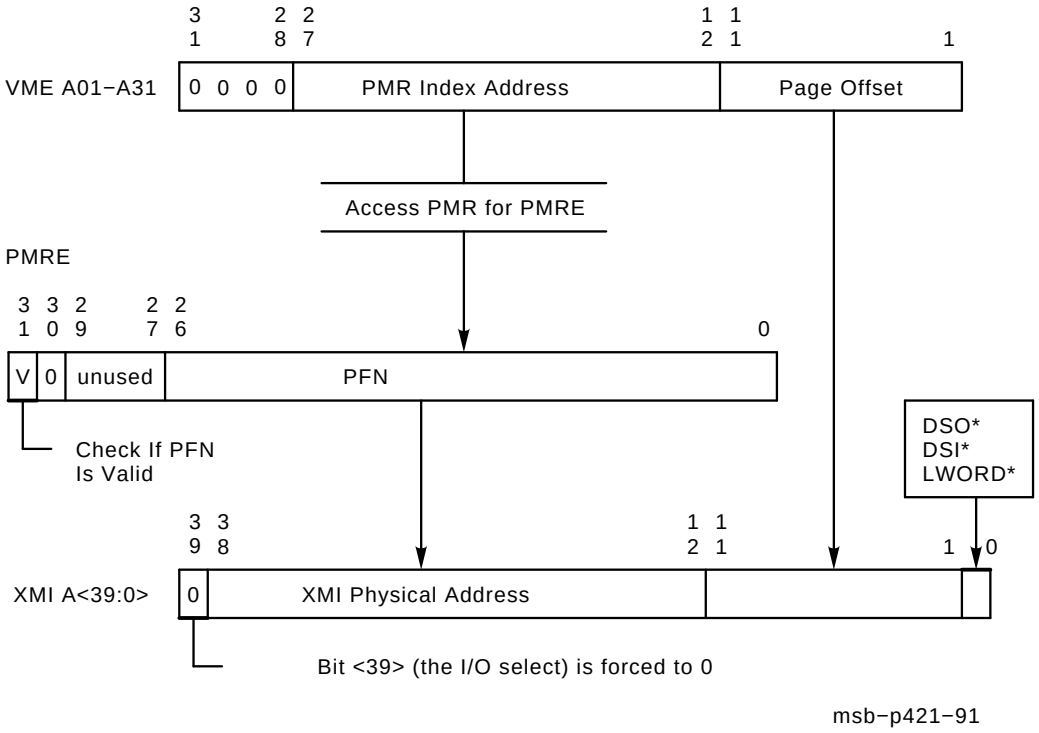
5 Generate XMI Address

Load zero into XMI A<39>.
Load PMRE<29:0> into XMI A<38:9>.
Load VME A0–A08 into XMI A<8:0>.

2.5.3.2 4-Kbyte Page Size

When using a 4-Kbyte page size in 40-bit VAX address translation mode, the DWMVA maps only the first 256 Mbytes of VME memory address space to XMI memory address space (see Figure 2–9). This is the maximum range that can be mapped with 64K PMR entries.

Figure 2–9 40-Bit VAX Address Translation Using 4-Kbyte Page Size



The 40-bit translation of a VME DMA address using 4-Kbyte page sizes uses VME address bits VME A12–A27 as an index into the PMRs. These bits select the PMRE that contains the required PFN. Because in this mode the DWMVA only maps the first 256 Mbytes of VME memory address space, the upper address bits of the VME address, VME A28–A31, must be zero. The validity of the PFN is checked and if good the PFN is used to complete the DMA address translation. The 40-bit physical address is obtained by combining the PFN field of the PMRE (PMRE<26:0>) with VME address bits VME A0–A11. The steps used for 40-bit address translation using 4-Kbyte page sizes are as follows:

1 Check Upper Address Bits

VME A28–A31 must all be zero.

2 Access PMR for PMRE

VME address VME A12–A27 used as an index into the PMR to fetch the PMRE.

3 Check PMRE Valid Bit

If PMRE<31> = 1, then PFN is valid.
If PMRE<31> = 0, then PFN is invalid and transaction is aborted.

4 ECC Check

If no error or correctable error, then PFN is good.
If uncorrectable error, then PFN is bad and the transaction is aborted.

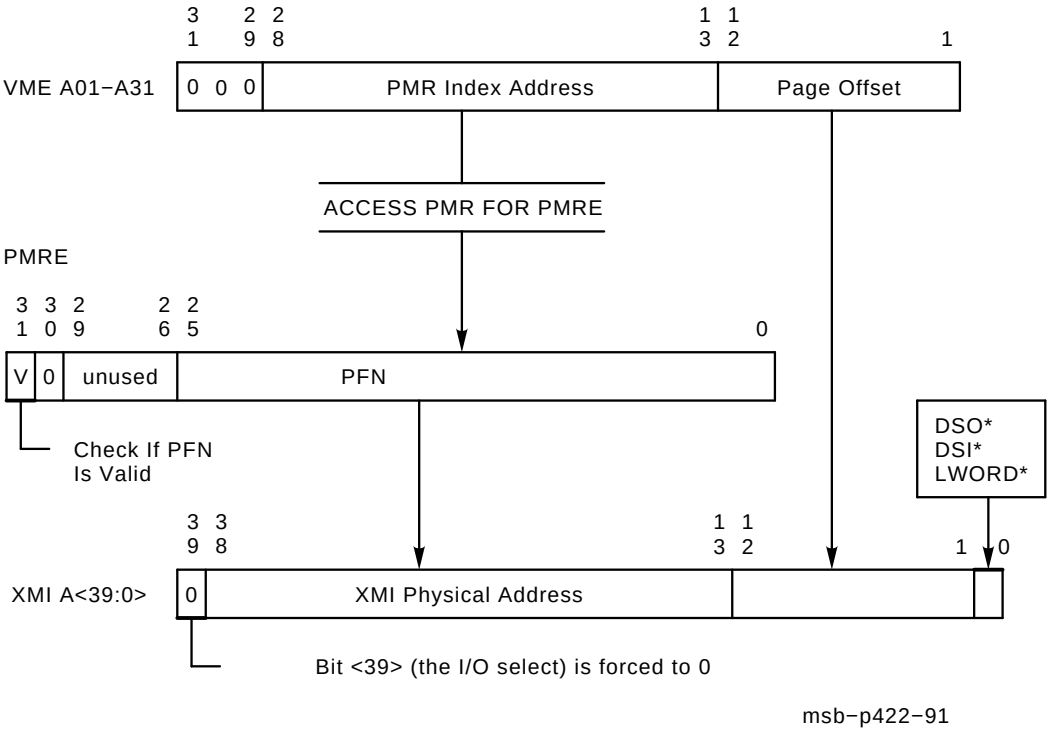
5 Generate XMI Address

Load zeros into XMI A<39>.
Load PMRE<26:0> into XMI A<38:12>.
Load VME A0–A11 into XMI A<11:0>.

2.5.3.3 8-Kbyte Page Size

When using an 8-Kbyte page size in 40-bit VAX address translation mode, the DWMVA can map 512 Mbytes of VME memory address space to XMI memory address space (see Figure 2–10). This is the maximum range that can be mapped with 64K PMR entries.

Figure 2–10 40-Bit VAX Address Translation Using 8-Kbyte Page Size



The 40-bit translation of a VME DMA address using 8-Kbyte page sizes uses VME address bits VME A13–A28 as an index into the PMRs. The validity of the PFN is checked and, if good, the PFN is used to complete the DMA address translation. The 40-bit physical address is obtained by combining the PFN field of the PMRE (PMRE<25:0>) with VME address bits VME A0–A12. The steps used for 40-bit address translation using 8-Kbyte page sizes are as follows:

1 Check Upper Address Bits

VME A29–A31 must be zero.

2 Access PMR for PMRE

VME address VME A13–A28 used as an index into the PMR to fetch the PMRE.

3 Check PMRE Valid Bit

If PMRE<31> = 1, then PFN is valid.
If PMRE<31> = 0, then PFN is invalid and transaction is aborted.

4 ECC Check

If no error or correctable error, then PFN is good.
If uncorrectable error, then PFN is bad and the transaction is aborted.

5 Generate XMI Address

Load zeros into XMI A<39>.
Load PMRE<24:0> into XMI A<38:13>.
Load VME A0–A12 into XMI A<12:0>.

3 VME System Control

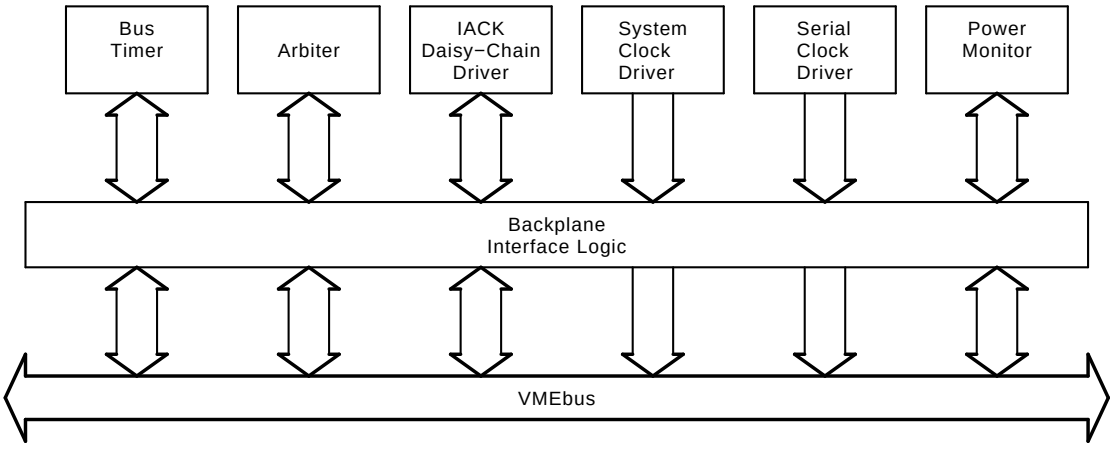
The C3200 module contains the VME system controller (VSC) that includes all the hardware necessary to provide timing and control to the VME system. The VSC consists of the following elements:

- Bus timer
- VME arbitration
- IACK daisy-chain driver
- System clock driver
- Serial clock driver

In addition, the DWMVA requires an external power monitor provided by the system integrator.

Figure 3–1 shows a block diagram of the VME system controller, including the power monitor.

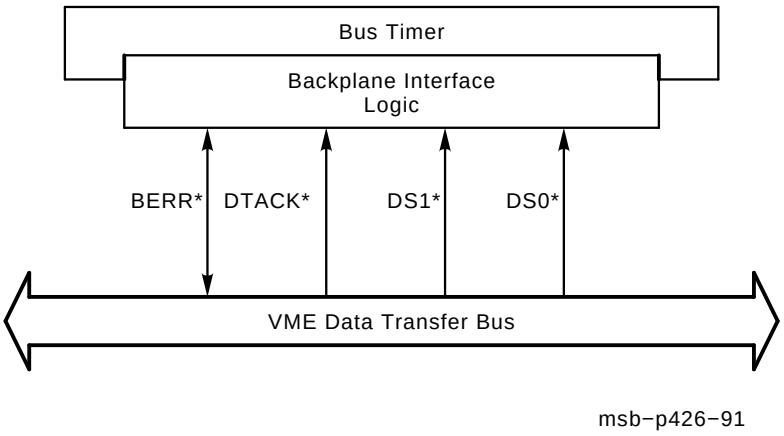
Figure 3–1 VME System Controller Block Diagram



3.1 Bus Timer

The bus timer asserts BERR*[1], the bus error line, to indicate to the master that the data transfer was not completed. BERR* is asserted when the first data strobe (DS0* or DS1*) stays asserted for longer than the bus timeout period, and DTACK* and BERR* are deasserted. Figure 3–2 shows a block diagram of the bus timer.

Figure 3–2 Bus Timer Block Diagram



The timeout value is set by software. The timeout period is programmed in bits <18:16> of the DWMVA Device/Configuration Register, as shown in Table 3–1.

Table 3–1 VME Transaction Timeout Selection

VDCR<18:16>	Timeout Value
111	Timeouts disabled
110	3.28 ms
101	819 μs
100	128 μs
011	64.0 μs
010	32.0 μs
001	12.8 μs
000	800 ns

The transaction timeout period causes an interrupt if the Enable VME Transaction Timeout Interrupt bit (VICR<21>) is set.

[1] An asterisk (*) appended to a VME signal name indicates a low true signal.

3.2 Arbitration

The VME system controller contains an arbitration subsystem that supports arbitration algorithms and timeouts. The type of arbitration is selected by software.

The arbiter is responsible for allocating the data transfer bus to optimize bus usage and prevent two or more masters from using the bus simultaneously.

3.2.1 Arbitration Subsystem Input/Output

The arbitration subsystem of the VME system controller uses the following signals:

Bused signals

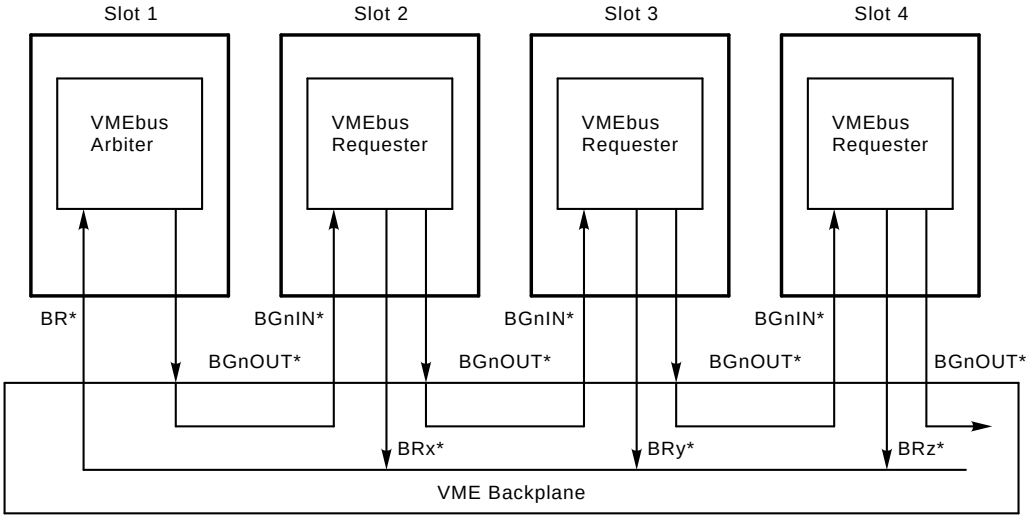
- BR0* through BR3*
- BBSY*
- BCLR*

Daisy-chained signals

- BG0IN* through BG3IN*
- BG0OUT* through BG3OUT*

The bus request lines, BR0* through BR3*, are asserted by a requester of the data transfer bus. These lines are monitored by the arbiter, which in turn asserts the appropriate bus grant line BG0OUT* through BG3OUT*. The bus grant signals are propagated down the backplane in a daisy-chained manner. The bus grant line, BGxOUT*, asserted by the arbiter, is monitored by the option in slot 2 on the BGxIN* line of the option. If this device is not currently requesting the bus, it passes the bus grant to the next device on the backplane by asserting its BGxOUT* line, which is received on the next module via the BGxIN* signal, and so on. If the device receiving BGxIN* has a request pending at that particular level, BBSY* is asserted by the device and all other devices are prevented from using the bus. The logical connections of the arbitration bus are shown in Figure 3–3.

Figure 3–3 VME Arbitration Bus



x,y,z −− One and only one of value 0, 1, 2, or 3.
n −− One of each line of value 0, 1, 2, and 3.

msb−p427−91

Other arbitration signals are BBSY*, the bus busy line, and BCLR*, the bus clear line. After receiving a bus grant, a requester asserts BBSY* to inform the arbiter that it has received the bus grant and is using the bus. See Chapter 5 for a complete description of VMEbus signals.

3.2.2 Arbitration Algorithms

The arbiter logic on the C3200 module supports four arbitration algorithms. The first three of these are defined in the VME specification; the fourth is DWMVA specific. The type of arbitration is determined by bits <30:29> of the DWMVA Device/Configuration Register. The default arbitration set at power-up and at node reset is round robin. The algorithms are described in Table 3–2.

Table 3–2 VME Arbitration Algorithms

VDCR <30:29>	Algorithm	Description
00	Round robin	Grants the bus on a rotating basis. When the bus is granted to requester BR(n)*, the highest priority requester for the next bus cycle becomes BR(n-1)*. BR(n)* now becomes the lowest priority device. BR(n)* is only allowed access to the bus after all devices currently requesting the bus have received bus grants, in descending order.
01	Prioritized	Assigns the bus on a fixed priority basis, with BR3* having the highest priority and BR0* the lowest. If a higher priority device requests the bus while a lower priority device is using it, the arbiter asserts BCLR*, requesting that the low-priority device relinquish the bus to the higher priority device.
10	Prioritized and round robin	Combines the prioritized and round robin arbitration algorithms. The BR3* line has the highest priority, while BR2*–BR0* are granted in a round robin fashion.
11	Single	Accepts only requests on BR3* and relies on the BG3OUT*/BG3IN* daisy-chain to arbitrate as well as grant the requests.

The VMEbus implements an additional level of arbitration that is based on placement of VME devices in the backplane. For example, if two devices are configured to request the bus at BR3, the device in the lower numbered slot (physically closer to the arbiter) has priority, because any device receiving the BG3IN signal can choose to not propagate the signal to the next slot through BG3OUT, if it is currently requesting the bus (both devices are asserting BR3).

Bus request conflicts can be minimized by judicious assignment of BR levels and backplane slots to the VME devices. In addition, the selection of appropriate requester types enables the VME system integrator to eliminate any lockout possibilities that the conflict condition may cause. Refer to Section 3.2.5 for additional discussion on requester types.

3.2.3 Bus Request Level Assignment

The bus request levels for the C3200 module are determined by bits <25:24> of the DWMVA Device/Configuration Register (see Table 3–3). The bus request level for other VME devices is typically configured using jumpers on the module.

Table 3–3 VME Bus Request Level Codes

VDCR<25:24>	Selected Bus Request Level
00	Bus Request Level 0 (BR0)
01	Bus Request Level 1 (BR1)
10	Bus Request Level 2 (BR2)
11	Bus Request Level 3 (BR3)

3.2.4 Arbitration Timeout Counter

The arbitration timeout counter prevents the VME from hanging in the event of a failure by the DWMVA adapter. The timer causes the arbiter to stop driving BGxOUT* if, after a period of time, the requester has not asserted BBSY*. The arbitration timeout period is determined by bits <21:19> in the DWMVA Device/Configuration Register (see Table 3–4). The arbitration timeout causes an interrupt if the Enable VME Arbitration Timeout Interrupt bit (VICR<21>) is set.

Table 3–4 VME Arbitration Timeout Selection

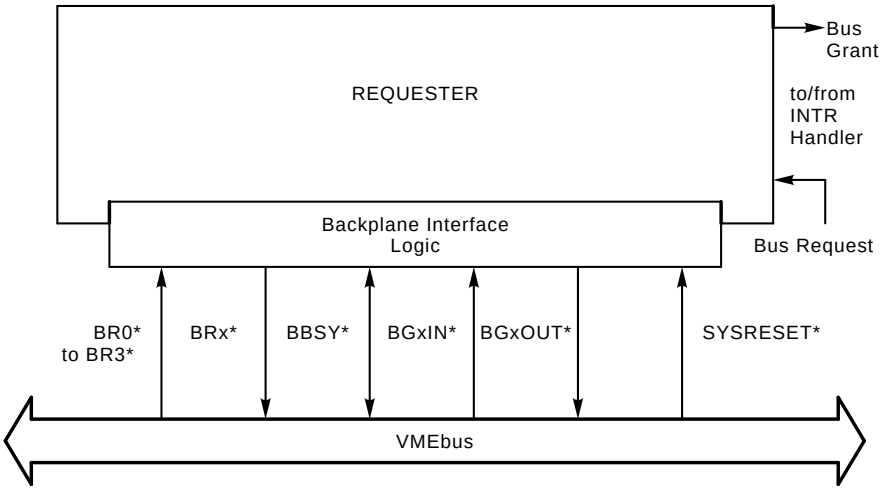
VDCR<21:19>	Timeout Value
111	Timeouts disabled
110	3.28 ms
101	819 μs
100	128 μs
011	64.0 μs
010	32.0 μs
001	12.8 μs
000	800 ns

3.2.5 Data Transfer Bus Requesters

A requester is a functional block on the C3200 module that is responsible for requesting the VME bus for CPU writes and reads to VME slaves. The requester does not physically form part of the VSC. However, it is functionally related to the VME arbitration, and is described here to complete the arbitration discussion. Figure 3–4 shows a block diagram of a VMEbus requester.

The requester is notified by on-board logic that the VMEbus will be required to complete the current transaction. As a result, the requester asserts BRx* on the VMEbus. The pending transaction halts until the arbiter grants permission to use the VMEbus. The requester then monitors the BGxIN* signals. When it detects an asserted BGxIN* signal at the same level as the BRx it sent, it does not pass on that BGxIN*

Figure 3–4 VMEbus Requester Block Diagram



along the arbitration daisy-chain, but uses the bus grant to drive the bus and complete its pending transaction.

The VME specification defines three types of requesters:

- Release when done (RWD)
- Release on request (ROR)
- FAIR

An RWD requester releases the bus by deasserting BBSY* when its master no longer needs the bus for its current data transfer. An RWD requester need not monitor BR3*–BR0*, since it will release the bus upon completion of its transaction regardless of the values of the bus request lines.

An ROR requester does not deassert BBSY* when its master no longer needs the bus, but instead holds the bus until it detects another requester asserting a BRx* signal. This type of requester, therefore, monitors BR3*–BR0* continuously once it has ownership of the bus. The release of the bus upon detection of BRx* by another requester reduces the amount of arbitration on the bus when the master of the ROR requester is generating a large percentage of the bus traffic.

A FAIR requester is used in the case of more than one master sharing the same bus request level. After it has been granted the bus, a FAIR requester will not request the bus again as long as there are any active bus requests pending at its bus request level. To implement a FAIR requester, the bus requester logic must be able to monitor at least its own bus request level line.

NOTE: The RWD and ROR capabilities describe the conditions under which a requester relinquishes control of the data transfer bus. The FAIR capability describes under what condition a requester

will request control of the data transfer bus. Therefore, RWD and ROR requesters can include the FAIR capability as well.

All three types of requesters are supported by the DWMVA. However, the preferred type is an ROR FAIR requester. The DWMVA requester type is RWD FAIR.

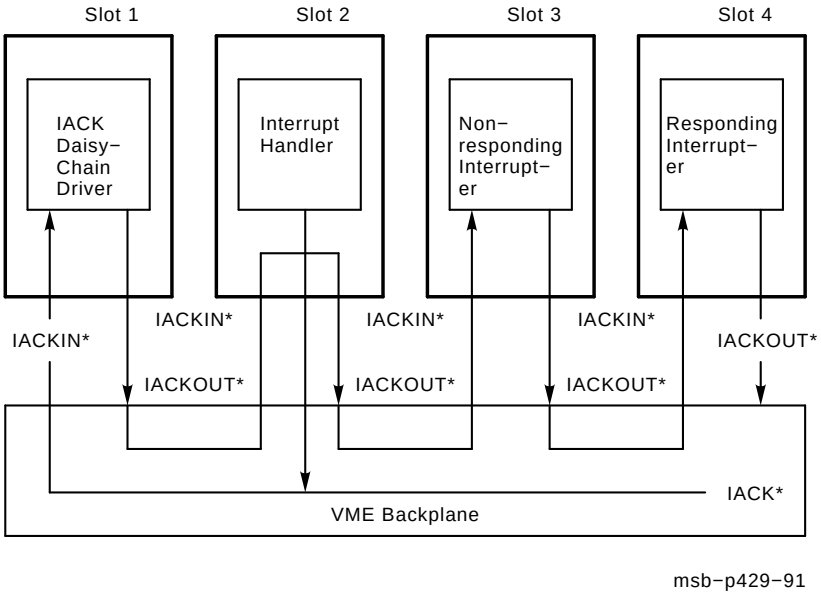
3.3 IACK Daisy-Chain Driver

The DWMVA provides an IACK daisy-chain driver as required by all slot 1 VME devices. The IACK daisy-chain driver generates the signal IACKOUT* each time an interrupt handler initiates an interrupt acknowledge cycle by asserting IACK*. The IACKOUT* signal propagates to the module in slot 2 of the VME backplane as IACKIN*. This module propagates the interrupt acknowledge on its IACKOUT* line if it does not have an interrupt pending at the level present on A01–A03. This IACKOUT* enters the module at VME slot 3 as IACKIN*, and so on down the backplane.

When the IACKIN* reaches the module with the current interrupt pending at the correct level, that module does not propagate IACKOUT*. Instead, the interrupting device returns its vector to the interrupt handler in response to the interrupt acknowledge cycle.

The IACK daisy-chain driver is illustrated in Figure 3–5.

Figure 3–5 IACK Daisy-Chain Driver



3.4 System Clock Driver

The system clock is an independent, nongated, fixed-frequency, 16 MHz, 50% (nominal) duty cycle signal (SYSCLK). The system clock driver is located on the system controller module. SYSCLK is always driven by the C3200 module, which must be installed in slot 1 of the VMEbus backplane.

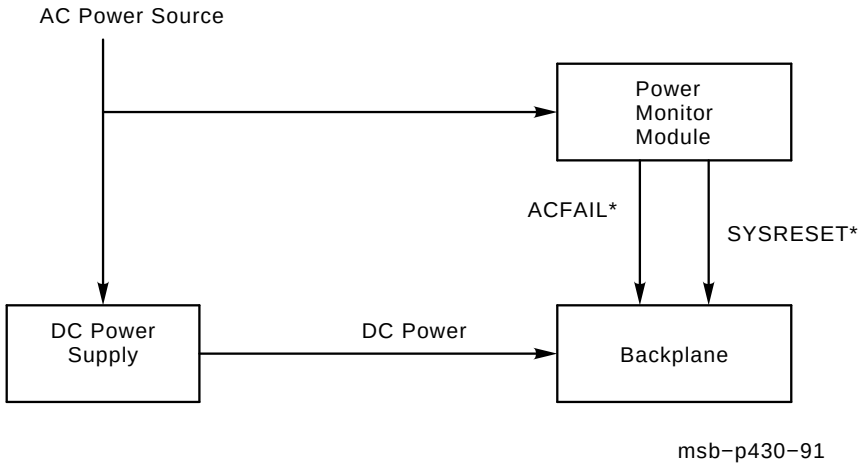
3.5 Serial Clock Driver

The serial clock driver provides a programmable, special waveform signal used by serial modules that reside on VME-compatible boards. SERCLK in conjunction with SERDAT* provides a serial communication link between boards. The C3200 module does not drive the SERDAT* line, but it does provide the serial clock for any module on the VME backplane that needs it. The clock source is software programmable to 32, 16, 8, and 4 MHz (see description of VDCR in Chapter 7).

3.6 Power Monitor

The power monitor detects power failures and signals the system by issuing an IVINTR (see descriptions of AREAR and AESR in Chapter 7). When power is then reapplied to the system, the power monitor ensures that all other modules are initialized. Whenever any board asserts SYSRESET*, the power monitor holds the signal asserted for a minimum of 200 ms. Figure 3–6 shows a block diagram of the VME power supply.

Figure 3–6 Power Supply Block Diagram



NOTE: The power monitor is provided by the system integrator through an external module.

4 Transactions

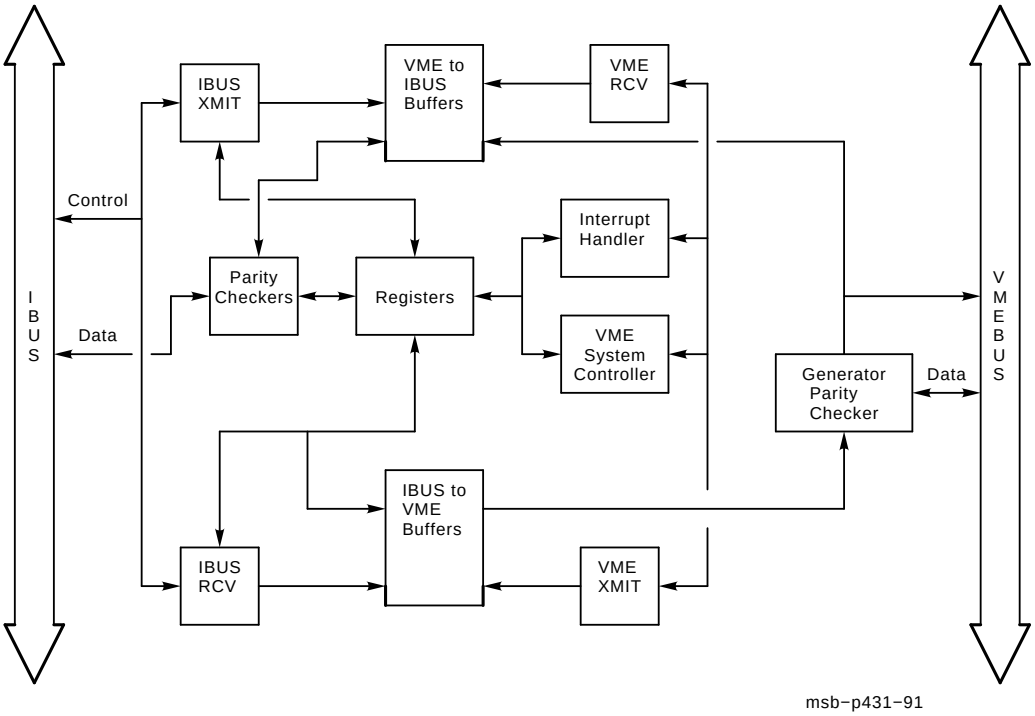
The DWMVA performs two types of transactions:

- CPU transactions
- DMA transactions

A CPU transaction is initiated by a processor on the XMI bus. The processor is the commander. The DWMVA becomes the responder. A DMA transaction begins on the VMEbus and targets XMI memory through the DWMVA adapter.

This chapter explains how the two types of transactions are processed through the C3200 module, between the IBUS and the VMEbus. Figure 4–1 shows the data paths and the major logic sections on the C3200 module.

Figure 4–1 C3200 Block Diagram



4.1 Command Translation

The XMI and the VMEbus use different commands. A translation of commands must take place over the XMI-to-VME data path for commands initiated on the XMI to be executed on the VME, and vice versa.

4.1.1 XMI-to-VME Translation

The DWMVA generates VME commands when it is acting as the responder to an XMI-initiated transaction. The DWMVA accepts only longword CPU transactions. Hexword, octaword, and quadword Write Mask transactions are illegal when targeted at I/O space.

Interlock Read/Unlock Write pairs on the XMI are translated into Read Modify Write (RMW) commands on the VME. Due to differences in protocol between the XMI and VME, some problems may occur and the Interlock Read/Unlock Write may get separated into distinct read and write transactions on the VMEbus, as explained in Section 4.2.2.4. If this condition occurs, the C3200 sets an error bit and generates an interrupt, if enabled to do so. The C3200 does not generate interrupts to the VME. Table 4–1 shows the XMI-to-VME command translations.

Table 4–1 XMI-to-VME Command Translations

XMI	VME
Longword Read	Byte/Word/Longword Read
Quadword Read	Illegal
Octaword Read	Illegal
Hexword Read	Illegal
Longword Masked Write	Byte/Word/Longword Write
Quadword Masked Write	Illegal
Octaword Masked Write	Illegal
Hexword Masked Write	Illegal
Longword Interlock Read	Byte/Word/Longword Read - Start of VME RMW
Quadword Interlock Read	Illegal
Octaword Interlock Read	Illegal
Hexword Interlock Read	Illegal
Longword Unlock Masked Write	Byte/Word/Longword Write - End of VME RMW
Quadword Unlock Masked Write	Illegal
Octaword Unlock Masked Write	Illegal
Hexword Unlock Masked Write	Illegal
IDENT	Interrupt Acknowledge

4.1.2 VME-to-XMI Translation

When the DWMVA processes a DMA transaction, it generates the corresponding XMI command.

A VME read or write can be 1, 2, 3, or 4 bytes of data. The smallest unit of data that can be addressed in XMI memory space is a quadword. Therefore, VME reads translate into XMI quadword reads, and VME writes into quadword masked writes.

Because VME block transfers are assumed to be quite long, the C3200 always issues octaword transactions when it decodes a VME block transfer.

Table 4–2 shows the VME-to-XMI command translations.

Table 4–2 VME-to-XMI Command Translations

VME	XMI
Read	Quadword Read
Write	Quadword Write Mask
Block Read	Octaword Read
Block Write	Octaword Write Mask
Read Modify Write	A VME RMW translates to the following XMI sequence: 1 Quadword Interlock Read 2 Quadword Unlock Write
Address Only	No-op
Interrupt	INTR

4.2 CPU Transaction Process

The CPU initiates the following transactions:

- DWMVA register transactions
- VME device transactions

When a CPU targets the VMEbus as the destination of a transaction, the T2018 accepts the command and stores it in its internal data buffer. The T2018 then informs the C3200 module that it has a CPU transaction in its buffer that is ready to be transmitted to the VMEbus.

The C3200 checks the status of the IBUS. If the IBUS is available, the C3200 loads the CPU transaction into its internal CPU data buffer. Once the buffer is loaded, the C3200 begins requesting the VMEbus. When the C3200 receives the bus grant indicating that it is the VMEbus master, it drives the CPU transaction onto the VME. The targeted VME slave

responds with an acknowledgment. The CPU transaction is complete at this point if it is a CPU write.

If the transaction is a CPU read, the C3200 latches read return data from the VME slave into its internal buffer. The C3200 checks the status of the IBUS. When the IBUS is available, the C3200 causes the read return data to be loaded over the IBUS into the T2018's CPU buffer. When the T2018 receives the data, it arbitrates for the XMI and returns the data to the processor.

4.2.1 DWMVA Register Transactions

DWMVA registers can be read or written by an XMI processor. They are not accessible by VME devices.

4.2.1.1 T2018 Register Read/Write

If the CPU transaction is a write to a T2018 register, the T2018 receive state machine writes the data into the addressed control/status register (CSR).

In a read transaction, the T2018 receive state machine sets its Busy flag after the T2018 acknowledges a read command from an XMI commander. The T2018 then arbitrates for the XMI as a responder. When granted use of the bus, the T2018 sends the data from its addressed register to the XMI commander. The T2018 receive state machine then clears its Busy flag and returns to its idle state.

4.2.1.2 C3200 Register Read/Write

When the T2018 decodes a valid C3200 register address, it encodes the CSR to be accessed on the address lines, loads the CPU buffer, and signals the C3200 that a valid transaction is in the buffer. When the C3200 fetches the command and address over the IBUS, it determines if the transaction being sent across the IBUS is destined for a device on the VMEbus or is an access to one of the C3200's internal CSRs.

In a write transaction to a C3200 register, handshaking takes place between the T2018 and C3200 across the IBUS. First the T2018 receive state machine loads command, address, and data into the T2018 CPU buffer. Following this operation, the T2018 receive state machine sets the CPU Busy flag. The C3200 reads the data and checks parity. If the parity is good, the data is written into the addressed C3200 register. The C3200 then signals the processor the termination of the transaction. This clears the Busy flag in the T2018's CSR. If the C3200 detects bad parity, it asserts an error signal and does not write the data into the addressed register.

In a read transaction, the command and address are received by the C3200 in the same way as for a C3200 register write. The data is fetched from the addressed register on the C3200 and sent to the T2018 CPU buffer. The C3200 notifies the T2018 transmit state machine that the T2018 CPU buffer contains new data. The T2018 arbitrates for the XMI bus as a responder, sends the data to the XMI commander, and clears its Busy flag. The DWMVA has now completed the transaction and is ready to process

another. If the C3200 detects bad parity during the IBUS transfer, it does not return read data to the T2018 and asserts an error signal.

NOTE: Certain DWMVA register transactions cause address-only cycles on the VME.

4.2.2 CPU-to-VME Device Transactions

The processor targets a VME device for a read or write transaction. There are three modes of CPU transactions. The transaction mode is specified by the address modifier code attached to the VME address. The address modifier codes used by the C3200 during CPU (XMI-to-VME) transactions are listed in Table 4–3.

Table 4–3 Address Modifier Codes for CPU Transactions

Code	Access Type
2D	Short supervisory access
3E	Standard supervisory program access
0E	Extended supervisory program access

4.2.2.1 VME Device Write

When a CPU transaction targets a node on the VMEbus, the data is written into the C3200 internal buffer the same way as for a C3200 register write. Once the data is stored in the CPU buffer, the C3200 requests the VMEbus. After being granted use of the VMEbus, the C3200 broadcasts the address, address modifier, and data to be written over the VMEbus. The C3200 waits for DTACK*, which indicates that the slave successfully received the data over the VMEbus. When the transaction is complete, the C3200 notifies the T2018 transmit state machine, which in turn clears the Busy flag. The transaction is now complete, and the DWMVA is ready to process a new transaction.

4.2.2.2 VME Device Read

A VME device read is similar to a device write. The sequence of events is the same except that data is supplied to the DWMVA after the DWMVA has broadcast the address and address modifier. In addition, the following events must take place to complete the VMEbus read. The C3200 (master) monitors DTACK* to detect if the VME device (slave) has placed valid data on the VMEbus. The C3200 coordinates the flow of data from the VMEbus to the DMA buffer on the C3200. The C3200 releases the VMEbus when it has finished reading data from the VME device. The C3200 controls the data flow from its DMA buffer over the IBUS to the T2018 and notifies the T2018 transmit state machine that the data is read. Finally, the T2018 arbitrates for the XMI bus as a responder, returns the data to the commander, and clears its Busy flag, indicating that it is ready to accept a new transaction.

4.2.2.3 CPU Reads and Masked Writes

CPU reads and masked writes are used to select the specific byte or bytes to be read from the DWMVA or to be written to VME space.

CPU masked writes to VME space can take place when the appropriate mask bits are asserted on the 4-bit mask field over the IBUS as shown in Table 4–4.

Table 4–4 CPU Masked Writes to VME Space

IBUS Mask Field <3:0>	Masked Write Command
0001	Write byte 0
0010	Write byte 1
0100	Write byte 2
1000	Write byte 3
0011	Write word 0
1100	Write word 1
0111[1]	Write triple byte 0–2
1110[1]	Write triple byte 1–3
1111	Write longword

[1]These commands are not supported on the VAX 6000.

When the CPU reads data from the DWMVA, a longword of data is returned. Depending on the VME device and the address being read, only specific bytes within the longword are guaranteed to be valid.

For example, any given VME device can have an 8-bit, 16-bit, or 32-bit data path. A device with an 8-bit data path can return only one byte of data for each read transaction. The same device can store only one byte for each CPU write. Similarly, a 16-bit device returns a word, and a 32-bit device returns one longword of valid data per CPU read transaction. If a device is requested to read or write data wider than its data path, the transaction times out and does not complete.

The two least significant bits of the CPU address and the VME data length (8-, 16-, or 32-bit) information determine which bytes of CPU read return data will be valid. The data length information is stored in the C3200 CPU Transaction Address Offset Register.

Table 4–5 shows how CPU masked read commands are selected.

Table 4–5 CPU Reads of DWMVA

Data Length VAOR<17:16>	CPU A<1:0>	Read Command
01 (8-bit device)	00	Read byte 0
01 (8-bit device)	01	Read byte 1
01 (8-bit device)	10	Read byte 2
01 (8-bit device)	11	Read byte 3
10 (16-bit device)	00	Read word 0
10 (16-bit device)	10	Read word 1
11 (32-bit device)	00	Read longword

4.2.2.4 CPU Interlocks

The VMEbus has no transactions equivalent to Interlock Read/Unlock Write on the XMI. Instead, the VMEbus implements RMW transactions in response to XMI originated Interlock Reads and Unlock Writes. The RMW on the VMEbus is an atomic operation; that is, no other transaction is allowed on the bus between the read and write. The Interlock Read/Unlock Write transactions on the XMI, however, can be separated.

An Interlock Read to VMEbus space could proceed in the following manner. The C3200 would initiate a RMW cycle on the VME. Since there is no restriction on the XMI that the Unlock Write must immediately follow the Interlock Read, it is possible that another VME reference could be addressed to the DWMVA before the DWMVA receives the Unlock Write. The DWMVA would try to process that transaction and find the VME hung in the middle of an incomplete RMW transaction.

The C3200 attempts to perform a RMW on the VME when it receives an Interlock Read from the XMI. If, however, the next transaction it receives over the IBUS after the Interlock Read is anything but an Unlock Write, the C3200 releases the bus without doing the write portion of the RMW. If this condition occurs, the C3200 correctly performs the Interlock Read/Unlock Write (though as a separate read and write, not RMW), and any intervening transactions. The C3200 sets an error bit and interrupts the processor, if interrupts are enabled.

4.3 DMA Transaction Process

A DMA transaction is initiated by a VME device. The initiating device becomes the master, the DWMVA becomes the slave, and the targeted XMI memory becomes the responder. Only XMI memory can respond to a DMA transaction. DMA transactions can consist of 1-, 2-, 3-, or 4-byte single-access transfers or 1-, 2-, or 4-byte block reads and writes. Block reads and writes can transfer up to 256 bytes and store the transferred data in contiguous locations in XMI memory.

XMI memory supports quadword, octaword, and hexword reads and writes. The DWMVA is optimally designed to transfer octawords. During DMA writes, large block transfers from the VME are sent to XMI memory in octaword segments. In DMA block reads, the T2018 reads data in octaword blocks from XMI memory. The C3200 transfers this data in bytes, words, or longwords, depending on the transfer size requested by the VME master.

VME protocol allows up to 256 bytes to be transferred during a single block transfer. The C3200 contains two sets of buffers, referred to as the VTI (VME-to-IBUS) and ITV (IBUS-to-VME). The VTI buffer can hold two octawords of write data along with a command/address for each write transaction. The ITV buffer can hold two octawords of read return data.

A DMA transaction begins on the VMEbus and targets XMI memory through the DWMVA adapter. The C3200 monitors the VMEbus and if it detects a transaction that falls within its address range, it accepts the transaction and loads it in its internal buffer. The C3200 then transfers the transaction over the IBUS to the T2018, provided the IBUS is not busy.

In the case of a DMA write, the T2018 is the commander and arbitrates for the XMI bus. The responder is memory on the XMI. If the transaction is a read, the description above takes place with the addition of the following: before memory can send the data that has been requested, the memory board must arbitrate for the XMI. The T2018 receives the return data and stores it within its data file. The T2018 notifies the C3200 that the read return data is available. Provided the IBUS is not busy, the C3200 asserts control signals on the IBUS to cause the T2018 data file to be read into the buffers of the C3200. The C3200 then returns data to the VME master that requested it. At this point, the VME master releases control of the bus so that other devices can begin data transfers.

4.3.1 VME-to-XMI Memory Write

When the C3200 is addressed, it stores the write data in the C3200 DMA buffer and acknowledges the master by asserting DTACK*. If the transaction is a block write, consecutive data transfers are made by the VME master. Each transfer is acknowledged with assertion of DTACK*. If the transaction is a single-access write, 1, 2, 3, or 4 bytes of data are sent to the DWMVA in a single data cycle. For single-access writes, the C3200 issues a quadword-length masked write and sends the data to the T2018 over the IBUS. If the VME transaction is a block transfer, 1, 2, or 4 bytes of data are transferred during each VME cycle. In this case, the C3200 builds octaword-length transfers in its DMA buffer and sends the octaword masked write over the IBUS to the T2018 only when it has completed filling data on the current octaword address boundary.

The data is sent from the C3200 DMA buffer to the available DMA buffer on the T2018 upon notification from the C3200. After the T2018 DMA buffer has been loaded by the C3200 data buffer, the T2018 transmit state machine arbitrates as a commander on the XMI. After the T2018 is granted the bus, it sends the data from its DMA buffer to XMI memory. The T2018 Busy flag, which had been set once the T2018 had started

accepting IBUS data, is now cleared. At this point the IBUS octaword write transaction is complete. The VME block write can still be ongoing, requiring additional octaword writes over the IBUS.

4.3.2 VME-to-XMI Memory Read

The XMI memory read is similar to the XMI memory write, except that in an XMI read only the address is sent to initiate the transaction. Whether the transaction is a block or single-access read is determined from the signals on the address modifier lines AM01–AM05. If a single-access read is detected, the C3200 issues a quadword read over the IBUS. If a block read was decoded, the C3200 instead requests an octaword of data, since it expects the next consecutive addresses to be read during the block transfer.

The XMI memory is the responder on the XMI and returns the requested data to the T2018, the commander. The T2018 accepts the data into one of its DMA buffers. The T2018 notifies the C3200 that return data is available. The C3200 accepts the data and stores the data into its ITV buffer. The VMEbus has been stalled waiting for the return data and has been dedicated to the DWMVA since the beginning of the transaction. The C3200 sends the data over the VMEbus under the control of the DWMVA master.

4.3.3 DMA Interlocks

The VME initiates RMW transactions in the same manner that it initiates a normal read. The VME slave (DWMVA in this case) is unaware that the intended transaction is a RMW until the VME master holds the bus following the read return data and issues the corresponding write.

The resulting XMI transactions to a read followed by a write would be an XMI read followed by an XMI write, because the RMW is an atomic VME transaction, even though the intent was to do an Interlock Read/Unlock Write pair on the XMI.

Since the DWMVA has no indication that the VME master intends to do a RMW, it provides a mechanism that enables the DWMVA to cause reads and writes to specific addresses to translate into Interlock Reads and Unlock Writes, respectively. This is done by writing to the Byte Swap RAM Access Register with the RMW bit set. See Chapter 7 for more details.

If a VME master initiates a read that is not part of a RMW transaction to a page set up for RMWs, the DWMVA issues the following sequence of instructions:

1. Interlock Read
2. Unlock Write (with all data bits masked)
3. Set RMW Error II bit (VESR<28>)
4. Interrupt, if enabled

In this manner, the read transaction will be completed and an interrupt will occur to indicate that an interlock instruction was executed at an unintended location.

If, on the other hand, a VME master initiates a RMW transaction to a page that was not set up for RMWs, the DWMVA issues the following sequence of instructions:

1. Read (not Interlock Read)
2. Write (not Unlock Write)
3. Set RMW Error I bit (VESR<27>)
4. Interrupt, if enabled

In this manner, both the read and write will be executed. However, since the two instructions will not result in an Interlock Read/Unlock Write pair, the DWMVA will issue an interrupt.

5 VMEbus Interface

The VMEbus is a high-performance bus for use in microcomputer systems that employ single or multiple microprocessors. It is the bus that interconnects the DWMVA and VME devices.

The VMEbus includes four substructures:

- Data transfer bus
- Arbitration bus
- Priority interrupt bus
- Utility bus

This chapter discusses the VME interface of the DWMVA subsystem. The material presented here is limited to the DWMVA implementation of the VMEbus. Refer to IEEE Standard 1014 for a comprehensive discussion of the VMEbus.

5.1 Data Transfer Bus

The data transfer bus (DTB) is a high-speed asynchronous parallel bus used for nonmultiplexed address/data transfers. Masters use the DTB to select storage locations provided by slaves and to transfer data to or from those locations. Some masters and slaves use all of the DTB lines, while others use only a subset.

After a master initiates a data transfer cycle, it waits for the addressed slave to respond before terminating the cycle. The asynchronous definition of the bus allows a slave to take all the time it needs to respond. When a slave fails to respond because of some malfunction, or the master addresses a location where there is no slave, the bus timer intervenes, allowing the cycle to be terminated and freeing the bus for subsequent transactions.

Table 5–1 shows the address, data, and control lines of the data transfer bus.

Table 5–1 Data Transfer Bus Signals

Address Lines	Data Lines	Control Lines
A01–A31	D0–D31	AS*
AM0–AM5		DS0*
DS0*		DS1*
DS1*		BERR*
LWORD*		DTACK*
		WRITE*

5.1.1 Address Lines

The smallest accessible unit of storage is a byte location. Each byte location corresponds to a unique address and can be assigned to one of four categories, according to the two least significant bits of its address, as shown in Table 5–2.

Table 5–2 Categories of Byte Locations

Category	Byte Address
Byte(0)	...XXXXXX00
Byte(1)	...XXXXXX01
Byte(2)	...XXXXXX10
Byte(3)	...XXXXXX11

The four byte locations in the same longword are referred to as a 4-byte group or a Byte(0–3) group. Some, or all, of the bytes in a naturally aligned longword can be accessed in a single DTB cycle.

Masters use address lines A02–A31 to select the longword to be accessed. Four additional lines, DS1*, DS0*, A01, and LWORD*, are then used to select the byte location(s) within the 4-byte group to be accessed during the data transfer. Using these four lines, a master can access 1-, 2-, 3-, or 4-byte locations simultaneously, as shown in Table 5–3.

Table 5–3 Selecting Byte Locations Within Longwords

Access Type	Bytes Selected	DS1*[1]	DS0*[1]	A01[1]	LWORD*[1]
Single Byte	Byte(0)	0	1	0	1
	Byte(1)	1	0	0	1
	Byte(2)	0	1	1	1
	Byte(3)	1	0	1	1
Double Byte	Byte(0–1)	0	0	0	1
	Byte(1–2)	0	0	1	0
	Byte(2–3)	0	0	1	1
Triple Byte	Byte(0–2)	0	1	0	0
	Byte(1–3)	1	0	0	0
Quad Byte	Byte(0–3)	0	0	0	0

[1]A value of 0 indicates low voltage level; a value of 1 indicates high voltage level.

The six address modifier lines of the VMEbus allow the master to pass additional information to the slave during DTB cycles. The address modifier function codes fall into three categories as follows:

- Short addressing AM codes indicate that address lines A02–A15 are being used to select a Byte(0–3) group.

- Standard addressing AM codes indicate that address lines A02–A23 are being used to select a Byte(0–3) group.
- Extended addressing AM codes indicate that address lines A02–A31 are being used to select a Byte(0–3) group.

Table 5–4 lists the address modifier codes accepted by the DWMVA. No additional function codes are supported. All codes are treated the same.

Table 5–4 Address Modifier Codes

AM0–AM5[1]	AM0–AM5 (hex)	Function
111111	3F	Standard supervisory block transfer
111110	3E	Standard supervisory program access
111101	3D	Standard supervisory data access
111011	3B	Standard nonprivileged block transfer
111010	3A	Standard nonprivileged program access
111001	39	Standard nonprivileged data access
001111	0F	Extended supervisory block transfer
001110	0E	Extended supervisory program access
001101	0D	Extended supervisory data access
001011	0B	Extended nonprivileged block transfer
001010	0A	Extended nonprivileged program access
001001	09	Extended nonprivileged data access

[1]The DWMVA does not respond to short address transactions.

5.1.2 Data Lines

The DWMVA has 32 data lines (D0–D31). When the master selects 1-, 2-, 3-, or 4-byte locations, using the method described in Table 5–3, it can transfer data between itself and those locations over the data bus. Table 5–5 shows how the data lines are used to access byte locations.

Table 5–5 Use of Data Lines to Access Byte Locations

Bytes Accessed	D24–D31	D16–D23	D08–D15	D0–D07
Byte(0)			Byte(0)	
Byte(1)				Byte(1)
Byte(2)			Byte(2)	
Byte(3)				Byte(3)
Byte(0–1)			Byte(0)	Byte(1)
Byte(1–2)		Byte(1)	Byte(2)	
Byte(2–3)			Byte(2)	Byte(3)
Byte(0–2)	Byte(0)	Byte(1)	Byte(2)	
Byte(1–3)		Byte(1)	Byte(2)	Byte(3)
Byte(0–3)	Byte(0)	Byte(1)	Byte(2)	Byte(3)

5.1.3 Control Lines

Table 5–6 lists the signal lines used to control the movement of data over the data transfer lines.

Table 5–6 Control Line Signals

Signal	Name
AS*	Address Strobe
BERR*	Bus Error
DS0*	Data Strobe Zero
DS1*	Data Strobe One
DTACK*	Data Transfer Acknowledge
WRITE*	Read/Write

See Section 5.5 for desriptions of the control signals.

5.2 Arbitration Bus

The arbitration bus controls the allocation of the data transfer bus in a multiple processor system. The arbitration system on the VMEbus:

- Prevents simultaneous use of the bus by two masters
- Schedules requests from multiple masters for optimum bus use

The arbitration bus allocates bus mastership based on implementation of round robin and prioritized arbitration algorithms.

5.3 Priority Interrupt Bus

The priority interrupt bus provides the signal lines needed to generate and service interrupts. Interrupters use the priority interrupt bus to send interrupt requests to interrupt handlers. In a single-handler system, the supervisory processor is the destination for all bus interrupts, servicing them in a prioritized manner. In distributed systems, each processor services only those interrupts directed to it, establishing dedicated paths among all processors.

5.4 Utility Bus

The utility bus provides signal lines used to control utility functions such as periodic timing, initialization, and diagnostics. The utility bus is used for system power-up and power-down synchronization.

Three functional modules on the VME system controller, located in slot 1, drive and receive the utility bus signals. These functional modules are the serial clock driver, the system clock driver, and the power monitor. The drivers are responsible for driving and meeting the required electrical specifications given by the IEEE 1014 standard. The system clock (SYSCLK) and the serial clock (SERCLK) are defined in Section 5.5.4.

The power monitor detects power failures and signals the system in time to effect an orderly shutdown. When power is then reapplied to the system, the power monitor ensures that all other modules are initialized.

5.5 VMEbus Signal Descriptions

This section provides descriptions of VMEbus signals.

NOTE: In adherence to the VMEbus conventions, low true VMEbus signals are marked with an asterisk (*).

5.5.1 Data Transfer Bus Signals

A01–A31

Masters broadcast A02–A31 over the VMEbus to select the 4-byte group to be accessed. A01 and three additional lines, DS0*, DS1*, and LWORD*, described further below in this section, are then used to select which byte location(s) within the 4-byte group are accessed during the data transfer.

AM0–AM5

The address modifier lines allow the master to pass additional information to the slave during DTB cycles. This information is related to short, standard, and extended addressing schemes and block versus nonblock transfers.

AS*

A falling edge on the address strobe, AS*, informs all slaves that the address is stable and can be captured.

BERR*

The Bus Error signal is asserted by the slave or by the bus timer to indicate to the master that the data transfer was unsuccessful. For example, when a master tries to write to a location that contains read-only memory, the responding slave can assert BERR*. Also, when the master tries to access a location that is not provided by any slave, the bus timer asserts BERR* after a specified period.

D0–D31

The VMEbus has 32 data lines. Devices can be configured to use either eight data lines (D0–D07), 16 data lines (D0–D15), or 32 data lines (D0–D31). Masters that have 16 data lines can access at the most two byte locations simultaneously, while those with 32 data lines can access all four bytes of a 4-byte group at one time.

DS0* and DS1*

The two data strobes are two of the four signals used to select the byte location(s) within the 4-byte group.

DS0* and DS1* also serve additional functions. On write cycles, the first falling edge of a data strobe indicates that the master has placed valid data on the data bus. On read cycles, the first rising edge informs the slave that it can remove its data from the data bus.

DTACK*

The slave asserts DTACK* (Data Acknowledge) to indicate that it has successfully received the data on a write cycle. On a read cycle, the slave asserts this signal to indicate that it has placed data on the data lines.

LWORD*

LWORD* is one of the four signals used to select the byte location(s) within the 4-byte group.

WRITE*

WRITE* is a level-significant signal that is strobed by the falling edge of the first data strobe. It is used by the master to indicate the direction of data transfers. When WRITE* is asserted, data is transferred from the master to the slave. When WRITE* is deasserted, data is transferred from the slave to the master.

5.5.2 Arbitration Bus Signals

BBSY*

Once a requester has been granted control of the data transfer bus by way of the bus grant daisy chain, it asserts BBSY* (Bus Busy). The requester then has control of the DTB. The arbiter can grant the DTB to some other requester only when the current requester releases the DTB by deasserting BBSY*.

BCLR*

The priority arbiter asserts BCLR* (Bus Clear) to inform the master, currently in control of the DTB, when a higher priority request is pending. The current master is not required to relinquish the bus within any prescribed time. It can continue transferring data until it reaches an appropriate stopping point and allow its on-board requester to deassert BBSY*.

BG0IN*–BG3IN*

A master that receives BGxIN* (Bus Grant In) , and has a request pending at the same level as the BGxIN, has access to the data transfer bus. Otherwise, the master passes on the BGxOUT* so that the next module in the VME card cage will receive BGxIN*. BGxIN* and BGxOUT* propagate in a daisy-chain fashion along the VME backplane.

BG0OUT*–BG3OUT*

A device passes on BGxOUT* (Bus Grant Out) if its master does not have a request pending at that particular level, BRx*.

BR0*–BR3*

Masters drive one of the four bus request levels, BR0*–BR3*, to gain access to the data transfer bus.

5.5.3 Priority Interrupt Bus Signals

IACK*

The IACK* (Interrupt Acknowledge) line runs the full length of the backplane and is connected to the IACKIN* pin of slot 1. When asserted, the IACKIN* pin causes the IACK daisy-chain driver, located in slot 1, to propagate a falling edge down the interrupt acknowledge daisy chain.

IACKIN*

Interrupters that receive IACKIN* (Interrupt Acknowledge In) and have a request pending at the same level encoded on A01–A03 can respond to the interrupt acknowledge cycle. If the pending request level does not match the encoded request level on A01–A03, the interrupter passes on IACKOUT* to the module in the next slot of the VME card cage.

IACKOUT*

IACKOUT* (Interrupt Acknowledge Out) is asserted by a VME device that does not have either an interrupt pending or a pending interrupt level matching the level encoded on A01–A03.

IRQ7*–IRQ1*

Interrupters request interrupts by asserting IRQx*. The interrupt handler receives the interrupt request and gives highest priority to IRQ7*.

5.5.4 Utility Bus Signals

ACFAIL*

ACFAIL* (AC power failure) is one of two signals (the other is SYSRESET*) used in a power-up/power-down sequence.

SERCLK

The serial clock driver provides a programmable, special waveform signal.

SERDAT*

SERDAT* is used for data transmission.

SYSCLK

The system clock is independent and nongated. It pulses at 16 MHz fixed-frequency and has a 50% (nominal) duty cycle. It provides a known time base that is useful for counting off time delays. SYSCLK has no fixed-phase relationship with other timings.

SYSFAIL*

SYSFAIL* is held low when the system is powered up and remains low until system self-tests are complete.

SYSRESET*

SYSRESET* is one of two signals (the other is ACFAIL*) used in a power-up/power-down sequence.

6 Interrupts

The DWMVA provides a path for VME devices to interrupt the host processor. The C3200 implements an interrupt handler that accepts and processes interrupts initiated from VME devices.

6.1 Error Interrupts

The C3200 module generates two types of interrupts to report errors to the system, INTR (interrupt) and IVINTR (implied vector interrupt). An INTR type of interrupt is usually associated with errors detected in read transactions. An IVINTR type of interrupt generally occurs during a write transaction. The following errors cause the C3200 to initiate interrupts to the XMI processor, if interrupts are enabled:

- VME system reset
- VME bus timeout
- VME arbitration timeout
- RMW error
- Interlock error
- Parity error

NOTE: When the C3200 module detects an error, it locks the data in the error registers, which cannot be updated until the corresponding error bit is cleared. If the C3200 module detects a subsequent error before the previous error bit is cleared, the status bit ME (AESR bit<14>) sets and the error registers remain locked with data from the first error.

VME interrupts must be longword-aligned. Note that the C3200 does not initiate any interrupts to the VMEbus, and that an XMI device cannot interrupt a VME device.

6.2 Interrupt Sequence

Table 6–1 shows the sequential events that take place in servicing a VME interrupt.

Table 6–1 VME-to-XMI Interrupt Progression

VME Device	DWMVA	XMI Processor
Generates IRQ* on VME		
	Generates INTR to XMI	
		Accepts INTR Generates IDENT
	Accepts IDENT Generates IACK* on VME	
Responds to IACK* with its vector		
	Passes VME device's vector to XMI processor in response to IDENT	
		Accepts vector and services interrupt

6.3 VME-to-XMI Interrupt Protocol

A VME device initiates an interrupt request to a host processor by asserting the IRQx signal, where *x* is a value from 1 to 7. As soon as an IRQx line is asserted, its corresponding pending bit is asserted in the C3200 Error Summary Register, bits <10:4>.

The interrupter is the logic module that asserts IRQ lines, and the interrupt handler is the logic module that monitors IRQ lines and manages the interrupts. The interrupt is translated to an XMI interrupt if the Enable VME Device Interrupt bits (<24:17>) and the corresponding VME Interrupt Request Level Mask bits (<31:25>) are cleared in the C3200 Interrupt Configuration Register.

Before proceeding with the interrupt, the C3200 must arbitrate for the VMEbus. The C3200 sends the interrupt request by issuing an INTR command to the T2018. The BRn Interrupt Sent bits (<3:0>) of the C3200 Error Summary Register are set when the C3200 issues an INTR command. The T2018 then issues an INTR command at the corresponding BRn level to the XMI. See Figure 6–1 for the XMI INTR command format.

An XMI processor issues an IDENT in response to the T2018 INTR command. The T2018 transmits the IDENT command to the C3200. When the IDENT command is received by the C3200, the BRn Interrupt Sent field in the C3200 Error Summary Register (bits <3:0>) is cleared. See Figure 6–2 for the XMI IDENT command format.

Figure 6–1 XMI INTR Command Format

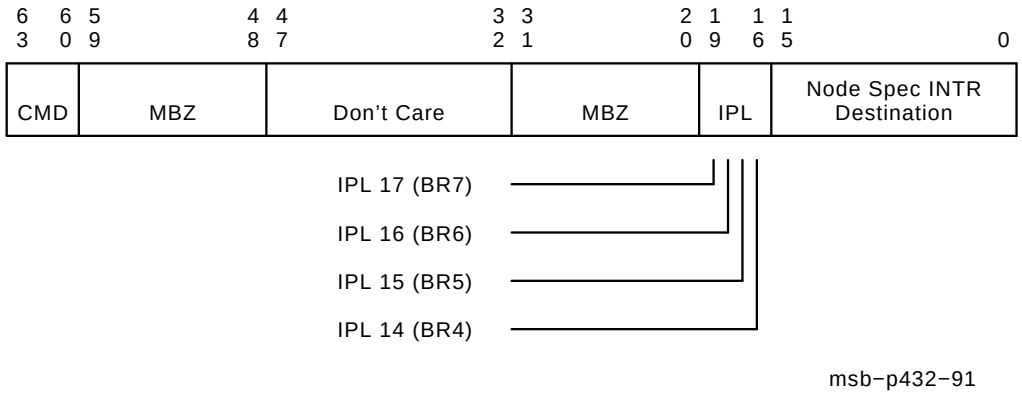
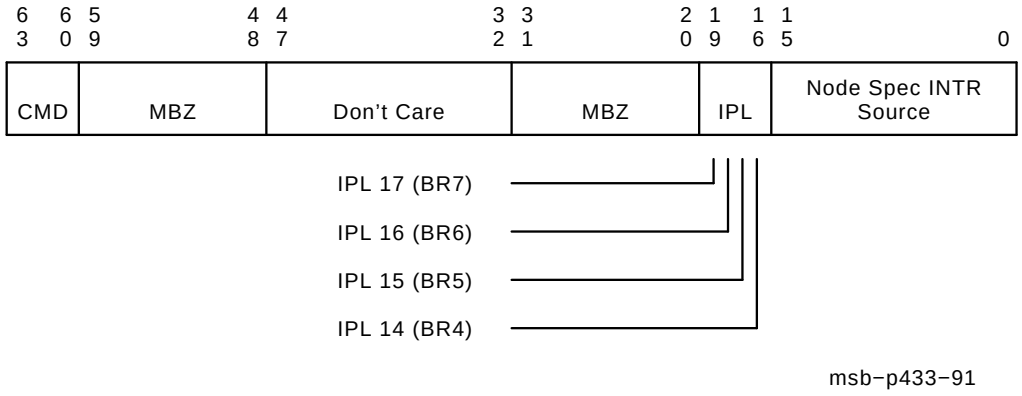


Figure 6–2 XMI IDENT Command Format



The interrupt handler asserts IACK*, acknowledging the selected interrupt level coded on the A01–A03 address lines. Upon acknowledgment, the corresponding IRQn Interrupt Pending bit in the C3200 Error Summary Register is cleared.

The interrupter then responds asserting DTACK*, signaling the interrupt handler that the status/ID is valid on the VMEbus D0–D07. This status/ID is appended to the C3200 Vector Offset Register bits <15:8> and transferred to the T2018 as the address vector of the interrupt routine to be executed to service the VME device over the DWMVA. Since the XMI vector must be longword-aligned, the two lower bits of the vector, D0–D01, are dropped. Figure 6–3 shows how the XMI vector is formed. The interrupt handler is of type D32, which means that it will generate 32-bit interrupt acknowledge cycles and reads an 8-bit status/ID from D0–D07.

Figure 6–3 Generating the XMI IDENT Response Vector

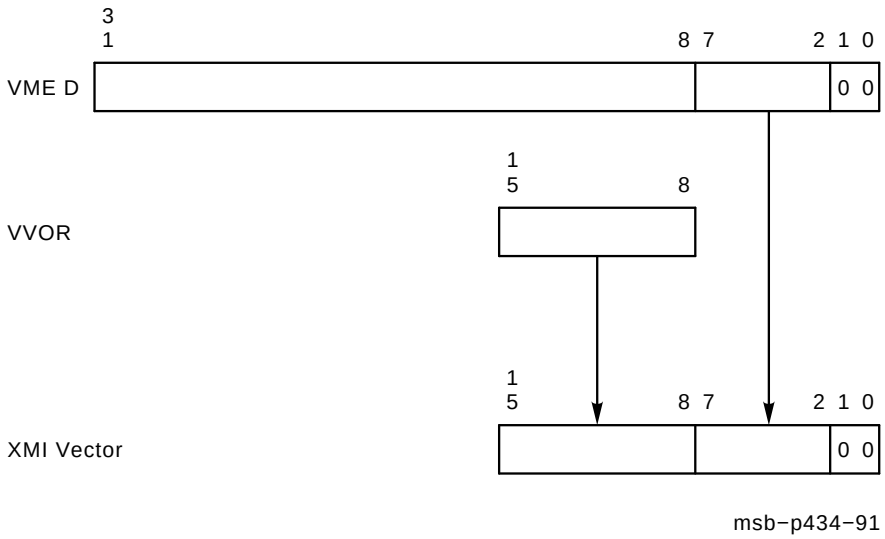
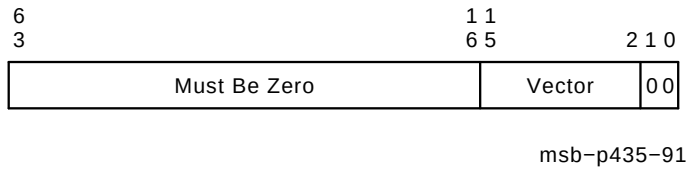


Figure 6–4 shows the XMI IDENT response format.

Figure 6–4 XMI IDENT Response Format



6.4 Interrupt Request Levels

The four bus request lines of the XMI, BR7–BR4, can accommodate four interrupt request signals (IRQx*) from the VMEbus on one-to-one correspondence. Since the VMEbus features seven interrupt request levels, four selected interrupt levels must be mapped to the four XMI bus request lines. The mapping can be random. The only condition is that each interrupt request level be mapped to a single BR line. The operating system can generate the desired mapping by configuring the mapping bits in the C3200 Interrupt Configuration Register. Following operating system initialization, the VME interrupt request levels are mapped to XMI interrupt priority levels, as shown in Table 6–2. The remaining three interrupt request levels must be disabled, as explained in the Interrupt Configuration Register.

Table 6–2 VME Interrupt Request Levels and XMI Defaults

VME Interrupt Request Level	Default XMI Interrupt Priority Level
IRQ7	BR7
IRQ6	BR6
IRQ5	BR5
IRQ4	BR4
IRQ3	Disabled
IRQ2	Disabled
IRQ1	Disabled

The C3200 allows software selection of its own interrupt request level through a write to bits <13:12> of the C3200 Device/Configuration Register. The default request level is BR7 (IPL17).

6.5 C3200 Interrupter/Interrupt Handler Selection

The VME protocol allows multiple interrupt handlers on the VMEbus. This is referred to as a distributed handler system. The VME also permits single-handler systems for the case where only one interrupt handler is on the bus. The DWMVA's interrupt handler allows the DWMVA adapter to respond to any of the VME interrupt request levels. This feature is necessary when the DWMVA is the only interrupt handler on the bus.

When the C3200 is configured as the only interrupt handler in the VME subsystem (the default configuration), all VME interrupts at selected levels are accepted by the C3200 and passed on to the XMI through an INTR transaction.

In a distributed handler system, the C3200 Interrupt Configuration Register bits <31:25> allow masking of any or all of the VME Interrupt Request Levels IRQ7*–IRQ1*. Depending on the state of these bits, the DWMVA can be made to accept only certain interrupts, letting another VME interrupt handler process others. The distributed handler system should be configured so that all interrupts that must be handled by the XMI processor have their corresponding mask bits set in the C3200 Interrupt Configuration Register. In this way, the C3200 interrupt handler can be configured to accept none or any combination of up to four VME interrupts.

6.6 VME Interrupter Types

The VME specification (IEEE P1014) defines two types of interrupters: RORA and ROAK. A RORA interrupter releases its interrupt request line following an access to its internal register in response to the interrupt. A ROAK interrupter releases its interrupt request line following the interrupt acknowledge cycle that acknowledges its interrupt. The C3200 module supports both types of interrupters, provided they are properly initialized in the Interrupt Configuration Register (see Chapter 7).

7 Registers

This chapter describes the DWMVA registers. The registers reside on both modules: T2018 and C3200. Registers required for an XMI interface reside on the T2018 module. Accordingly, the discussions are grouped in two sections:

- T2018 registers
- C3200 registers

Each section starts with a listing of the module registers, then proceeds to describe the individual registers. Table 7–1 indicates how the type of bits or fields is referred to in register descriptions.

Register addresses are referenced to a base address and stated as BB + *nn*, where BB is the nodespace starting address, and is computed by the equation:

BB = E180 0000 + (8 0000 * XMI Node ID)

for 32-bit addresses, or equation

BB = 2180 0000 + (8 0000 * XMI Node ID)

for 30-bit addresses.

Table 7–1 Types of Registers and Bits

Acronym	Type
RO	Read only.
R/W	Read/write
R/W, 0	Read/write; cleared on power-up.
R/W1	Read/write one to set; self-cleared; cannot be cleared by a write of zero.
R/W1C	Read/write one to clear; unaltered by a write of zero.
R/W1C, 0	Read/write one to clear; unaltered by a write of zero; cleared on power-up.
R/W1C, 1	Read/write one to clear; unaltered by a write of zero; set on power-up.
R0/W1	Read as zero/write one to set; self-cleared; cannot be cleared by a write of zero.
WO	Write only

7.1 T2018 Registers

The DWMVA registers on the T2018 module fall into two categories:

- XMI required registers
- T2018 specific registers

The XMI required registers must be implemented on each XMI node to establish communication between the node and the XMI. The unique registers implement node-specific functions.

Table 7–2 lists the T2018 registers and gives their offsets from the base address. Table 7–3 gives the values that should appear in the T2018 registers at power-up or following a hardware or software reset.

Table 7–2 T2018 Registers

Name	Mnemonic[1]	Address[2]
Device Register	XDEV	BB + 0000 0000
Bus Error Register	XBER	BB + 0000 0004
Failing Address Register	XFADR	BB + 0000 0008
Responder Error Address Register	AREAR	BB + 0000 000C
Error Summary Register	AESR	BB + 0000 0010
Interrupt Mask Register	AIMR	BB + 0000 0014
Implied Vector Interrupt Destination/Diagnostic Register	AIVINTR	BB + 0000 0018
Diagnostic 1 Register	ADG1	BB + 0000 001C
Utility Register	AUTLR	BB + 0000 0020
Control and Status Register	ACSR	BB + 0000 0024
Return Vector Register	ARVR	BB + 0000 0028
Failing Address Extension Register	XFAER	BB + 0000 002C
VME Error Address Register	ABEAR	BB + 0000 0030
Page Map Register (first location)	PMR	BB + 0000 0200
:	:	:
Page Map Register (last location)	PMR	BB + 0004 01FC

[1]X used as the first letter of the mnemonic indicates an XMI required register.

[2]BB refers to the base address of an XMI node (the address of the first location in the nodespace).

Table 7–3 Initialization Values of the T2018 Registers

Register	Initialization Bit States (Bin)	(Hex)
AREAR	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
AESR	1000 0000 0000 0000 0000 0000 0010 0000	8000 0020
AIMR	0000 0000 0000 X000 0000 0000 000X 0X00	000X 00XX
AIVINTR	XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX	XXXX XXXX
ADG1	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
AUTLR	0100 0011 1111 0000 0000 0000 0000 0000	43F0 0000
ACSR	0000 0000 0000 0000 0000 0001 1000 0000	0000 0180
ARVR	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
XFAER	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
ABEAR	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
ADG1	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
PMR	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000

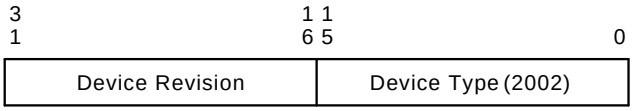
X denotes an indeterminate bit state.

Device Register (XDEV)

The Device Register contains information to identify the node and is loaded during node initialization. A zero value indicates an uninitialized node. This register should not be modified by the operating system.

ADDRESS

XMI nodespace base address + 0000 0000



msb−p412−91

bits<31:16>

Name: Device Revision

Mnemonic: DREV

Type: RO

DREV identifies the functional revision level of the module in hexadecimal. This field always reflects the letter revision of the module as follows:

T2018 Revision	DREV (decimal)	DREV (hex)
A*n*	1	0001
B*n*	2	0002
C*n*	3	0003
D*n*	4	0004
E*n*	5	0005
F*n*	6	0006
Not used	7	0007
H*n*	8	0008
Not used	9	0009
J*n*	10	000A
K*n*	11	000B
L*n*	12	000C
M*n*	13	000D
N*n*	14	000E
Not used	15	000F

bits<15:0>

Name: Device Type

Mnemonic: DTYPE

Type: RO, 2002 (hex)

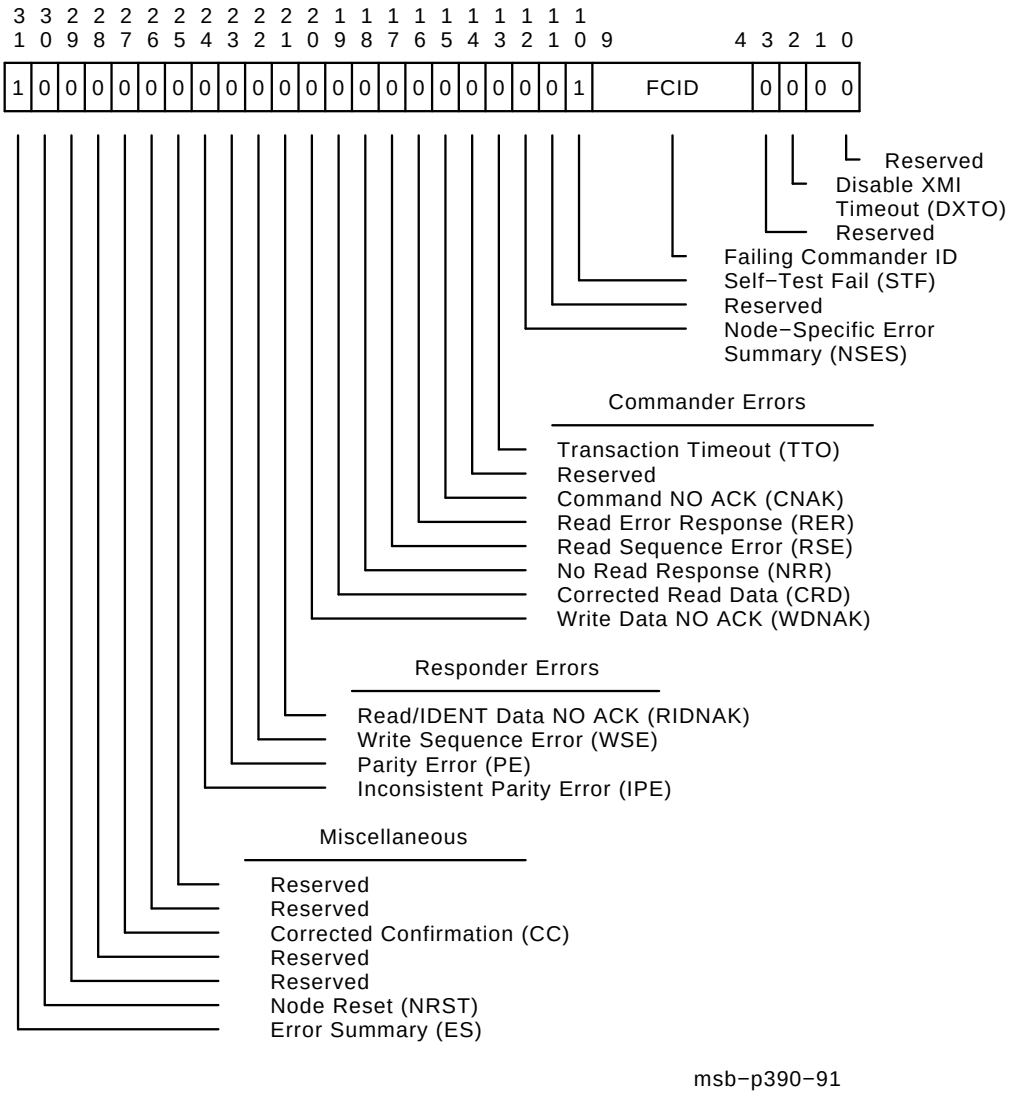
DTYPE identifies the type of node on the XMI. This field is 2002 (hex) for the DWMVA.

Bus Error Register (XBER)

The Bus Error Register stores the error bits flagged in operations involving the DWMVA and logs the failing commander ID. This register includes an Error Summary bit that is the logical OR of all the other error bits.

The status of this register remains locked up until software resets the error bit(s).

ADDRESS *XMI nodespace base address + 0000 0004*



msb−p390−91

bit<31>

Name: Error Summary

Mnemonic: ES

Type: RO, 1

ES represents the logical OR of the error bits in this register. It is set whenever any error bit listed in the following table is set.

XBER Bit	Mnemonic	Name
<27>	CC	Corrected Confirmation
<24>	IPE	Inconsistent Parity Error
<23>	PE	Parity Error
<22>	WSE	Write Sequence Error
<21>	RIDNAK	Read/IDENT Data NO ACK
<20>	WDNAK	Write Data NO ACK
<19>	CRD	Corrected Read Data
<18>	NRR	No Read Response
<17>	RSE	Read Sequence Error
<16>	RER	Read Error Response
<15>	CNAK	Command NO ACK
<13>	TTO	Transaction Timeout
<12>	NSES	Node-Specific Error Summary
<10>	STF	Self-Test Fail

bit<30>

Name: Node Reset

Mnemonic: NRST

Type: R/W, 0

Writing a one to NRST initiates a power-up reset of the VME. Reads to this bit location return zero. When NRST has a one written to it, the DWMVA:

- Resets all logic on the T2018 module to an initialized (power-up) state, regardless of what state it is in.
- Causes the C3200 to reset to an initialized (power-up) state and assert SYSRESET* on the VMEbus, thus initializing all devices on the VME.

While performing its node reset, the DWMVA does not affect the operation of the XMI bus.

bit<29>

Name: Node Halt
Mnemonic: NHALT
Type: RO, 0

Reserved; reads as zero.

bit<28>

Name: XMI BAD
Mnemonic: XBAD
Type: RO, 0

Reserved; reads as zero.

bit<27>

Name: Corrected Confirmation
Mnemonic: CC
Type: R/W1C, 0

CC sets when the DWMVA detects a single-bit CNF error (a single-bit CNF error is corrected automatically by the XCLOCK chip in the XMI Corner). If CC is set, ES (XBER<31>) is also set.

bit<26>

Name: XMI Trigger
Mnemonic: XTRIG
Type: R/W1C, 0

This bit indicates the state of the XMI TRIGGER line and is used by Digital during development.

bit<25>

Name: Write Error Interrupt
Mnemonic: WEI
Type: RO, 0

Reserved; reads as zero.

bit<24>

Name: Inconsistent Parity Error

Mnemonic: IPE

Type: R/W1C, 0

IPE sets when the DWMVA detects a parity error on an XMI cycle and at least one other node (the responder) detected good parity during the cycle (the confirmation for the cycle was ACK). This bit sets for all XMI inconsistent parity errors, whether the DWMVA is the target of the current XMI cycle or not.

bit<23>

Name: Parity Error

Mnemonic: PE

Type: R/W1C, 0

When set, PE bit indicates that the DWMVA detected a parity error on an XMI cycle.

bit<22>

Name: Write Sequence Error

Mnemonic: WSE

Type: R/W1C, 0

When set, WSE indicates that the DWMVA aborted a write transaction directed to it due to missing data cycles.

bit<21>

Name: Read/IDENT Data NO ACK

Mnemonic: RIDNAK

Type: R/W1C, 0

When set, RIDNAK indicates that a Read or IDENT data cycle (GRD*n*, CRD*n*, LOC, RER) transmitted by the DWMVA received a NO ACK confirmation.

bit<20>

Name: Write Data NO ACK

Mnemonic: WDNAK

Type: R/W1C, 0

When set, WDNAK indicates that a Write data cycle (GRD*n*, CRD*n*, LOC, RER) transmitted by the DWMVA received a NO ACK confirmation.

bit<19>

Name: Corrected Read Data

Mnemonic: CRD

Type: R/W1C, 0

When set, CRD bit indicates that the DWMVA received a CRD*n* read response.

bit<18>

Name: No Read Response

Mnemonic: NRR

Type: R/W1C, 0

When set, NRR indicates that a read transaction initiated by the DWMVA failed due to a read response timeout.

bit<17>

Name: Read Sequence Error

Mnemonic: RSE

Type: R/W1C, 0

When set, RSE indicates that a transaction initiated by the DWMVA failed due to a read sequence error.

bit<16>

Name: Read Error Response

Mnemonic: RER

Type: R/W1C, 0

When set, RER indicates that the DWMVA received a Read Error Response.

bit<15>

Name: Command NO ACK

Mnemonic: CNAK

Type: R/W1C, 0

When set, CNAK indicates that a command/address cycle transmitted by the DWMVA received a NO ACK confirmation and all reattempts have failed (retry timeout). This can be caused by either a reference to a nonexistent memory location or a command cycle parity error. This bit is set only if the reattempts fail.

CNAK does not set unless all retries have failed and TTO (XBER<13>) is set.

bit<14>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; reads as zero.

bit<13>

Name: Transaction Timeout

Mnemonic: TTO

Type: R/W1C, 0

When set, TTO indicates that one of the following has occurred:

- The DWMVA did not receive an XMI grant before the timeout period expired.
- The DWMVA received a NO ACK response to a command/address cycle and all reattempts have failed (CNAK set).
- The DWMVA did not receive read data in response to an ACKed read command before the timeout period expired (NRR set).

bit<12>

Name: Node-Specific Error Summary

Mnemonic: NSES

Type: RO, 0

The NSES sets when the DWMVA detects a node-specific error condition. The exact nature of the error is contained in the Error Summary Register (AESR) bits listed in the following table.

AESR Bit	Mnemonic	Name
<31>	None	DWMVA Cable OK
<14>	ME	Multiple Errors
<13>	CORR PMR ECC ERR	Correctable PMR ECC Error
<12>	UNCORR PMR ECC ERR	Uncorrectable PMR ECC Error
<11>	IPFN	Invalid PFN
<10>	CORR DMA ECC ERR	Correctable DMA ECC Error
<9>	UNCORR DMA ECC ERR	Uncorrectable DMA ECC Error
<8>	INV VME ADR	Invalid VME Address
<7>	IE	Internal Error
<6>	None	I/O Write Failure
<5>	None	VME AC LO
<4>	IBUS DMA-A DATA PE	IBUS DMA-A Data Parity Error
<3>	IBUS DMA-A C/A PE	IBUS DMA-A Command/Address Parity Error
<2>	IBUS DMA-B DATA PE	IBUS DMA-B Data Parity Error
<1>	IBUS DMA-A C/A PE	IBUS DMA-B Command/Address Parity Error
<0>	IBUS I/O RD PE	IBUS I/O Read Data Parity Error

bit<11>

Name: Extended Test Fail

Mnemonic: ETF

Type: RO, 0

Reserved; reads as zero.

bit<10>

Name: Selt-Test Fail

Mnemonic: STF

Type: R/W1C, 1

When set, STF indicates that the DWMVA has not yet passed its self-test. The CPU node clears this bit upon successful completion of the DWMVA self-test.

bits<9:4>

Name: Failing Commander ID

Mnemonic: FCID

Type: RO, 0

FCID logs the commander ID of a failing transaction. FCID is set only if all reattempts fail.

bit<3>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; reads as zero.

bit<2>

Name: Disable XMI Timeout

Mnemonic: DXTO

Type: R/W, 0

When set, DXTO disables the transaction timeout counter, causing Timeout Limit (AUTLR<23:20>) to be ignored. The DWMVA either retries a transaction on the XMI or waits for returning DMA read data in response to a successful XMI read for an indefinite period. The DWMVA never aborts the transaction or sets TTO.

bits<1:0>

Name: Reserved

Mnemonic: None

Type: RO, 0

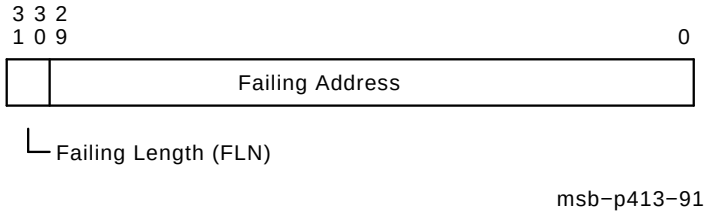
Reserved; read as zero.

Failing Address Register (XFADR)

The Failing Address Register logs address and length information associated with a failing transaction. The following error bits, when set, lock this register and XFAER:

- Write Data NO ACK, XBER<20>
- No Read Response, XBER<18>
- Read Sequence Error, XBER<17>
- Read Error Response, XBER<16>
- Command NO ACK, XBER<15>
- Transaction Timeout, XBER<13>
- Internal Error, AESR<7>

ADDRESS *XMI nodespace base address + 0000 0008*



msb−p413−91

bits<31:30>

Name: Failing Length

Mnemonic: FLN

Type: RO, 0

The FLN logs the value of XMI D<31:30> during the command/address cycle of a failed XMI commander transaction. This field is loaded on every command/address cycle issued by the DWMVA. It is locked, however, only after all retries of the transaction fail. FLN unlocks when the error that caused the lock is cleared.

bits<29:0>

Name: Failing Address

Mnemonic: None

Type: RO, 0

The Failing Address field logs the value of XMI D<29:0> during the command/address cycle of a failing transaction. Failing Address is loaded on every command/address cycle issued by the DWMVA. It is locked, however, only after all retries of the transaction fail. FLN unlocks when the error that caused the lock is cleared.

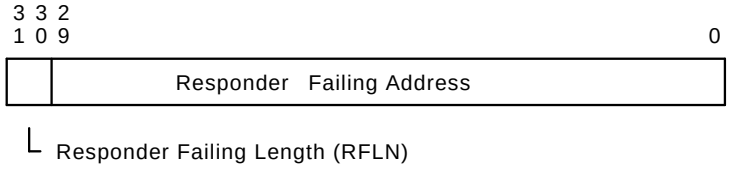
Responder Error Address Register (AREAR)

The Responder Error Address Register (AREAR) logs the failing address of an I/O write, read, or IDENT from an XMI commander node directed to the DWMVA or the VME. AREAR is loaded when the DWMVA ACKs the XMI's command/address cycle.

AREAR is locked when the DWMVA is unable to complete the requested operation because of a detected error. The following error bits, when set, lock this register, the Responder Failing ID (AESR<25:20>), and the Responder Failing Command (AESR<19:16>):

- Write Sequence Error, XBER<22>
- Read/IDENT Data NO ACK, XBER<21>
- Uncorrectable PMR ECC Error, AESR<13>
- Correctable PMR ECC Error, AESR<12>
- Internal Error, AESR<7>
- I/O Write Failure, AESR<6>
- IBUS I/O Read Data Parity Error, AESR<0>

ADDRESS *XMI nodespace base address + 0000 000C*



msb−p414−91

bits<31:30>

Name: Responder Failing Length

Mnemonic: RFLN

Type: RO, 0

RFLN loads XMI D<31:30> during the cycle that the DWMVA accepts the command/address from an XMI commander. This field locks only if the transaction fails and unlocks when all the error conditions clear.

bits<29:0>

Name: Responder Failing Address

Mnemonic: None

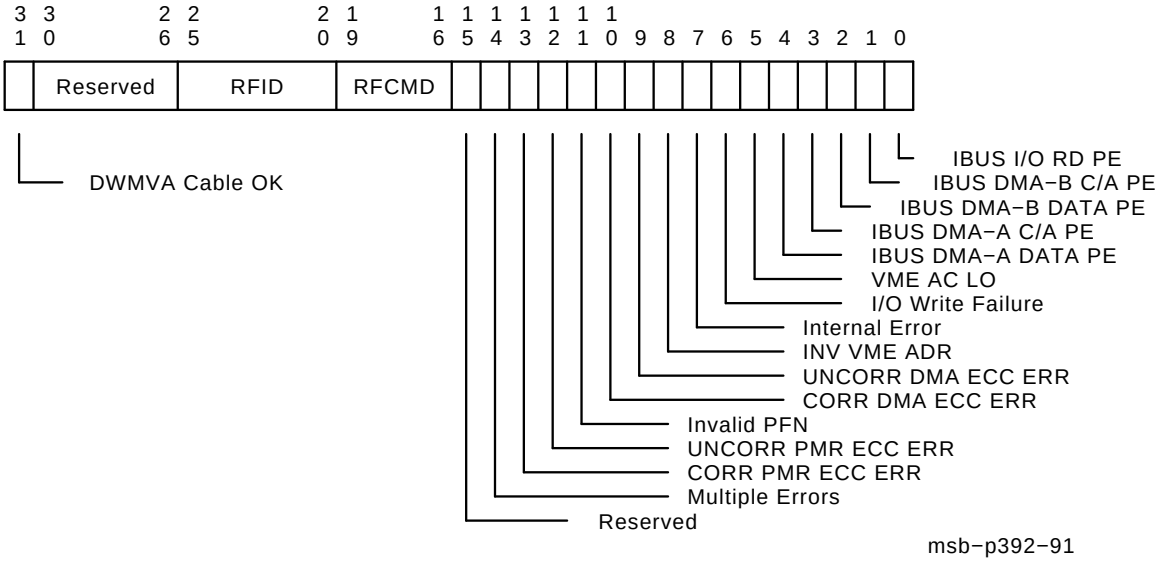
Type: RO, 0

Responder Failing Address logs the value of XMI D<29:0> during the cycle that the DWMVA accepts the command/address from an XMI commander. It locks only if the transaction fails and unlocks when all the error conditions clear.

Error Summary Register (AESR)

The Error Summary Register logs error conditions related to the T2018 module. This register also captures the Failing Command, Failing Commander ID, and flags on an unexplained error detection on the T2018.

ADDRESS *XMI nodespace base address + 0000 0010*



bit<31>

Name: DWMVA Cable OK

Mnemonic: None

Type: RO, 1

DWMVA Cable OK sets to one on initialization if the four IBUS cables are correctly connected and if the C3200 has dc power from the VME backplane. If DWMVA Cable OK clears and the C3200 has VME dc power, then one or more of the cables is not connected or is incorrectly installed.

bits<30:26>

Name: Reserved
Mnemonic: None
Type: RO, O

Reserved; read as zero.

bits<25:20>

Name: Responder Failing ID
Mnemonic: RFID
Type: RO, 0

RFID logs the XMI node ID of a failed DWMVA I/O write, I/O read, or XMI IDENT transaction. The DWMVA loads this field every time it accepts a command/address cycle. This field locks if the transaction fails and unlocks when the error condition clears.

bits<19:16>

Name: Responder Failing Command
Mnemonic: RFCMD
Type: RO, 0

RFCMD logs the XMI command of a failed DWMVA I/O write, I/O read, or XMI IDENT transaction. The DWMVA loads this field every time it accepts a command/address cycle. This field locks if the transaction fails and unlocks when the error condition clears.

bit<15>

Name: Reserved
Mnemonic: None
Type: RO, 0

Reserved; reads as zero.

bit<14>

Name: Multiple Errors
Mnemonic: ME
Type: R/W1C, 0

When set, ME indicates that an error(s) occurred in a second transaction before software acknowledged and cleared the error(s) from the first transaction. The following bits have no effect on ME:

- VME AC LO, AESR<5>
- Self-Test Fail, XBER<10>

bit<13>

Name: Correctable PMR ECC Error

Mnemonic: CORR PMR ECC ERR

Type: R/W1C, 0

When set, CORR PMR ECC ERR indicates that a correctable ECC error occurred during an I/O read access to a PMR. The set state of this bit locks the AREAR and generates an interrupt if INTR CORR ECC ERR (AIMR<10>) is set.

bit<12>

Name: Uncorrectable PMR ECC Error

Mnemonic: UNCORR PMR ECC ERR

Type: R/W1C, 0

When set, UNCORR PMR ECC ERR indicates that an uncorrectable ECC error occurred during an I/O read access to a PMR. The set state of this bit locks the AREAR and generates an interrupt if INTR UNCORR ECC ERR (AIMR<9>) is set.

bit<11>

Name: Invalid PFN

Mnemonic: IPFN

Type: R/W1C, 0

When set, IPFN indicates that the Valid bit of a PMRE accessed during a DMA transaction was not a one. The set state of IPFN causes ABEAR to lock the VME address of the failed DMA transaction and an interrupt request is generated if INTR IPFN (AIMR<11>) is set.

If the transaction was a DMA write, or otherwise might cause a data loss, an IVINTR is generated if Enable IVINTR Transactions (AIMR<31>) is set.

bit<10>

Name: Correctable DMA ECC Error

Mnemonic: CORR DMA ECC ERR

Type: R/W1C, 0

When set, CORR DMA ECC ERR indicates that a fetch from the PMR during a DMA address translation detected and corrected an error. The set state of this bit locks the ABEAR. CORR DMA ECC ERR sets only when the DWMVA operates in an address translation mode. When this bit sets, an interrupt is generated if INTR CORR ECC ERR (AIMR<10>) is set.

bit<9>

Name: Uncorrectable DMA ECC Error

Mnemonic: UNCORR DMA ECC ERR

Type: R/W1C, 0

When set, UNCORR DMA ECC ERR indicates that a fetch from the PMR during a DMA address translation detected an uncorrectable error. The set state of this bit locks the ABEAR. UNCORR DMA ECC ERR sets only when the DWMVA operates in an address translation mode. When this bit sets, an interrupt is generated if INTR UNCORR ECC ERR (AIMR<9>) is set.

If the transaction was a DMA write, or otherwise might cause a data loss, an IVINTR is generated if Enable IVINTR Transactions (AIMR<31>) is set.

bit<8>

Name: Invalid VME Address

Mnemonic: INV VME ADR

Type: R/W1C, 0

When set, INV VME ADR indicates that the VME address for the requested DMA transaction is invalid (not in memory space).

In no translation mode or 40-bit address translation mode using 8-Kbyte page size, a DMA transaction is invalid if VME address bit A29 equals one.

In 40-bit address translation mode using 4-Kbyte page size, a DMA transaction is invalid if VME address bits A28–A29 do not equal zero.

In 40-bit address translation mode, a DMA transaction is invalid if VME address bits A25–A28 do not equal zero.

The set state of INV VME ADR causes the ABEAR to lock the VME address of the failed transaction. An interrupt request is generated if INTR INV VME ADR (AIMR<8>) is set.

If the transaction was a DMA write, or otherwise might cause a data loss, an IVINTR with WRT ERROR INT set in the Type field is generated if Enable IVINTR Transactions (AIMR<31>) is set.

bit<7>

Name: Internal Error

Mnemonic: IE

Type: R/W1C, 0

IE is set when an UNEXPLAINED internal error to the T2018 gate array is detected. This error generally indicates a hardware problem where control logic has encountered UNDEFINED conditions. When IE is set, the DWMVA issues an IVINTR transaction with WRT ERROR INT set in the Type field, if Enable IVINTR Transactions (AIMR<31>) is set.

The following conditions cause IE to set:

- A state machine in the T2018 gate array reaches an illogical state.
- A parity error is detected internal to the gate array on the transfer of PMR write data for a PMR write request, indicating that the PMR location's data is corrupt. This error condition also causes I/O Write Failure (AESR<6>) to set.
- A parity error is detected on the transfer of write data for a loopback write command during a loopback mode. This also causes the loopback write transaction to abort and I/O Write Failure (AESR<6>) to set.
- A parity error is detected on the return of DMA read data that is looped back as CPU read data during a loopback mode. This also causes the loopback read transaction to abort.

bit<6>

Name: I/O Write Failure

Mnemonic: None

Type: R/W1C, 0

The I/O Write Failure bit sets if the C3200 module is unable to complete an I/O write transaction to either its register space or to VME address space. The set state of this bit causes the generation of an IVINTR transaction with WRT ERROR INT set in the Type field, if Enable IVINTR Transactions (AIMR<31>) is set. Software uses this bit and other error bits to determine the cause of a DWMVA-generated IVINTR transaction.

When I/O Write Failure is set, the contents of the T2018 Responder Error Address Register lock.

bit<5>

Name: VME AC LO

Mnemonic: None

Type: R/W1C, 1

The VME AC LO bit sets when the AC FAIL L signal is asserted, indicating that the VME power has fallen below specifications. The DWMVA issues an IVINTR with WRT ERROR INT set in the Type field when AC FAIL L is asserted, if Enable IVINTR Transactions (AIMR<31>) is set, so that software can determine the cause of this IVINTR transaction. Software then clears VME AC LO in the interrupt service routine that executes as a result of the IVINTR.

The following conditions cause VME AC LO to set:

- An XMI power-up sequence.
- Software sets NRST (XBER<30>) to initiate a node reset.
- Software sets Control Reset (ACSR<30>) to initiate a diagnostics node reset.
- VME power falls below specifications, causing a VME power failure.
- Software causes a VME node reset to execute a remote booting routine.

This bit is cleared by self-test at power-up.

bit<4>

Name: IBUS DMA-A Data Parity Error

Mnemonic: IBUS DMA-A DATA PE

Type: R/W1C, 0

IBUS DMA-A DATA PE sets when the T2018 module detects a parity error on the IBUS when the C3200 module was loading a DMA-A data buffer location. When this bit is set, the DWMVA issues an IVINTR with WRT ERROR INT set in the Type field, if Enable IVINTR Transactions (AIMR<31>) is set.

bit<3>

Name: IBUS DMA-A C/A Parity Error

Mnemonic: IBUS DMA-A CA PE

Type: R/W1C, 0

IBUS DMA-A C/A PE bit sets when the T2018 module detects a parity error on the IBUS when the C3200 module was loading a DMA-A data buffer command/address location. When this bit is set, and the failing DMA transaction is a write or interrupt, the DWMVA issues an IVINTR with WRT ERROR INT set in the Type field. The DWMVA issues an error interrupt if INTR DMA-A CA PE (AIMR<3>) is set.

bit<2>

Name: IBUS DMA-B Data Parity Error

Mnemonic: IBUS DMA-B DATA PE

Type: R/W1C, 0

IBUS DMA-B DATA PE sets when the T2018 module detects a parity error on the IBUS when the C3200 module was loading a DMA-B data buffer location. When this bit is set, the DWMVA issues an IVINTR with WRT ERROR INT set in the Type field, if Enable IVINTR Transactions (AIMR<31>) is set.

bit<1>

Name: IBUS DMA-B C/A Parity Error

Mnemonic: IBUS DMA-B CA PE

Type: R/W1C, 0

IBUS DMA-B CA PE sets when the T2018 module detects a parity error on the IBUS when the C3200 module was loading a DMA-B data buffer command/address location. When this bit is set, and the failing DMA transaction is a write or interrupt, the DWMVA issues an IVINTR with WRT ERROR INT set in the Type field. The DWMVA issues an error interrupt if this error bit is set and INTR DMA-B CA PE (AIMR<1>) is also set.

bit<0>

Name: IBUS I/O Read Data Parity Error

Mnemonic: IBUS I/O RD PE

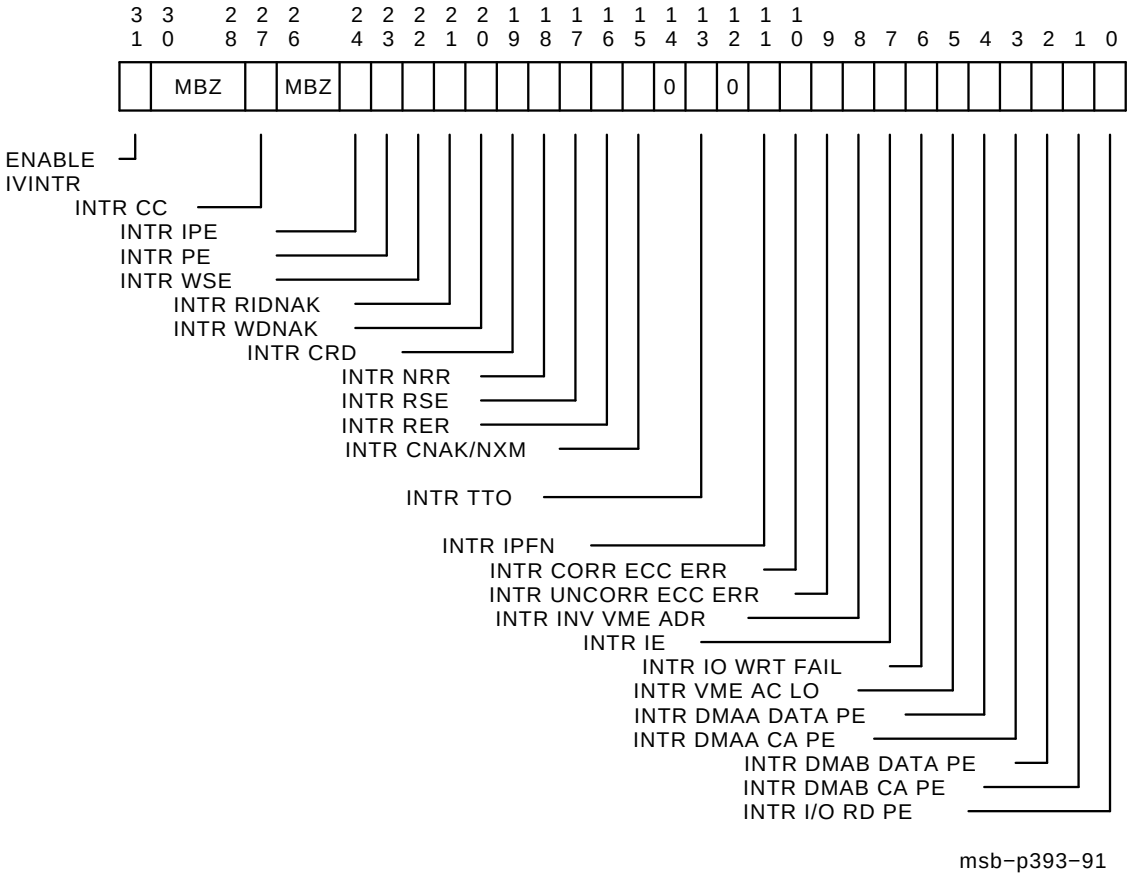
Type: R/W1C, 0

IBUS I/O RD PE sets when the T2018 module detects a parity error on the IBUS when the C3200 module was loading the I/O data location during an XMI commander-initiated I/O read or IDENT. The DWMVA issues a Read Error Response (RER) to the commander when the error occurs during an I/O read transaction. If the error occurs during an IDENT transaction, the DWMVA returns the contents of the Return Vector Register (ARVR) as the vector. The DWMVA issues an interrupt to the XMI when this bit is set, if INTR I/O RD PE (AIMR<0>) is also set.

Interrupt Mask Register (AIMR)

The Interrupt Mask Register enables/disables the generation of an error interrupt transaction when the corresponding error bit in either the Bus Error Register (XBER) or the Error Summary Register (AESR) is set.

ADDRESS *XMI nodespace base address + 0000 0014*



msb−p393−91

bit<31>

Name: Enable IVINTR Transactions

Mnemonic: ENABLE IVINTR

Type: R/W, 0

When ENABLE IVINTR is set and IVINTR Destination Register is properly configured, IVINTRs are enabled and can be issued on the XMI bus. The following error conditions generate IVINTRs:

- Invalid PFN, AESR<11>, only if the failing transaction was a DMA write
- Uncorrectable DMA ECC error, AESR<9>, only if the failing transaction was a DMA write
- Invalid VME address, AESR<8>, only if the failing transaction was a DMA write
- Internal Error, AESR<7>
- I/O Write Failure, AESR<6>
- VME AC LO, AESR<5>
- IBUS DMA-A Data Parity Error, AESR<4>
- IBUS DMA-A C/A Parity Error, AESR<3>, only if the failing transaction was a DMA write
- IBUS DMA-B Data Parity Error, AESR<2>
- IBUS DMA-B C/A Parity Error, AESR<1>, only if the failing transaction was a DMA write
- Transaction Timeout, XBER<13>, only if the failing transaction was a DMA write

CAUTION: This bit must be set to ensure proper error reporting in the case of asynchronous write failures and occurrence of a pending VME powerfail (VME power failure not initiated by XMI AC LO, XMI DC LO, or DWMVA node reset).

bits<30:28>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; must be zero.

bit<27>

Name: Interrupt on Corrected Confirmation
Mnemonic: INTR CC
Type: R/W, 0

If INTR CC IS SET, the DWMVA generates an interrupt when Corrected Confirmation (XBER<27>) sets.

bits<26:25>

Name: Reserved
Mnemonic: None
Type: RO, 0

Reserved; must be zero.

bit<24>

Name: Interrupt on Inconsistent Parity Error
Mnemonic: INTR IPE
Type: R/W, 0

If INTR IPE is set, the DWMVA generates an interrupt when Inconsistent Parity Error (XBER<24>) sets.

bit<23>

Name: Interrupt on Parity Error
Mnemonic: INTR PE
Type: R/W, 0

If INTR PE is set, the DWMVA generates an interrupt when Parity Error (XBER<23>) sets.

bit<22>

Name: Interrupt on Write Sequence Error
Mnemonic: INTR WSE
Type: R/W, 0

If INTR WSE is set, the DWMVA generates an interrupt when Write Sequence Error (XBER<22>) sets.

bit<21>

Name: Interrupt on Read/IDENT NO ACK
Mnemonic: INTR RIDNAK
Type: R/W, 0

If INTR RIDNAK is set, the DWMVA generates an interrupt when Read/IDENT NO ACK (XBER<21>) sets.

bit<20>

Name: Interrupt on Write Data NO ACK

Mnemonic: INTR WDNAK

Type: R/W, 0

If INTR WDNAK is set, the DWMVA generates an interrupt when Write Data NO ACK (XBER<20>) sets.

bit<19>

Name: Interrupt on Corrected Read Data

Mnemonic: INTR CRD

Type: R/W, 0

If INTR CRD is set, the DWMVA generates an interrupt when Corrected Read Data (XBER<19>) sets.

bit<18>

Name: Interrupt on No Read Response

Mnemonic: INTR NRR

Type: R/W, 0

If INTR NRR is set, the DWMVA generates an interrupt when No Read Response (XBER<18>) sets.

bit<17>

Name: Interrupt on Read Sequence Error

Mnemonic: INTR RSE

Type: R/W, 0

If INTR RSE is set, the DWMVA generates an interrupt when Read Sequence Error (XBER<17>) sets.

bit<16>

Name: Interrupt on Read Error Response

Mnemonic: INTR RER

Type: R/W, 0

If INTR RER is set, the DWMVA generates an interrupt when Read Error Response (XBER<16>) sets.

bit<15>

Name: Interrupt on Command NO ACK

Mnemonic: INTR CNAK

Type: R/W, 0

If INTR CNAK is set, the DWMVA generates an interrupt when Command NO ACK (XBER<15>) sets.

bit<14>

Name: Reserved
Mnemonic: None
Type: RO, 0

Reserved; must be zero.

bit<13>

Name: Interrupt on Transaction Timeout
Mnemonic: INTR TTO
Type: R/W, 0

If INTR TTO is set, the DWMVA generates an interrupt when Transaction Timeout (XBER<13>) sets.

bit<12>

Name: Reserved
Mnemonic: None
Type: RO, 0

Reserved; must be zero.

bit<11>

Name: Interrupt on Invalid PFN
Mnemonic: INTR IPFN
Type: R/W, 0

If INTR IPFN is set, the DWMVA generates an interrupt when Invalid PFN (AESR<11>) sets.

bit<10>

Name: Interrupt on Correctable ECC Error
Mnemonic: INTR COR ECC ERR
Type: R/W, 0

If INTR COR ECC ERR is set, the DWMVA generates an interrupt when Correctable PMR ECC Error (AESR<13>) or Correctable DMA ECC Error (AESR<10>) sets.

bit<9>

Name: Interrupt on Uncorrectable ECC Error
Mnemonic: INTR UNCOR ECC ERR
Type: R/W, 0

If INTR UNCOR ECC ERR is set, the DWMVA generates an interrupt when Uncorrectable PMR ECC Error (AESR<12>) or Uncorrectable DMA ECC Error (AESR<9>) sets.

bit<8>

Name: Interrupt on Invalid VME Address

Mnemonic: INTR INV VME ADR

Type: R/W, 0

If INTR INV VME ADR is set, the DWMVA generates an interrupt when Invalid VME Address (AESR<8>) sets.

bit<7>

Name: Interrupt on Internal Error

Mnemonic: INTR IE

Type: R/W, 0

If INTR IE is set, the DWMVA generates an interrupt when Internal Error (AESR<7>) sets.

bit<6>

Name: Interrupt on I/O Write Failure

Mnemonic: INTR IO WRT FAIL

Type: R/W, 0

If INTR IO WRT FAIL is set, the DWMVA generates an interrupt when I/O Write Failure (AESR<6>) sets.

bit<5>

Name: Interrupt on VME AC LO

Mnemonic: INTR VME AC LO

Type: R/W, 0

If INTR VME AC LO is set, the DWMVA generates an interrupt when VME AC LO (AESR<5>) sets.

bit<4>

Name: Interrupt on DMA-A Data Parity Error

Mnemonic: INTR DMA-A DATA PE

Type: R/W, 0

If INTR DMA-A DATA PE is set, the DWMVA generates an interrupt when IBUS DMA-A Data Parity Error (AESR<4>) sets.

bit<3>

Name: Interrupt on IBUS DMA-A C/A Parity Error

Mnemonic: INTR DMA-A CA PE

Type: R/W, 0

If INTR DMA-A CA PE is set, the DWMVA generates an interrupt when IBUS DMA-A C/A Parity Error (AESR<3>) sets.

bit<2>

Name: Interrupt on DMA-B Data Parity Error

Mnemonic: INTR DMA-B DATA PE

Type: R/W, 0

If INTR DMA-B DATA PE is set, the DWMVA generates an interrupt if IBUS DMA-B Data Parity Error (AESR<2>) sets.

bit<1>

Name: Interrupt on IBUS DMA-B C/A Parity Error

Mnemonic: INTR DMA-B CA PE

Type: R/W, 0

If INTR DMA-B CA PE is set, the DWMVA generates an interrupt if IBUS DMA-B C/A Parity Error (AESR<1>) sets.

bit<0>

Name: Interrupt on IBUS I/O Read Data Parity Error

Mnemonic: INTR I/O RD PE

Type: R/W, 0

If INTR I/O RD PE is set, the DWMVA generates an interrupt if IBUS I/O Read Data Parity Error (AESR<0>) sets.

Implied Vector Interrupt Destination/Diagnostic Register (AIVINTR)

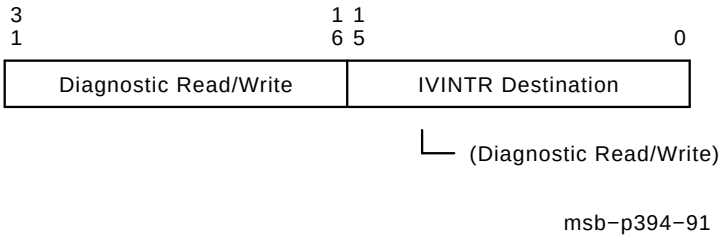
The Implied Vector Interrupt Destination/Diagnostic Register is used for two different purposes:

- As a mask during DWMVA-initiated transactions
- As a data path tester during diagnostics

DWMVA-initiated transactions use bits <15:0> only of the register to define the Implied Vector Interrupt (IVINTR) destination. Diagnostics use all 32 bits.

ADDRESS

XMI nodespace base address + 0000 0018



bits<31:0>

Name: Diagnostic Read/Write

Mnemonic: None

Type: R/W, Undefined

The Diagnostic Read/Write bits are used by diagnostics to verify the integrity of the T2018 main data path. Diagnostics should ensure that the processor's IPL level is raised above IPL 30, so that in case the occurrence of an error causes the T2018 to issue an IVINTR transaction, an unexpected interrupt does not result.

bits<15:0>

Name: IVINTR Destination

Mnemonic: None

Type: R/W, 0

The IVINTR Destination mask field determines which nodes on the XMI will be targeted by the DWMVA when the DWMVA issues an IVINTR transaction. Each bit, when set, selects the corresponding node to participate in the IVINTR transaction. Multiple bits can be set to engage simultaneously as many XMI nodes as desired. When bits <15:0> are used as an IVINTR destination mask, bits <31:16> must be written as zero.

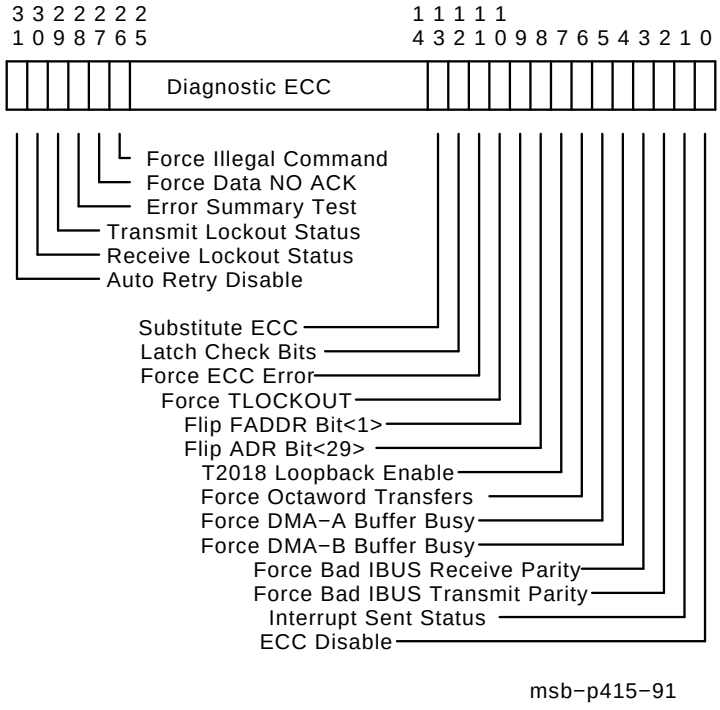
Diagnostic 1 Register (ADG1)

The Diagnostic 1 Register is used by diagnostics to test parity and other logic on the T2018 module and the IBUS.

CAUTION: This register must NOT be accessed by the user. It is reserved for use by Digital service personnel.

ADDRESS

XMI nodespace base address + 0000 001C



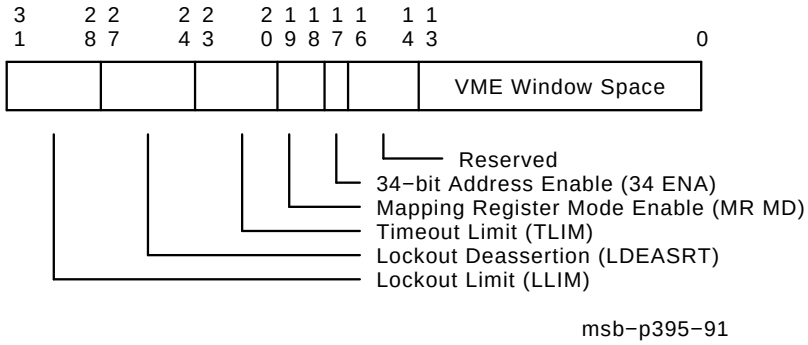
msb−p415−91

Utility Register (AUTLR)

The Utility Register contains fields for software programmable selection of timeout values and for moving the DWMVA window address space to an I/O address range other than the power-up or reset default values.

ADDRESS

XMI nodespace base address + 0000 0020



msb−p395−91

bits<31:28>

Name: Lockout Limit

Mnemonic: LLIM

Type: R/W, 4 (hex)

The value loaded into LLIM determines the maximum number of consecutive IREADS that the DWMVA retries, before it asserts the XMI LOCKOUT L signal.

The default value loaded into this field at power-up and at node reset is 4 (hex). Software can load the field with a value between 0 and F (hex) at system initialization. The values for this field are as follows:

LLIM(hex)	IREAD Attempts	LLIM(hex)	IREAD Attempts
0	1	8	8
1	1	9	9
2	2	A	10
3	3	B	11
4	4 (default)	C	12
5	5	D	13
6	6	E	14
7	7	F	15

bits<27:24>

Name: Lockout Deassertion

Mnemonic: LDEASRT

Type: R/W, 3 (hex)

The value loaded into LDEASRT determines the maximum time XMI LOCKOUT L can remain asserted on the XMI. This field enables the lockout deassertion time to vary between 1 to 15 ms. The default value at power-up and at node reset is 2 to 3 ms. Software can load this field with a value between 1 and F (hex) at system initialization. The values for this field are as follows:

LDEASRT(hex)	Timeout (ms)	LDEASRT(hex)	Timeout (ms)
0	00-1	8	7–8
1	0–1	9	8–9
2	1–2	A	9–10
3	2–3 (default)	B	10–11
4	3–4	C	11–12
5	4–5	D	12–13
6	5–6	E	13–14
7	6–7	F	14–15

bits<23:20>

Name: Timeout Limit

Mnemonic: TLIM

Type: R/W, F (hex)

The value loaded into TLIM determines the time that the DWMVA retries a transaction on the XMI or waits for returning read data in response to a successful XMI read command before aborting the transaction and setting the Transaction Timeout (TTO) bit in the XBER.

The DWMVA has two timeout limits, a normal timeout limit that ranges from 0 to 15 ms, and a short timeout limit that ranges from 0 to 960 μs. The value of Short Timeout Enable (ACSR<9>) determines whether the DWMVA uses the normal or short timeout. Software can load the field with a value between 0 and F (hex) at system initialization. The default value at power-up and at node reset is 3 (hex), for a timeout of 14 to 15 ms.

The programmable values of timeout are as follows:

TLIM (hex)	ACSR<9>=0 Normal Timeout (ms)	ACSR<9>=1 Short Timeout (μs)
0	0–1	0–64
1	0–1	0–64
2	1–2	64–128
3	2–3	128–192
4	3–4	192–256
5	4–5	256–320
6	5–6	320–384
7	6–7	384–448
8	7–8	448–512
9	8–9	512–576
A	9–10	576–640
B	10–11	640–704
C	11–12	704–768
D	12–13	768–832
E	13–14	832–896
F	14–15 (default)	896–960

bits<19:18>

Name: Mapping Register Mode Enable

Mnemonic: MR MD

Type: R/W, 0

MR MD selects the translation mode (including no translation) to convert a VME address into an XMI physical address. Software sets up this field at system initialization. The T2018 defaults to no translation mode after a power-up or node reset. This field selects the translation mode as follows:

MR MD<19:18>	Translation Mode
00	No translation (Default)
01	40-bit address translation using 512-byte page sizes
10	40-bit address translation using 4-Kbyte page sizes
11	40-bit address translation using 8-Kbyte page sizes

The 34-bit translation mode is selected through bit <17> of this register.

bit<17>

Name: 34-Bit Address Enable

Mnemonic: 34 ENA

Type: R/W, 0

When set, 34 ENA masks the upper 5 bits of the translated address. Thus, the T2018 transmits a 34-bit VAX address with a DMA command.

This option is only valid while the T2018 is in the 40-bit VAX address translation mode using 512-byte page sizes (AUTLR<19:18> = 1). This bit has no effect if set during another translation mode.

bits<16:14>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; read as zero.

bits<13:0>

Name: VME Window Space

Mnemonic: VWS

Type: R/W, 0

The 14-bit VWS enables software to reconfigure the DWMVA I/O address space to any 32-Mbyte address region within the 512-Mbyte range of the I/O adapter address space.

Referenced to the VME window address space are stated as bb + *nn*, where bb is the VME I/O window space base address.

VME I/O window space is normally accessed if VWS ENA (bit <5>) is clear in the Control and Status Register (ACSR). The VME I/O window space base address (bb) is computed from the equation:

bb = E000 0000 + (200 0000*XMI node ID) + (2000*VME node ID)

If VWS ENA is set, the VME I/O window space base address is computed from the equation:

bb = E000 0000 + (200 0000*AUTLR<13:0>) + (2000*VME node ID)

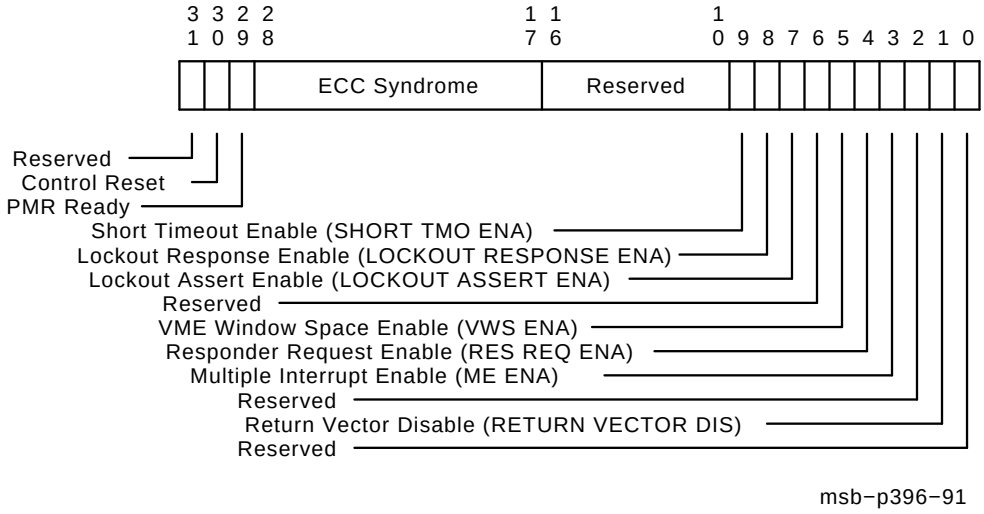
Note that using node 0 in VME window space is illegal. Therefore, AUTLR<13:0> should be set up before VWS ENA is set in the ACSR.

If an I/O command directed to the VME is not targeted for a VME CSR, and ACSR<5> is set while AUTLR<13:0> field equals zero, the I/O command is NO ACKed by the DWMVA.

Control and Status Register (ACSR)

The Control and Status Register contains control and status information for the T2018 module operation.

ADDRESS *XMI nodespace base address + 0000 0024*



msb−p396−91

bit<31>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; reads as zero.

bit<30>

Name: Control Reset

Mnemonic: CTL RESET

Type: WO, 0

CTL RESET is used for diagnostic purposes only. Writing a one to this bit initiates a partial node reset. Reads of this bit location return zero.

Writing a one to this bit causes the DWMVA to execute a control reset, even if it is in a hung state or busy processing another transaction.

When this bit is set to one, the DWMVA:

- Resets all logic on the T2018 module, including the PMRs (except the I/O registers), to an initialized (power-up) state.
- Resets the C3200 module and the VME.

- Disables IVINTRs by resetting AIMR<31>, the IVINTR enable bit.

The state of CTL RESET does not affect XMI operations.

bit<29>

Name: PMR Ready

Mnemonic: None

Type: RO, 0

When clear, PMR Ready prevents access of the PMRs from the XMI and VME (that is, address translation is disabled).

This bit is set when PMR INIT IN PROG H is deasserted. It is cleared on power-up, XMI node reset, or upon assertion of PMR INIT IN PROG H.

bits<28:17>

Name: ECC Syndrome

Mnemonic: None

Type: RO, 0

The ECC Syndrome field is loaded with the ECC syndrome bits when an ECC error is detected. This field is locked when an ECC error is detected and remains locked until the error condition has been cleared. Error bits that lock this field are given in the following table.

Register<bit>	Name
AESR<13>	Correctable PMR ECC Error
AESR<12>	Uncorrectable PMR ECC Error
AESR<10>	Correctable DMA ECC Error
AESR<9>	Uncorrectable DMA ECC Error

bits<16:10>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; read as zero.

bit<9>

Name: Short Timeout Enable

Mnemonic: SHORT TMO ENA

Type: R/W, 0

When set, SHORT TMO ENA enables the DWMVA to use a smaller timeout range, from 0 to 960 μs, instead of the normal timeout range of 0 to 15 ms.

bit<8>

Name: Lockout Response Enable

Mnemonic: LOCKOUT RESPONSE ENA

Type: R/W, 1

When set, LOCKOUT RESPONSE ENA enables the DWMVA to respond to the XMI LOCKOUT L signal on the XMI. The DWMVA defaults to the Full XMI Lockout mode after a power-up or a node reset.

bit<7>

Name: Lockout Assert Enable

Mnemonic: LOCKOUT ASSERT ENA

Type: R/W, 1

When set, LOCKOUT ASSERT ENA enables the DWMVA to assert the XMI LOCKOUT L signal. The DWMVA defaults to the Full XMI Lockout mode after a power-up or a node reset.

bit<6>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; reads as zero.

bit<5>

Name: VME Window Space Enable

Mnemonic: VWS ENA

Type: R/W, 0

When set, VWS ENA enables the VME Window Space field (AUTLR<13:0>), allowing software to reconfigure the VME I/O address space into any 32-Mbyte region of the 512-Mbyte I/O address space.

bit<4>

Name: Responder Request Enable

Mnemonic: RES REQ ENA

Type: R/W, 0

When set, RES REQ ENA bit causes the DWMVA to arbitrate for the XMI as a commander using XMI RES(n) REQ L instead of XMI CMD(n) REQ L.

If XMI SUP L is asserted when the DWMVA wins the XMI, it aborts the transaction and retries again when XMI SUP L is deasserted, allowing the DWMVA to gain a higher priority than other XMI commander nodes.

bit<3>

Name: Multiple Interrupt Enable

Mnemonic: ME ENA

Type: R/W, 0

When set, ME ENA allows INTRs to be issued, if enabled, upon the logging of every error detected by the DWMVA, regardless of the current state of the Error Summary bit, XBER<31>. Self-Test Fail, XBER<10>, does not affect ME ENA.

When this bit is clear (the default), one INTR is issued, if enabled, upon detection of an error, if the Error Summary bit is currently clear. If a subsequent error occurs, a second INTR is not issued while the first error is outstanding. After an INTR is issued for the first error detected, further INTRs are disabled, and remain disabled until the Error Summary bit is cleared. Software reads the XBER after servicing the INTR to ensure that all errors have been detected.

bit<2>

Name: Reserved
Mnemonic: None
Type: RO, 0

Reserved; reads as zero.

bit<1>

Name: Return Vector Disable
Mnemonic: RETURN VECTOR DIS
Type: R/W, 0

When set, RETURN VECTOR DIS prevents the DWMVA from returning the contents of the DWMVA Return Vector Register in response to an unsolicited or failed IDENT. Instead, the DWMVA issues a Return Error Response to the XMI.

bit<0>

Name: Reserved
Mnemonic: None
Type: RO, 0

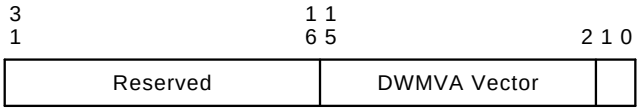
Reserved; reads as zero.

Return Vector Register (ARVR)

The DWMVA returns the vector in ARVR<15:2> when the module either receives an unsolicited IDENT or receives an IDENT that fails on the VMEbus. This feature of the DWMVA is controlled by the Return Vector Disable bit in the Control and Status Register (ACSR <1>). When the Return Vector Disable bit is set, the DWMVA responds with an RER.

ADDRESS

XMI nodespace base address + 0000 0028



msb−p397−91

bits<31:16>

Name: Reserved
Mnemonic: None
Type: RO, 0

Reserved; read as zero.

bits<15:2>

Name: DWMVA Vector
Mnemonic: None
Type: R/W, 0

The DWMVA Vector field is loaded by software at system initialization. The value in this field should be the same as the value stored in the Vector Register (VVR).

bits<1:0>

Name: Reserved
Mnemonic: None
Type: RO, 0

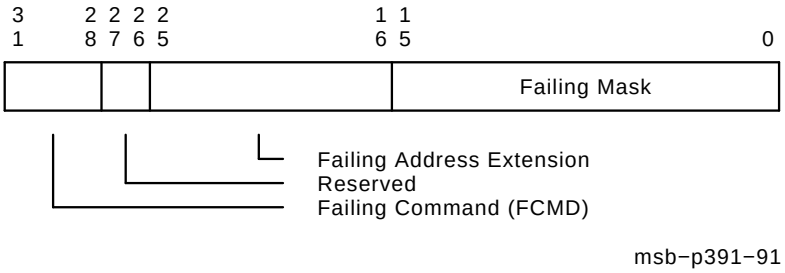
Reserved; read as zero.

Failing Address Extension Register (XFAER)

The Failing Address Extension Register logs the address extension, command, and mask information associated with a failed XMI commander transaction. The DWMVA locks XFAER only if the transaction fails. The following error bits, when set, lock this register and XFADR:

- Write Data NO ACK, XBER<20>
- No Read Response, XBER<18>
- Read Sequence Error, XBER<17>
- Read Error Response, XBER<16>
- Command NO ACK, XBER<15>
- Transaction Timeout, XBER<13>
- Internal Error, AESR<7>

ADDRESS *XMI nodespace base address + 0000 002C*



msb−p391−91

bits<31:28>

Name: Failing Command

Mnemonic: FCMD

Type: RO, 0

FCMD logs XMI D<63:60> during the command/address cycle of a failed XMI commander transaction. This field is loaded on every command/address cycle issued by the DWMVA, but locks only if the transaction fails and unlocks when the error that caused the lock is cleared.

bits<27:26>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; read as zero.

bits<25:16>

Name: Failing Address Extension

Mnemonic: None

Type: RO, 0

Failing Address Extension logs XMI D<57:48> during the command/address cycle of a failed XMI commander transaction or bits<38:29> of the address specified in the transaction for DMA reads and DMA writes.

Failing Address Extension is loaded on every command/address cycle issued by the DWMVA, but locks only if the transaction fails and unlocks when the error that caused the lock is cleared.

bits<15:0>

Name: Failing Mask

Mnemonic: None

Type: RO, 0

Failing Mask logs XMI D<47:32> during the command/address cycle of a failed XMI commander transaction or the write mask for DMA writes. The field is undefined for other transactions.

Failing Mask is loaded on every command/address cycle issued by the DWMVA, but locks only if the transaction fails and unlocks when the error that caused the lock is cleared.

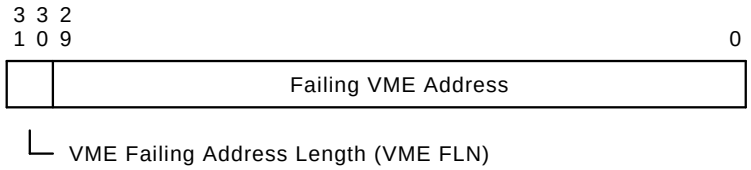
VME Error Address Register (ABEAR)

The VME Error Address Register logs address and length information of failed IBUS DMA and interrupt transactions that are detected by the T2018 module. The logged addresses are in VME format. The invalid VME command/address is logged on the first occurrence of one of the following errors:

- Invalid PFN, AESR<11>
- Correctable DMA ECC Error, AESR<10>
- Uncorrectable DMA ECC Error, AESR<9>
- Invalid VME Address, AESR<8>
- Internal Error, AESR<7>
- IBUS DMA-A Data Parity Error, AESR<4>
- IBUS DMA-A C/A Parity Error, AESR<3>
- IBUS DMA-B Data Parity Error, AESR<2>
- IBUS DMA-B C/A Parity Error, AESR<1>

The ABEAR locks the VME address until the error status bit is cleared by software. Once the error status bit is cleared, another VME error causes the overwrite of the previous error address.

ADDRESS *XMI nodespace base address + 0000 0030*



msb−p398−91

bits<31:30>

Name: VME Failing Address Length

Mnemonic: VME FLN

Type: RO, 0

VME FLN logs IBUS D<31:30> during a failed IBUS DMA or interrupt transaction. This field is locked with IBUS D<31:30> anytime one of the AESR error bits is set, and the register is not already locked.

bits<29:0>

Name: Failing VME Address

Mnemonic: None

Type: RO, 0

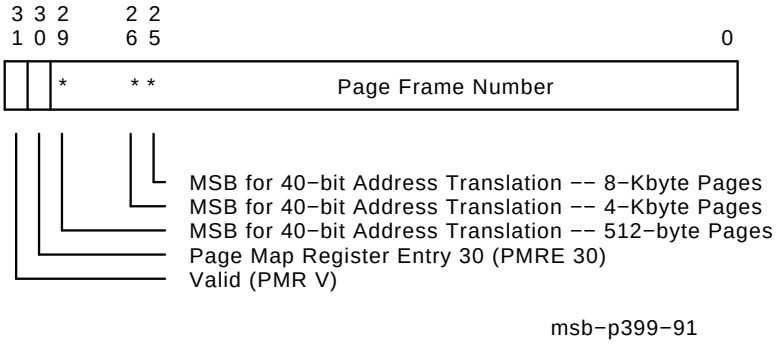
The Failing VME Address field logs IBUS D<29:0> during a failed IBUS DMA or interrupt transaction. This field is locked with IBUS D<29:0> anytime one of the AESR error bits is set, and the register is not already locked.

Page Map Registers (PMRs)

The T2018 module contains 64K page map registers (PMRs) which are used to store page frame numbers (PFNs) for extended address translation. The format of the PMRs is identical.

ADDRESS

XMI nodespace address BB + 0000 0200 to BB + 0004 01FC



msb−p399−91

bit<31>

Name: Valid

Mnemonic: PMR V

Type: R/W, 0

System software sets PMR V when it loads a valid PFN into the PFN field of the PMR. The bit is used by the DWMVA during address translation to determine the validity of the PFN stored in the PMR.

bit<30>

Name: Page Map Register Entry Bit 30

Mnemonic: PMRE 30

Type: R/W, 0

PMRE 30 is undefined for normal operation. Diagnostics uses this bit to write an entire 32-bit page map register entry.

bits<29:0>

Name: Page Frame Number

Mnemonic: PFN

Type: R/W, 0

If the DWMVA is configured in any address translation mode for DMA operations, system software must load a valid PFN entry into this field for the associated PMR of every VME page it queues for transfer.

The following table indicates which VME address bits are concatenated with the appropriate PFN bits to generate the required XMI address in DMA transactions.

Address Mode	Page Size (Bytes)	XMI Address	Bits Forced to Zero
34-bit	512	PFN<24:0> + VME A0–A8 = XMI A<33:0>	XMI A<39:34>
40-bit	512	PFN<29:0> + VME A0–A8 = XMI A<38:0>	XMI A<39>
40-bit	4K	PFN<26:0> + VME A0–A11 = XMI A<38:0>	XMI A<39>
40-bit	8K	PFN<25:0> + VME A0–A12 = XMI A<38:0>	XMI A<39>

7.2 C3200 Registers

The registers on the C3200 module fall into two categories:

- C3200 error and configuration registers
- C3200 VME initialization/test registers

Error registers report error conditions that may occur on the C3200 module during various transactions. Initialization registers are used to set up initial conditions for the operation of the DWMVA. The test registers are used by diagnostics.

Table 7–4 lists all registers on the C3200 module.

Table 7–4 C3200 Registers

Name	Mnemonic	Address[1]
Error and Configuration Registers		
Device/Configuration Register	VDCR	BB + 0000 0040
VME Error Summary Register	VESR	BB + 0000 0044
VME Failing Address Register	VFADR	BB + 0000 0048
Interrupt Configuration Register	VICR	BB + 0000 004C
Vector Offset Register	VVOR	BB + 0000 0050
Vector Register	VVR	BB + 0000 0054
Byte Swap RAM Access Register	RAR	BB + 0000 0058
VME Initialization/Test Registers		
CSR Access Register	VCAR	BB + 0000 005C
VME Address Range Enable Register	VAER	through VCAR[2]
Diagnostic Register	VDR	through VCAR[2]
Failing Data Register	VFDR	through VCAR[2]
CPU Transaction Address Offset Registers	VAOR	through VCAR[2]

[1]BB refers to the base address of an XMI node (the address of the first location in the nodespace).

[2]The address of the register is the contents of the Register Select field of the VCAR. The code stored in the Register Select field must select the register to be accessed for the current transaction.

Table 7–5 gives the values that should appear in the C3200 registers following a hardware or software reset.

Table 7–5 Initialization Values of the C3200 Registers

Register	Initialization Bit States (Bin)	(Hex)
VDCR	1000 0011 0011 1111 0000 HHHH 1110 1000	833F 0XE8
VESR	0000 0000 0000 1111 1100 0000 0000 0000	000F C000
VFADR	XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXX0	XXXX XXXX
VICR	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
VVOR	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
VVR	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
RAR	0000 0000 0000 0000 0000 0000 0000 XXXX	0000 000X
VCAR	XXXX XXXX XXXX XXXX XX00 0000 0000 0000	XXXX XX00
VAER	0000 0000 0000 0000 0000 0000 0000 0000	0000 0000
VDR	0000 0000 1000 0000 0000 0000 0000 0000	0080 0000
VFDR	0000 0000 X000 0000 0000 0000 0000 0000	00X0 0000
VAOR	0000 0000 0000 0000 XXXX XXXX XXXX XXXX	0000 XXXX

H - Denotes a hardwired bit. The state of the bit depends on the revision level of the C3200 module.
X - Denotes an indeterminate bit state.

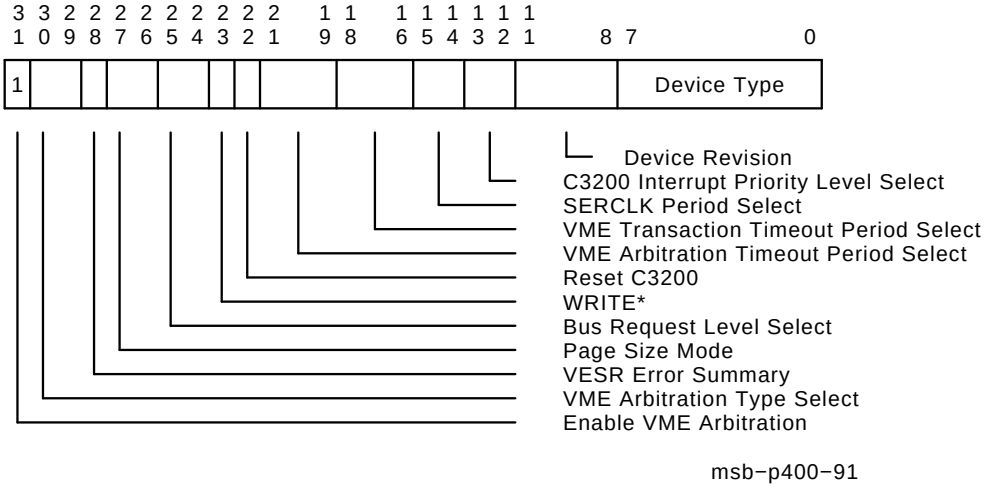
Device/Configuration Register (VDCR)

The Device/Configuration Register defines the operating modes and selects the operating parameters of the C3200 module such as:

VME arbitration type
VME arbitration timeout period
VME transaction timeout period
C3200 interrupt priority level
C3200 VMEbus request level
Page size mode

This register also contains an Error Summary bit that is the logical OR of all the error bits in the VME Error Summary Register.

ADDRESS *XMI nodespace base address + 0000 0040*



msb−p400−91

bit<31>

Name: Enable VME Arbitration

Mnemonic: None

Type: R/W, 1

When set, the Enable VME Arbitration bit causes the C3200 to act as an arbiter for the VME subsystem. This bit should always be set.

bits<30:29>

Name: VME Arbitration Type Select

Mnemonic: None

Type: R/W, 00

The VME Arbitration Type Select bits determine the algorithm that the C3200 VME arbiter uses to arbitrate VMEbus requests. The states of these bits define the algorithm to be used as follows:

VDCR <30:29>	Arbitration Type
00	Round robin (RRS)
01	Priority (PRI)
10	Prioritized round-robin (PRS)
11	Single-level (SGL)

NOTE: When the SGL arbitration algorithm is selected and the C3200 is enabled as the VME arbiter, the C3200 VME requester requests the bus through BR3*. This means that bits <25:24> of this register must be set to 11.

bit<28>

Name: VESR Error Summary

Mnemonic: None

Type: RO, 0

The state of the VESR Error Summary bit reflects the logical OR of all error bits in the VESR. The error bits that contribute to the state of the VESR Error Summary bit are:

Interlock Error
RMW Error
VME Transmit Parity Error
IBUS Transmit Parity Error
IBUS Receive Parity Error
VME Transaction Timeout
VME Arbitration Timeout
BERR*

bits<27:26>

Name: Page Size Mode

Mnemonic: None

Type: R/W, 00

The Page Size Mode field determines the page size that the C3200 will use in accessing its byte swap RAM. The bits are decoded as follows:

VDCR<27:26>	Page Size
00	4 Kbytes
01	8 Kbytes
10	512 bytes
11	Undefined

bits<25:24>

Name: Bus Request Level Select

Mnemonic: None

Type: R/W, 11

The Bus Request Level Select bits determine the arbitration level at which the C3200 will request the VMEbus. The bus request levels are coded as follows:

VDCR<25:24>	Selected Bus Request Level
00	BR0 (Level 0)
01	BR1 (Level 1)
10	BR2 (Level 2)
11	BR3 (Level 3)

NOTE: If the C3200 is enabled as arbiter of the VMEbus and the SGL arbitration algorithm is selected, the C3200 must request the bus through BR3*. This means that bits <25:24> must be set to 11.

bit<23>

Name: WRITE*

Mnemonic: None

Type: RO, 0

The WRITE* bit holds the state of the VME WRITE* signal at the moment a timeout occurs. This bit is loaded on every cycle and locked upon detection of a timeout.

NOTE: The WRITE* signal is asserted low on the VMEbus but is represented as a high state bit (set to 1) in this register.

bit<22>

Name: Reset C3200

Mnemonic: None

Type: R0/W1, 0

A one written to the Reset C3200 bit causes the C3200 to reset its state machines. All registers on the C3200 retain their values. This bit always reads as zero and does not cause a VME system reset.

bits<21:19>

Name: VME Arbitration Timeout Period Select

Mnemonic: None

Type: R/W, 7 (hex)

The VME Arbitration Timeout Period Select field determines the period of the VME Arbitration Timeout Counter. These bits are only valid if VME Arbitration is enabled for the C3200 (bit <31> of this register is set). The timeout period is decoded as follows:

VDCR<21:19>	Arbitration Timeout Period
111	Timeouts disabled
110	3.28 ms
101	819 μs
100	128 μs
011	64.0 μs
010	32.0 μs
001	12.8 μs
000	800 ns

NOTE: This field MUST be set to a value by operating system software to assure proper VME error reporting.

bits<18:16>

Name: VME Transaction Timeout Period Select

Mnemonic: None

Type: R/W, 7 (hex)

The VME Transaction Timeout Period Select field determines the timeout value for the VME Transaction Bus Timer. The bits are decoded as follows:

VDCR<18:16>	Transaction Timeout Period
111	Timeouts disabled
110	3.28 ms
101	819 μs
100	128 μs
011	64.0 μs
010	32.0 μs
001	12.8 μs
000	800 ns

NOTE: This field MUST be set to a value by operating system software to assure proper VME error reporting.

bits<15:14>

Name: SERCLK Period Select

Mnemonic: None

Type: R/W, 0

The SERCLK Period Select field determines the base frequency to be used for the SERCLK signal. The SERCLK line is driven by the C3200 module. The base frequency is determined as follows:

VDCR<15:14>	SERCLK Frequency
00	32 MHz
01	16 MHz
10	8 MHz
11	4 MHz

bits<13:12>

Name: C3200 Interrupt Priority Level Select

Mnemonic: None

Type: R/W, 00

The C3200 Interrupt Priority Level Select field selects the IPL to be used by the C3200 to interrupt the XMI processor for internal DWMVA error conditions. The field is decoded as follows:

VDCR<13:12>	Bus Line	Selected IPL
00	BR4	IPL14
01	BR5	IPL15
10	BR6	IPL16
11	BR7	IPL17

bits<11:8>

Name: Device Revision

Mnemonic: DREV

Type: RO, Hardwired

DREV indicates the revision level of the C3200 module. The value in this field is hardwired on the module with jumpers.

bit<7:0>

Name: Device Type

Mnemonic: None

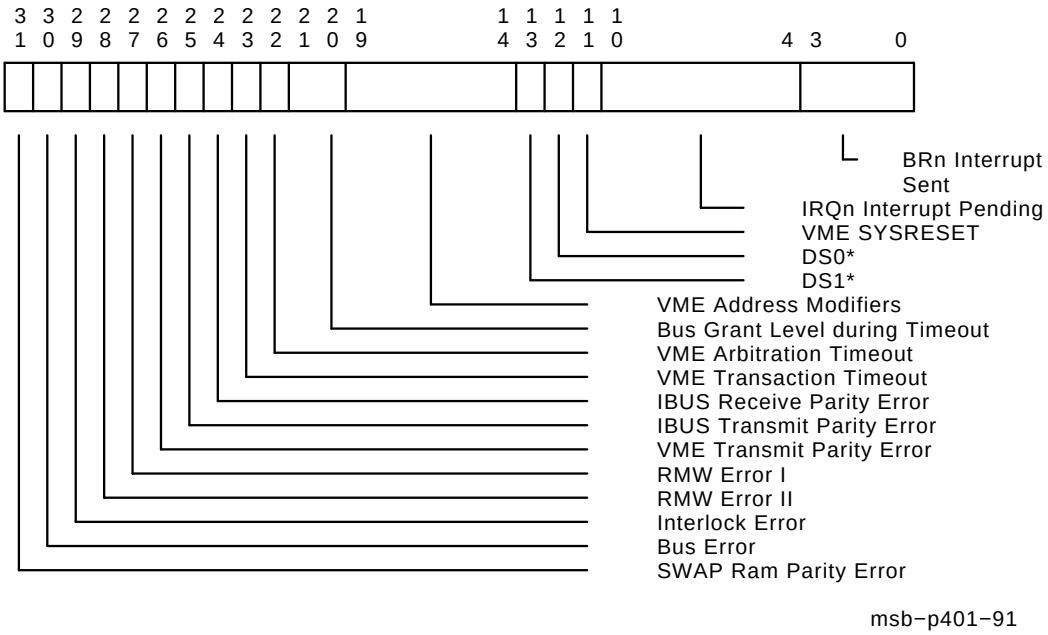
Type: RO, Hardwired to E8

The Device Type field identifies the device as the C3200 part of a DWMVA subsystem. This field is hardwired to a value of E8 (hex).

VME Error Summary Register (VESR)

The VME Error Summary Register reports error conditions that occur during the operation of the C3200 module. This register also reflects the current state of the BERR* signal on the VMEbus.

ADDRESS *XMI nodespace base address + 0000 0044*



msb−p401−91

bit<31>

Name: Swap RAM Parity Error

Mnemonic: None

Type: R/W1C, 0

When set, the Swap RAM Parity Error bit indicates that a parity error was detected while reading the contents of the byte swap RAM, which causes the C3200 to issue an IBUS INTR transaction, if the interrupt is enabled. This bit logs the first occurrence of a parity error and must be cleared by software before it can log another.

bit<30>

Name: Bus Error

Mnemonic: BERR*

Type: R/W1C, 0

BERR* reflects the current state of the BERR* signal on the VMEbus. It is set when BERR* is low and cleared when BERR* is high. This bit is writeable to allow diagnostics to cause the assertion of BERR*.

bit<29>

Name: Interlock Error

Mnemonic: None

Type: R/W1C, 0

The Interlock Error bit is set when an interlock error is detected by the C3200. An interlock error occurs when an Interlock Read, issued by an XMI device, is not followed immediately by an Unlock Write. The C3200 performs an Interlock Read, the first part of an Interlock Read/Unlock Write transaction pair, as the start of a Read Modify Write operation on the VME. If the next XMI transaction the C3200 receives is not the corresponding Unlock Write, the C3200 sets this bit, interrupts the XMI processor if the interrupt is enabled, and relinquishes control of the VMEbus. The C3200 continues to execute on the VME all XMI transactions it receives, including an Unlock Write transaction that does not immediately follow an Interlock Read.

bit<28>

Name: RMW Error II

Mnemonic: None

Type: R/W1C, 0

The RMW Error II bit is set when the C3200 detects a normal read transaction to an address that is set up in the swap RAM to expect Read Modify Write transactions. The C3200 issues an Interlock Read in response to the VME read command. After the C3200 returns read data to the VME master, the C3200 issues an Unlock Write with 0 data and all mask bits cleared to unlock the XMI, if it finds out that the transaction was not a Read Modify Write. At this point the C3200 issues an interrupt, if the interrupt is enabled.

bit<27>

Name: RMW Error I

Mnemonic: None

Type: R/W1C, 0

The RMW Error I bit is set when the C3200 detects an attempt by a VME master to perform a Read Modify Write transaction to an address that is not set up for RMWs. The bit is set during the write portion of the RMW transaction pair. The C3200 completes both transactions and issues an interrupt, if the interrupt is enabled.

bit<26>

Name: VME Transmit Parity Error

Mnemonic: None

Type: R/W1C, 0

The VME Transmit Parity Error bit is set when the C3200 detects a parity error on the output of the IBUS to VMEbus buffer prior to transmitting data on the VMEbus. The transaction is aborted if a parity error is detected. If no parity error is detected, the parity bit is dropped after this checker, since there is no parity protection on the VMEbus.

bit<25>

Name: IBUS Transmit Parity Error

Mnemonic: None

Type: R/W1C, 0

The IBUS Transmit Parity Error is set when the C3200 detects a parity error at the output lines of the VMEbus to IBUS buffer prior to transmitting data on the IBUS. When this bit is set, the C3200 issues an interrupt, if the interrupt is enabled. This interrupt cannot be disabled.

bit<24>

Name: IBUS Receive Parity Error

Mnemonic: None

Type: R/W1C, 0

The IBUS Receive Parity Error bit is set when the C3200 detects a parity error on data it has received from the T2018 across the IBUS. The C3200 aborts the transaction when this bit is set and issues an interrupt, if the interrupt is enabled.

bit<23>

Name: VME Transaction Timeout

Mnemonic: None

Type: R/W1C, 0

The VME Transaction Timeout bit is set by the VMEbus timer logic on the C3200 module when a VMEbus timeout occurs. When this bit is set, the address of the transaction is stored in the C3200 VME Failing Address Register if the C3200 was the bus master when the timeout was detected.

The set state of the VME Transaction Timeout bit also indicates that the VMEbus timer has asserted BERR* on the VMEbus, causing the timed-out transaction to be removed from the bus and allowing bus access to other VME devices. When this bit is set, an interrupt is issued, if the interrupt is enabled.

bit<22>

Name: VME Arbitration Timeout

Mnemonic: None

Type: R/W1C, 0

The VME Arbitration Timeout bit is set by the arbitration timer logic when an arbitration timeout occurs. The set state of this bit causes the arbiter to deassert the current bus grant level it is driving and rearbitrate all pending bus requests. When this bit is set, an interrupt is issued, if the interrupt is enabled.

bits<21:20>

Name: Bus Grant Level During Timeout

Mnemonic: None

Type: RO, 0

The Bus Grant Level During Timeout bits hold the encoded value of the BGn* being driven when a timeout occurs. These bits are loaded on every arbitration cycle and locked upon detection of a timeout.

bits<19:14>

Name: VME Address Modifiers

Mnemonic: None

Type: RO, 3F

The VME Address Modifiers field holds the state of the VME AM0–AM5 lines when a timeout occurs. This field is loaded on every cycle and locked upon detection of a timeout.

bit<13>

Name: DS1*

Mnemonic: None

Type: RO, 0

The DS1* bit holds the state of the VME DS1* signal when a timeout occurs. This bit is loaded on every cycle and locked upon detection of a timeout.

NOTE: The VME DS1* signal is asserted low on the VMEbus. The low state of the signal is reflected as a state of 1 on this bit.

bit<12>

Name: DS0*

Mnemonic: None

Type: RO, 0

The DS0* bit holds the state of the VME DS0* signal when a timeout occurs. This bit is loaded on every cycle and locked upon detection of a timeout.

NOTE: The VME DS0* signal is asserted low on the VMEbus. The low state of the signal is reflected as a state of 1 on this bit.

bit<11>

Name: VME SYSRESET

Mnemonic: None

Type: R/W1C, 0

The VME SYSRESET bit is set when the C3200 receives a SYSRESET* from the VMEbus. When set, this bit specifically indicates to the interrupt service routine that the C3200 has issued an interrupt in response to a VME-originated SYSRESET*. This interrupt is not maskable.

NOTE: The VME SYSRESET* signal is asserted low on the VMEbus. The low state of the signal is reflected as a state of 1 on this bit.

bits<10:4>

Name: IRQn Interrupt Pending

Mnemonic: None

Type: RO, 0

The IRQn Interrupt Pending field is used to indicate the status of VMEbus-originated interrupt requests. The seven bits of the field, bit <10> to bit <4>, are in one-to-one correspondence, respectively, with the seven VME interrupt request levels, IRQ7*–IRQ1*. The particular bit in the field is set when the corresponding interrupt request level is asserted on the VMEbus. The bit is cleared automatically for a ROAK (Release On Acknowledge) interrupter when an IACK* to that request level is issued on the VMEbus. However, when the interrupter is a RORA (Release On Register Access) type, the bit must be cleared by the operating system software.

bits<3:0>

Name: BRn Interrupt Sent

Mnemonic: None

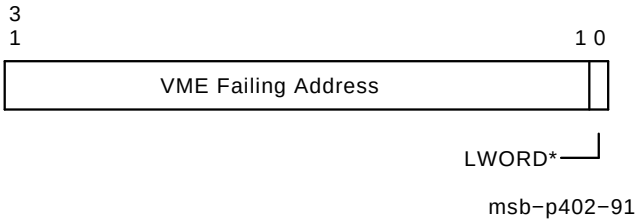
Type: RO, 0

The BRn Interrupt Sent field is used to indicate the status of XMI interrupts sent by the C3200 module. The four bits of the field, bit <3> to bit <0>, are in one-to-one correspondence, respectively, with BR7—BR4. The particular bit is set when an interrupt is generated at the corresponding level to the XMI. The bit is cleared upon receiving the associated IDENT transaction.

VME Failing Address Register (VFADR)

The VME Failing Address Register reflects the state of VME A01–A31 address lines in a timed-out transaction.

ADDRESS *XMI nodespace base address + 0000 0048*



bits<31:1>

Name: VME Failing Address

Mnemonic: None

Type: RO, Undefined

A valid VME Failing Address field holds the state of VME A01–A31, the address being accessed by the C3200 in a timed-out transaction. This field is only valid if the VME Transaction Timeout bit (VESR<23>) is set and the C3200 was the VMEbus master during the timeout.

NOTE: This field is undefined if read when the VME Transaction Timeout bit is not set.

bit<0>

Name: LWORD*

Mnemonic: None

Type: RO, 0

The LWORD* bit indicates the state of the VME LWORD* signal at the moment of a timeout. This bit is latched on every cycle and locked upon detection of a timeout.

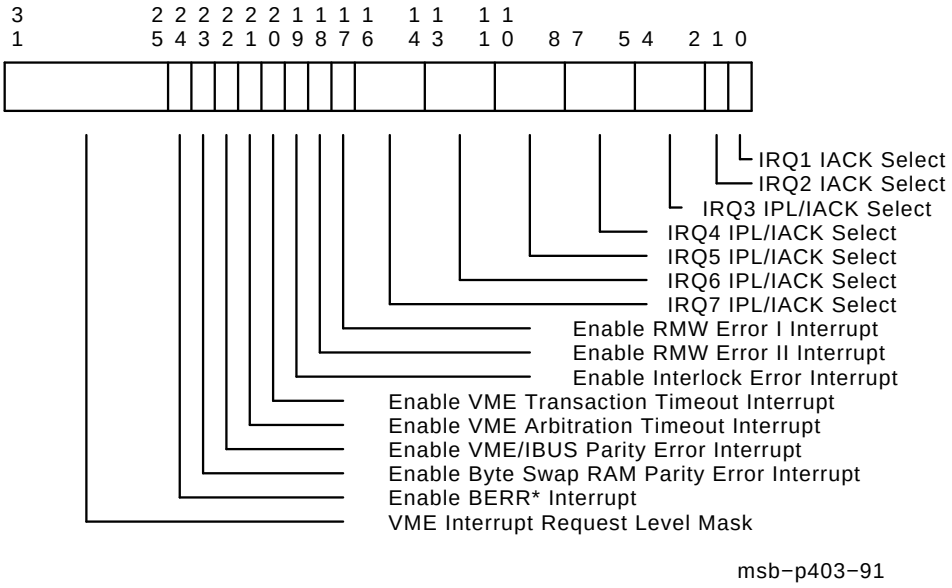
NOTE: The VME LWORD* signal is asserted low on the VMEbus. The low state of the signal is reflected as a state of 1 on this bit.

Interrupt Configuration Register (VICR)

The Interrupt Configuration Register holds the VME Interrupt Request Level Mask and defines the priority levels of all C3200-generated interrupts. This register also contains the Enable bits for the C3200 interrupts.

The four bus request lines of the XMI, BR7–BR4 can accommodate four interrupt request signals (IRQX*) from the VMEbus. Since the VMEbus has seven interrupt request levels, four selected interrupt levels must be mapped to the four XMI bus request lines. The mapping can be random. However, each interrupt request level must be mapped to a single BR line. At system initialization the operating system generates the default mapping shown in Table 6–2 and explained in the following bit descriptions.

ADDRESS *XMI nodespace base address + 0000 004C*



msb−p403−91

bits<31:25>

Name: VME Interrupt Request Level Mask

Mnemonic: None

Type: R/W, 0

The VME Interrupt Request Level Mask field is used to set up the VME interrupt levels to be received by the C3200 and passed to the processor in the form of XMI interrupt transactions. The seven bits in the field, bit <31> to bit <25> are in one-to-one correspondence, respectively, with the VME IRQ7*–IRQ1* signals. Any interrupt request level (one or more) can be independently enabled or disabled

without affecting the operation of the pending interrupts at other interrupt request levels.

NOTE: This field must be set by the operating system software to enable interrupts from VME devices. The operating system sets this field to a value of 111 1000, which disables IRQ3*–IRQ1*.

bit<24>

Name: Enable BERR* Interrupt

Mnemonic: None

Type: R/W, 0

Setting the Enable BERR* Interrupt bit causes an interrupt to be generated on the XMI when an asserted BERR* signal is detected.

bit<23>

Name: Enable Byte Swap RAM Parity Error Interrupt

Mnemonic: None

Type: R/W, 0

When the Enable Byte Swap RAM Parity Error Interrupt bit is set, the DWMVA generates an XMI interrupt upon detecting a byte swap RAM parity error.

bit<22>

Name: Enable VME/IBUS Parity Error Interrupt

Mnemonic: None

Type: R/W, 0

When the Enable VME/IBUS Parity Error Interrupt bit is set, the DWMVA generates an XMI interrupt upon detecting one of the following conditions:

- VME Transmit Parity Error
- VME Receive Parity Error
- IBUS Transmit Parity Error
- IBUS Receive Parity Error

bit<21>

Name: Enable VME Arbitration Timeout Interrupt

Mnemonic: None

Type: R/W, 0

When the Enable VME Arbitration Timeout Interrupt bit is set, the DWMVA generates an XMI interrupt when the VME arbitration timer detects an arbitration timeout.

bit<20>

Name: Enable VME Transaction Timeout Interrupt

Mnemonic: None

Type: R/W, 0

When the Enable VME Transaction Timeout Interrupt bit is set, the DWMVA generates an XMI interrupt when the VME bus timer detects a transaction timeout on the VMEbus.

bit<19>

Name: Enable Interlock Error Interrupt

Mnemonic: None

Type: R/W, 0

When the Enable Interlock Error Interrupt bit is set, the DWMVA generates an XMI interrupt when it detects an interlock error.

bit<18>

Name: Enable RMW Error II Interrupt

Mnemonic: None

Type: R/W, 0

When the Enable RMW Error II Interrupt bit is set, the DWMVA generates an XMI interrupt upon detecting a Read Modify Write transaction issued to the DWMVA by a VME device. For details on this error see the VME Error Summary Register, bit <28>.

bit<17>

Name: Enable RMW Error I Interrupt

Mnemonic: None

Type: R/W, 0

When the Enable RMW Error I Interrupt bit is set, the DWMVA generates an XMI interrupt upon detecting a Read Modify Write transaction issued to the DWMVA by a VME device. For details on this error see the VME Error Summary Register, bit <27>.

bits<16:14>

Name: IRQ7 Interrupt Priority Level/IACK Select

Mnemonic: None

Type: R/W, 0

The IRQ7 Interrupt Priority Level/IACK Select bits can be written to cause IRQ7* to translate to an interrupt priority level on the XMI other than the default (IPL14). These bits are also used to determine if the device interrupting at IRQ7* is a RORA or ROAK type interrupter. The bits are decoded as follows:

VICR<16:14>	Bus Line	IPL	Interrupter Type
000	BR4	IPL14	ROAK
001	BR4	IPL14	RORA
010	BR5	IPL15	ROAK
011	BR5	IPL15	RORA
100	BR6	IPL16	ROAK
101	BR6	IPL16	RORA
110	BR7	IPL17	ROAK
111	BR7	IPL17	RORA

The operating system initializes this field to a value of 110.

NOTE: See Chapter 6 for more details on RORA/ROAK devices.

bits<13:11>

Name: IRQ6 Interrupt Priority Level/IACK Select

Mnemonic: None

Type: R/W, 0

The IRQ6 Interrupt Priority Level/IACK Select bits can be written to cause IRQ6* to translate to an interrupt priority level on the XMI other than the default (IPL14). These bits are also used to determine if the device interrupting at IRQ6* is a RORA or ROAK type interrupter. The bits are decoded as follows:

VICR<13:11>	Bus Line	IPL	Interrupter Type
000	BR4	IPL14	ROAK
001	BR4	IPL14	RORA
010	BR5	IPL15	ROAK
011	BR5	IPL15	RORA
100	BR6	IPL16	ROAK
101	BR6	IPL16	RORA
110	BR7	IPL17	ROAK
111	BR7	IPL17	RORA

The operating system initializes this field to a value of 100.

NOTE: See Chapter 6 for more details on RORA/ROAK devices.

bits<10:8>

Name: IRQ5 Interrupt Priority Level/IACK Select

Mnemonic: None

Type: R/W, 0

The IRQ5 Interrupt Priority Level/IACK Select bits can be written to cause IRQ5* to translate to an interrupt priority level on the XMI other than the default (IPL14). These bits are also used to determine if the device interrupting at IRQ5* is a RORA or ROAK type interrupter. The bits are decoded as follows:

VICR<10:8>	Bus Line	IPL	Interrupter Type
000	BR4	IPL14	ROAK
001	BR4	IPL14	RORA
010	BR5	IPL15	ROAK
011	BR5	IPL15	RORA
100	BR6	IPL16	ROAK
101	BR6	IPL16	RORA
110	BR7	IPL17	ROAK
111	BR7	IPL17	RORA

The operating system initializes this field to a value of 010.

bits<7:5>

Name: IRQ4 Interrupt Priority Level/IACK Select

Mnemonic: None

Type: R/W, 0

The IRQ4 Interrupt Priority Level/IACK Select bits can be written to cause IRQ4* to translate to an interrupt priority level on the XMI other than the default (IPL14). These bits are also used to determine if the device interrupting at IRQ4* is a RORA or ROAK type interrupter. The bits are decoded as follows:

VICR<7:5>	Bus Line	IPL	Interrupter Type
000	BR4	IPL14	ROAK
001	BR4	IPL14	RORA
010	BR5	IPL15	ROAK
011	BR5	IPL15	RORA
100	BR6	IPL16	ROAK
101	BR6	IPL16	RORA
110	BR7	IPL17	ROAK
111	BR7	IPL17	RORA

The operating system initializes this field to a value of 000.

bits<4:2>

Name: IRQ3 Interrupt Priority Level/IACK Select

Mnemonic: None

Type: R/W, 000

The IRQ3 Interrupt Priority Level/IACK Select bits can be written to cause IRQ3* to translate to an interrupt priority level on the XMI other than the default (IPL14). These bits are also used to determine if the device interrupting at IRQ3* is a RORA or ROAK type interrupter. The bits are decoded as follows:

VICR<4:2>	Bus Line	IPL	Interrupter Type
000	BR4	IPL14	ROAK
001	BR4	IPL14	RORA
010	BR5	IPL15	ROAK
011	BR5	IPL15	RORA
100	BR6	IPL16	ROAK
101	BR6	IPL16	RORA
110	BR7	IPL17	ROAK
111	BR7	IPL17	RORA

bit<1>

Name: IRQ2 IACK Select

Mnemonic: None

Type: R/W, 0

The IRQ2 IACK Select bit is used to define what type of interrupter interrupts at IRQ2. The Bus Request level for IRQ2 is fixed at BR4 (IPL14).

VICR<1>	Selected Interrupter Type
0	ROAK
1	RORA

bit<0>

Name: IRQ1 IACK Select

Mnemonic: None

Type: R/W, 0

The IRQ1 IACK Select bit is used to define what type of interrupter interrupts at IRQ1. The Bus Request level for IRQ1 is fixed at BR4.

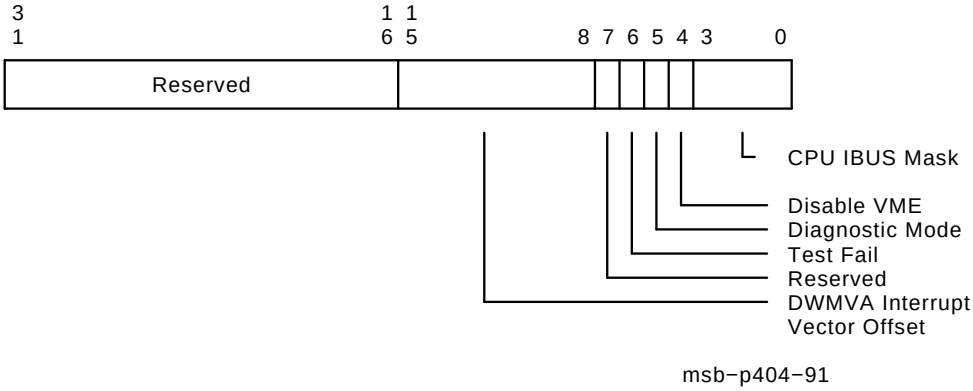
VICR<0>	Selected Interrupter Type
0	ROAK
1	RORA

Vector Offset Register (VVOR)

The Vector Offset Register stores the offset value to be appended to the vector returned by a VME device in response to an IACK cycle on the VME. This register also allows selection of diagnostic mode for the operation of the C3200 module.

ADDRESS

XMI nodespace base address + 0000 0050



bits<31:16>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; read as zero.

bits<15:8>

Name: DWMVA Interrupt Vector Offset

Mnemonic: None

Type: R/W, 0

The DWMVA Interrupt Vector Offset field provides the offset value to be appended to the vector returned by a VME device in response to an IACK cycle on the VME. The VME vector thus formed is transmitted to the XMI in response to an IDENT cycle targeting the VME device. This register allows multiple DWMVAs to reside in a system and provides a logical partition for storing their interrupt service routines.

bit<7>

Name: Reserved
Mnemonic: None
Type: RO, 0

Reserved; reads as zero.

bit<6>

Name: Test Fail
Mnemonic: None
Type: R/W, 0

The Test Fail bit is set by diagnostics at the beginning of a diagnostic test suite. It clears automatically when all tests pass.

bit<5>

Name: Diagnostic Mode
Mnemonic: None
Type: R/W, 0

The Diagnostic Mode bit is used to select loopback mode for diagnostics.

bit<4>

Name: Disable VME
Mnemonic: None
Type: R/W, 0

The Disable VME bit disables VME drivers to ensure that diagnostics do not corrupt devices on the VME.

bits<3:0>

Name: CPU IBUS Mask
Mnemonic: None
Type: RO/Write through IBUS I field, 0

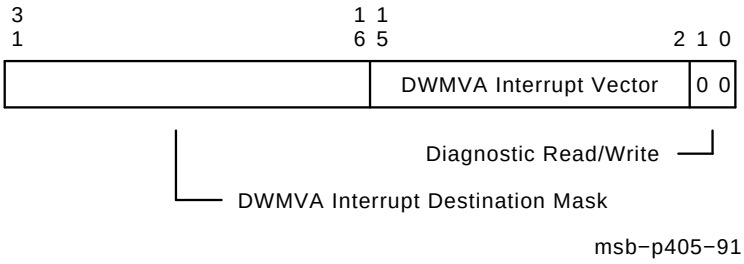
The CPU IBUS Mask field is reserved for diagnostic use.

Vector Register (VVR)

The Vector Register contains the DWMVA Interrupt Destination Mask and the DWMVA Interrupt Vector. It includes two diagnostic read/write bits that allow a 32-bit vector to be captured in the VVR for diagnostic reads and writes.

ADDRESS

XMI nodespace base address + 0000 0054



bits<31:16>

Name: DWMVA Interrupt Destination Mask

Mnemonic: None

Type: R/W, 0

The DWMVA Interrupt Destination Mask field determines which XMI nodes will be targeted when the DWMVA issues an INTR transaction. Each bit in the field corresponds to one of the XMI nodes. When a bit in this field is set, the corresponding node will be targeted for the interrupt. Any number of XMI nodes can be set for interrupts.

bits<15:2>

Name: DWMVA Interrupt Vector

Mnemonic: None

Type: R/W, 0

The DWMVA Interrupt Vector field provides the vector to be returned by the C3200 in response to an IDENT command issued as a result of an INTR transaction generated by an error condition on the C3200 module. Interrupts generated by VME devices other than the DWMVA do not return this vector. Instead, they supply their own vector, which is appended to the value in the C3200 Vector Offset Register.

bits<1:0>

Name: Diagnostic Read/Write

Mnemonic: None

Type: R/W, 0

The Diagnostic Read/Write field allows a 32-bit vector to be captured in the C3200 Vector Register for diagnostic reads and writes.

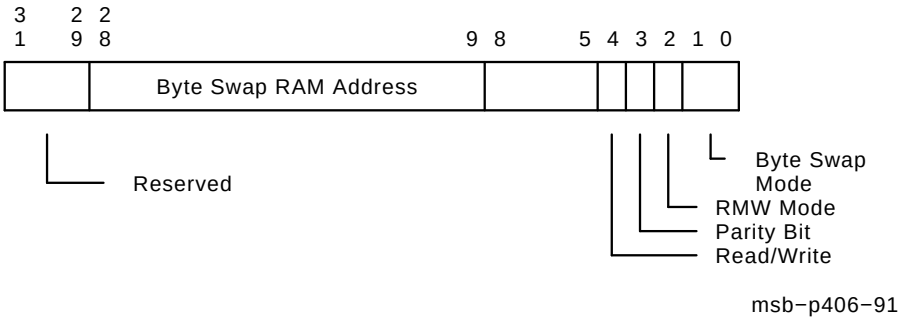
Byte Swap RAM Access Register (RAR)

The Byte Swap RAM Access Register is a pseudo-register that maps all references to its address to a 64K by 4 RAM on the C3200 module.

To write the byte swap RAM with byte swapping and Read Modify Write information, the RAR must be written with the address of the page to be mapped, the Read/Write bit (RAR<4>) must be set, and RAR<2:0> must be set to the appropriate RMW and byte swap values.

A read of the byte swap RAM requires two transactions to the Byte Swap RAM Access Register. First, the RAR must be written with the address of the page to be read and the Read/Write bit (RAR<4>) cleared. Next, the RAR should be read. In response to this read, the RMW and byte swapping mode fields will appear as bits <2:0> of the read return data.

ADDRESS *XMI nodespace base address + 0000 0058*



bits<31:29>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; read as zero.

bits<28:9>

Name: Byte Swap RAM Address

Mnemonic: None

Type: R0/W, 0

The Byte Swap RAM Address field provides the address of the byte swap RAM location to be read or written. The address is decoded on the module in conjunction with the Page Size bits (<27:26>) in the Device Configuration Register. The options are summarized as follows:

VDCR <27:26>	Page Size	RAR BITS for Swap RAM Address
00	4 Kbytes	<27:12>
01	8 Kbytes	<28:13>
10	512 bytes	<24:9>

bits<8:5>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; read as zero.

bit<4>

Name: Read/Write

Mnemonic: None

Type: WO, 0

The Read/Write bit is used to select a read of the RAM or a write to it for the current transaction. If this bit is set, the data in bits <2:0> of this register will be written to the RAM address decoded by bits <28:9> of the register. If, however, the Read/Write bit is cleared with a write to this register, the address written will be applied to the RAM, but the write enable line will not be asserted, thus causing the RAM to place the addressed location's data on its data outputs. To complete the read transaction, a read must be performed to the Byte Swap RAM Access Register. This will return the data back to the XMI host.

bit<3>

Name: Parity Bit

Mnemonic: None

Type: RO, Undefined

The Parity bit is meaningful only when reading the register. When the register is written to cause a byte swap RAM write, the hardware generates even parity on the lower three bits of this register and stores all four bits in the RAM. When data is read back, the parity bit is returned to the host along with the data.

bit<2>

Name: RMW Mode

Mnemonic: None

Type: R/W, Undefined

The RMW (Read Modify Write) Mode bit is written by software to indicate whether a given page should translate VME-originated reads (start of RMW) into Interlock Reads to the XMI. If this bit is set, reads generated on the VME are sent to the XMI as normal reads. If it is cleared, any read generated to this page from the VME is translated to an Interlock Read on the XMI.

bits<1:0>

Name: Byte Swap Mode

Mnemonic: None

Type: R/W, Undefined

The Byte Swap Mode field selects the byte swap mode to be used for VME-initiated transactions for a given page. The bits decode as follows:

Bits<1:0>	Swap Mode
00	No swap
01	Byte swap
10	Word swap
11	Longword swap

NOTE: Details of byte swapping are given in Appendix B.

CSR Access Register (VCAR)

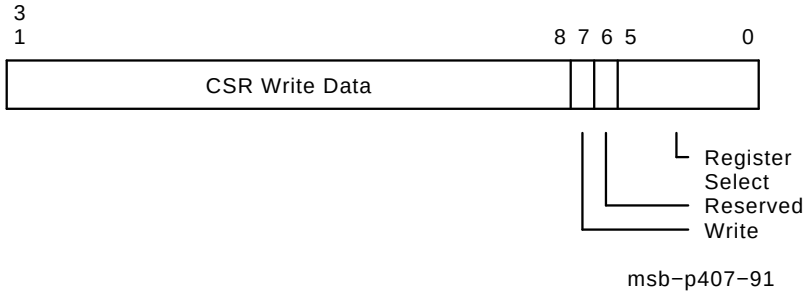
The CSR Access Register provides indirect access to some C3200 registers not directly accessible through a CPU address. The VCAR must first be written with the code corresponding to the address of the targeted register (CSR).

A write transaction writes the upper 18 bits of the data to the destination register specified in the Register Select field (VCAR<5:0>). The Write bit (VCAR<7>) must be set when writing the code to the VCAR.

A read to one of these registers requires two transactions to complete, a write followed by a read. During the write transaction, the address of the register to be read is written to the VCAR Register Select field with the Write bit clear. To obtain read return data from the selected register, the CPU requesting the data must next perform a read transaction on the VCAR. The read return data from this access will originate from the register selected in the write phase of the transaction.

To read the contents of the VCAR itself, first a write must be performed to the register with the Write bit (VCAR<7>) set. A subsequent read to VCAR causes the contents of the register itself to be returned. (A read of the VCAR is used for diagnostics.)

ADDRESS *XMI nodespace base address + 0000 005C*



msb−p407−91

bits<31:8>

Name: CSR Write Data

Mnemonic: None

Type: WO, Undefined

The CSR Write Data field is used as the 18-bit data path for a write transaction to the C3200 register selected by the code written to bits <5:0> of this register. The 18 bits of data will only be written to the selected register if the Write bit (VCAR<7>) is set while writing to the VCAR.

bit<7>

Name: Write

Mnemonic: None

Type: R/W, 0

Setting the Write bit causes the data written into VCAR<31:14> to be loaded into the register specified by the Register Select field (VCAR<5:0>). This bit must be written concurrently with the data and Register Select bits to cause the correct write transaction.

Writing a zero to this bit indicates that a read is to be performed to the register specified by the Register Select field. The read return data is supplied by the selected register in response to a read to the VCAR.

In the case where the Write bit is set and a VCAR read is performed, the data returned in bits <31:14> will be the contents of the register decoded in bits <5:0> and the data returned in bits <13:0> will be the contents of the VCAR itself. If no register is indicated in bits <5:0>, the response to a read VCAR will be zero.

bit<6>

Name: Reserved

Mnemonic: None

Type: RO, 0

Reserved; reads as zero.

bits<5:0>

Name: Register Select

Mnemonic: None

Type: R/W, 0

The Register Select field indicates which CSR is being accessed for a given transaction. The register must be selected in the same cycle that the Write bit is written.

The CSRs accessed through the VCAR are listed in the following table:

VCAR <5:0>	Register Decoded
0–4	Reserved
5	VME Address Range Enable Register
6	Diagnostic Register
7	VME Failing Data Register
8–F	Reserved
10–1F	Reserved for diagnostic use
20	CPU Transaction Offset / Length Register 00
21	CPU Transaction Offset / Length Register 01
22	CPU Transaction Offset / Length Register 02
23	CPU Transaction Offset / Length Register 03
24	CPU Transaction Offset / Length Register 04
25	CPU Transaction Offset / Length Register 05
26	CPU Transaction Offset / Length Register 06
27	CPU Transaction Offset / Length Register 07
28	CPU Transaction Offset / Length Register 08
29	CPU Transaction Offset / Length Register 09
2A	CPU Transaction Offset / Length Register 10
2B	CPU Transaction Offset / Length Register 11
2C	CPU Transaction Offset / Length Register 12
2D	CPU Transaction Offset / Length Register 13
2E	CPU Transaction Offset / Length Register 14
2F	CPU Transaction Offset / Length Register 15
30	CPU Transaction Offset / Length Register 16
31	CPU Transaction Offset / Length Register 17
32	CPU Transaction Offset / Length Register 18
33	CPU Transaction Offset / Length Register 19
34	CPU Transaction Offset / Length Register 20
35	CPU Transaction Offset / Length Register 21
36	CPU Transaction Offset / Length Register 22
37	CPU Transaction Offset / Length Register 23
38	CPU Transaction Offset / Length Register 24
39	CPU Transaction Offset / Length Register 25
3A	CPU Transaction Offset / Length Register 26
3B	CPU Transaction Offset / Length Register 27
3C	CPU Transaction Offset / Length Register 28
3D	CPU Transaction Offset / Length Register 29
3E	CPU Transaction Offset / Length Register 30
3F	CPU Transaction Offset / Length Register 31

VME Address Range Enable Register (VAER)

The VME Address Range Enable Register contains the Enable bits for the selection of the VME address range. It is addressed through the Register Select field of the CSR Access Register (VCAR) with a code of 05 (hex). This register is used in VME-generated transactions only.

NOTE: The DWMVA does not respond to short address transactions.

ADDRESS *VCAR Register Select Code 05*

bits<31:27>

Name: Must be zero

Mnemonic: MBZ

Type: R/W, 0

Reserved; must be zero.

bit<26>

Name: VME Extended Address Range Enable

Mnemonic: None

Type: R/W, 0

The VME Extended Address Range Enable bit controls DMA transactions in extended address space from VME masters to the XMI through the DWMVA. When this bit is set, the DWMVA accepts all VME transactions with VME A29–A31 equal to zero, provided they have valid address modifier codes (see Table 5–4). When this bit is clear, the DWMVA does not accept extended address transactions.

bit<25>

Name: VME Standard Address Range Enable

Mnemonic: None

Type: R/W, 0

The VME Standard Address Range Enable bit controls DMA transactions in standard address space from VME masters to the XMI through the DWMVA. When this bit is set, the DWMVA accepts all VME transactions with VME A23 equal to zero, provided the transactions have valid address modifier codes (see Table 5–4). When this bit is clear, the DWMVA does not accept standard address transactions.

bits<24:0>

Name: Must Be Zero

Mnemonic: None

Type: R/W, 0

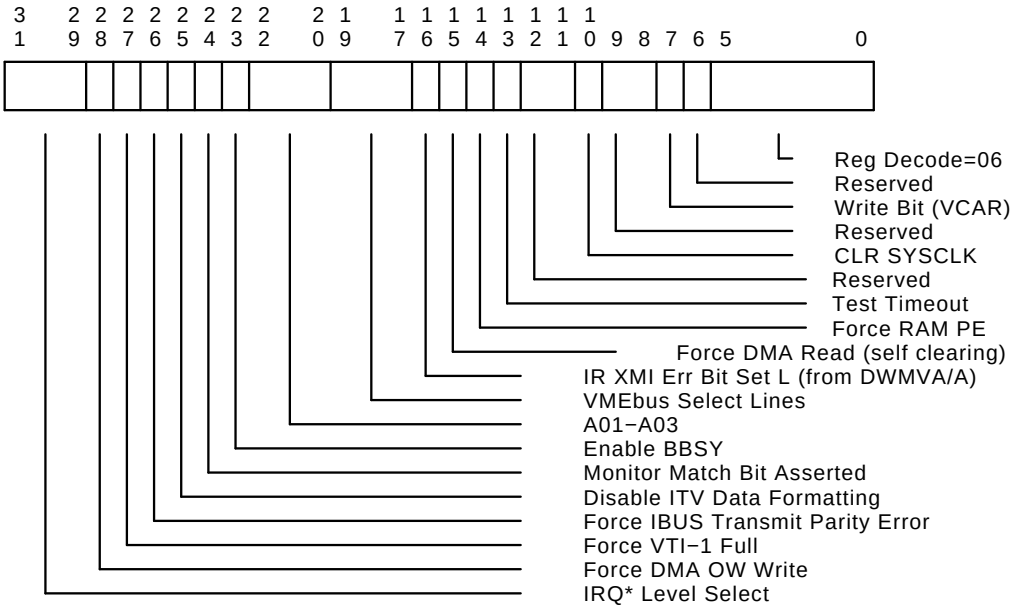
Reserved; must be zero.

Diagnostic Register (VDR)

The Diagnostic Register controls diagnostic operations performed on the C3200 module.

NOTE: This register is reserved for use by Digital.

ADDRESS *VCAR Register Select Code 06*

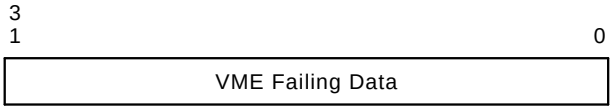


msb−p409−91

Failing Data Register (VFDR)

The Failing Data Register holds the VME failing data bits at transaction timeout.

ADDRESS *VCAR Register Select Code 07*



msb−p410−91

bits<31:0>

Name: VME Failing Data

Mnemonic: None

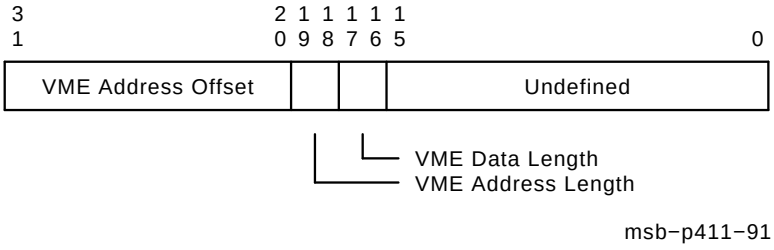
Type: RO, 0

The VME Failing Data field holds data of the transaction at transaction timeout. Data is latched to this register on every cycle and is locked on transaction timeout. The data is unlocked when the VME Transaction Timeout bit (AESR<23>) is cleared.

CPU Transaction Address Offset Registers (VAOR)

The 32 CPU Transaction Address Offset Registers contain the VME address offsets to be appended to XMI <19:0> to form a VME address (see Figure 2–5). These registers also define the VME data lengths and address lengths.

ADDRESS *VCAR Register Select Codes 20–3F*



bits<31:20>

Name: VME Address Offset

Mnemonic: None

Type: R/W, 0

The contents of the VME Address Offset field is appended to XMI <19:0> to form a VME address.

bits<19:18>

Name: VME Address Length

Mnemonic: None

Type: R/W, 0

The VME Address Length field indicates the address length of a VME transaction to be generated in response to a CPU transaction to a given address. These bits are encoded as follows:

VAOR <19:18>	VME Address Length
00	Extended address
01	Invalid
10	Short address[1]
11	Standard address

[1]The DWMVA does not respond to short address transactions.

bits<17:16>

Name: VME Data Length

Mnemonic: None

Type: R/W, 0

The VME Data Length field indicates the length of a VME transaction to be generated in response to a CPU read transaction to a given address. These bits are encoded as follows:

VAOR <17:16>	VME Data Length
00	Invalid
01	Byte
10	Word (2 bytes)
11	Longword (4 bytes)

The data length of a write transaction is determined by the mask field of the IBUS.

bits<15:0>

Name: Reserved

Mnemonic: None

Type: RO, undefined

Reserved; initial states are undefined.

8 Initialization

The DWMVA subsystem can be initialized in two ways:

- As an XMI node
- As a specific I/O adapter

As an XMI node, the DWMVA can be initialized at one of two levels:

- System level—Through system power-down/power-up or XMI power-up sequence emulation. When the system is powered up, XMI AC LO L and XMI DC LO L are sequenced so that all XMI nodes are reset. The XMI emulates a power-up when software asserts the XMI RESET L line (by writing to IPR55), causing the power supply to sequence XMI AC L and XMI DC L as in hardware power-up.
- Node level—Through a node reset caused by writing bit <30> of the DWMVA Bus Error Register.

The C3200 state machines can be initialized locally by writing one to bit <22> of the Device/Configuration Register.

Following DWMVA initialization:

- All DWMVA logic is reset to a known state.
- The DWMVA asserts XMI STF L as required by the XMI specification. This signal remains asserted until self-test completes successfully.
- The DWMVA asserts SYSRESET* to cause all devices on the VME to perform initialization.
- With the deassertion of the reset condition, all DWMVA registers assume their default state. Any desired nondefault values must be written to the registers.

A VME Interface Signal List

The VME interface is made up of two 96-pin connectors, referred to as J1 and J2. Table A–1 and Table A–2 list the pin assignments for these connectors.

Table A–1 DWMVA-to-VME Interface (J1 Connector)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SETCLK	A17
22	IACKOUT*	SERDAT	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5V STDBY	+12V
32	+5V	+5V	+5V

Table A–2 DWMVA-to-VME Interface (J2 Connector)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	User-defined	+5V	User-defined
2	User-defined	GND	User-defined
3	User-defined	RESERVED	User-defined
4	User-defined	A24	User-defined
5	User-defined	A25	User-defined
6	User-defined	A26	User-defined
7	User-defined	A27	User-defined
8	User-defined	A28	User-defined
9	User-defined	A29	User-defined
10	User-defined	A30	User-defined
11	User-defined	A31	User-defined
12	User-defined	GND	User-defined
13	User-defined	+5V	User-defined
14	User-defined	D16	User-defined
15	User-defined	D17	User-defined
16	User-defined	D18	User-defined
17	User-defined	D19	User-defined
18	User-defined	D20	User-defined
19	User-defined	D21	User-defined
20	User-defined	D22	User-defined
21	User-defined	D23	User-defined
22	User-defined	GND	User-defined
23	User-defined	D24	User-defined
24	User-defined	D25	User-defined
25	User-defined	D26	User-defined
26	User-defined	D27	User-defined
27	User-defined	D28	User-defined
28	User-defined	D29	User-defined
29	User-defined	D30	User-defined
30	User-defined	D31	User-defined
31	User-defined	GND	User-defined
32	User-defined	+5V	User-defined

B VME-to-XMI Byte Swapping

The DWMVA adapter implements byte swapping in hardware to allow XMI devices to interpret data previously stored on the VMEbus. Byte swapping is necessary for three reasons:

1. The VMEbus format is *big endian*, while the XMI bus format is *little endian*.
2. The format used by big endian processors (common on VME devices) to store data depends on the type of data—character or integer.
3. The VME protocol allows a given byte to be transferred on different VME byte lanes in VME transfers of different lengths.

The DWMVA adapter swaps bytes so that VME data appears correctly on the XMI bus and XMI data appears correctly on the VMEbus.

VME data is justified to the low-order byte lanes. XMI data, however, is relative only to its position within a quadword, not the length of the transfer.

The architecture of big endian processors introduces another level of complexity. The order in which bytes are stored in memory is different depending upon the type of data stored. Integer bytes, words, and longwords are all stored differently while character data is represented in still another fashion. This requires that the DWMVA subsystem understand what type of data it is transferring so that data can be swapped correctly.

Figure B–1 illustrates the big endian byte lane formats.

B.1 Definition of Terms

The following definitions apply to terms used in this appendix.

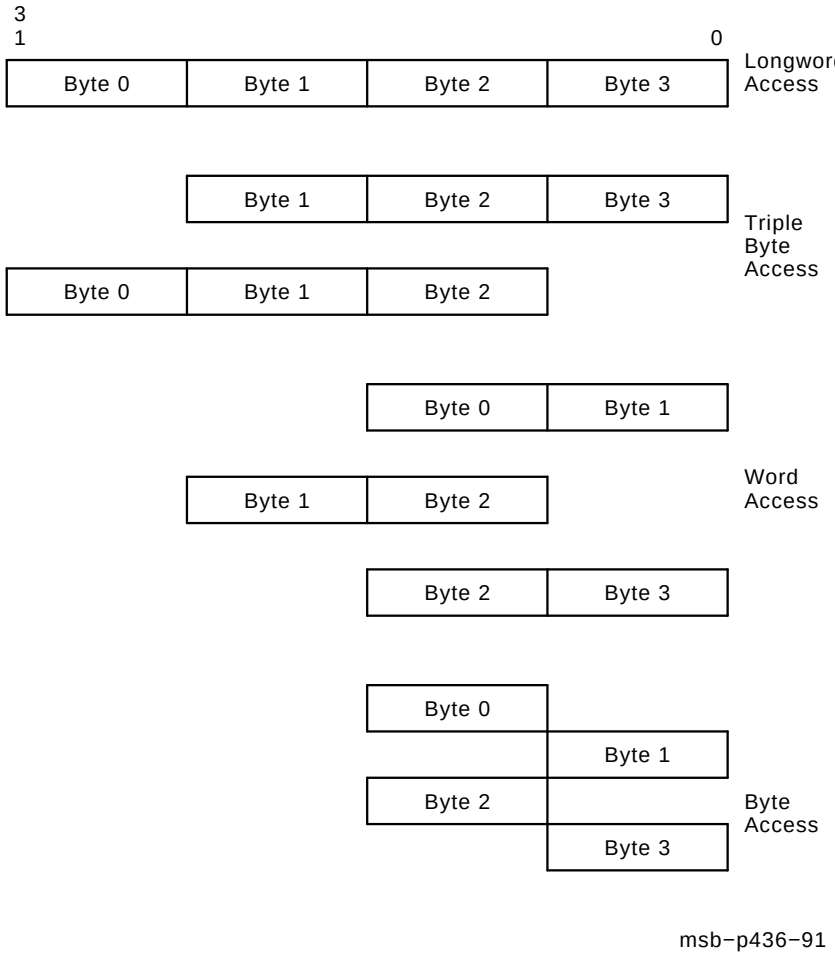
Byte 0 - The byte in a longword that is addressed when the two lowest order address lines (XMI A<1:0> and VME A01,DS1*) are 00.

Byte 1 - The byte in a longword that is addressed when the two lowest order address lines (XMI A<1:0> and VME A01,DS1*) are 01.

Byte 2 - The byte in a longword that is addressed when the two lowest order address lines (XMI A<1:0> and VME A01,DS1*) are 10.

Byte 3 - The byte in a longword that is addressed when the two lowest order address lines (XMI A<1:0> and VME A01,DS1*) are 11.

Figure B–1 Big Endian VME Byte Lane Formats



Byte Lanes - The VME data lines used for a given length transfer. Different bytes (*Byte 0–Byte 3*) will be transferred on different byte lanes depending on the size of the transfer. Table B–1 illustrates the relationship of data lines to byte lanes.

Byte Swapping - The process of changing the address at which a byte is referenced relative to the base address, where the base address is the lowest byte address within a set of contiguous bytes used to store data.

Table B–1 Byte Lanes for Different Sizes of VME Transfers

Byte Lanes	VME D24–D31	VME D16–D23	VME D08–D15	VME D0–D07
Byte Access				
Byte0			Byte0	
Byte1				Byte1
Byte2			Byte2	
Byte3				Byte3
Word Access				
Byte0–1			Byte0	Byte1
Byte1–2		Byte1	Byte2	
Byte2–3			Byte2	Byte3
Triple Byte Access				
Byte0–2	Byte0	Byte1	Byte2	
Byte1–3		Byte1	Byte2	Byte3
Longword Access				
Byte0–3	Byte0	Byte1	Byte2	Byte3

B.2 Byte Swapping in Data Storage

Big endian and little endian processors store data in architecturally different ways. The big endian processor architecture defines the byte stored at the higher address to be the least significant, and the byte stored at the lowest address to be the most significant. Little endian processor's storage convention is the opposite. Therefore, integer longwords stored by either processor are transposed for the other. The byte order for word data is transposed within words but words are not transposed within the longword.

Both processors store character byte strings the same way, beginning at byte address 0. Therefore, passing character data between processors requires no translation. Clearly, a hardware solution for aligning integer bytes will not work for byte strings.

Figure B–2 and Figure B–3 illustrate the conventions used by little endian and big endian processors for word and longword integer data. Storage of byte string data is shown in Figure B–4.

Figure B–2 Little Endian Integer Data Storage

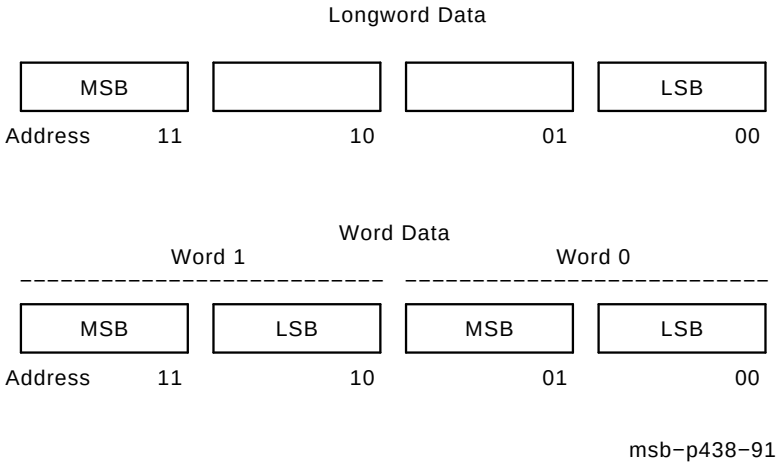


Figure B–3 Big Endian Integer Data Storage

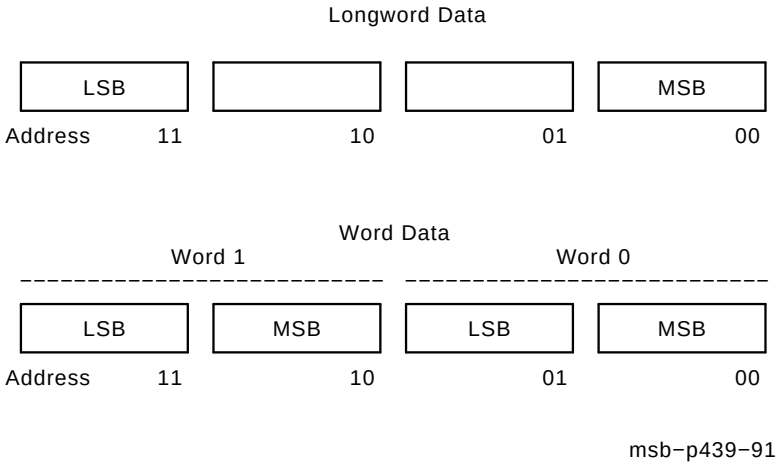
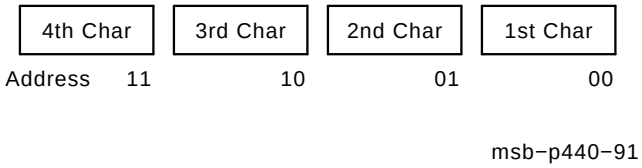


Figure B–4 Byte String Storage



The fundamental issue of sharing data over the VMEbus is the requirement of the big endian data format that any type of processor have complete knowledge of the data structure to interpret the data correctly. Different translation algorithms are required for integer bytes and integer words stored as 16-bit values, integer words packed as longwords, and longwords. Furthermore, correct interpretation of byte streams requires yet another algorithm. Thus, an application must recognize the data type to make the right translation.

The term "byte swapping" reflects the differences in the way that little endian and big endian processors reference bytes within longwords. Little endian processors store data in 32-bit longwords, with the most significant byte addressable at byte(3) and the least significant at byte(0). Word data is stored as if arrived in a stream. The MSB of the first word is stored at byte(1) and the LSB at byte(0). The MSB of the second word is stored at byte(3) and the LSB at byte(2). Bytes that are stored individually appear as if they were stored as a longword.

Big endian processors store longword integer data with the most significant byte at byte(0) and the least significant at byte(3). Word data is stored with the MSB of the first word at byte(0) and the LSB at byte(1). The MSB of the second word appears at byte(2) and the LSB at byte(3). Bytes stored individually appear as if they were stored as a longword.

B.3 DWMVA Byte Swapping Requirements

Four categories of swapping are required for VME-to-XMI and XMI-to-VME transactions. These categories are referred to as Mode 0, Mode 1, Mode 2, and Mode 3. The swapping mode is selected by configuring bits <1:0> of the DWMVA Byte Swap RAM Access Register. Table B–2 shows the units swapped with the four swapping modes.

Table B–2 Byte Swapping Modes

RAR<1:0>	Mode	Swapped Unit
00	0	No Swap
01	1	Byte Swap
10	2	Word Swap
11	3	Longword Swap

B.3.1 Mode 0—No Swap

Mode 0 does not perform any swapping. For example, if the longword 0123 4567 hex (byte 67 = MSB) is written on the VMEbus and Mode 0 is selected on the DWMVA, the data pattern that appears on the XMI bus will be 6745 2301 (byte 67 = MSB). Figure B–5 shows Mode 0 swapping for 4-byte, 3-byte, 2-byte, and 1-byte transfers.

Figure B–5 Mode 0 (No Swap) Transfers

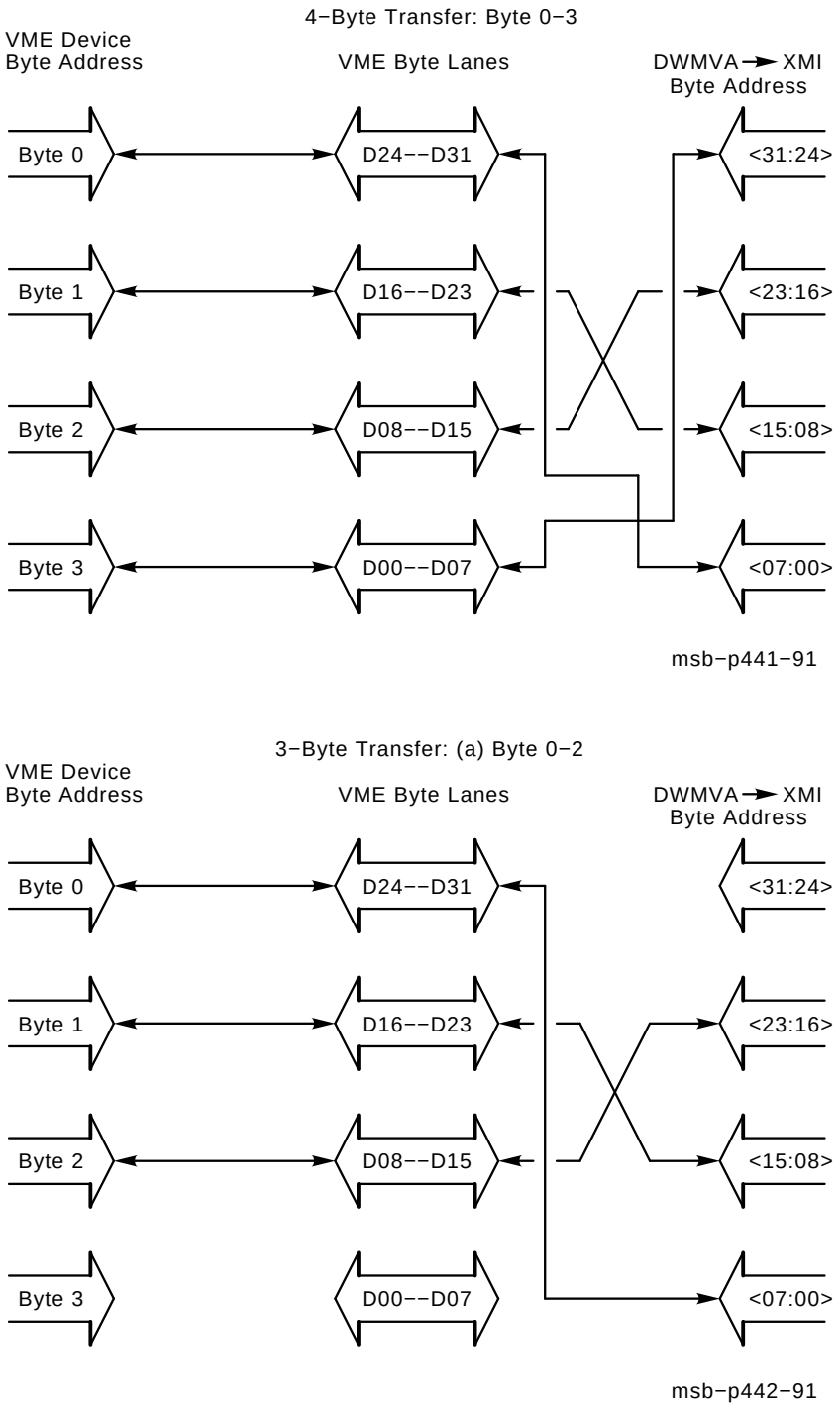


Figure B–5 Cont’d on next page

Figure B–5 (Cont.) Mode 0 (No Swap) Transfers

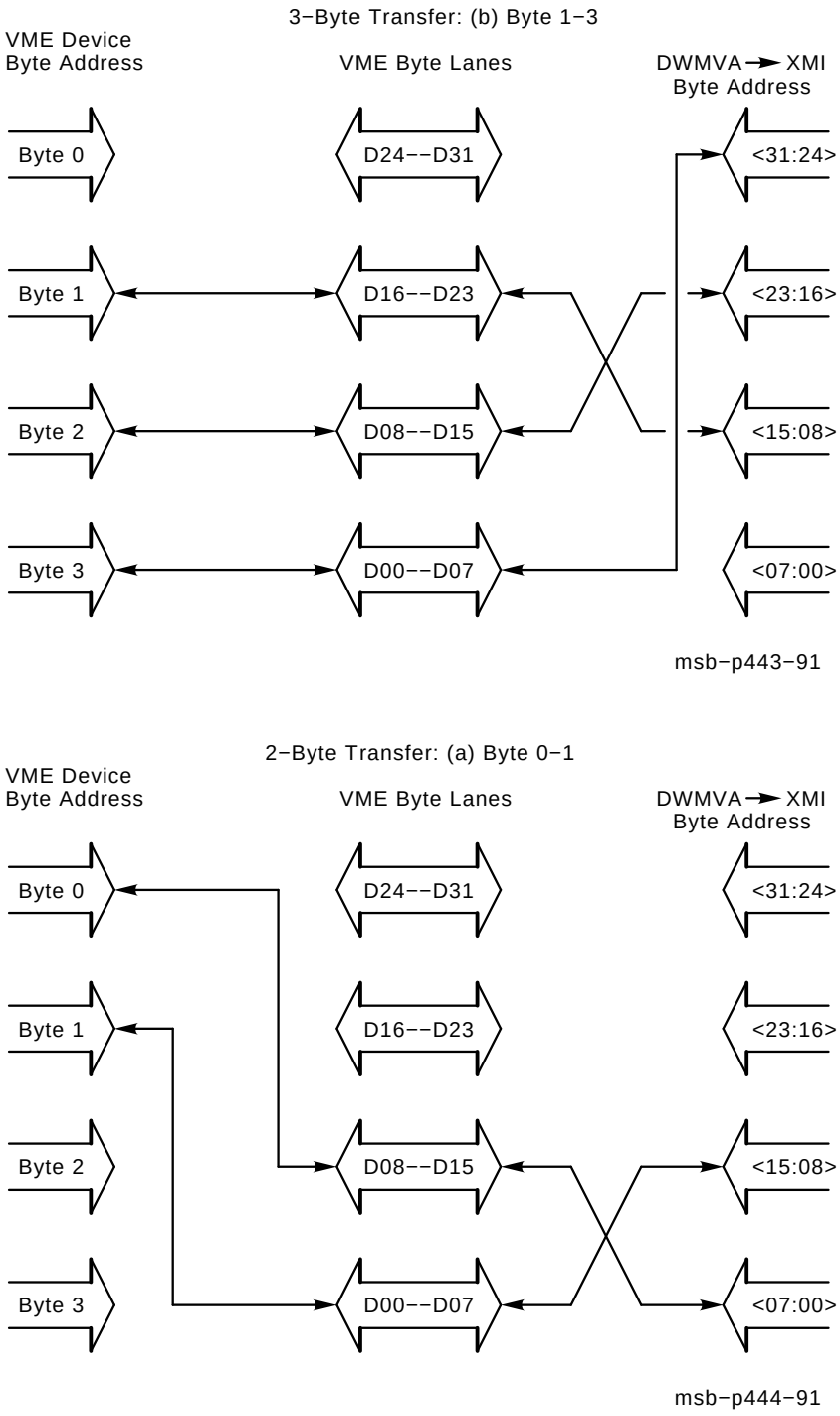


Figure B–5 Cont'd on next page

Figure B–5 (Cont.) Mode 0 (No Swap) Transfers

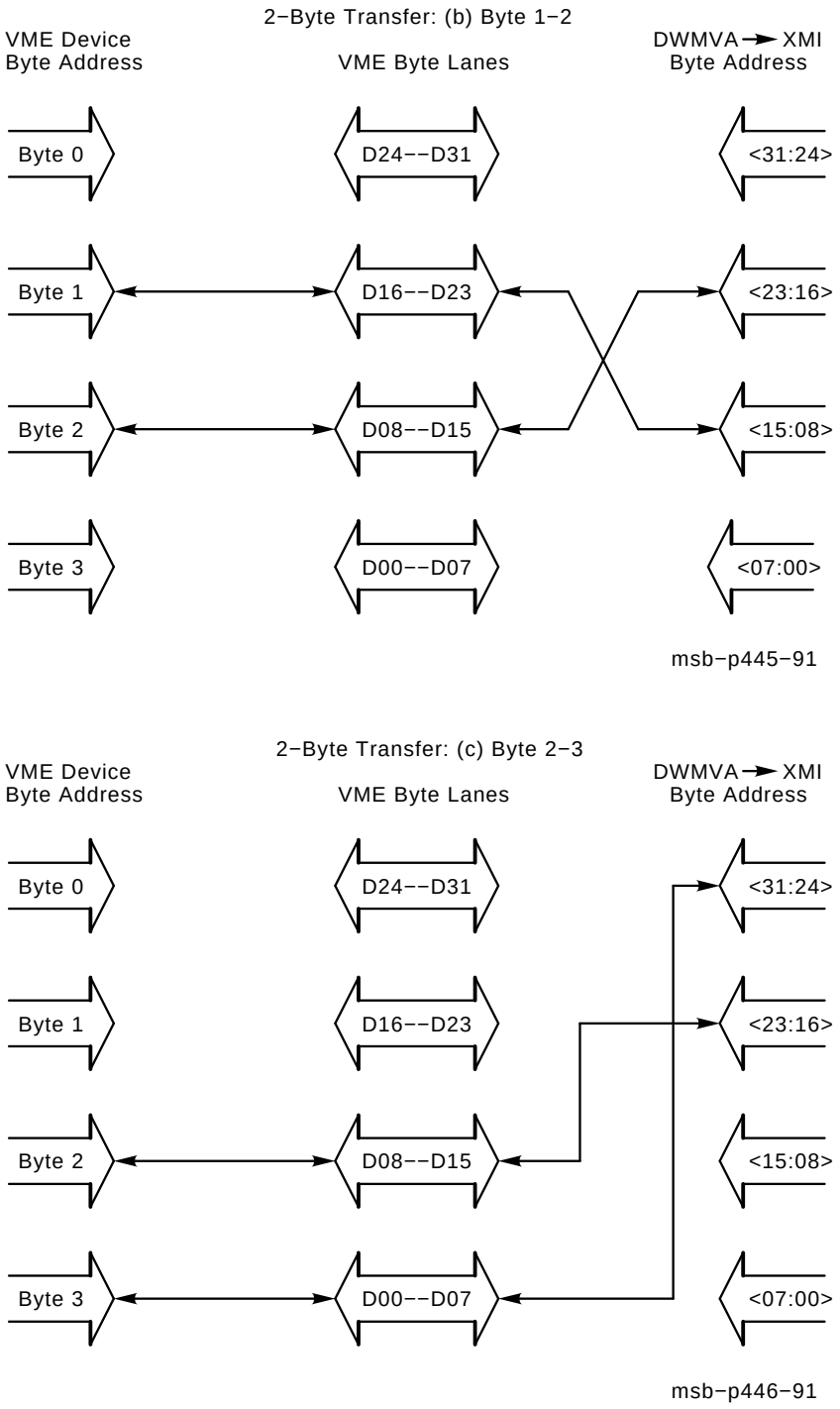


Figure B–5 Cont'd on next page

Figure B–5 (Cont.) Mode 0 (No Swap) Transfers

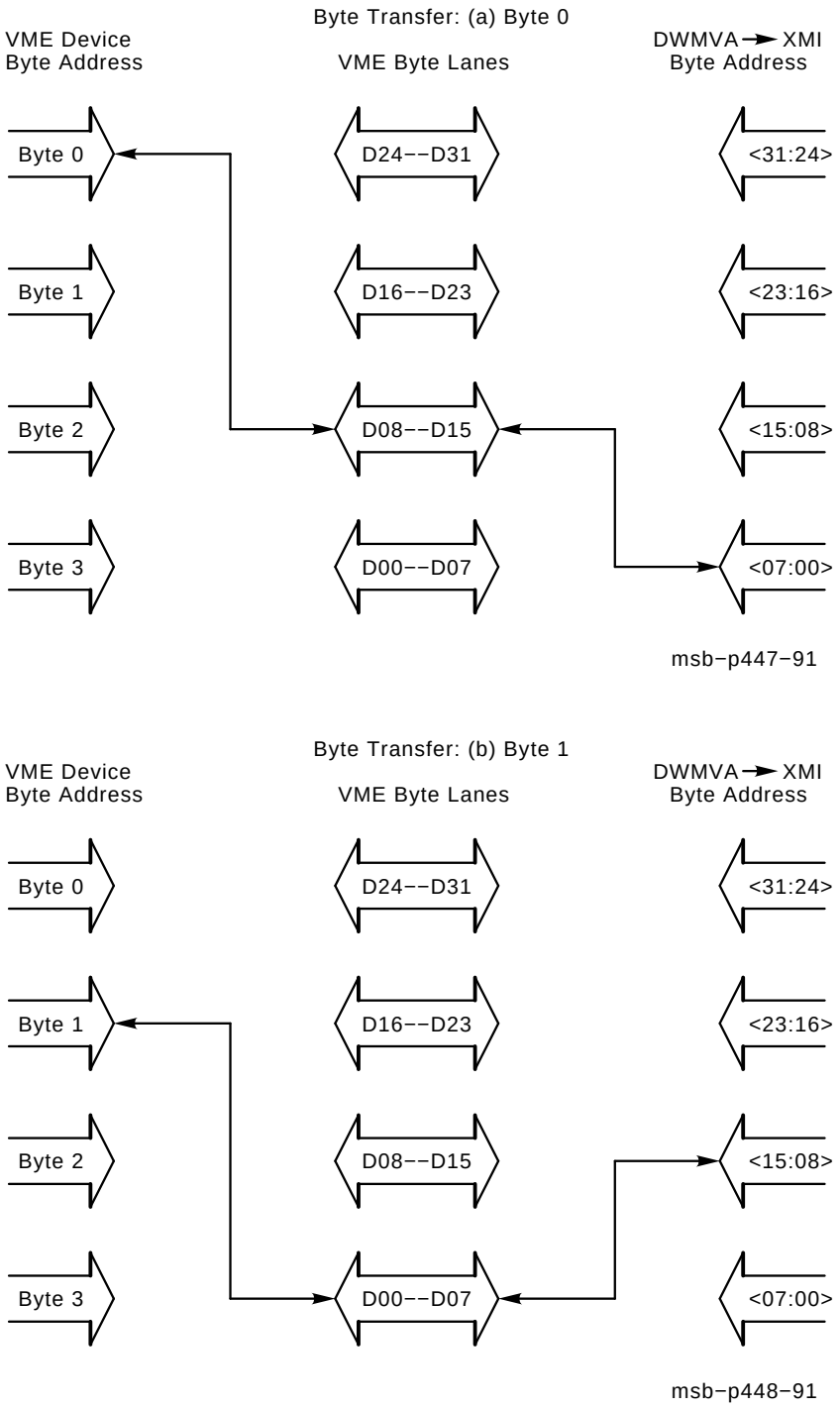
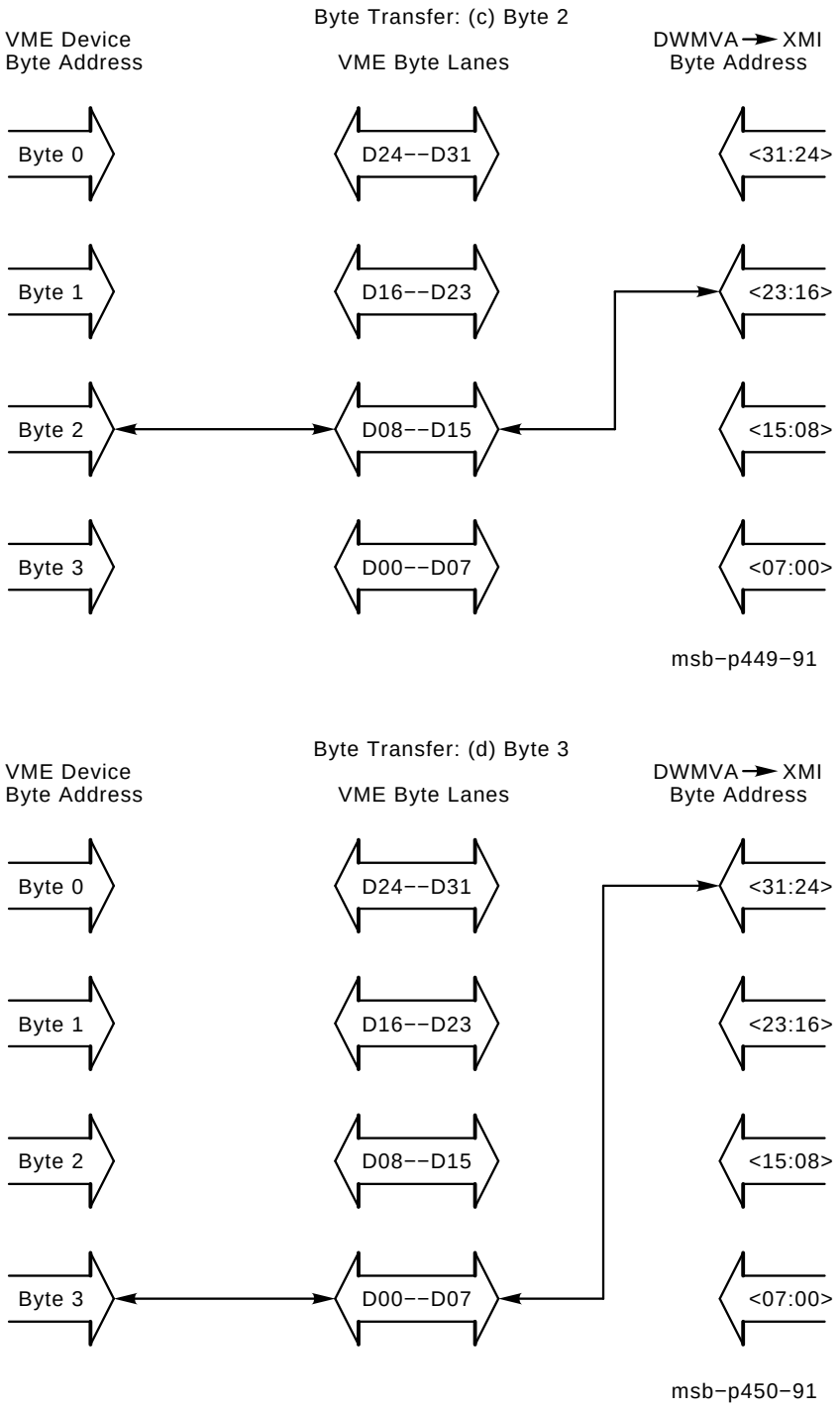


Figure B–5 Cont'd on next page

Figure B–5 (Cont.) Mode 0 (No Swap) Transfers



B.3.2 Mode 1—Byte Swap

Mode 1 swaps the bytes within each of the two words making up a longword of data. For example, if the longword 0123 4567 hex (byte 67 = MSB) is written on the VMEbus and Mode 2 is selected on the DWMVA, the data pattern that appears on the XMI bus will be 4567 0123 (byte 45 = MSB). Figure B–6 shows Mode 1 swapping for 4-byte, 3-byte, 2-byte, and 1-byte transfers.

Figure B–6 Mode 1 (Byte Swap) Transfers

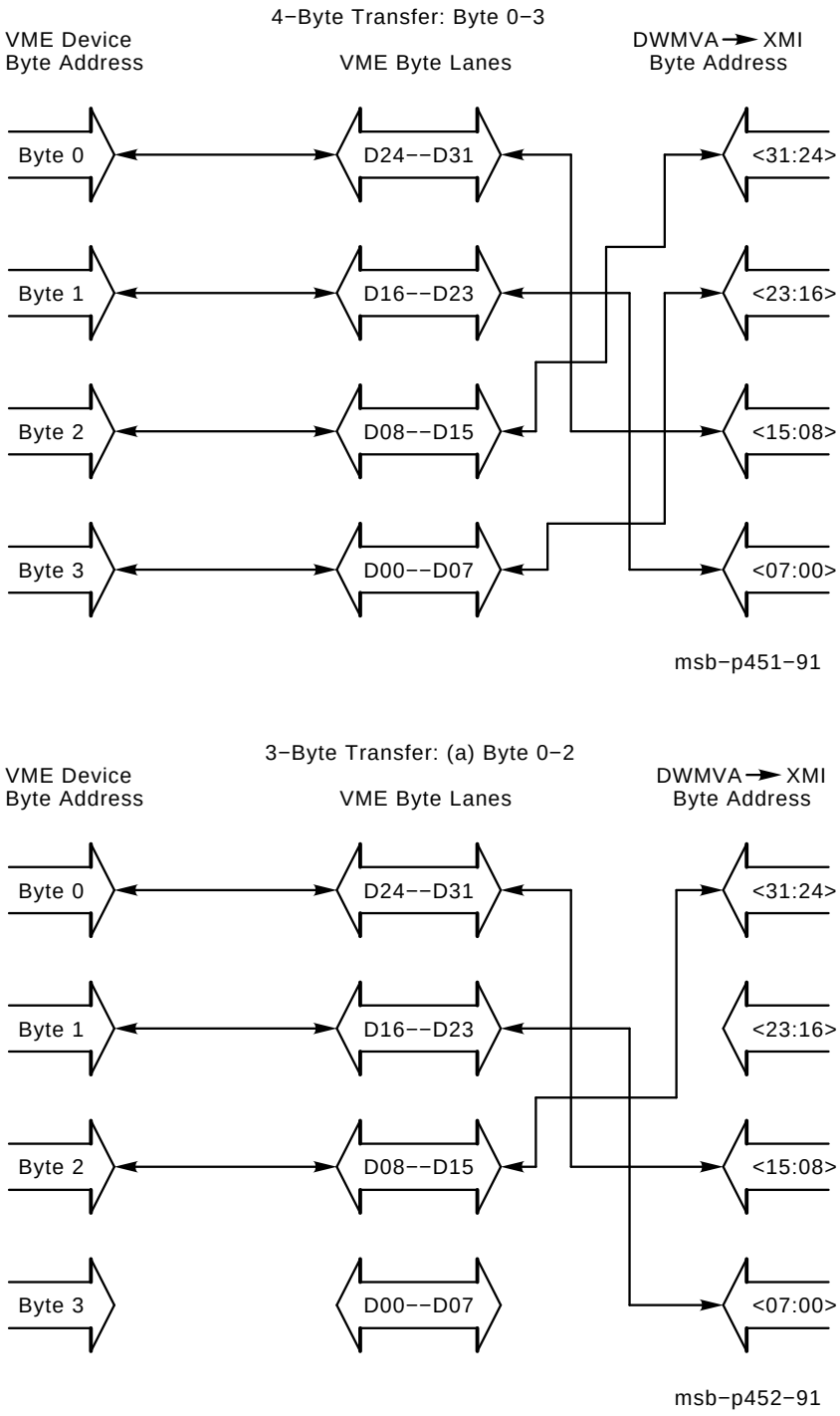


Figure B–6 Cont'd on next page

Figure B–6 (Cont.) Mode 1 (Byte Swap) Transfers

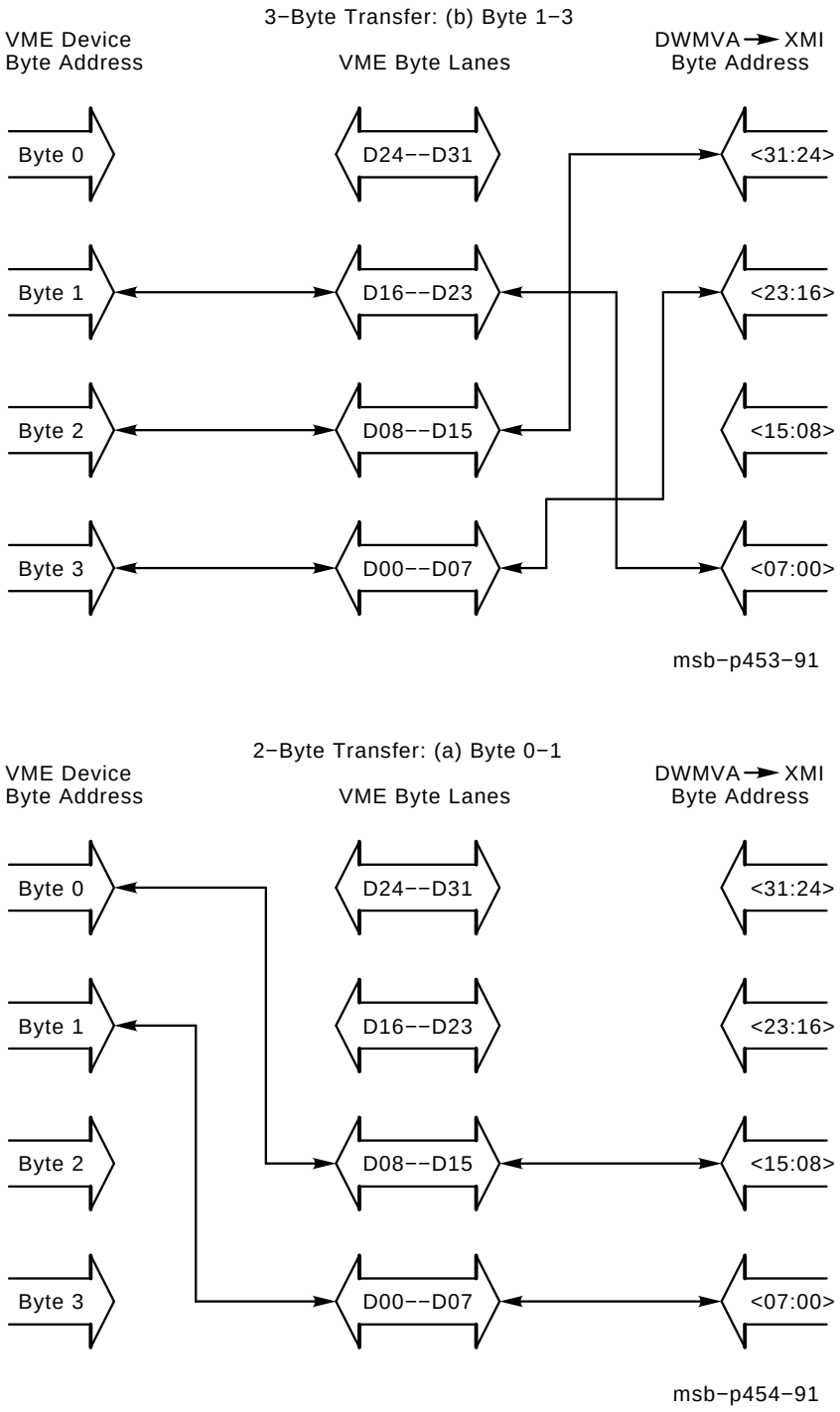


Figure B–6 Cont'd on next page

Figure B–6 (Cont.) Mode 1 (Byte Swap) Transfers

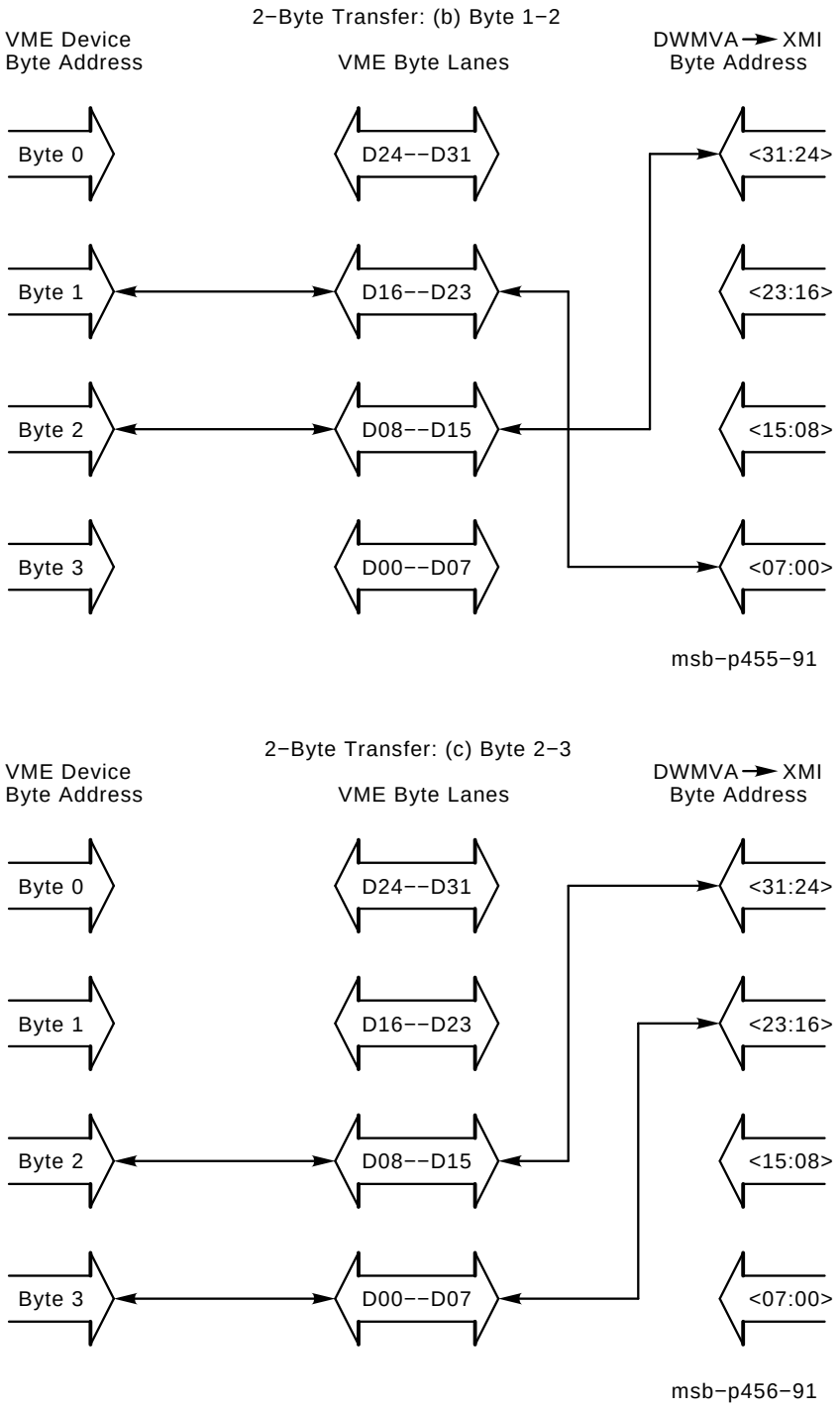


Figure B–6 Cont'd on next page

Figure B–6 (Cont.) Mode 1 (Byte Swap) Transfers

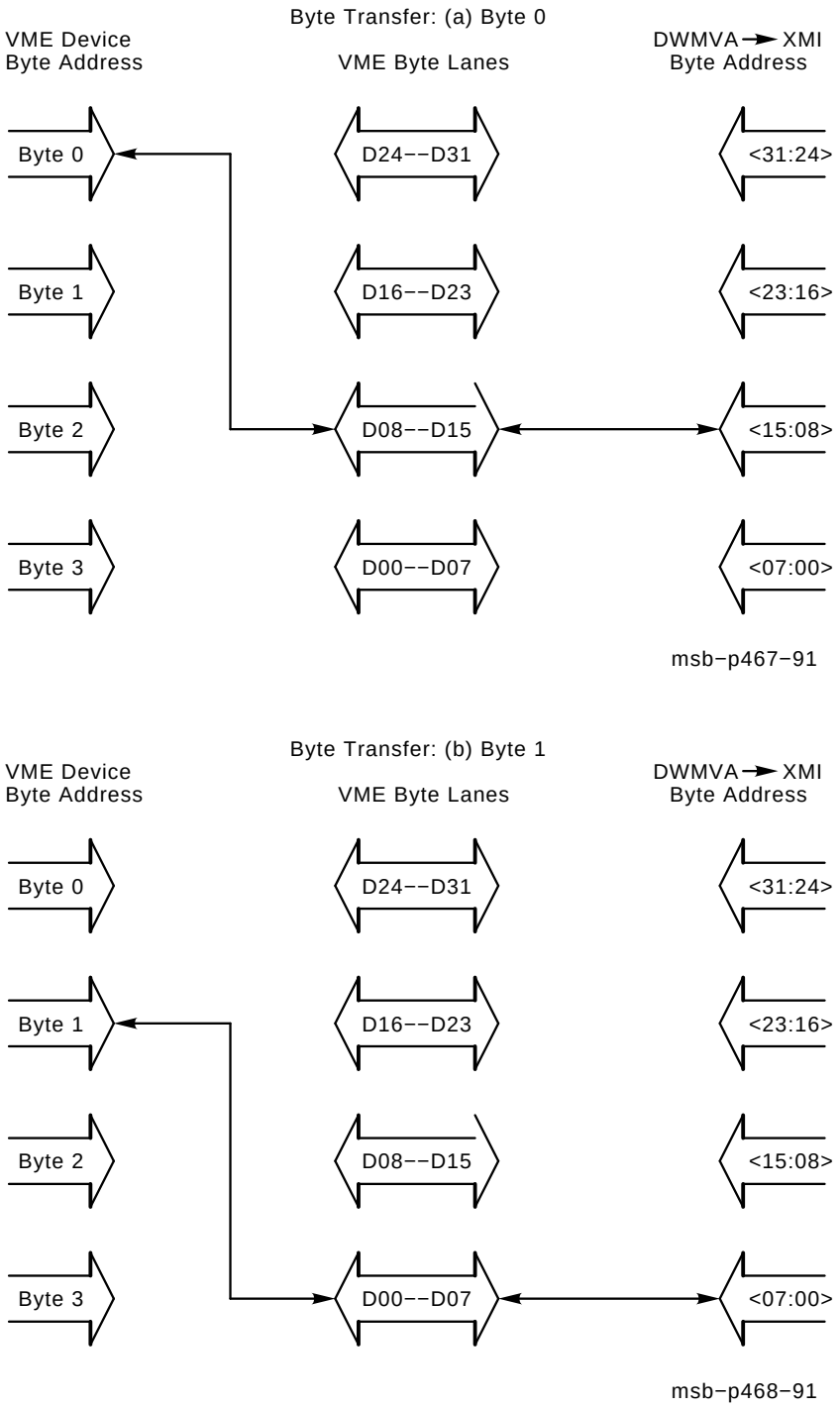
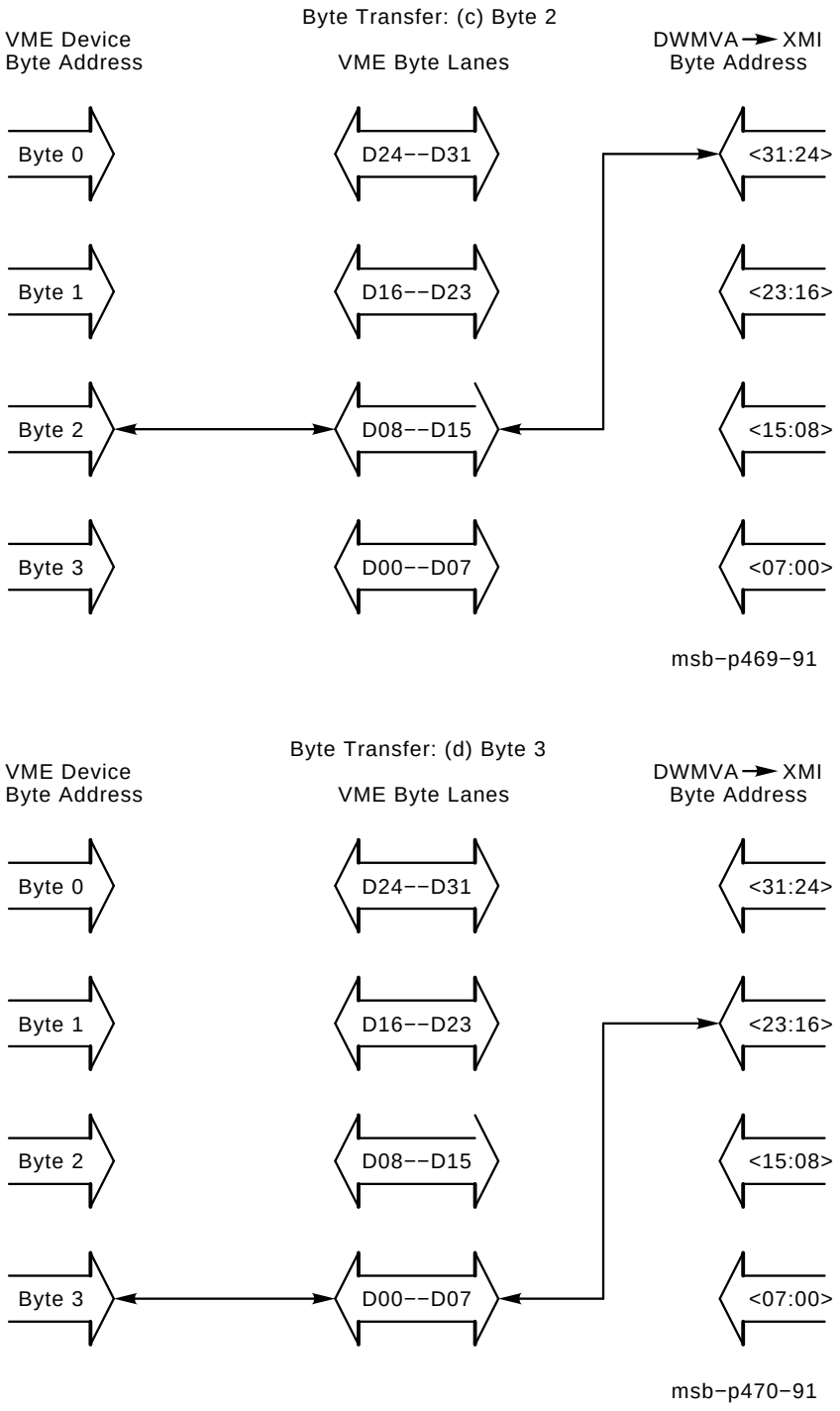


Figure B–6 Cont'd on next page

Figure B–6 (Cont.) Mode 1 (Byte Swap) Transfers



B.3.3 Mode 2—Word Swap

Mode 2 swaps the words within a longword. For example, if the longword 0123 4567 hex (byte 67 = MSB) is written on the VMEbus and Mode 2 is selected on the DWMVA, the data pattern that appears on the XMI bus will be 2301 6745 (byte 23 = MSB). Figure B–7 shows Mode 2 swapping for 4-byte, 3-byte, 2-byte, and 1-byte transfers.

Figure B–7 Mode 2 (Word Swap) Transfers

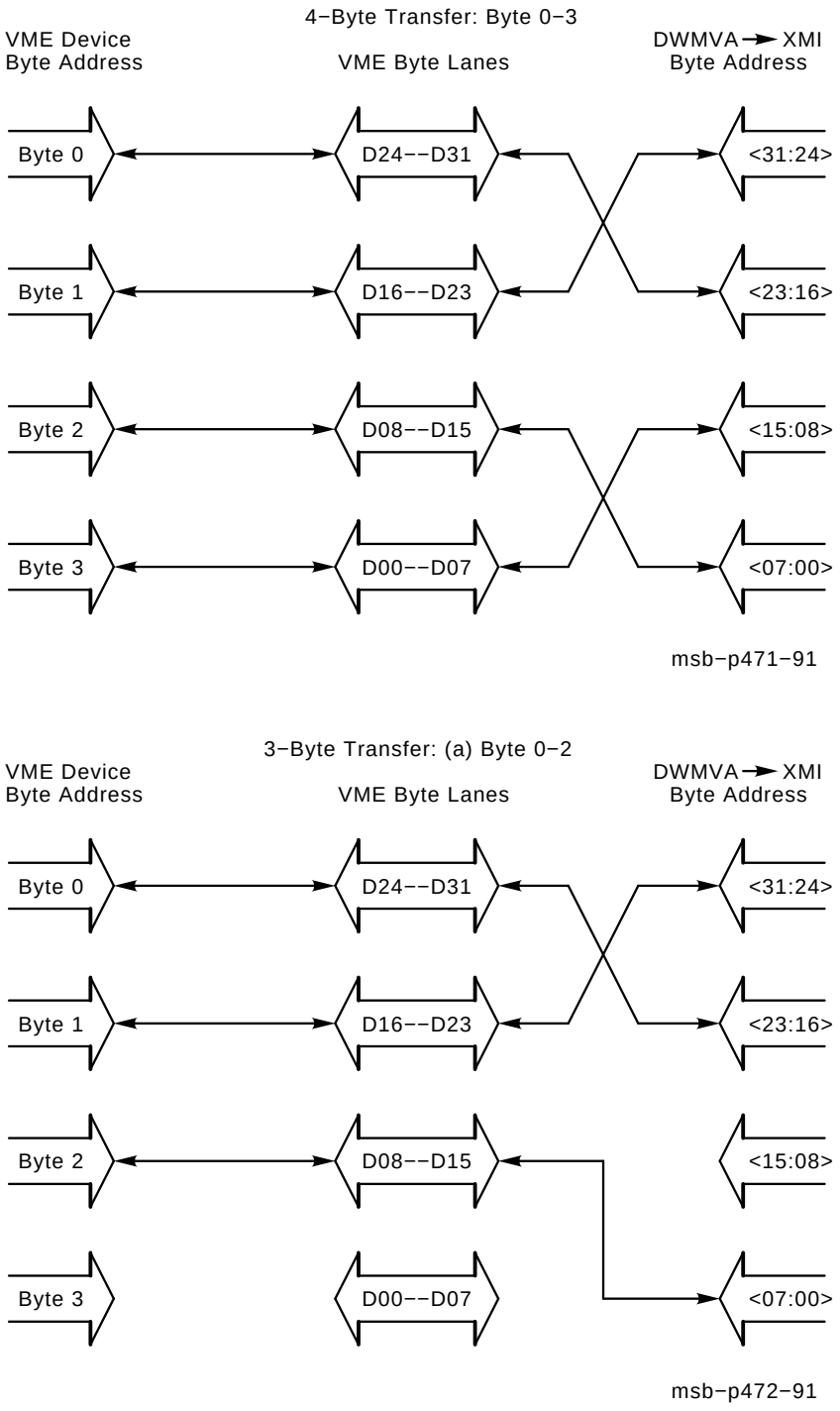


Figure B–7 Cont'd on next page

Figure B–7 (Cont.) Mode 2 (Word Swap) Transfers

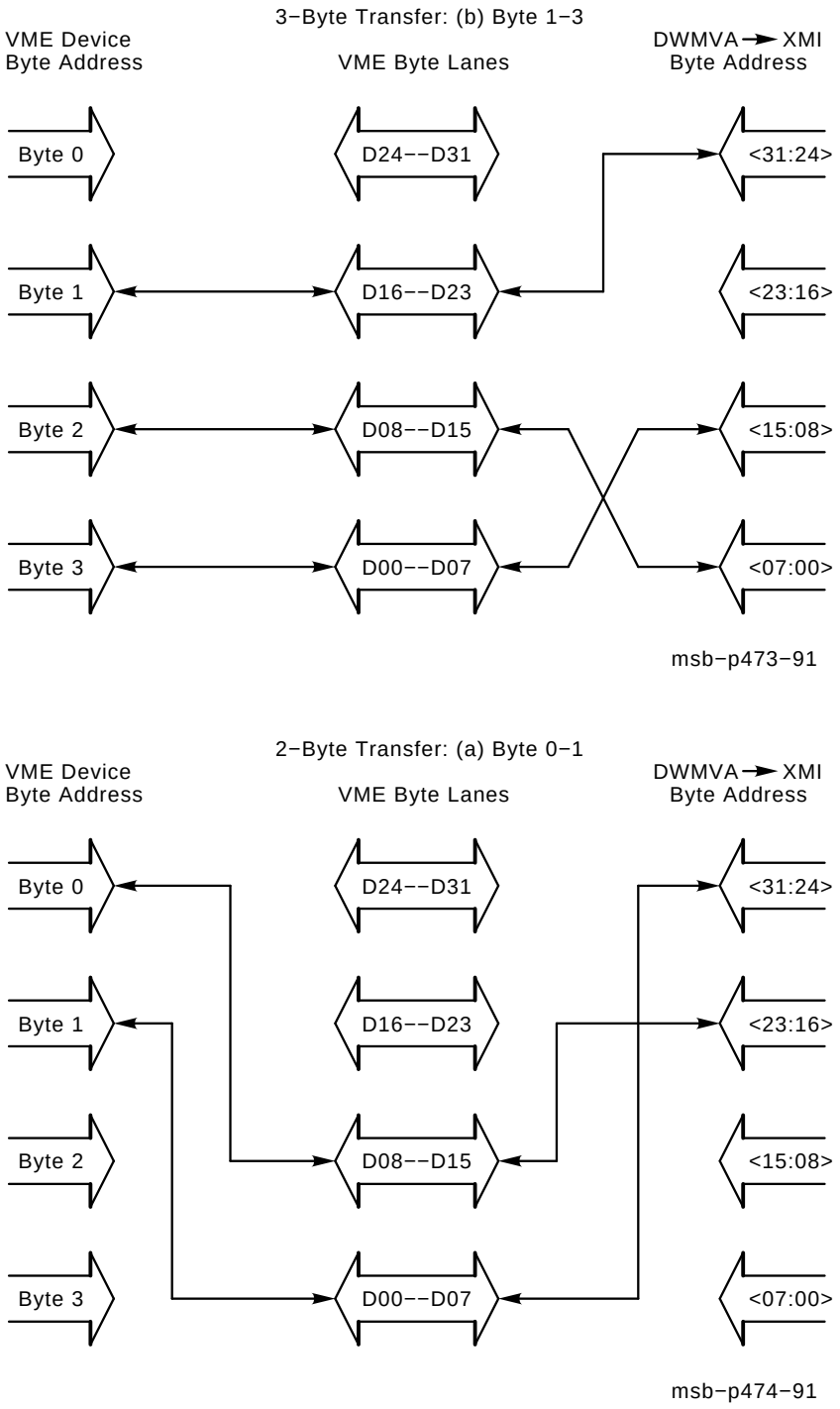


Figure B–7 Cont'd on next page

Figure B–7 (Cont.) Mode 2 (Word Swap) Transfers

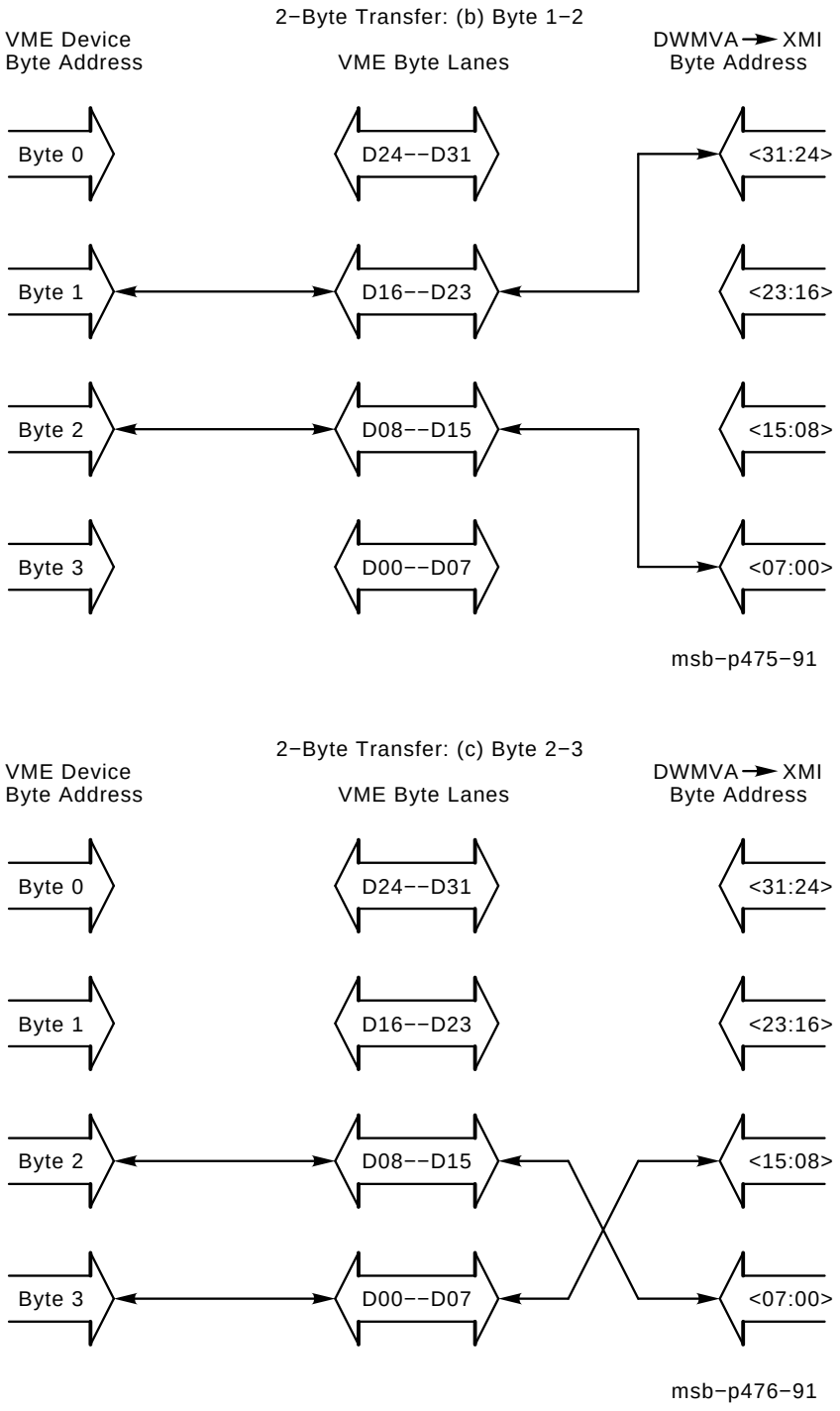


Figure B–7 Cont'd on next page

Figure B–7 (Cont.) Mode 2 (Word Swap) Transfers

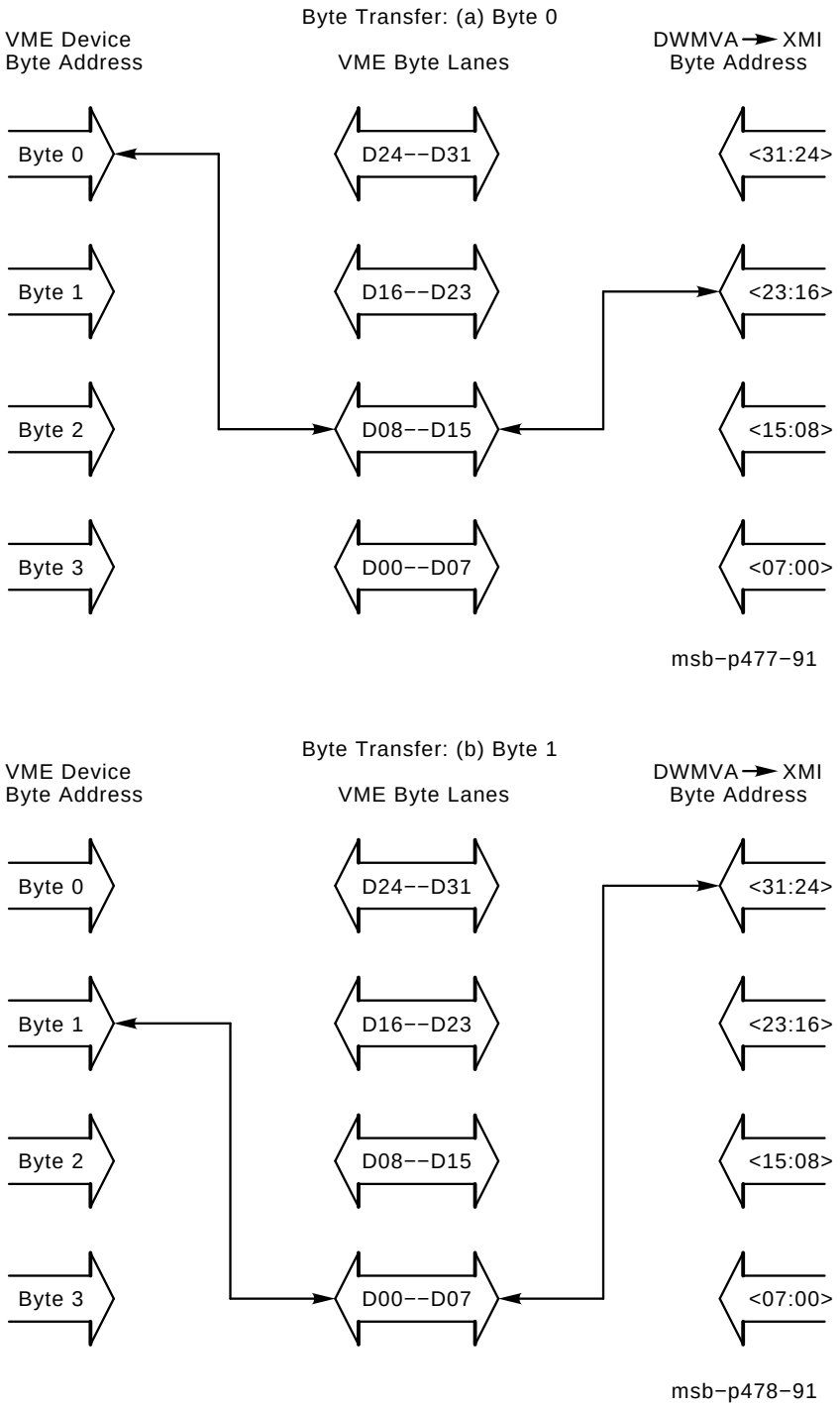
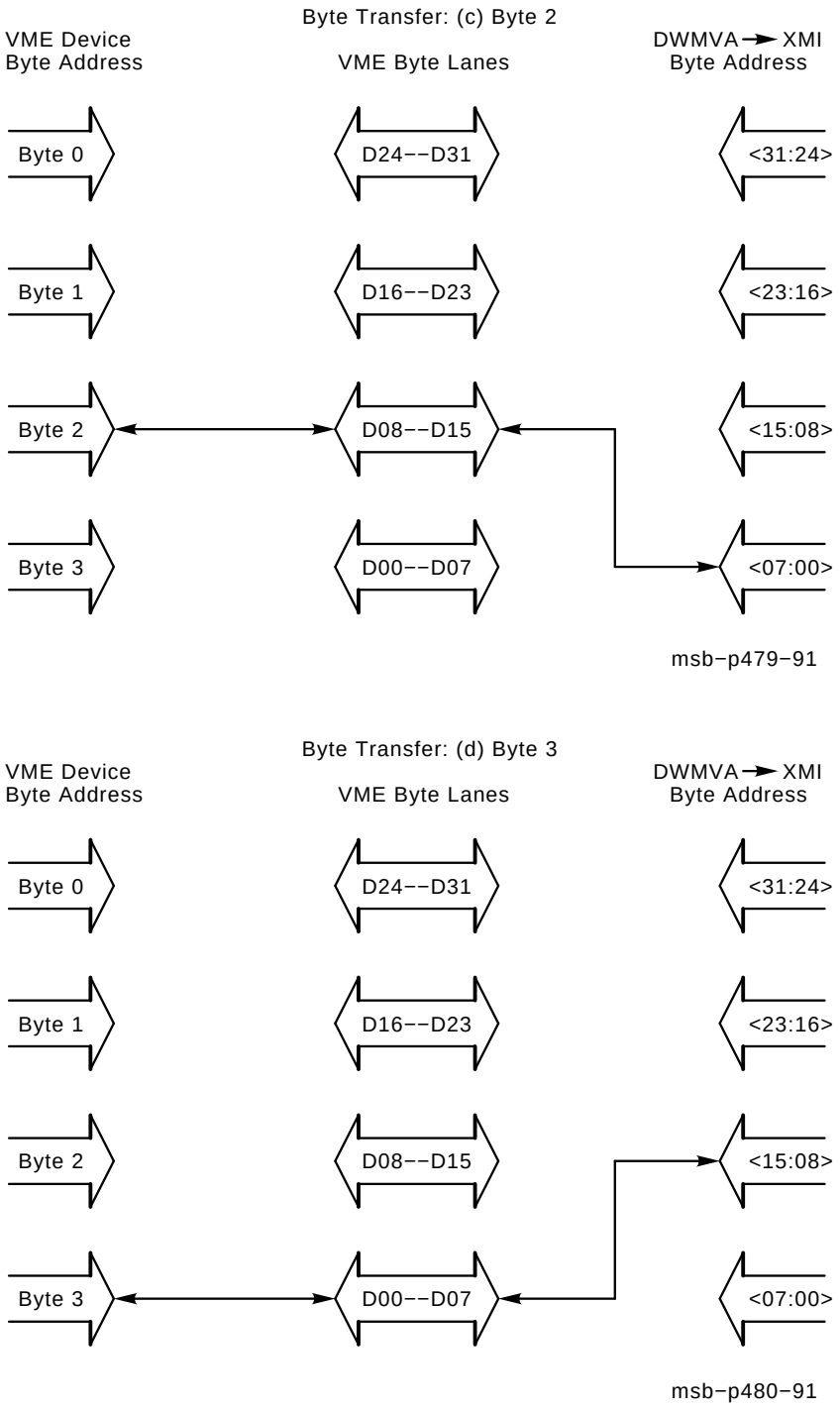


Figure B–7 Cont'd on next page

Figure B–7 (Cont.) Mode 2 (Word Swap) Transfers



B.3.4 Mode 3—Longword Swap

Mode 3 swaps longwords. For example, if the longword 0123 4567 hex (byte 67 = MSB) is written on the VMEbus and Mode 3 is selected on the DWMVA, the data pattern that appears on the XMI bus will be 0123 4567 (byte 01 = MSB). Figure B–8 shows Mode 3 swapping for 4-byte, 3-byte, 2-byte, and 1-byte transfers.

Figure B–8 Mode 3 (Longword Swap) Transfers

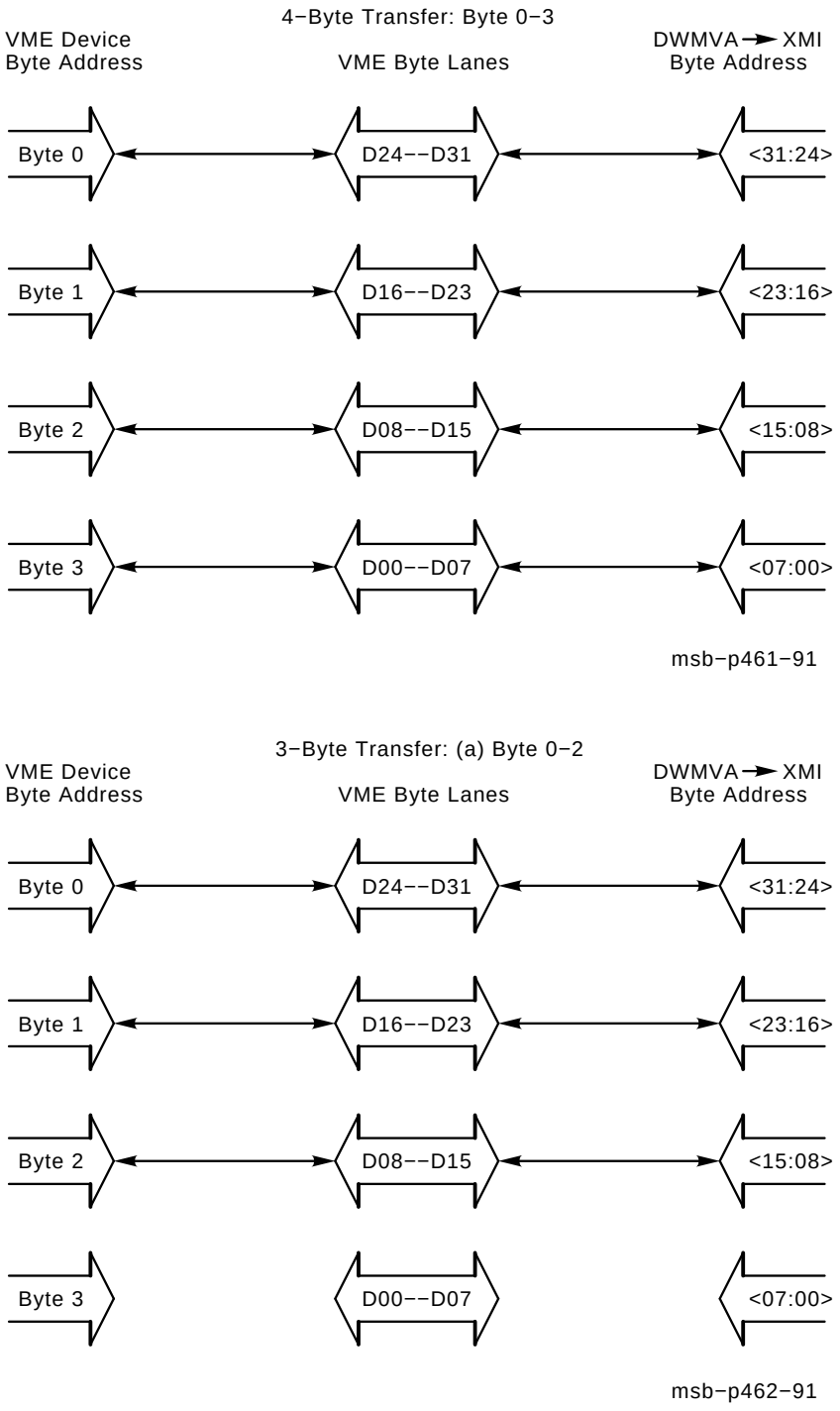


Figure B–8 Cont’d on next page

Figure B–8 (Cont.) Mode 3 (Longword Swap) Transfers

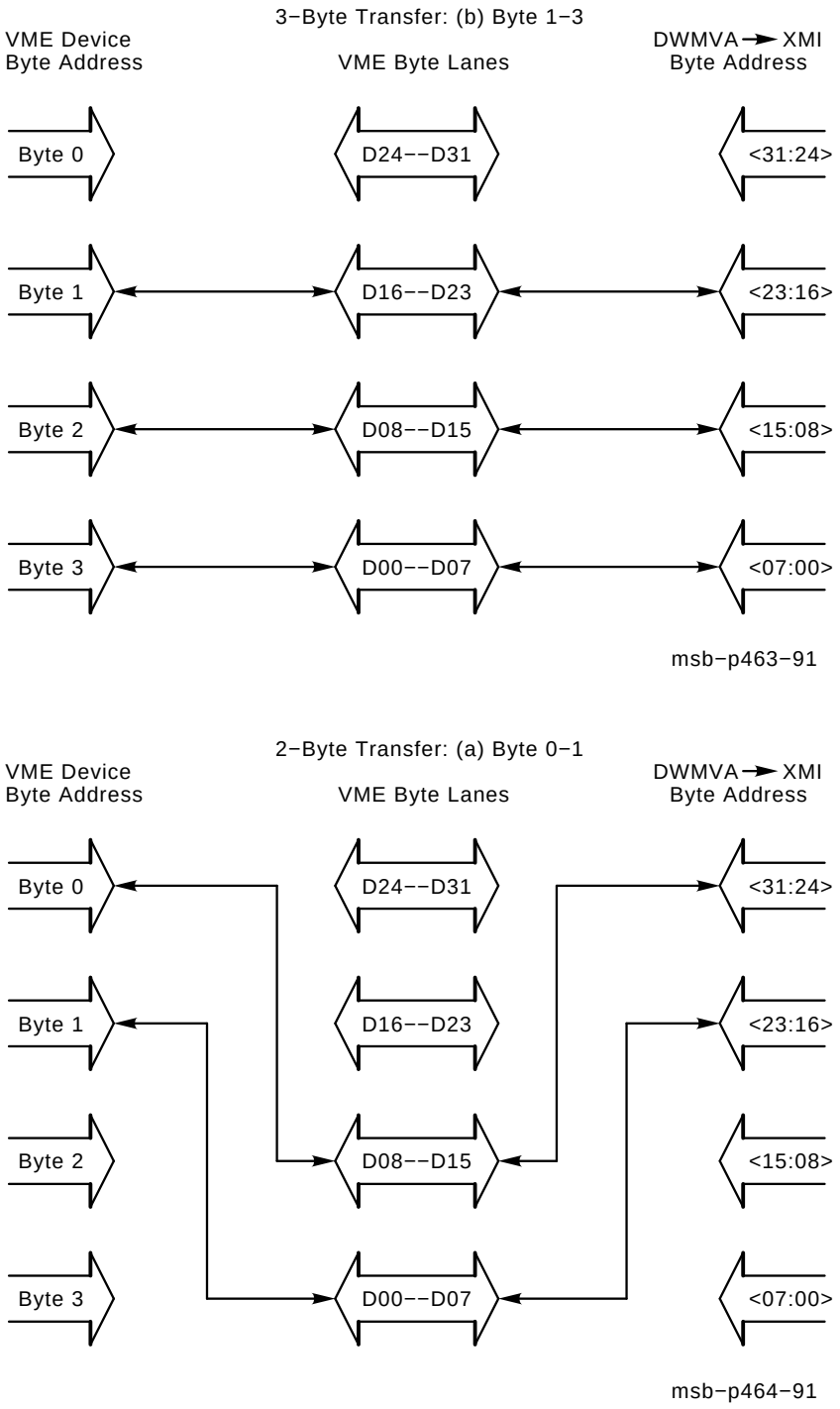


Figure B–8 Cont'd on next page

Figure B–8 (Cont.) Mode 3 (Longword Swap) Transfers

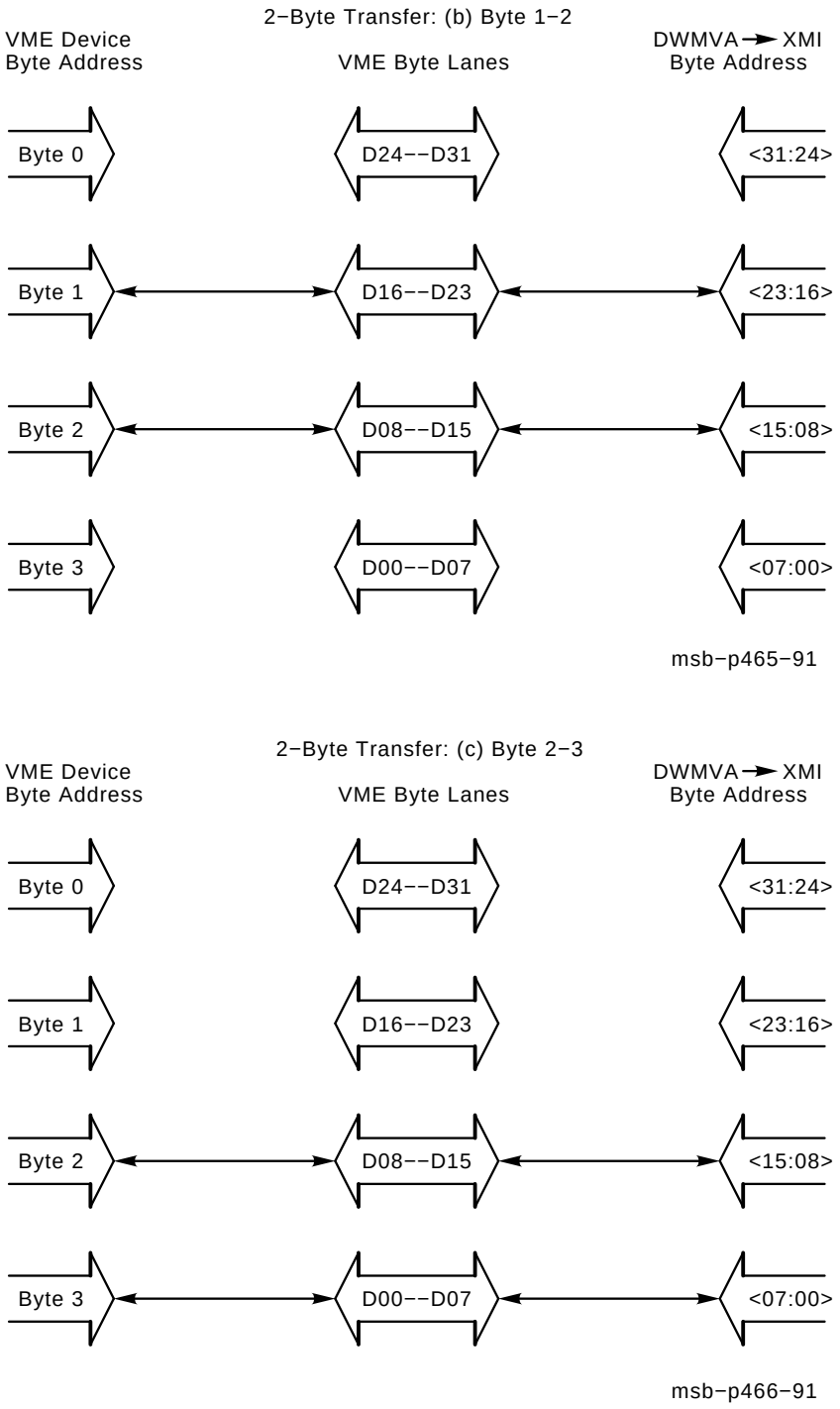


Figure B–8 Cont'd on next page

Figure B–8 (Cont.) Mode 3 (Longword Swap) Transfers

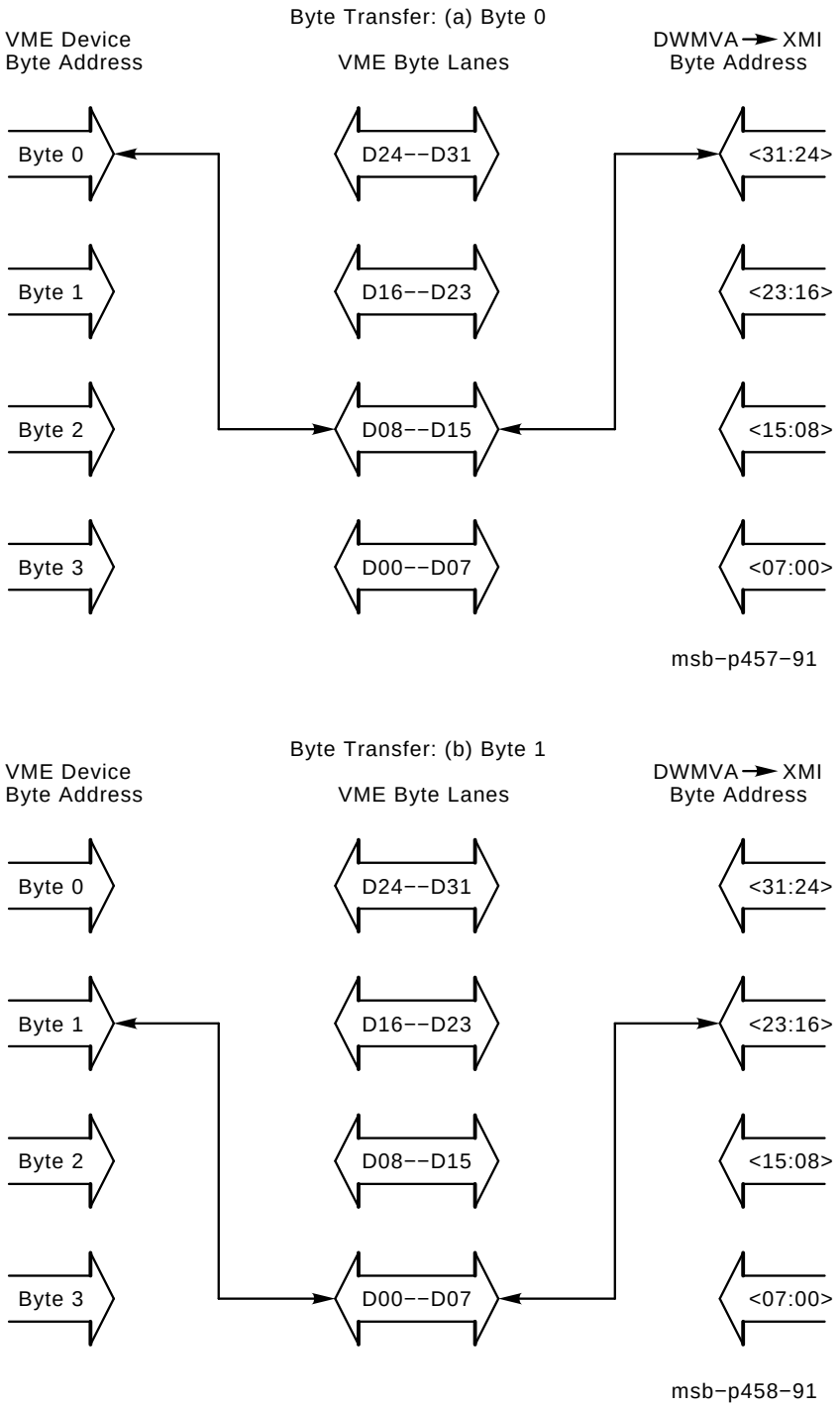
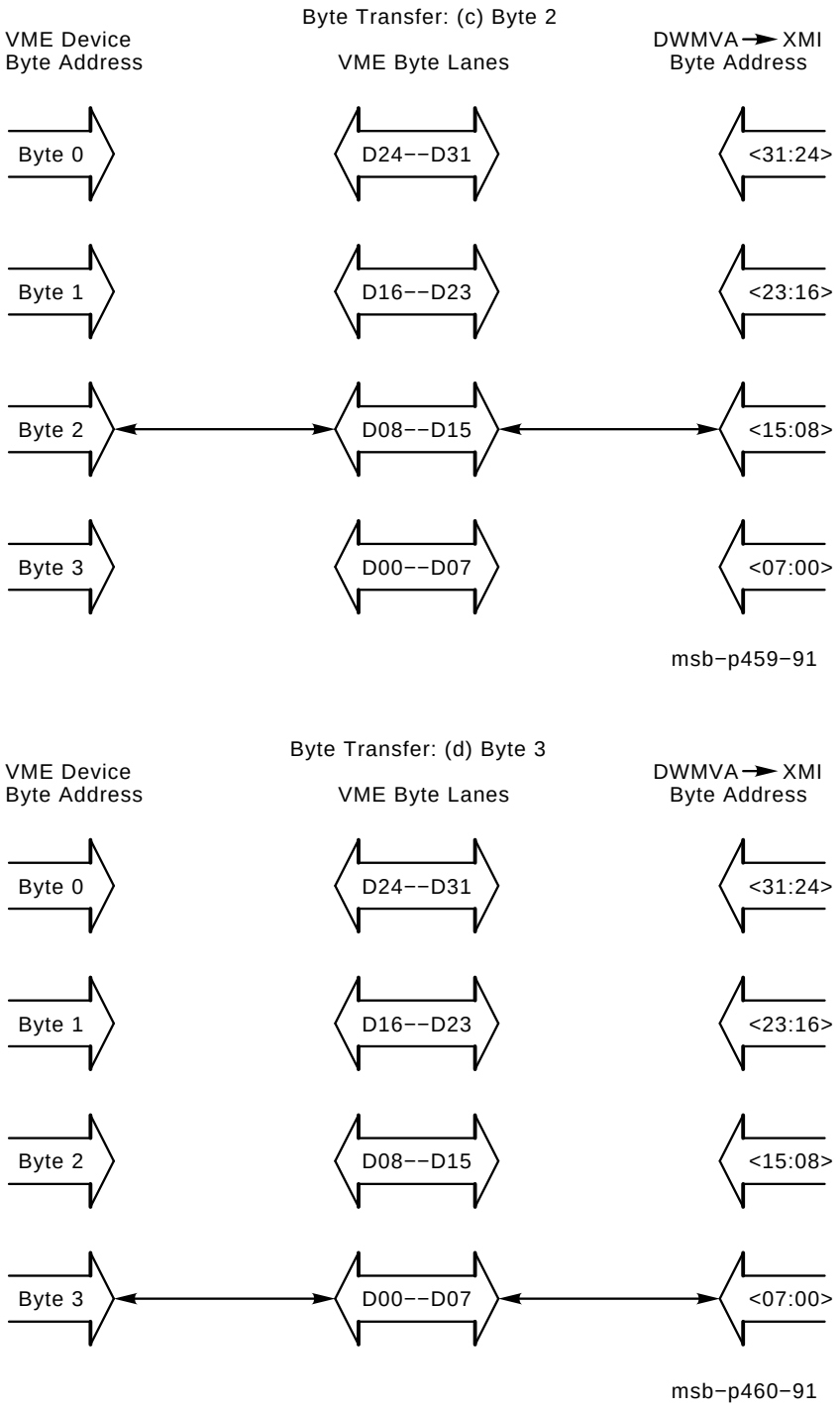


Figure B–8 Cont'd on next page

Figure B–8 (Cont.) Mode 3 (Longword Swap) Transfers



GLOSSARY

arbiter: In VMEbus terminology, a functional module that accepts bus requests from the requester modules and grants control of the data transfer bus to one requester at a time.

arbitration: In VMEbus terminology, the process of assigning control of the data transfer bus to a requester.

backplane: The area in a computer that connects circuit boards. When used in reference to the VMEbus, the backplane is a printed circuit board with 96-pin connectors and signal paths.

block read cycle: A data transfer bus cycle used to transfer a block of 1 to 256 bytes from a slave to a master. The transaction uses a string of 1, 2, or 4-byte data transfers. Once the block transfer is started, the master does not release the data transfer bus until all bytes have been transferred. The block read cycle differs from a string of read cycles in that the master broadcasts only one address and address modifier. The slave then increments this address on each transfer so that the data for the next transfer is retrieved from the next higher location.

block write cycle: A data transfer bus cycle used to transfer a block of 1 to 256 bytes from a master to a slave. The transaction uses a string of 1, 2, or 4-byte data transfers. Once the block transfer is started, the master does not release the data transfer bus until all bytes have been transferred. The block write cycle differs from a string of write cycles in that the master broadcasts only one address and address modifier. The slave then increments this address on each transfer so that the data for the next transfer is stored in the next higher location.

bus timer: A functional module that measures the duration of each data transfer on the data transfer bus and terminates the data transfer bus cycle if a transfer takes too long. Without this module, if the master tries to transfer data to or from a nonexistent slave location, it could wait forever for a slave to respond. The bus timer prevents this by terminating the cycle.

C3200: One of two modules that make up the DWMVA. The C3200 module (DWMVA/B) is on the VMEbus.

commander: A device on the XMI that initiates a transaction, whether read or write. During a write, the commander supplies the data, while in a read transaction, the commander receives the read return data. The device that initiates the transaction will be the commander for the duration of the transaction.

daisy chain: A type of signal line that is used to propagate a signal level from board to board, starting with the first slot and ending with the last slot. There are four bus grant daisy chains and one interrupt acknowledge daisy chain on the VME backplane.

data transfer bus: One of the four buses provided by the VME backplane. The data transfer bus (DTB) allows masters to direct the transfer of binary data between themselves and slaves.

data transfer bus cycle: A sequence of level transitions on the signal lines of the data transfer bus that result in the transfer of an address and data between a master and a slave.

DTB: See *data transfer bus*.

destination: The receiver of information during a transfer on either the XMI or VMEbus.

DWMVA: An I/O option consisting of a T2018 (DWMVA/A) module, a C3200 (DWMVA/B) module and a set of three cable assemblies, which provides a logical path between the XMI bus and the VMEbus through the IBUS.

DWMVA/A: See *T2018*.

DWMVA/B: See *C3200*.

hexword: 256 bits of data; XMI defined.

IACK: Interrupt Acknowledge.

IACK daisy-chain driver: A functional module that activates the interrupt acknowledge daisy chain whenever an interrupt handler acknowledges an interrupt request. The daisy chain ensures that only one interrupter will respond with status/ID when more than one has generated an interrupt request on the same level.

IBUS: Bus that connects the T2018 module to the C3200 module.

IDENT: XMI transaction generated by the XMI commander in response to the interrupt request on the XMI.

interrupt acknowledge cycle: A data transfer bus cycle, initiated by an interrupt handler, that reads a status/ID from an interrupter. An interrupt handler generates this cycle when it detects an interrupt request from an interrupter and it has control of the data transfer bus.

interrupter: A functional module that generates an interrupt request on the interrupt bus and then provides status/ID information when the interrupt handler requests it.

interrupt handler: A functional module that detects interrupt requests generated by interrupters and responds to those requests by asking for status/ID information.

INTR: XMI Interrupt transaction for device interrupts.

IVINTR: Implied Vector Interrupt transaction. A single-cycle interrupt command on the XMI bus.

master: A VME specification term for a device that performs the same role as a commander on the XMI.

octaword: 128 bits of data; XMI defined.

priority interrupt bus: One of the four buses provided by the VME backplane. The priority interrupt bus allows interrupter modules to send interrupt requests to interrupt handler modules and interrupt handler modules to acknowledge these interrupt requests.

quadword: 64 bits of data; XMI defined.

read cycle: A data transfer bus cycle used to transfer 1, 2, 3, or 4 bytes from a slave to a master. The cycle begins when the master broadcasts an address and an address modifier. Each slave captures this address and address modifier and checks to see if it is to respond to the cycle. If the slave is to respond, then it retrieves the data from its internal storage, places it on the data bus, and acknowledges the transfer. Then the master terminates the cycle.

Read Modify Write cycle: In VMEbus terminology, a data transfer bus cycle that is used both to read from and write to a slave's byte location(s) without permitting any other master to access the same location during that cycle. This cycle is most useful in multiprocessing systems where certain memory locations are used to control access to certain system resources. (For example, semaphore locations.)

requester: A functional module that resides on the same board as a master or interrupt handler and requests use of the data transfer bus whenever its master or interrupt handler needs it.

responder: A device on the XMI that is targeted by the commander. The device acts as a responder for the duration of the transaction. See *slave*.

slave: A VME term for a device that performs the same role as a responder on the XMI.

source: A source is the provider of information during a transfer on either the XMI or VME.

T2018: One of two modules that make up the DWMVA. The T2018 module (DWMVA/A) is on the XMI bus.

transaction: An operation consisting of single or multiple data transfers. CPU reads, CPU writes, DMA reads, and DMA writes are transaction types. Each transaction begins with a command and address transfer. During writes, the command/address transfer is followed by data transfers from the commander (in VME terminolgy, master) initiating the transaction. Data transfers are performed by the responder (in VME terminolgy, slave) during read transactions.

transfer: Command/address or data that is sent over a bus from the source to the destination. A change in the command/address or data defines the end of a particular transfer and the beginning of another. Transactions can consist of multiple transfers.

utility bus: One of four buses provided by the VME backplane. This bus includes signals that provide periodic timing and coordinate the power-up and power-down of the system.

VMEbus: An industry-standard bus defined by IEEE 1014. It is an asynchronous interlocked bus with separate data and address lines.

VSC: VME system controller. Performs functions defined by IEEE 1014. It is a protocol for the VMEbus that describes the clock drivers, power monitor, bus arbiter, IACK driver, and bus timer.

write cycle: A data transfer bus cycle used to transfer 1, 2, 3, or 4 bytes from a master to a slave. The cycle begins when the master broadcasts an address and address modifier and places data on the data transfer bus. Each slave captures this address and modifier and checks to see if it is to respond to the cycle. If so, the slave stores the data and then acknowledges the transfer. The master then terminates the cycle.

XMI: A synchronous, pended bus used in VAX 6000 systems. It has multiplexed data and address lines. The XMI is the interconnect between CPU modules, memory, and I/O adapters.

XMI Corner: An area on an XMI module that connects to the backplane and provides an electrically identical interface for every XMI node.

Index

A

B

C

D

E

F

L

M

N

O

P

R

S

T

U

V

W

X