XMI BUS OVERVIEW

Document Title: XMI BUS OVERVIEW HANDBOOK

Order Number: EK-XMIOV-HB-002

This handbook is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The handbook can be ordered separately or as part of the set.

The *XMI Adapters Handbook Documentation Set* is a dynamic document which will be periodically updated as new XMI adapters are announced. The first release of the set includes the following handbooks:

Order Number	Title
EK-XMIOV-HB	XMI Bus Overview Handbook
EK-CIXCD-HB	CIXCD Handbook
EK-DEMNA-HB	DEC LANcontroller 400 (DEMNA) Handbook
EK-DWMBA-HB	DWMBA Handbook

This handbook and the document set are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

XMI Bus Overview Handbook

Order Number EK-XMIOV-HB-002

This document is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The document can be ordered separately or as part of the set. The *XMI Bus Overview Handbook* and the *XMI Adapters Handbook Documentation Set* are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

Revision/Update Information: Revision 2.0

Digital Equipment Corporation

June, 1991

This document was prepared and published by Educational Services Development and Publishing, Digital Equipment Corporation.

Contents

3 XMI BUS FUNCTIONAL DESCRIPTION

Figures

Tables

About This Manual

Intended Audience

This handbook is part of a series of handbooks which comprise the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). This handbook and the handbook set are for VAX system trained Digital customer service personnel who service XMI-based systems and subsystems. Users of the handbook set should be familiar with the XMI bus architecture (either through the *XMI Bus Concepts* course or through practical experience) and have a minimum of level 1 hardware maintenance training on one or more VAX systems (for example, VAX 6000 or VAX 9000 systems).

Document Scope and Structure

Several I/O adapters have been developed to interface the XMI bus to devices which employ different bus structures and protocols. These adapters are available as stand-alone options and may be installed on a variety of systems or subsystems.

The *XMI Adapters Handbook Documentation Set* provides a single, quick reference source to the type of information most frequently required to service XMI adapters. This handbook contains general information about the XMI bus and applies to all XMI adapters covered in the document set.

This handbook is divided into five chapters:

Chapter 1, XMI BUS OVERVIEW outlines the XMI bus architecture and describes XMI bus implementations, terms, and specifications.

Chapter 2, XMI BUS PHYSICAL DESCRIPTION reviews the XMI bus physical characteristic and general configuration rules.

Chapter 3, XMI BUS FUNCTIONAL DESCRIPTION presents the bus functional characteristics including bus addressing, function codes, and data transfer transactions.

Chapter 4, WRITE-BACK CACHE SUPPORT AND THE XMI+ PROTOCOL overviews the implementation of write-back cache in XMI-based systems.

Chapter 5, DIAGNOSING XMI BUS RELATED ERRORS overviews the XMI bus error reporting and error handling mechanisms.

Conventions

addresses	All addresses are given in hexadecimal (hex).
bits	All bit numbers are given in decimal with the bit(s) enclosed in angle brackets; for example <31>.
	Multiple individual bits or bit fields are separated by commas with bit fields indicated by two numbers separated by a colon. For example <31:24,20,18,14:10> indicates bits 31 through 24 (inclusive), bit 20, bit 18, and bits 14 through 10 (inclusive).
$\boxed{\text{CTRL/x}}$	Specifies to press and hold the $\boxed{\text{Ctrl}}$ key while pressing the $\boxed{\text{x}}$ key; for example, $\boxed{\text{CTRL/C}}$.
[item] . . .	Indicates the item is optional. The horizontal ellipsis indicates that additional optional items can be entered.
. . .	Vertical ellipsis in examples, tables, or figures, indicates that not all information is shown.

Related Documents

The *XMI Adapters Handbook Documentation Set* was written in conjunction with XMI adapter specific user's guides and technical descriptions. Each adapter specific handbook in the set contains a list of documents which the user can reference for more detailed information.

CHAPTER 1

1 XMI BUS OVERVIEW

1.1 INTRODUCTION

The XMI bus is a limited length, synchronous, high-speed bus with centralized arbitration. The XMI is a pended bus (XMI nodes do not hold the bus waiting for a response) with multiplexed address and data lines.

Multiple transactions can be in progress at any time on the XMI, with arbitration and data transfer transactions occurring simultaneously.

1.2 IMPLEMENTATIONS

The XMI bus can be implemented as the primary system bus (for example, in most VAX 6000 systems) or as an I/O bus (for example, in VAX 9000 systems).

When used as the primary system bus, the XMI can support multiple processors, memory subsystems, and I/O adapters, in a variety of configurations, to a maximum of 14 nodes.

When used as an I/O bus, the XMI is interfaced to the system by way of an adapter. For example, in VAX 9000 systems, the XMI bus interfaces to the system by way of the XJA adapter, JXDI bus, and an I/O control unit.

Figures 1–1, 1–2, and 1–3 show typical XMI bus implementations.

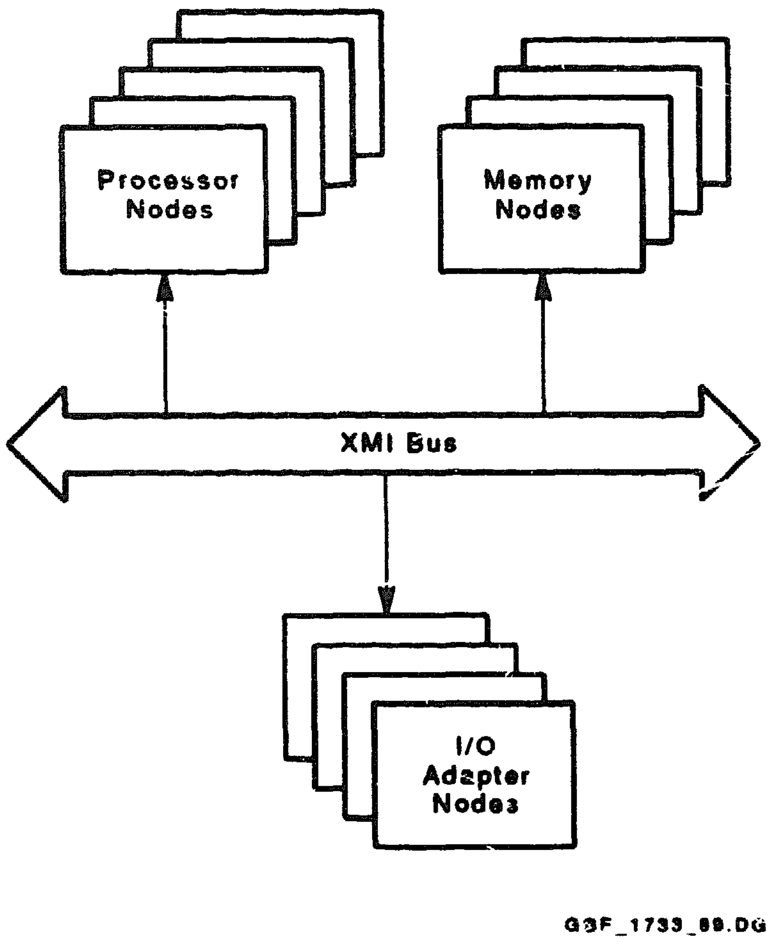


Figure 1–1 XMI Bus as the Primary System Bus

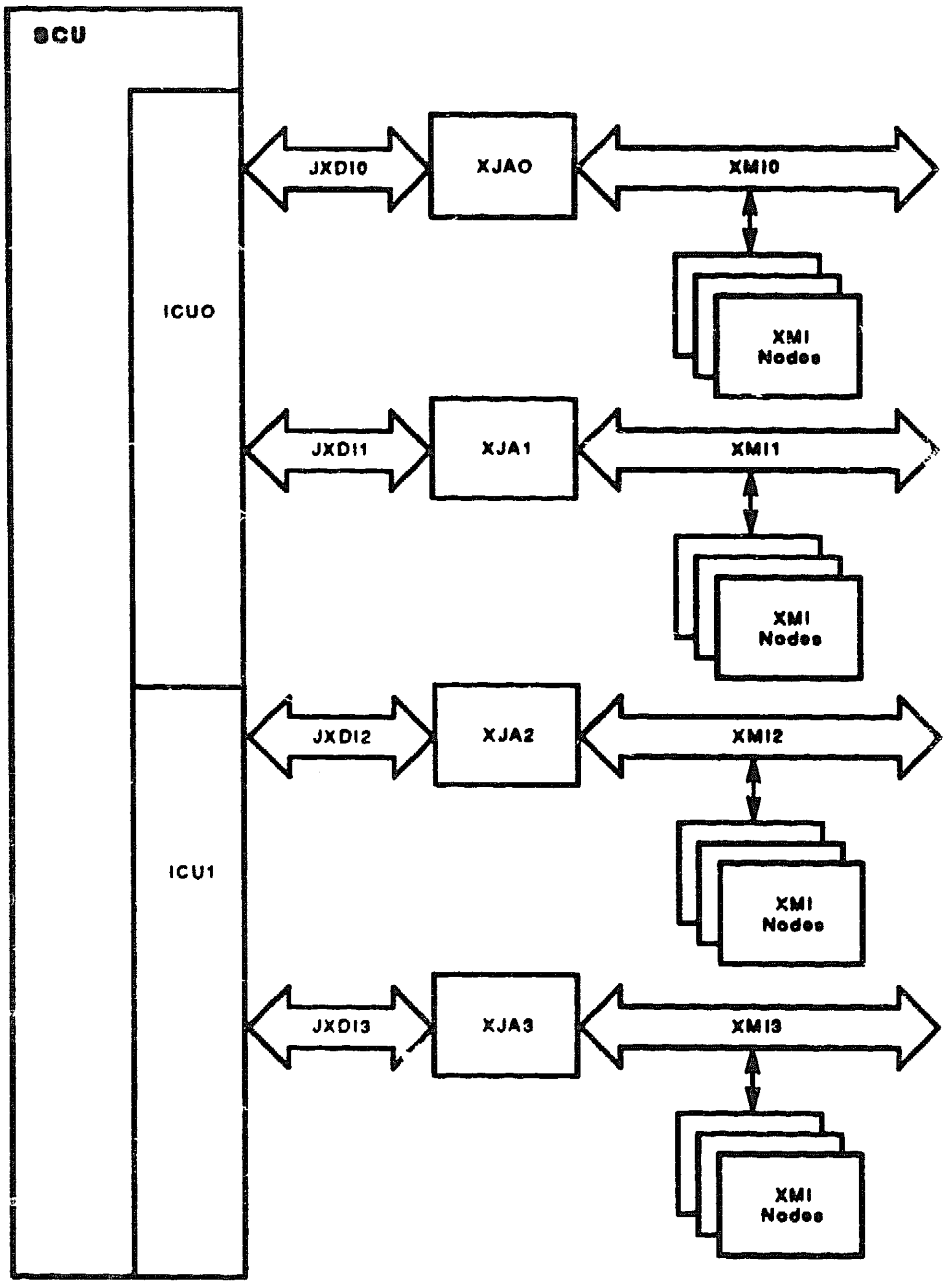


Figure 1-2 XMI Bus as an I/O Bus

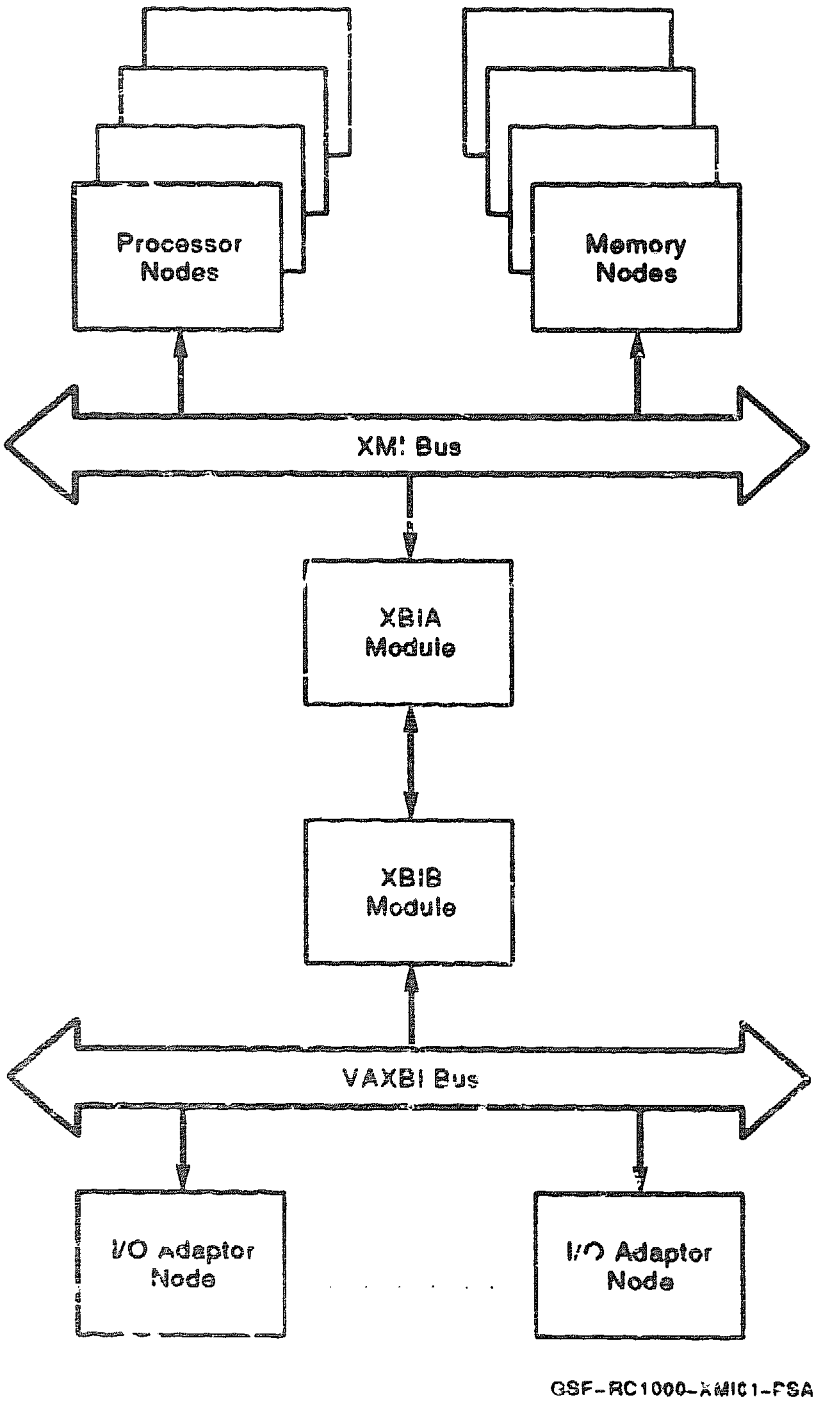


Figure 1–3 XMI-Based System with VAXBI-Based I/O

1.3 BUS PROTOCOL

XMI bus operations are controlled by one of two protocols:

Table 1-1 XMI Bus Protocols

Name	Description
XMI	Implements the basic set of XMI bus transactions:
	Type / **Transactions**
	Data Transfer: Read (READ); Interlocked read (IREAD); Write masked (WMASK); Unlock write masked (UWMASK)
	Interrupt: Interrupt request (INTR); Interrupt acknowledge (IDENT); Implied vector interrupt (IVINTR)
XMI+	Superset of the XMI protocol. Includes three additional data transfer transactions to support nodes which implement write-back cache: Ownership read (OREAD); Disown write masked (DWMASK); Tag bad data (TBDATA)

Type	Transactions
Data Transfer	Read (READ)
	Interlocked read (IREAD)
	Write masked (WMASK)
	Unlock write masked (UWMASK)
Interrupt	Interrupt request (INTR)
	Interrupt acknowledge (IDENT)
	Implied vector interrupt (IVINTR)

The bus protocol used on a given system depends on the type of nodes present in the XMI backplane. Systems which implement write-back cache use the XMI+ protocol.

1.4 TERMINOLOGY

Table 1-2 XMI Architecture Terms

Term	Definition
Node	Hardware device that connects to the XMI backplane. A node can consist of one or more modules, but only one module may have an XMI corner.
Transfer	Smallest unit of information exchange that occurs on the XMI bus. For example, the command and data cycles of a read transaction and the command and data cycles of a write transaction are transfers.
Transaction	One or more transfer cycles which comprise the XMI task being performed. For example, a read transaction consists of a command transfer followed some time later by a return data transfer.
Commander	Node that initiated the current transaction. In a write transaction, the commander node is the source of the data to be transferred. In a read transaction, the commander is the node that requested the data. A node which initiates a transaction is considered to be the commander for the duration of the transaction. For example, on a read transaction, the commander initiates the transaction and the responder returns the data. During the return data transfer, the requesting node is still considered the commander.
Responder	Node which is the target of a transaction request.
Transmitter	Node that is the source of the data on the bus. For example, on a read transaction, the commander is the transmitter during the command transfer and the responder is the transmitter on the return data transfer.
Receiver	Node that is the target of the data on the bus.
Naturally aligned	Data quantity whose address is an offset, from the beginning of memory, of an integral number of data elements of the same size. The low order address bits of naturally aligned data items are always zero. All XMI bus read and write transfers occur on naturally aligned blocks of data.

Table 1-2 (Cont.) XMI Architecture Terms

Term	Definition
Wraparound read	Octaword or hexword read operation in which read data is returned so that the originally requested quadword is returned first, independent of alignment. The remaining data in the naturally aligned block of data which contains the addressed quadword is returned in subsequent transfers. XMI bus protocol requires that all octaword and hexword reads, both normal and interlocked, be wraparound reads.
Ownership	On systems which implement the XMI+ (write-back cache) protocol, each 32 byte block of data in memory has two associated status bits. These bits are used by memory to implement the XMI+ protocol. A node gains ownership of a block by performing an ownership read transaction (OREAD). After gaining ownership, the node is free to write to its local cache without transmitting a write to main memory. The node that owns a block must monitor the bus for attempts to read or write the block. If a WMASK reference is detected, the node must return the data to main memory using a disown write mask transaction (DWMASK).

1.5 BUS INTEGRITY FEATURES

The XMI bus contains several features that enhance the integrity and reliability of the bus.

- Parity protection for all bus information transfer lines
- ECC protection on all bus confirmation signals
- Bus protocol permits detection and correction of single-bit errors
- Defined timeout conditions to detect and diagnose faults

1.6 SPECIFICATIONS

Table 1-3 XMI Bus Specifications

Physical	
Backplane	14-slot card cage
Nodes[1]	Maximum of 14
Node ID	Hardwired to physical slot
Bus length	Fixed, nonexpandable
Technology	CMOS
Performance	
Bus type	Synchronous
Bus cycle	64 ns
Address/data lines[2]	64
Address bits	40
Address space	1 terabyte (2^{40} bytes)
Data transfer size	64 bits/cycle
Data transfer type[3]	Pended
Bandwidth	See Table 1-4
Arbitration	
Type	Centralized
Algorithm	Modified round-robin
Cycles[4]	Concurrent with data transfers
Queues	Two — commander, responder

[1]The XMI architecture allows for up to 16 nodes, but current physical constraints limit the bus to 14 nodes.

[2]Multiplexed address/data lines.

[3]Read- and interrupt-type transactions are pended (node does not hold bus while waiting for a response).

[4]No dedicated arbitration cycles.

Table 1-4 XMI Bandwidth

Data Size	Read[1]	Write[1]
Longword[2]	31.25	31.25
Quadword	62.5	62.5
Octaword	83.3	83.3
Hexword	100.0	100.0

[1]Bandwidth values given in Mbytes/s.

[2]The XMI architecture allows data transfers of all data sizes to both memory and I/O space. However, some implementations may not support transfers greater than a longword to I/O space.

CHAPTER 2

2
XMI BUS PHYSICAL DESCRIPTION

2.1 XMI CARD CAGE

The XMI card cage consists of a 14-slot backplane, connectors, card guides, and structural members. Each slot has a ZIF connector which is opened and closed by a cam actuator mechanism. This mechanism is sometimes called the card cage handle.

2.1.1 Backplane

There are two basic types of XMI backplanes:

- XMI-1, Figure 2–1
- XMI-2, Figure 2–2

The main differences between the two backplanes are that the XMI-2:

- Supplies more +3.3 Vdc current than does the XMI-1

  The increased current is provided to support more nodes with CMOS III logic. The additional current is supplied to each node by way of pins which are grounded on the XMI-1 backplane.

- Supplies battery backup power (if battery backup is present) for all backplane voltages

  The XMI-1 supplies battery backup only at +5.0 Vbb (see Figure 2–1). The +5.0 Vbb is separate from the +5.0 Vdc to allow backup power to memory and clocks without the need for supporting all XMI node logic.

  On the XMI-2, if battery backup is present, it is available at all backplane voltages for all nodes.

  Implementation of battery backup on either backplane is system specific.

- **Includes two additional signals: XMI BP ID L and XMI PS EN H**

  These signals are both grounded on the XMI-1 backplane. See Chapter 3 for signal descriptions.

Refer to system specific documentation for the type of backplane installed in a given system.

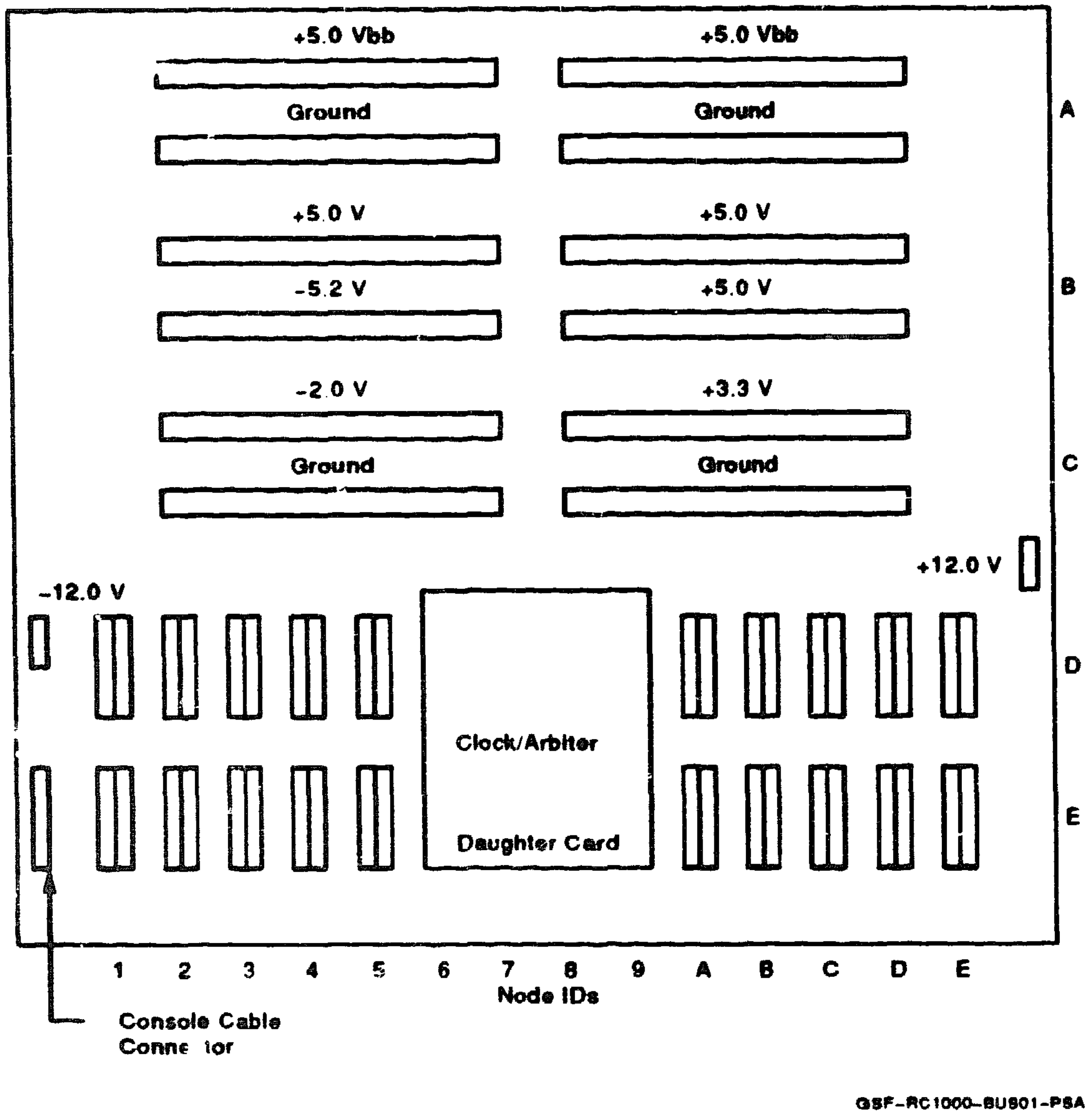


Figure 2–1 XMI-1 Backplane (+5.0 Vbb Power)

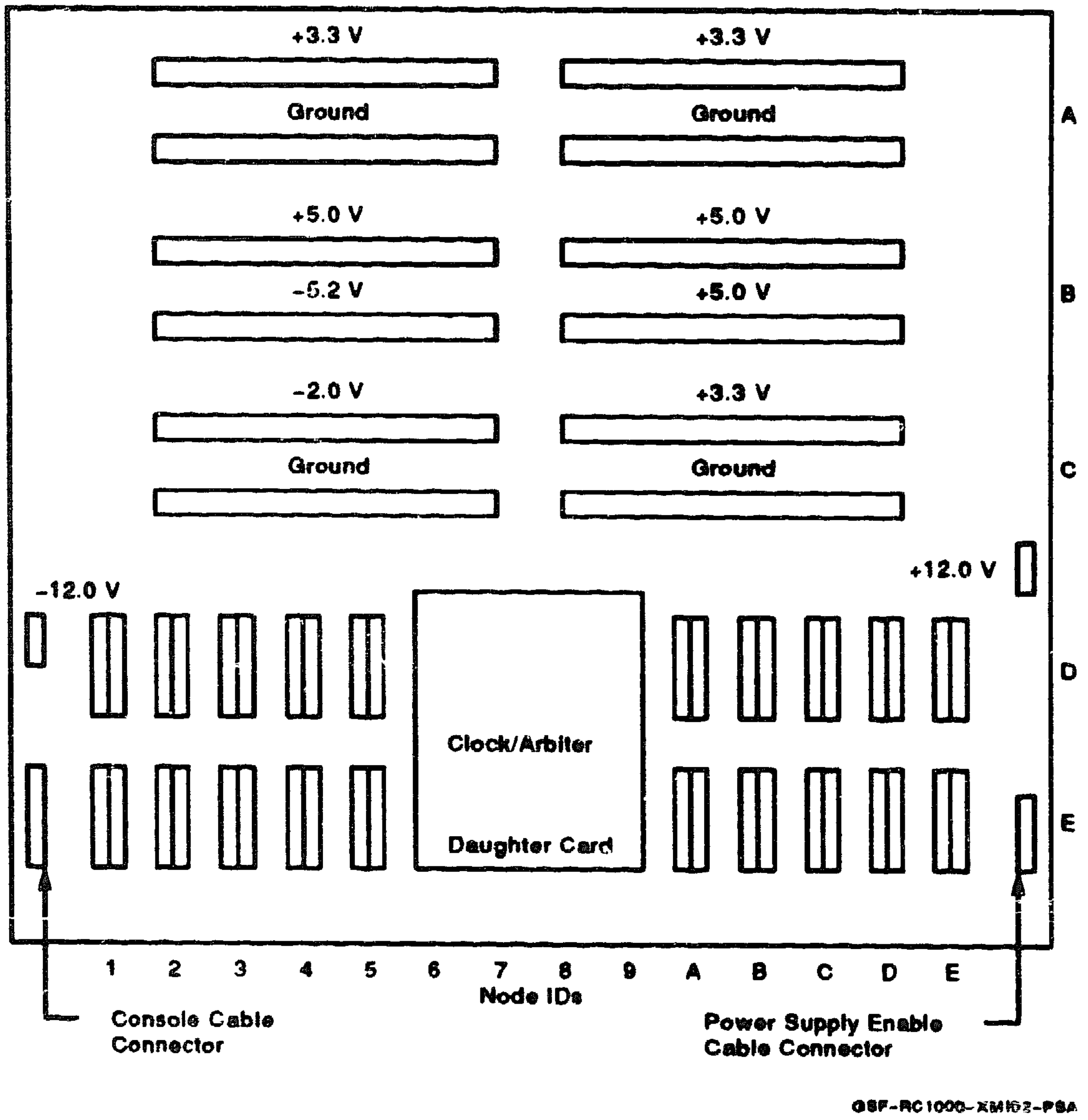


Figure 2–2 XMI-2 Backplane (+3.3 Vdc Power)

2.1.2 Node ID Numbers

XMI node ID numbers are hardwired on the XMI backplane. The node numbers correspond to the backplane slot numbers as shown in Figures 2–1 and 2–2.

2.1.3 Clock/Arbiter Card

In some implementations, the XMI system clock and bus arbiter are located on a daughter card which is attached to the XMI backplane. For example, Figures 2–1 and 2–2 show the location of the daughter card (DCARD) on VAX 6000 systems.

In other implementations, the clock/arbiter module is mounted in the card cage. Figure 2–3 shows the location of the clock/arbiter (CCARD) on VAX 9000 systems.

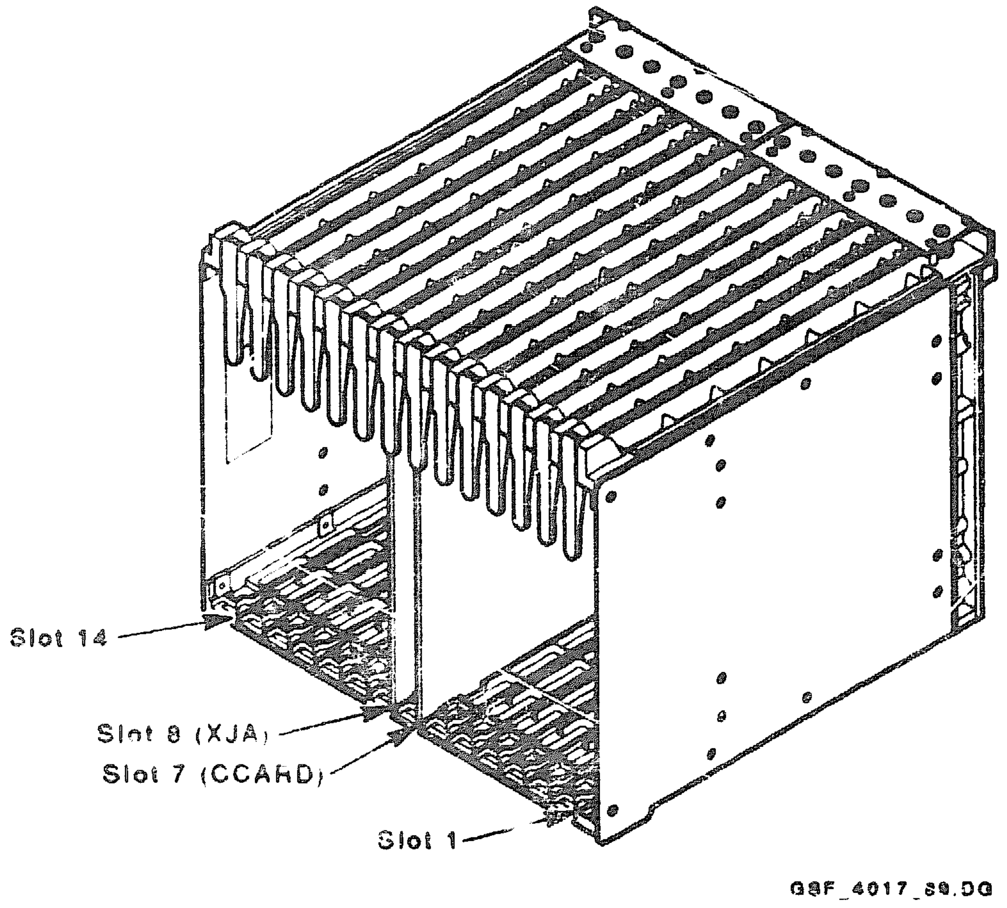


Figure 2–3 XMI Card Cage, VAX 9000 System

2.1.4 I/O Connector Pins

Segments D and E of each XMI backplane slot have connector pins for attaching I/O cables. Figure 2-4 shows the pin numbering layout. Refer to Section 2.3 for I/O node placement restrictions.

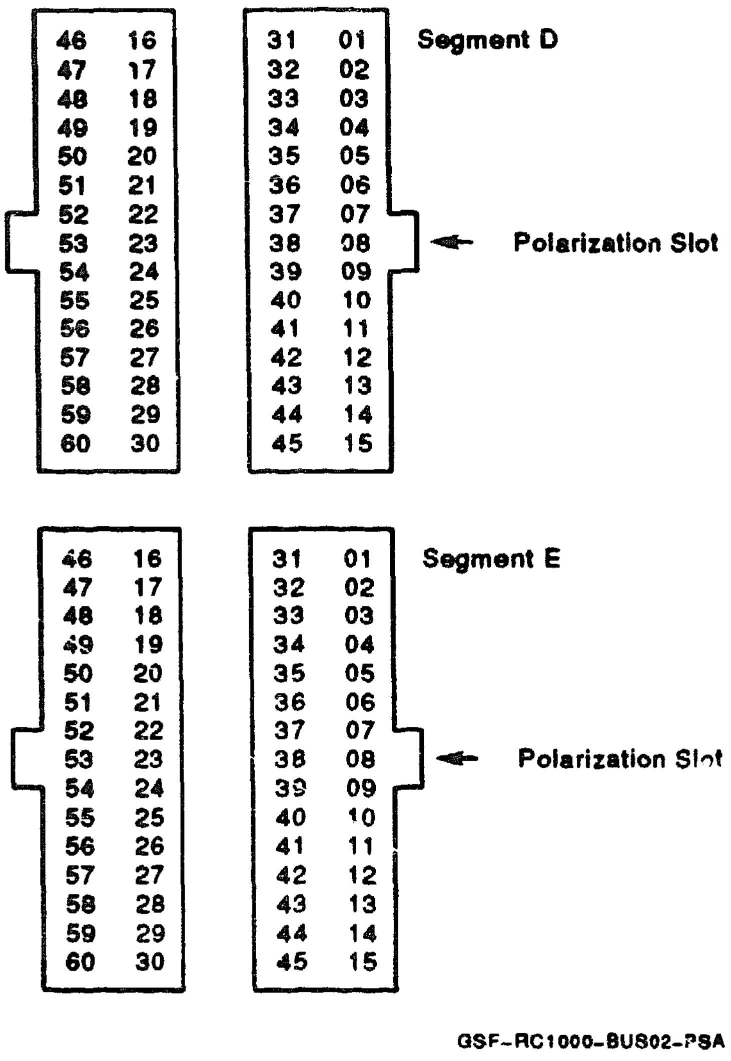


Figure 2-4 I/O Connector Pins

2.2 XMI NODE

An XMI node consists of one or more modules mounted in the card cage that interface to the bus via an XMI corner. If the node consists of more than one module, only one may have an XMI corner.

2.2.1 XMI Corner

The XMI corner (Figure 2-5) contains custom logic that:

- Assures a standard electrical interface to the bus
- Buffers drive signals to the bus
- Buffers receive signals from the bus

The XMI corner occupies an area of approximately 4.45 cm (1.75 in) × 12.70 cm (5.00 in) on the module and consists of eight Digital custom CMOS chips:

- Seven XLATCH chips
- One XCLOCK chip

Note that the XMI corner does not perform any node control functions. All node control functions are performed by the node-specific logic.

2.2.2 Self-Test LED

XMI nodes are required to have one yellow self-test pass (STP) LED to indicate the result of the node's self-test. This LED lights when self-test passes. The LED is located on the front edge of the module (the edge opposite the connectors) and is the only yellow LED on the module.

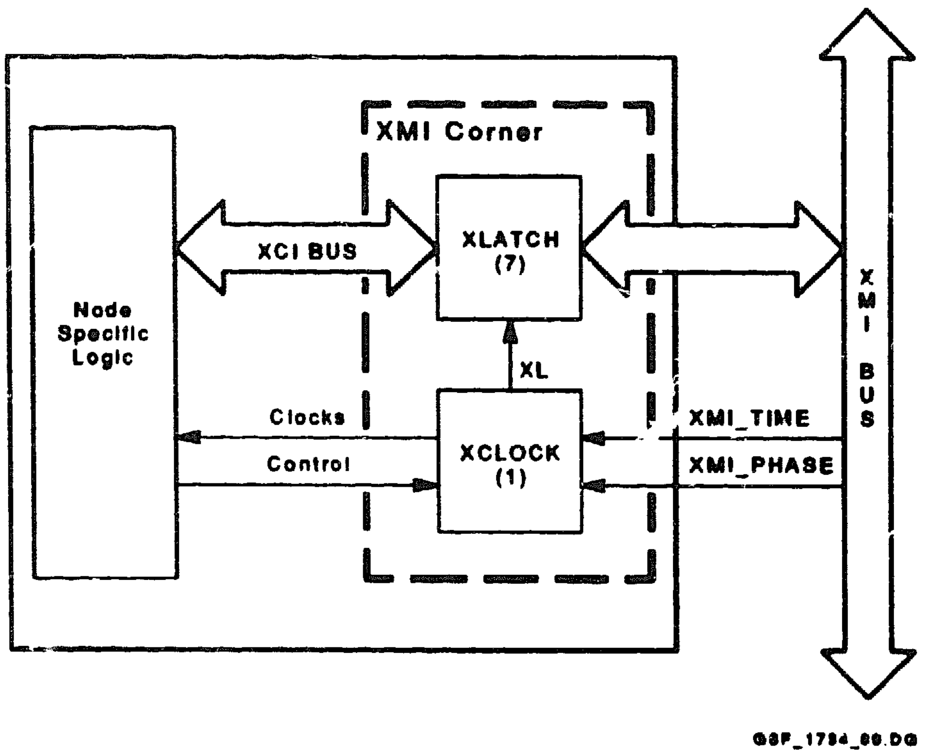


Figure 2-5 XMI Corner

2.3 CONFIGURATION RULES

The XMI backplane design places certain restrictions on the placement of modules in the XMI card cage.

Some implementations (for example, VAX 6000 systems) prohibit the placement of I/O nodes in slots 6 to 9 (or 5 to A, depending on the backplane type; see below) as the I/O connector pins for these slots are covered by the clock/arbiter card. In other implementations (for example, VAX 9000 systems) the clock/arbiter plugs into the card cage as any other node, allowing greater flexibility in the placement of I/O nodes.

The module placement restrictions for VAX 6000 and VAX 9000 systems are listed below. Refer to system specific documentation for details.

VAX 6000 Systems

- Slot 1 or E must contain a (non-memory) module
- No I/O adapter modules in slots 5 to A (XMI-1 backplane), slots 6 to 9 (XMI-2 backplane)
- CPU modules are typically installed beginning with slot 1
- Memory modules are placed in slots A to 5, then in slots B and C
- DWMBA adapters are installed in the left side of the card cage, beginning with slot E

VAX 9000 Systems

- Slot 1 or E must contain a (non-memory) module
- CCARD module is installed in slot 7
- XJA module is installed in slot 8
- First I/O adapter is installed in slot 1 or E
- Additional I/O adapters may be located in any other slot

2.4 MODULE PLACEMENT PRECAUTIONS

Some XMI modules (for example, the T2017 XRV) have components with heat sinks mounted on side 2 of the module. These components may cause mechanical or thermal interference problems with the module placed in the slot directly to their left in the card cage. Observe caution when installing or replacing a module adjacent to one with side 2 components.

CHAPTER 3

3 XMI BUS FUNCTIONAL DESCRIPTION

3.1 ADDRESSING

The XMI bus supports one terabyte (2^{40} bytes) of address space which is divided into physical memory space and I/O space (Figure 3–1).

On a command/address cycle, XMI D<63:00>, which are multiplexed address/data lines, carry the forty bits of addressing, A<39:00>, as follows:

XMI D<63:00>	Address bits
D<29>	A<39>
D<57:48>	A<38:29>
D<28:00>	A<28::00>

The most significant bit of the address, A<39>, selects between memory and I/O space:

A<39> = 0 memory space
A<39> = 1 I/O space

NOTE
Some implementations of the XMI only support 30-bit addressing. These systems use internal address bit A<29>, which is output to the bus as XMI D<29>, to distinguish memory space from I/O space.

I/O space is further divided into private space, nodespace, and 15 I/O adapter address space regions. Figure 3–2 shows the I/O space divisions and Tables 3–1 and 3–2 describe the regions.

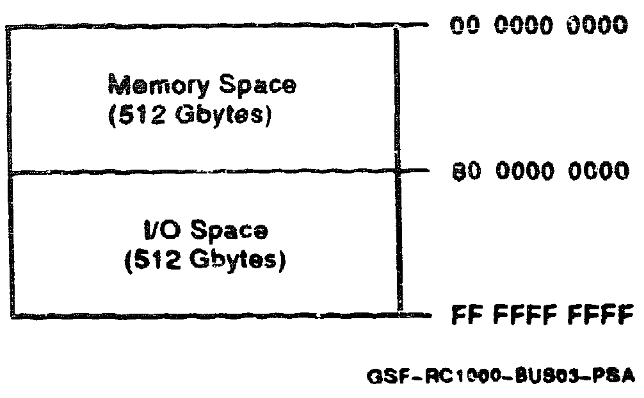
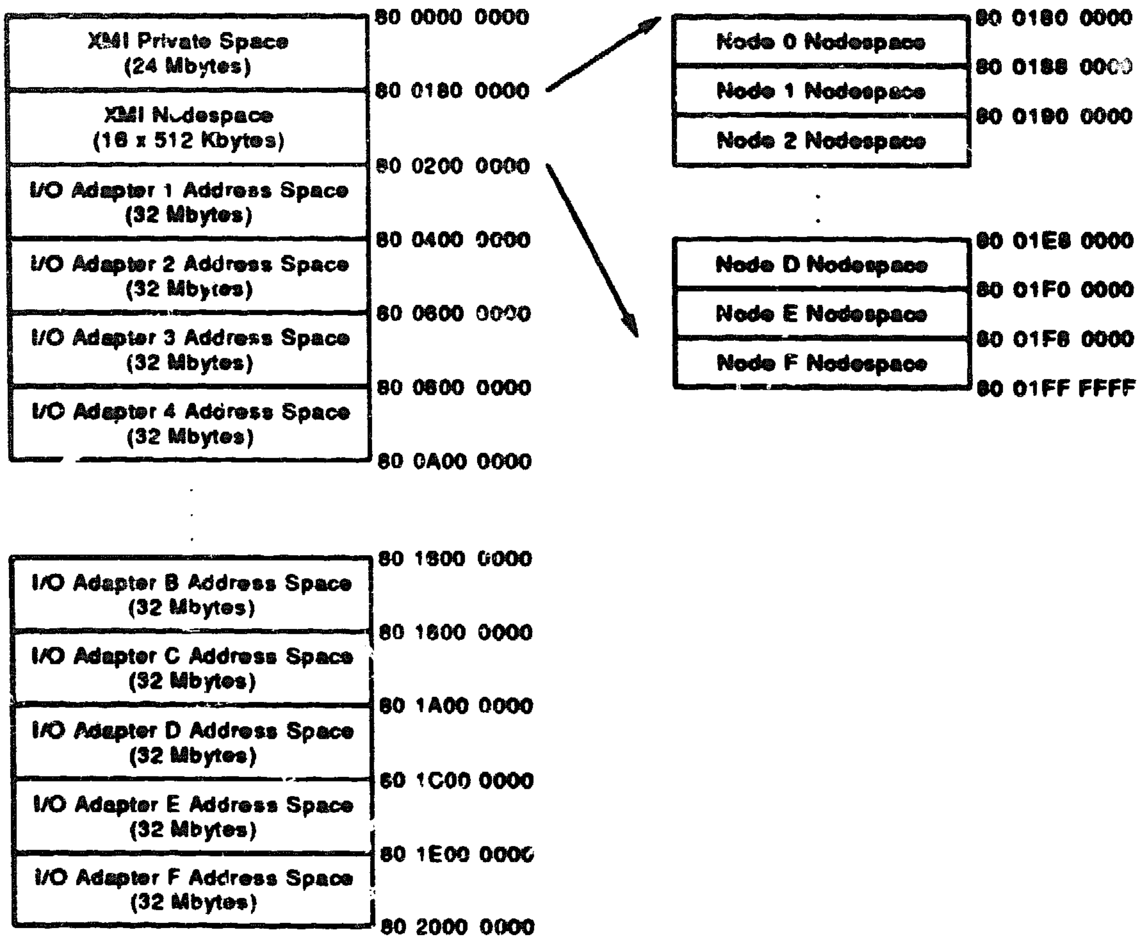


Figure 3-1 XMI Address Space

Table 3-1 XMI I/O Space Regions

Region	Description
Private space	24-Mbyte region reserved for operations local to the nodes. References to private space are serviced by resources local to a node, such as local device control and status registers (CSRs) and boot ROM. These references are not broadcast on the XMI bus.
Nodespace	16, 512-Kbyte regions for node control and status registers. Table 3-2 shows the address ranges.
I/O adapter address space	15, 32-Mbyte regions used for accessing XMI I/O adapters. Table 3-2 shows the address ranges. Note that node 0 does not have an I/O adapter address region.


XMI Private Space (24 Mbytes)
XMI Nodespace (16 x 512 Kbytes)
I/O Adapter 1 Address Space (32 Mbytes)
I/O Adapter 2 Address Space (32 Mbytes)
I/O Adapter 3 Address Space (32 Mbytes)
I/O Adapter 4 Address Space (32 Mbytes)
I/O Adapter B Address Space (32 Mbytes)
I/O Adapter C Address Space (32 Mbytes)
I/O Adapter D Address Space (32 Mbytes)
I/O Adapter E Address Space (32 Mbytes)
I/O Adapter F Address Space (32 Mbytes)
80 0000 0000
80 0180 0000
80 0200 0000
80 0400 0000
80 0600 0000
80 0800 0000
80 0A00 0000
80 1600 0000
80 1800 0000
80 1A00 0000
80 1C00 0000
80 1E00 0000
80 2000 0000
Node 0 Nodespace
Node 1 Nodespace
Node 2 Nodespace
Node D Nodespace
Node E Nodespace
Node F Nodespace
80 0180 0000
80 0188 0000
80 0190 0000
80 01E8 0000
80 01F0 0000
80 01F8 0000
80 01FF FFFF

GSF-RC1000-BU064-PSA

Figure 3-2 I/O Space

Table 3–2 XMI Nodespace and I/O Space Allocations

	Nodespace		I/O Adapter Space	
Node	Begin	End	Begin	End
0[1,2]	80 0180 0000	80 0187 FFFF	See footnote 2	
1	80 0188 0000	80 018F FFFF	80 0200 0000	80 03FF FFFF
2	80 0190 0000	80 0197 FFFF	80 0400 0000	80 05FF FFFF
3	80 0198 0000	80 019F FFFF	80 0600 0000	80 07FF FFFF
4	80 01A0 0000	80 01A7 FFFF	80 0800 0000	80 09FF FFFF
5	80 01A8 0000	80 01AF FFFF	80 0A00 0000	80 0BFF FFFF
6	80 01B0 0000	80 01B7 FFFF	80 0C00 0000	80 0DFF FFFF
7	80 01B8 0000	80 01BF FFFF	80 0E00 0000	80 0FFF FFFF
8	80 01C0 0000	80 01C7 FFFF	80 1000 0000	80 11FF FFFF
9	80 01C8 0000	80 01CF FFFF	80 1200 0000	80 13FF FFFF
A	80 01D0 0000	80 01D7 FFFF	80 1400 0000	80 15FF FFFF
B	80 01D8 0000	80 01DF FFFF	80 1600 0000	80 17FF FFFF
C	80 01E0 0000	80 01E7 FFFF	80 1800 0000	80 19FF FFFF
D	80 01E8 0000	80 01EF FFFF	80 1A00 0000	80 1BFF FFFF
E	80 01F0 0000	80 01F7 FFFF	80 1C00 0000	80 1DFF FFFF
F[3]	80 01F8 0000	80 01FF FFFF	80 1E00 0000	80 1FFF FFFF

[1]Node 0 nodespace is reserved for future expansion.

[2]Node 0 does not have I/O adapter space; addresses in this range comprise XMI private space and XMI nodespace.

[3]Reserved for future expansion.

3.2 BUS SIGNALS

Table 3–3 XMI Bus Signals

Signal	Description
Arbitration	
XMI CMD REQn	Commander bus request lines (n = node number)
XMI RES REQn	Responder bus request lines (n = node number)
XMI GRANTn	Bus grant lines (n = node number)
XMI HOLD	Bus hold (multicycle transfers)
XMI SUP	Suppress initiation of new XMI transactions
XMI LOCKOUT[1]	Prevent resource starvation (forces sequential access to shared resources)
Information Transfer	
XMI D <63:00>	Data cycles: read or write data Command cycles: command, address, mask
XMI F <03:00>	Bus function (see Table 3–4)
XMI ID <05:00>	Commander ID. Field is of the form AAAAnn where AAAA is the commander's node ID and nn is the command ID (each node can have up to 4 outstanding transactions at the same time).
XMI P <02:00>	Parity of XMI D, XMI F, and XMI ID lines
Response	
XMI CNF <02:00>	Data transfer status confirmation (from receiver)

[1]Commander nodes assert XMI LOCKOUT when repeated attempts to perform hardware locks are denied, or repeated attempts to perform IDENTs or I/O space references are NOACKed. The assertion of LOCKOUT ensures fair access to resources by preventing nodes which have completed a lock, IDENT, or I/O reference from initiating another request while LOCKOUT is asserted. XMI LOCKOUT need only be generated and monitored by commander nodes that perform interlock reads, ownership reads, IDENTs, or I/O space references.

Table 3–3 (Cont.) XMI Bus Signals

Signal	Description
Control	
XMI AC LO	Low ac line voltage
XMI DC LO	Impending loss of dc power
XMI BAD	Node failure (asserted until all nodes pass self-test)
XMI DEF [A,B]	Defaults bus tristate lines during XMI idle cycles
XMI ERR DEF	Bus configuration defaulting check
XMI RESET	Initialize system to power-up state
XMI TRIGGER[2]	Node detected significant event (specific to node)
XMI TIME n	XMI clock reference (n = 1 to 15)
XMI PHASE n	XMI clock phase reference (n = 1 to 15)
XMI UPDATE EN	Modification control for EEPROM or other writeable, non-volatile storage devices. UPDATE EN must be observed by all nodes that implement on-board, writeable, non-volatile storage.
Console and Front Panel[3]	
XMI CON XMIT	Transmit data to console
XMI CON RECV	Receive data from console
XMI CON SECURE	Console secure (if XMI is system bus, disables CTRL/P detection)
XMI BOOT EN	Auto-boot control (if XMI is system bus)
XMI RUN	Front panel RUN LED control
XMI TOY BBU PWR	Time of year clock BBU power
XMI TOY BBU OK	TOY clock BBU status
XMI BP ID	Identifies the backplane type. Asserted on the XMI-1 backplane and deasserted on the XMI-2.

[2]This signal may be labeled as XMI FAULT on some implementations of the XMI bus.

[3]Used only by CPU nodes

Table 3–3 (Cont.) XMI Bus Signals

Signal	Description
Console and Front Panel[3]	
XMI PS EN	Ensures that the +3.3 Vdc power supply is not enabled if an XMI-1 style module is plugged into the XMI-2 style backplane (some pins used for +3.3 Vdc on the XMI-2 are grounded on XMI-1 style modules).
Miscellaneous	
XMI NODE ID <03:00>	Backplane wired node ID (for example, slot 1 = node 1)
XMI SPARE0	Reserved

[3]Used only by CPU nodes

3.3 BUS FUNCTION CODES

Table 3-4 Bus Function Codes

XMI F<3:0>					
3	**2**	**1**	**0**	**Mnemonic**	**Function**
0	0	0	0	NULL	Null cycle
0	0	0	1	CMD	Command cycle[1]
0	0	1	0	WDAT	Write data cycle
0	0	1	1		Reserved (decoded as NULL)
0	1	0	0	LOC	Lock response
0	1	0	1	RER	Read error response
0	1	1	0		Reserved (decoded as NULL)
0	1	1	1		Reserved (decoded as NULL)
1	0	0	0	GRD0	Good read data, cycle 0
1	0	0	1	GRD1	Good read data, cycle 1
1	0	1	0	GRD2	Good read data, cycle 2
1	0	1	1	GRD3	Good read data, cycle 3
1	1	0	0	CRD0	Corrected read data, cycle 0
1	1	0	1	CRD1	Corrected read data, cycle 1
1	1	1	0	CRD2	Corrected read data, cycle 2
1	1	1	1	CRD3	Corrected read data, cycle 3

[1]See Table 3-5 for the encoding of XMI D<63:60> when XMI F<3:0> specifies a command cycle.

3.4 BUS COMMAND CODES

Table 3–5 Bus Command Codes

XMI D<63:60>					
63	62	61	60	Mnemonic	Command
0	0	0	0	–	RESERVED (decode as NULL)
0	0	0	1	READ	Read
0	0	1	0	IREAD	Interlock read
0	0	1	1	OREAD[1]	Ownership read
0	1	0	0	DWMASK[1]	Disown write masked
0	1	0	1	–	Reserved (decoded as null)
0	1	1	0	UWMASK	Unlock write masked
0	1	1	1	WMASK	Write masked
1	0	0	0	INTR	Interrupt
1	0	0	1	IDENT	Interrupt acknowledge
1	0	1	0	–	RESERVED (decode as NULL)
1	0	1	1	TBDATA[1]	Tag bad data
1	1	0	0	–	RESERVED (decode as NULL)
1	1	0	1	–	RESERVED (decode as NULL)
1	1	1	0	–	RESERVED (decode as NULL)
1	1	1	1	IVINTR	Implied vector interrupt

[1]XMI+ protocol only.

3.5 DATA TRANSFER TRANSACTIONS

The XMI architecture supports data transfers of all data sizes to both memory and I/O space (some implementations may not support transfers greater than a longword to I/O space).

Table 3–6 overviews the data transfer transactions. Figures 3–3 to 3–8 show the command/address and data cycles of selected transactions.

Refer to Chapter 4 for memory response requirements on systems which support the XMI+ protocol.

Table 3–6 Data Transfer Transactions

Type	Description
Read (READ)	Moves a longword, quadword, octaword, or hexword of data from the responder to the commander. Data are naturally aligned and delivered in wraparound order. Multiple transfers may be necessary to transfer all quadwords of an octaword or hexword transaction.
Interlocked read (IREAD)	IREADs to memory space are similar to READs except that memory determines if the transfer should continue based on the state of the block referenced:

Block State	Memory Action
Free	Locks the location from future IREADs and OREADs[1] and returns the requested data to the commander. The commander must issue an UWMASK to release the lock.
Locked or owned[1]	Returns a LOC response to the requesting node and terminates the transaction; no data is transferred.

IREADs to I/O space are implementation dependent. Most I/O nodes treat IREADs the same as READs (and UWMASKs the same as WMASKs).

Interlock granularity is implementation dependent. The minimum supported granularity in memory space is hexword.

[1]XMI+ protocol.

Table 3–6 (Cont.) Data Transfer Transactions

Type	Description
Ownership read (OREAD)[1]	Moves a hexword block from memory to a caching node and flags the block as owned in memory. The owning node must issue a DWMASK to write the cached data back to memory and release ownership of the block. Write-back cache nodes use OREADs to perform both writes, and reads with modify intent. The XMI+ protocol supports hexword OREADs only.
Write masked (WMASK)	Moves specific bytes in a longword, quadword, octaword, or hexword data block from the commander to the responder. The data block is naturally aligned and the bytes to be transferred are identified by a byte mask field. Write transactions are performed with one, two, or four consecutive data transfer cycles with no NULL cycles in between.
Unlock write masked (UWMASK)	Complement of IREAD. Writes data to, and releases the lock on, a locked memory location. When a node issues an IREAD, it must unlock the memory structure when it is finished by issuing UWMASK with the data to be written. When memory receives the UWMASK, it unlocks the memory location and writes the data as requested. If UWMASK is directed to a currently unlocked location, memory performs a masked write operation.
Disown write masked (DWMASK)[1]	Complement of ownership read. Writes data to an owned hexword block and returns the block to the "free" state. The XMI+ protocol requires write-back cache nodes to monitor the bus for references to owned memory blocks. If a node detects a transaction (read or write) to a block it owns, the node is required to immediately issue a DWMASK to write the cached data back to memory and release ownership of the block. The XMI+ protocol supports quadword, octaword and hexword DWMASK transactions.

[1]XMI+ protocol.

Table 3-6 (Cont.) Data Transfer Transactions

Type	Description
Tag bad data (TBDATA)[1]	Used in place of DWMASK by write-back cache nodes if cached data is corrupted. Flags the corresponding memory location as bad. Write-back cache nodes implement error correcting code (ECC) on cached data. If a double bit error is detected, a TBDATA is issued in place of DWMASK to flag the data as bad to future references. Tagging a location as bad allows corrupted data to be more readily associated with an actual process since the first read reference to the location will fail. The XMI+ protocol supports quadword, octaword and hexword TBDATA transactions.

[1]XMI+ protocol.

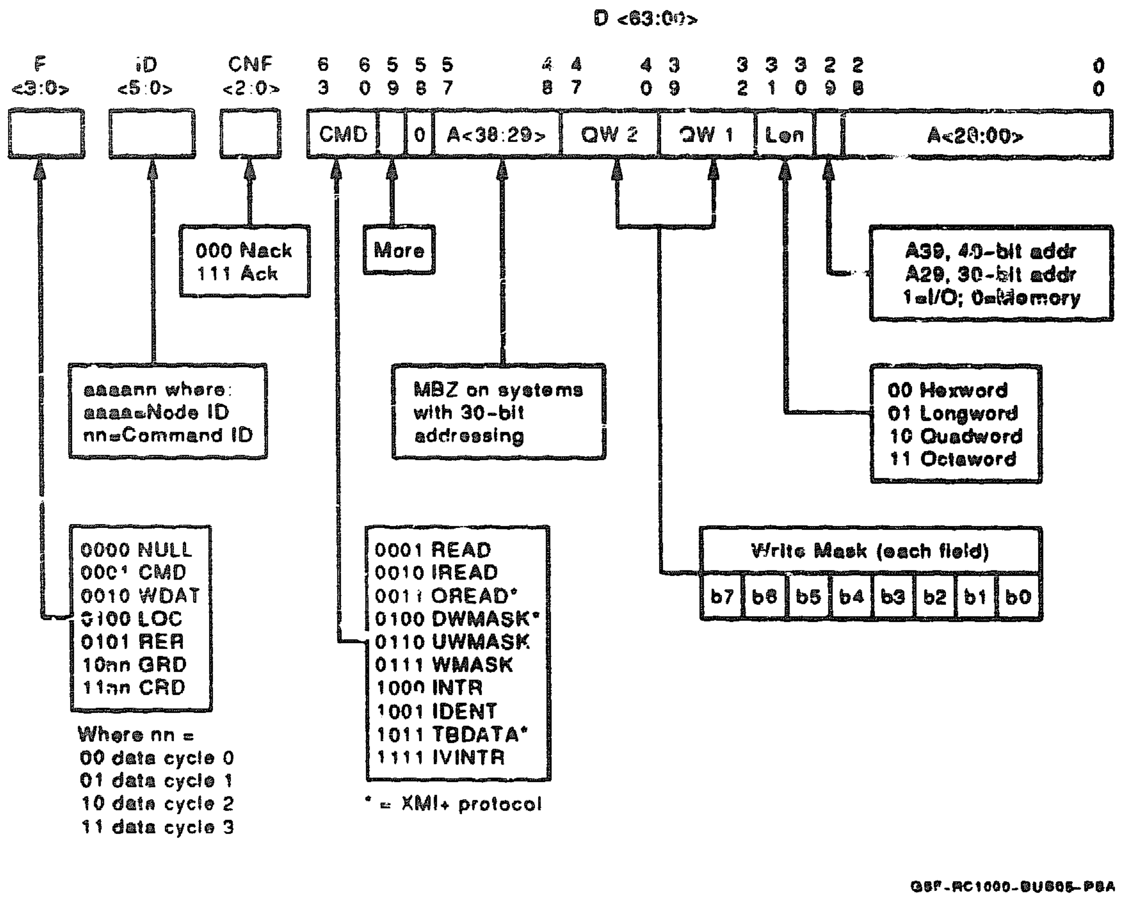
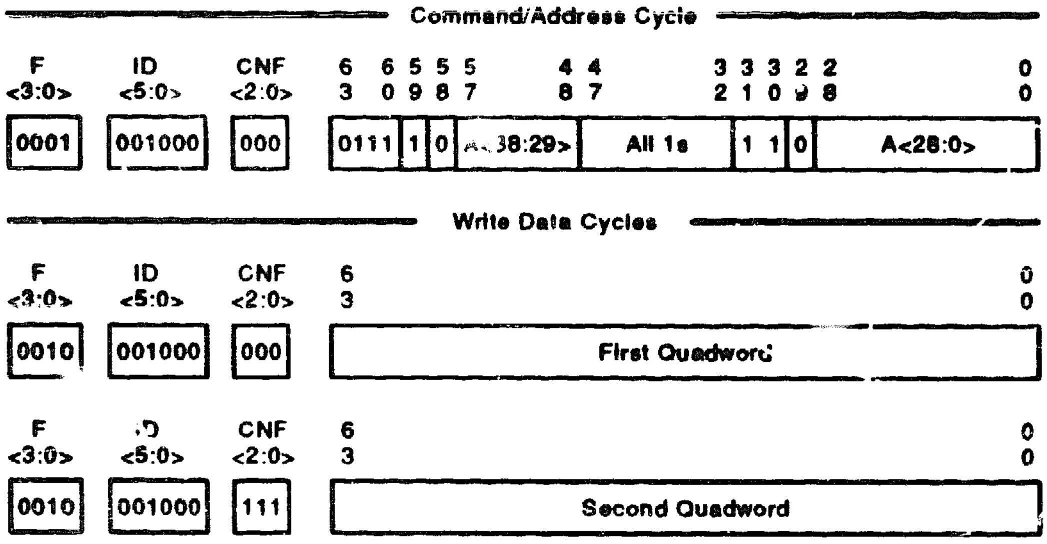


Figure 3–3 XMI Bus Signals for a Command/Address Cycle

Bus Cycles

	1	2	3	4	5	6
D <63:00>		WRITE MASKED COMMAND / MORE / WRITE ADDRESS / WRITE MASK / LENGTH	WRITE DATA 0	WRITE DATA 1		
F <3:0>		COMMAND	WRITE DATA	WRITE DATA		
ID <5:0>		CMDR				
CNF <2:0>				ACK	ACK	ACK
ARB	CMDR GRANT	HOLD	HOLD			

GSF-RC1000-BUS06-PSA



GSF-RC1000-BUS07-PSA

Figure 3–4 Octaword Write Bus Cycles

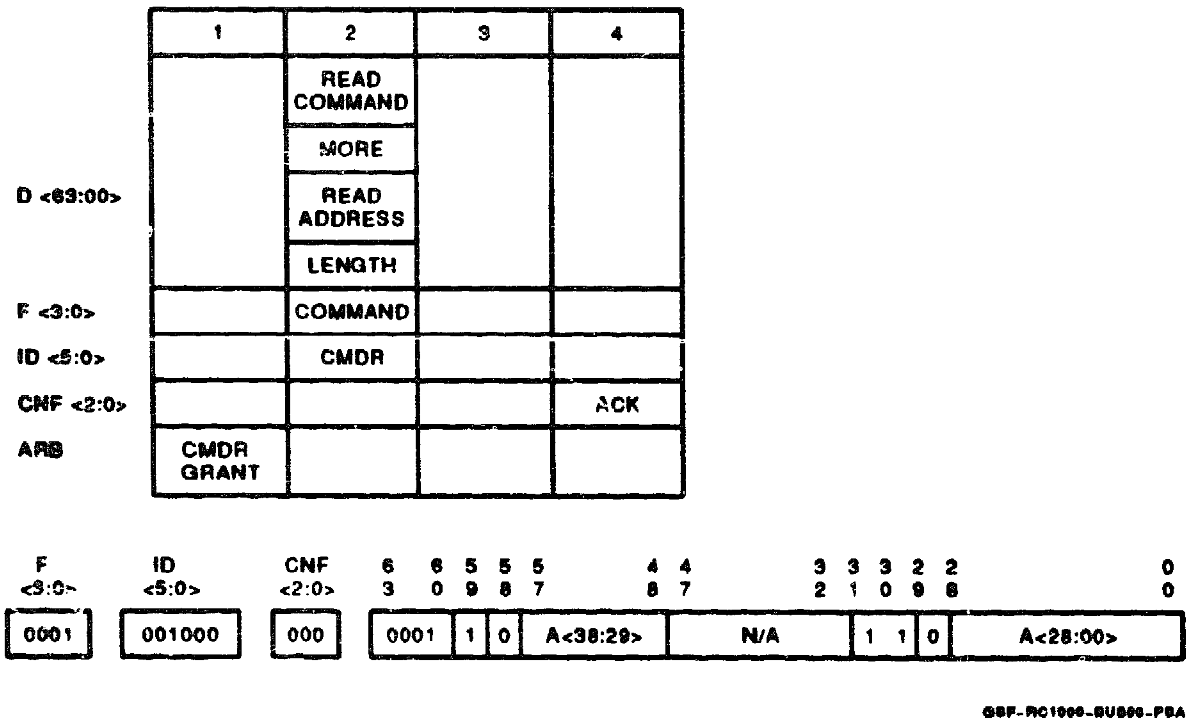
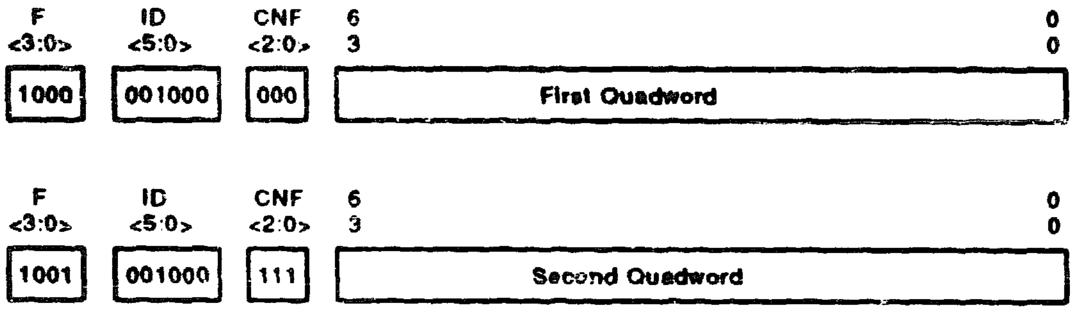


Figure 3–5 Octaword Read Command/Address Cycle

	n	n+1	n+2	n+3	n+4	n+5
D <63:00>		READ DATA 0		READ DATA 1		
F <3:0>		GRD0		GRD1		
ID <5:0>		CMDR		CMDR		
CNF <2:0>				ACK		ACK
ARB	RSPNDR GRANT		RSPNDR GRANT			



GSF-RC1000-BU009-PSA

Figure 3–6 Octaword Read, Read Response Data Cycles

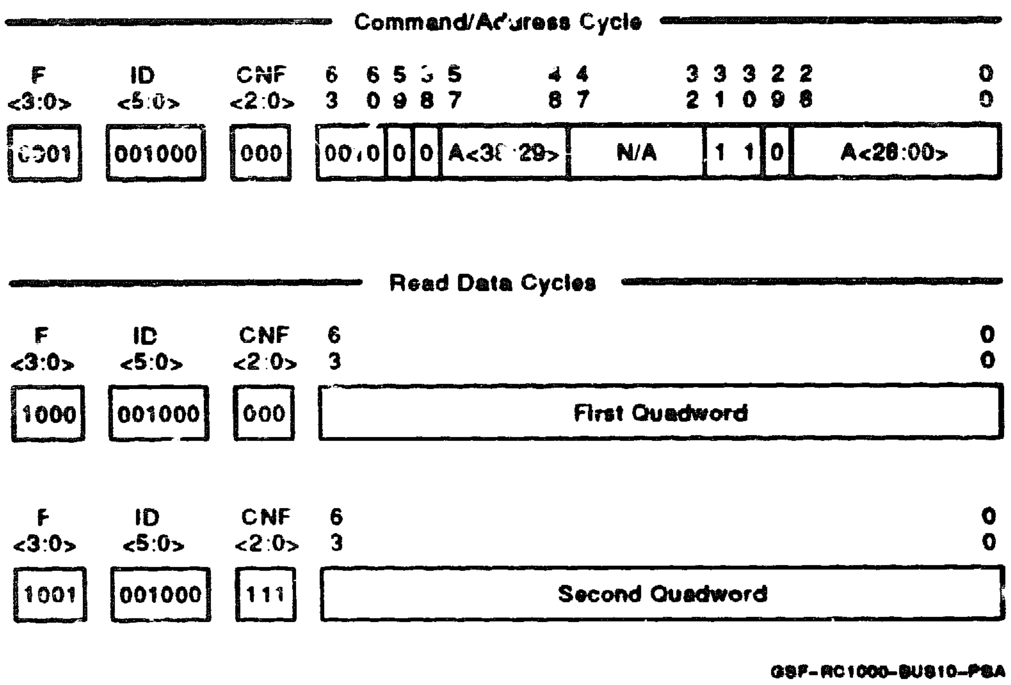


Figure 3–7 Interlock Read Transaction

NOTE
Interlock reads are similar to normal reads except that the memory location is locked and MORE is not allowed. See Figures 3–5 and 3–6.

If the target address is currently locked, memory will respond with the lock response (F <3:0> = 0100) and the XMI data lines (D <63:00>) are ignored.

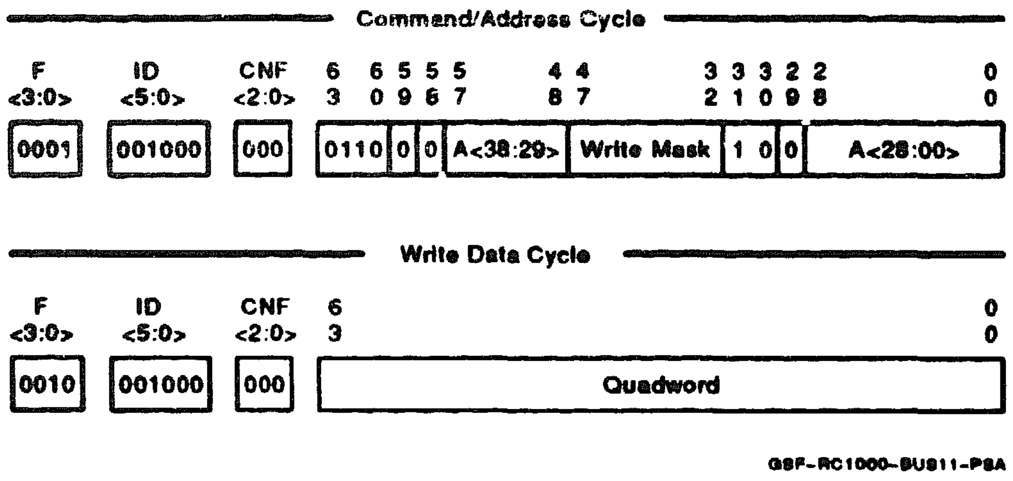


Figure 3-8 Unlock Write Transaction

3.6 INTERRUPT TRANSACTIONS

Table 3-7 Interrupt Transactions

Name	Mnem	Description
Interrupt request	INTR	Issued by I/O nodes to interrupt instruction execution in a processor (or processors) at a specified IPL. Interrupt requests can be broadcast to multiple processor nodes. The first processor responding with IDENT receives the interrupt vector; all other processors clear the interrupt pending condition.
Interrupt acknowledge	IDENT	Issued by a processor in response to an INTR transaction to request an interrupt vector. If IDENTs are issued simultaneously by two or more processors, the first to gain the bus services the interrupt; the other processors force a passive release.
Implied vector interrupt	IVINTR	Issued by a node to implement a single-cycle interrupt transaction. The interrupt priority and the interrupt vector value are implied by bits encoded in the interrupt type field. The IVINTR is used for interprocessor interrupts and write error interrupts. Since the interrupt priority and vector are indicated in the transaction, an IVINTR does not require a corresponding interrupt acknowledge cycle.

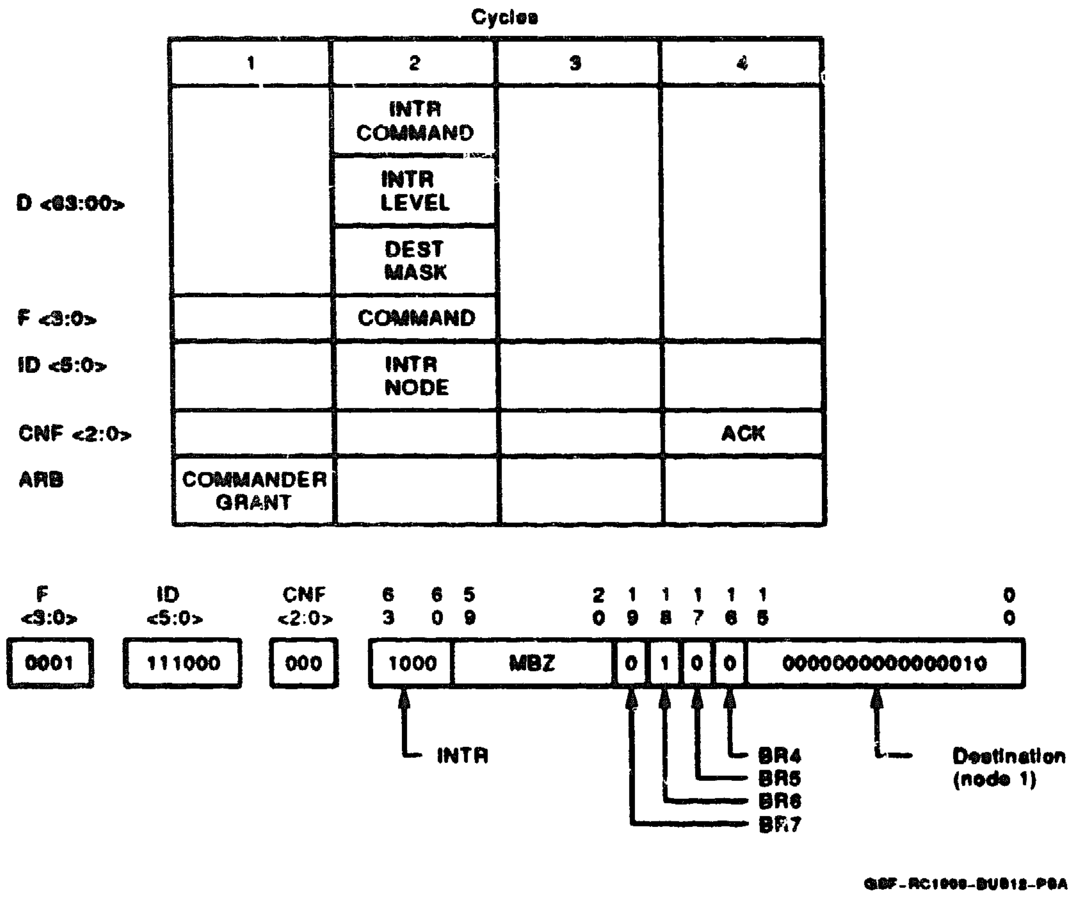


Figure 3-9 INTR Transaction

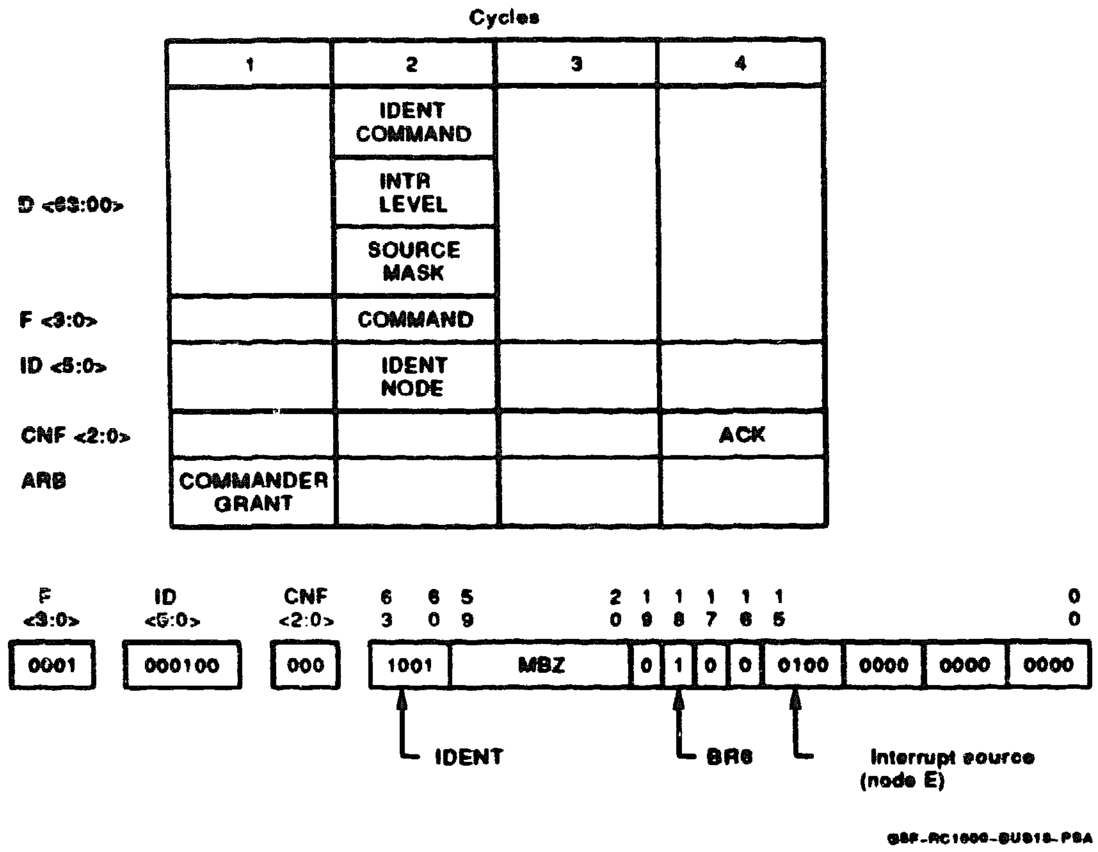


Figure 3-10 IDENT Transaction

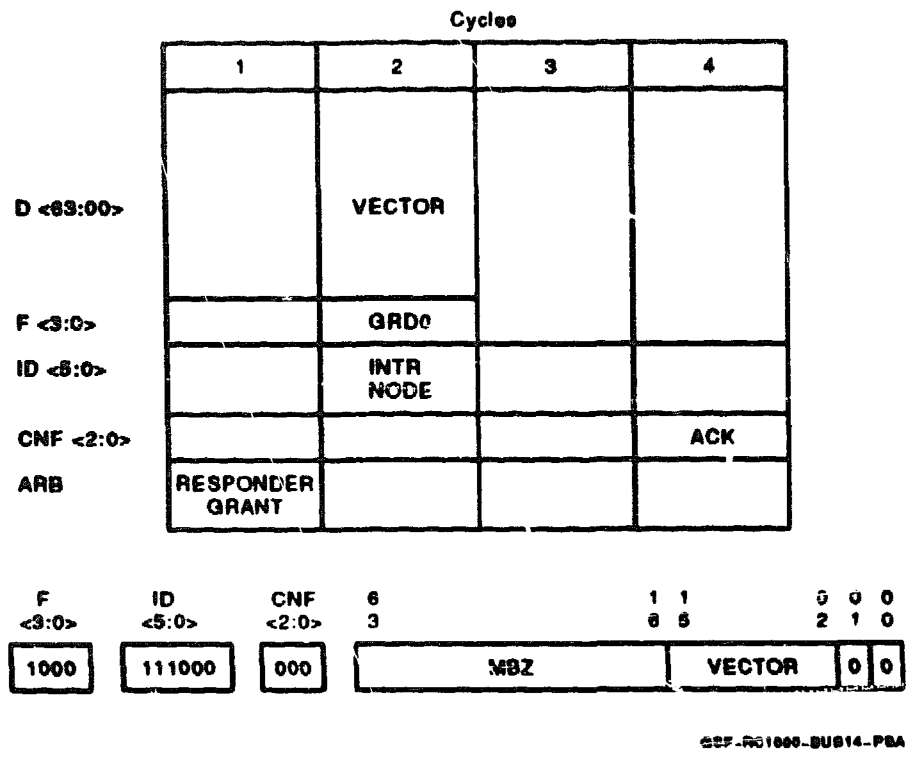


Figure 3-11 IDENT Response

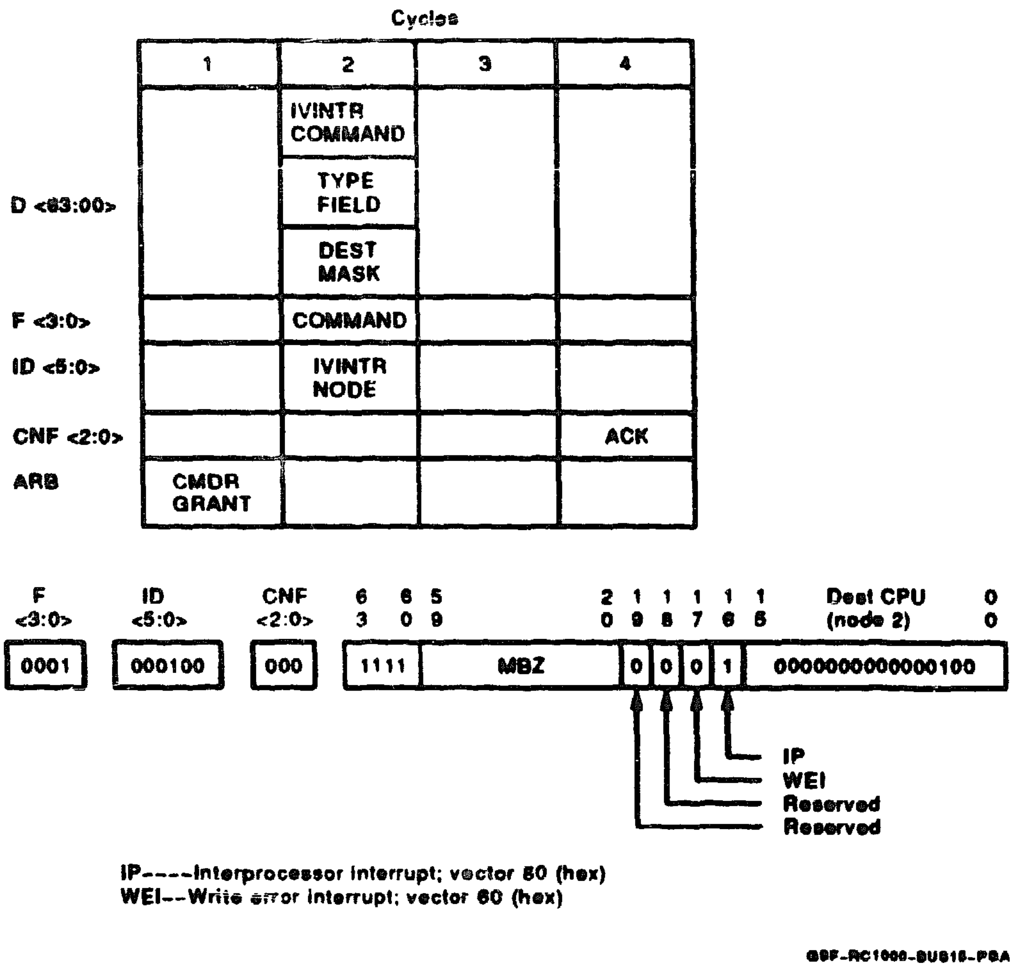


Figure 3–12 IVINTR Transaction

3.7 ARBITRATION

The XMI arbiter logic[1] has two independent arbitration queues: one for commanders and one for responders. Arbitration for each queue is performed in a round-robin manner, with responder requests receiving higher priority than commander requests.

Figure 3–13 shows the XMI arbitration logic. Note that with a set of dedicated arbitration lines, XMI bus arbitration cycles occur in parallel with data transfer cycles.

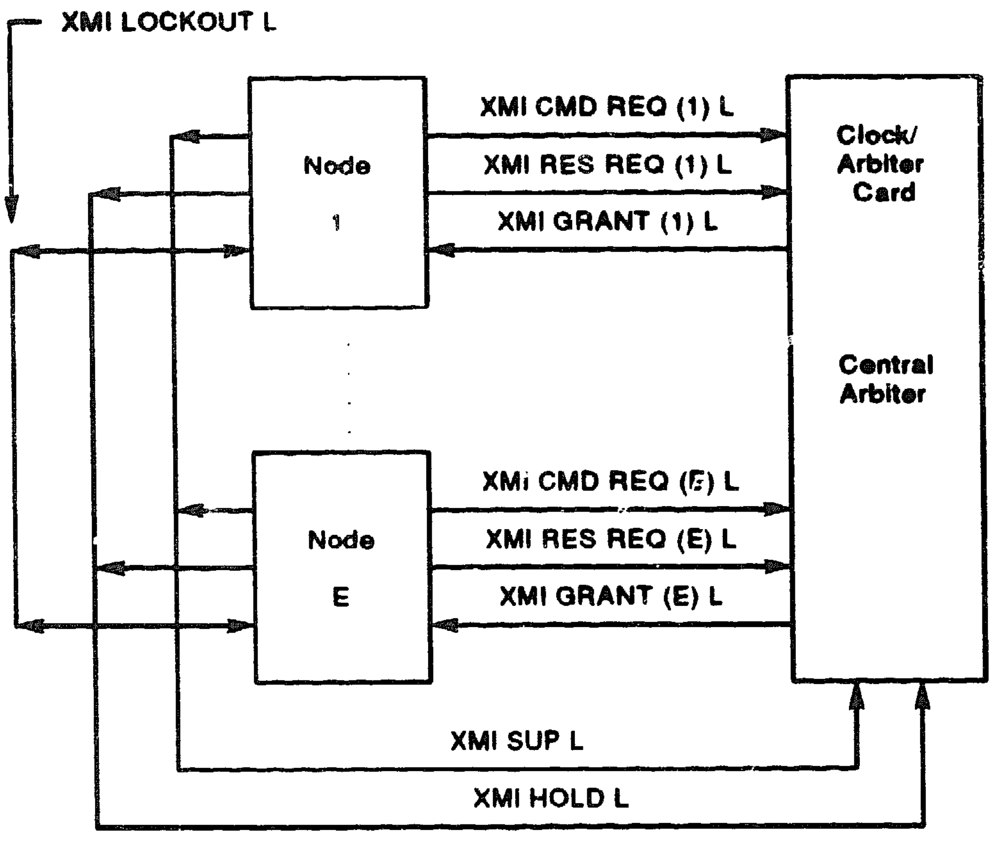


Figure 3–13 Arbitration Block Diagram

[1] DCARD on VAX 6000 systems; CCARD on VAX 9000 systems.

3.8 BUS INITIALIZATION

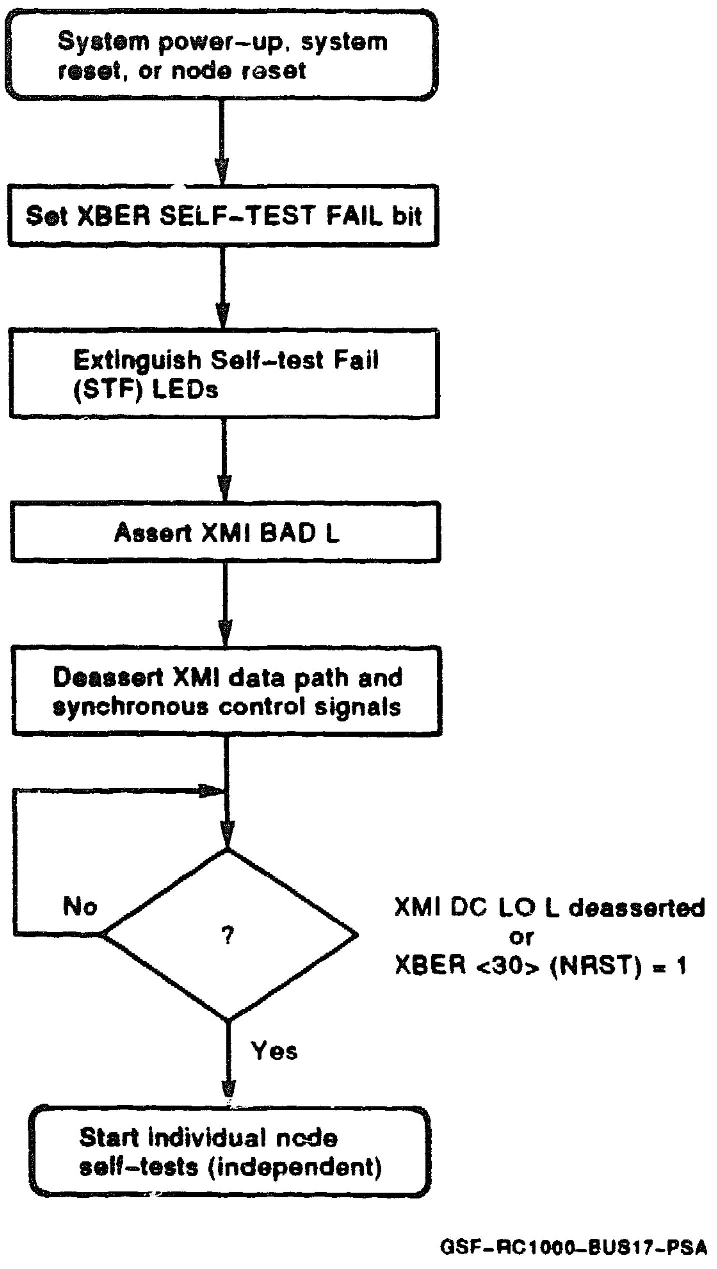

System power–up, system reset, or node reset
Set XBER SELF–TEST FAIL bit
Extinguish Self–test Fail (STF) LEDs
Assert XMI BAD L
Deassert XMI data path and synchronous control signals
?
No
XMI DC LO L deasserted or XBER <30> (NRST) = 1
Yes
Start individual node self–tests (independent)
GSF–RC1000–BUS17–PSA

Figure 3–14 Bus Initialization Flowchart

3.9 XMI NODE REQUIRED REGISTERS

The XMI architecture requires that each node include certain registers in specific locations in the node's nodespace:

Table 3–8 Required and Recommended Registers

Register Name	Mnem	Addr[1]	Status
Device type	XDEV	bb+00	Required
Bus error	XBE	bb+04	Required
Failing address	XFADR	bb+08	Required for commanders
Communication	XCOMM	bb+10	Recommended for RBD
Failing address extension	XFAER	bb+2C	Required for commanders

[1]Offset from base address. See Figure 3–2, and Tables 3–1 and 3–2.

Refer to the adapter specific handbooks of the *XMI Adapters Handbook Documentation Set* for descriptions of the XMI required and adapter specific registers.

CHAPTER 4

4 WRITE-BACK CACHE SUPPORT AND THE XMI+ PROTOCOL

4.1 WRITE-BACK CACHE OVERVIEW

Write-back cache enhances system performance by increasing the effective bandwidth of memory write operations. With write-back cache, memory writes, in addition to reads, can be serviced locally by a caching node without the need for generating immediate main memory references. This reduces the overall number of memory writes that would otherwise be required.

NOTE
In this chapter, the term "XMI+ memory" will be used to denote memory nodes which support the XMI+ protocol.

4.1.1 Block Ownership

The implementation of write-back cache on systems which support the XMI+ protocol is based on the "ownership" of data in memory.

XMI+ memory maintains two state bits for each 32 byte block (hexword) of data (see Section 4.2). At any given time, any node on the bus can own any block. The XMI+ protocol ensures that only one node will own a given block at a given time.

4.1.2 Gaining Block Ownership

A node gains block ownership by performing an ownership read (OREAD) transaction. During the time a node owns a block, it is free to write the block without generating main memory transactions. Once a node writes a block in its local cache, the block is considered "dirty" in that only the caching node has the updated data; main memory has the original data.

4.1.3 Bus Monitoring

The XMI+ protocol requires write-back cache nodes to monitor the bus for references to memory blocks owned by the node.

If an IREAD or OREAD is issued to a location owned by another node, XMI+ memory will return a locked (LOC) response to the commander. When the owning node detects a reference, it must immediately issue a DWMASK to write the cached data back to memory and release the block.

See Table 4-1 for XMI+ memory responses to bus transactions and to Table 4-3 for the actions taken by the owning node.

4.1.4 Releasing Ownership

A node releases block ownership by issuing a disown write mask (DWMASK) or tag bad data (TBDATA) transaction. Once the node writes the data back to memory and releases the block, read references to the block by other nodes result in the updated data being returned by memory. (This assumes that no other node wrote to the block between the OREAD and DWMASK; see Section 4.2.3.)

4.2 MEMORY REQUIREMENTS

The XMI+ protocol places certain constraints on memory to support write-back cache. This section overviews the major considerations.

4.2.1 Block State

The XMI+ protocol requires memory to maintain two state bits for each hexword block of data. At any given time, a block in memory can be in one of four states:

State	Block Status
Free	Neither owned or locked
Locked	Interlocked as a result of an IREAD
Owned	Owned by a write-back cache node as a result of an OREAD
Tagged bad	Contains corrupted data. (A caching node wrote back corrupted data and tagged the location as bad.)

4.2.2 Memory Response to Bus Transactions

The current block state determines memory's response to data transfer transactions. Table 4–1 lists the possible responses.

Table 4–1 Memory Response Requirements to Bus Transactions

	Block State[1]			
Command	**Free**	**Locked**	**Owned**	**Tagged Bad**
READ	GRD	GRD	GRD	RER
IREAD	GRD(L)	LOC	LOC	RER
OREAD	GRD(O)	LOC	LOC	RER
WMASK	Write	Write	Write	Write
UWMASK	Write[2]	Write(F)	Write[2]	Write[2]
DWMASK	Write(F)[2]	Write(F)[3]	Write(F)	Write[2]
TBDATA	Write(B)[2]	Write(B)[3]	Write(B)	Write[2]

[1]Letter in parentheses indicates the next state if it is different from the current state: F = free, L = locked, O = owned, and B = tag bad.

[2]Error condition.

4.2.3 Servicing READ and WMASK Transactions

Under the XMI+ protocol, owned memory locations cannot be read or written by another node until released (disowned) by the owning node. To accommodate READ and WMASK transactions to owned blocks, without the need for considering data ownership, XMI+ memory includes buffers and command queues. This allows memory to buffer writes, and defer reads, to owned locations until the data has been written back.

Table 4–2 lists the actions taken by XMI+ memory on READs and WMASKs to owned locations.

Table 4-2 Memory Actions on READs and WMASKs to Owned Blocks

Command	Memory Actions
READ	On a READ to an owned location, memory: 1. Stores the command, address, and ID in a deferred queue 2. Waits for the owning node to issue DWMASK 3. Processes the DWMASK with any required masking as indicated by the associated write buffer (if any; see WMASK actions below) 4. Processes all deferred reads that match the DWMASK address and returns the data with the appropriate response to the commander The order in which reads are processed is not critical except that all deferred reads to the DWMASK address must be processed before continuing with other commands.
WMASK	On a WMASK to an owned location, memory: 1. Immediately writes the data into the block 2. Stores the command, address, and mask bits in a buffer 3. Waits for the owning node to issue a DWMASK 4. Executes the DWMASK, but modifies the mask bits such that bytes written by the WMASK are not overwritten by the DWMASK. A mask buffer covers a hexword and, therefore, contains 32 mask bits (one bit for each byte). 5. Releases the block and the corresponding mask buffer If more than one write occurs to a given block between OREAD and DWMASK, XMI+ memory performs each write in order, accumulating a composite byte mask representing all bytes written by all WMASKs to the block. Memory uses the composite mask to modify the bytes not written by previous writes to that location.

4.2.4 Memory/Local Cache Consistency

To maintain consistency between local caches and main memory, all write-back cache nodes monitor the bus for potential references to memory blocks which may be owned by the node.

The action taken by a caching node on memory references by other nodes depends on the type of transaction and the result of comparing the bus address to entries in the local cache.

The following table lists the results possible from comparing the bus address to entries in a local cache. Table 4-3 lists the actions taken by a caching node in response to memory transactions by other nodes.

Compare Result	Indicates referenced location is
Miss	Not present in local cache
Clean hit	Present, marked valid, and unmodified (dirty bit clear)
Dirty hit	Present, marked valid, and modified (dirty bit set)

Table 4-3 Write-back Cache Action in Response to Bus Transactions

	Transaction Type				
Compare Result	READ	IREAD	OREAD	WMASK/ UWMASK	DWMASK
Miss	No action (all transaction types)				
Clean hit	None	Invalidate[1]	Invalidate	Invalidate	Error
Dirty hit	Write[2]	Write[2]	Write[3]	Write[3]	Error

[1]Node specific.

[2]Write-back cached data with DWMASK.

[3]Write-back cached data with DWMASK and invalidate cache.

4.3 I/O CONTROLLER SUPPORT

The XMI+ protocol supports I/O controllers that are compatible with the XMI protocol; I/O nodes do not perform OREAD and DWMASK transactions.

I/O Write to Memory

From the point of view of an I/O node, writes always complete just as they would with the XMI protocol. See Section 4.2.3 for the servicing of writes to owned blocks.

I/O Read From Memory

XMI+ memory services I/O reads of memory the same as CPU reads. If the read reference is to an owned location, the read command is deferred until the DWMASK is issued; see Section 4.2.3.

CHAPTER 5

5 DIAGNOSING XMI BUS RELATED ERRORS

5.1 NODE RBDs AND SELF-TESTS

XMI nodes include ROM-based diagnostics (RBDs) which are stored in non-volatile memory on the node module. The RBDs are run by calling the RBD user interface with the console "Z" command and then executing the appropriate RBD. The node self-tests are part of the RBD and are run when the node is initialized or when invoked by the operator.

Refer to the adapter specific handbooks of the *XMI Adapters Handbook Documentation Set* for adapter specific RBD and self-test descriptions.

5.2 ERROR CONDITIONS

Parity Error

All nodes monitor parity on the bus to detect single-bit errors. XMI receivers that detect bad parity ignore the current cycle and return a NOACK confirmation code.

Inconsistent Parity Error

Under certain conditions, some nodes may detect bad parity while others compute good parity. If the target of a transaction computes good parity, the cycle may be ACKed (and assumed good by the commander), even if other nodes ignore the cycle due to bad parity.

For memory-space write and OREAD (XMI+ protocol) transactions, this class of error may result in cache coherency problems due to cached processors failing to perform cache invalidates. CPU nodes running the XMI+ protocol recover from this type of error by executing error recovery software which flushes the cache (invalidating all unmodified blocks and writing back to memory all modified blocks).

For IVINTR transactions, some destinations of the IVINTR transaction may not receive the interrupt. All other XMI bus transactions are insensitive to this class of error.

Transaction Timeout

The XMI protocol supports three types of timeouts commanders may use to detect transaction failures. Responders must ensure that, under normal conditions, transactions do not exceed the timeout periods. XMI timeouts in a given node can be disabled by setting bit <2> of the XMI bus error (XBE) register.

Table 5-1 XMI Bus Transaction Timeouts

Type	Description
Response	During a READ, IREAD, or IDENT, if a commander does not receive all read responses within a certain number of cycles after the transaction is issued, the transaction is considered to have failed. Note that this does not imply that a responder is not functioning; XMI receivers ignore cycles with bad parity and response timeouts can occur as a result of ignored cycles.
Retry	An XMI commander may need to reissue an XMI transaction if it receives a NOACK or a LOC response. If the commander cannot successfully complete the transaction within the retry timeout period, the transaction is considered to have failed.
Lockout	When an XMI commander is repeatedly denied access to a shared resource, it must assert XMI LOCKOUT to ensure access to the resource. If the commander cannot successfully complete the transaction during the lockout timeout interval, the transaction is considered to have failed.

Sequence Error

Many transactions require that XMI cycles occur in a certain sequence. If the cycles occur out of sequence, the transaction is in error.

READ, IREAD, and IDENT transactions incorporate sequence IDs in the read data responses (GRDn, CRDn, and RER) to denote the read response cycle. The required order of sequence ID(s) for read responses is as follows:

Data Size	**Sequence ID(s)**
Longword [1]	0
Quadword	0
Octaword	0,1
Hexword	0,1,2,3

[1]Including IDENT.

The sequence ID for RER is implicitly 0.

For write transactions, the correct sequencing of cycles is determined by the location of the write data cycles relative to the write command cycle, rather than the use of sequence IDs. The write command cycle and associated write data cycles must occur in contiguous time slots. If a responder detects missing data cycles in a write transaction, the incorrect cycle (and subsequent write data cycles) are NOACKed.

5.3 ERROR HANDLING

XMI commanders and responders react to error conditions as follows:

- On an XMI bus error (denoted by the setting of XBE <31>), controllers/adapters do not clear the XMI visible XBE bits, but instead rely on the host (a CPU node) to clear the bits.
- Receivers that detect bad parity ignore the cycle.
- Responders ignore write transactions with a sequence or parity error (data at the referenced location is not modified since the entire write transaction is ignored).
- Responders receiving a NOACK to a read response do not transmit further read responses associated with that transaction within 10 XMI cycles of the NOACK.
- Memory nodes running the XMI protocol do not set a lock bit unless all read responses associated with an IREAD receive an ACK.

  Memory nodes running the XMI+ protocol set the lock or ownership bits on successful receipt of the IREAD or OREAD command/address cycle.
- Memory nodes do not clear a lock bit unless all write data cycles associated with the UWMASK are properly received.
- Cached processors detecting an inconsistent parity error either flush their cache, perform a machine check, or take appropriate action to maintain data consistency.

5.4 ERROR RECOVERY

Error recovery involves one or more repeat attempts of the failed transaction before reporting a hard error. Failed XMI transactions are retried under the following conditions:

- Any write to, or read from, memory space
- Any WMASK, UWMASK, READ, or IREAD to I/O space (OREAD, DWMASK, and TBDATA are not supported to I/O space)
- Any NOACK on the command cycle of any transaction. The transaction is automatically retried by hardware.

  NOACK can result from either a reference to nonexistent memory (NXM) or from a bus parity error. A transaction failing the retry is assumed to be an NXM.
- Any IDENT receiving a response timeout. This may result in a lost interrupt vector, the consequences of which may require servicing by software.

Note that on systems interfaced to a VAXBI bus, it may be unsafe to retry a READ or IREAD to VAXBI I/O space which resulted in an response timeout since some I/O devices may have read side effects.

5.5 ERROR REPORTING

The XMI bus protocol supports two mechanisms that signal error conditions to processors if normal transaction-level error reporting cannot be used.

Normal transaction-level error reporting mechanisms include NOACK, read error response (RER), and timeout. The mechanisms that signal error conditions to processors if normal transaction-level error reporting cannot be used are:

Mechanism	Description
Write error interrupt	This transaction is directed to one or more CPU nodes, resulting in each targeted CPU taking an IPL 1D (hex) error interrupt. The CPU then identifies the source of the write error interrupt.
XMI TRIGGER [1]	When XMI TRIGGER is asserted, all XMI CPUs take an IPL 1D (hex) error interrupt.

[1]This signal may be labeled XMI FAULT in some implementations. Use of XMI TRIGGER (or XMI FAULT) is system specific.

CIXCD

Document Title: CIXCD HANDBOOK

Order Number: EK-CIXCD-HB-001

This handbook is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The handbook can be ordered separately or as part of the set.

The *XMI Adapters Handbook Documentation Set* is a dynamic document which will be periodically updated as new XMI adapters are announced. The first release of the set includes the following handbooks:

Order Number	Title
EK-XMIOV-HB	XMI Bus Overview Handbook
EK-CIXCD-HB	CIXCD Handbook
EK-DEMNA-HB	DEC LANcontroller 400 (DEMNA) Handbook
EK-DWMBA-HB	DWMBA Handbook

This handbook and the document set are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

CIXCD Handbook

Order Numb·er EK-CIXCD-HB-001

This document is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The document can be ordered separately or as part of the set. The *CIXCD Handbook* and the *XMI Adapters Handbook Documentation Set* are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

Revision/Update Information: Revision 1.0

Digital Equipment Corporation

January, 1991

DEC	UNIBUS	VAX 9000
DECUS	VAX	VAXBI
PDP	VAX 6000	XMI

This document was prepared and published by Educational Services Development and Publishing, Digital Equipment Corporation.

Contents

Examples

Figures

Tables

About This Manual

Intended Audience

This handbook is part of a series of handbooks which comprise the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). This handbook and the handbook set are for VAX system trained Digital customer service personnel who service XMI-based systems and subsystems. Users of the handbook set should be familiar with the XMI bus architecture (either through the *XMI Bus Concepts* course or through practical experience) and have a minimum of level 1 hardware maintenance training on one or more VAX systems (for example, VAX 6000 or VAX 9000 systems).

Document Scope and Structure

Several I/O adapters have been developed to interface the XMI bus to devices which employ different bus structures and protocols. These adapters are available as stand-alone options and may be installed on a variety of systems or subsystems.

The *XMI Adapters Handbook Documentation Set* provides a single, quick reference source to the type of information most frequently required to service XMI adapters. This handbook contains information specific to the CIXCD option.

This handbook is divided into five chapters:

Chapter 1 introduces the CIXCD and overviews its physical and functional characteristics.

Chapter 2 indicates the configuration requirements for installing the option.

Chapter 3 describes the CIXCD's power-up self-tests and ROM-based diagnostics.

Chapter 4 reviews the CIXCD's macro-level diagnostics and support programs.

Chapter 5 describes the XMI required and CIXCD specific registers.

Conventions

addresses	All addresses are given in hexadecimal (hex).
bits	All bit numbers are given in decimal with the bit(s) enclosed in angle brackets; for example <31>.
	Multiple individual bits or bit fields are separated by commas with bit fields indicated by two numbers separated by a colon. For example <31:24,20,18,14:10> indicates bits 31 through 24 (inclusive), bit 20, bit 18, and bits 14 through 10 (inclusive).
CTRL/x	Specifies to press and hold the Ctrl key while pressing the x key; for example, CTRL/C.
[item] . . .	Indicates the item is optional. The horizontal ellipsis indicates that additional optional items can be entered.
. . .	Vertical ellipsis in examples, tables, or figures, indicate that not all information is shown.

CHAPTER 1

1 CIXCD INTERFACE OVERVIEW

1.1 INTRODUCTION

The CIXCD is a high-performance I/O interface which connects the XMI to the serial computer interconnect (CI) bus. The CIXCD implements the VAX-11 CI port architecture and incorporates resequencing dual path (RDP) protocol which supports simultaneous dual path operation of the CI. RDP protocol allows for independent operation of each CI path, enabling the CIXCD to transmit separate message packets over both CI paths simultaneously.

Figure 1–1 illustrates the dual pathing capability of the CIXCD in a VAXcluster.

1.2 FUNCTIONAL OVERVIEW

The CIXCD logic is partitioned into five major functional sections, implemented primarily by high-density gate arrays. Figure 1–2 shows the logic sections with gate array mnemonics given in the dashed boxes. Table 1–1 briefly describes each logic section.

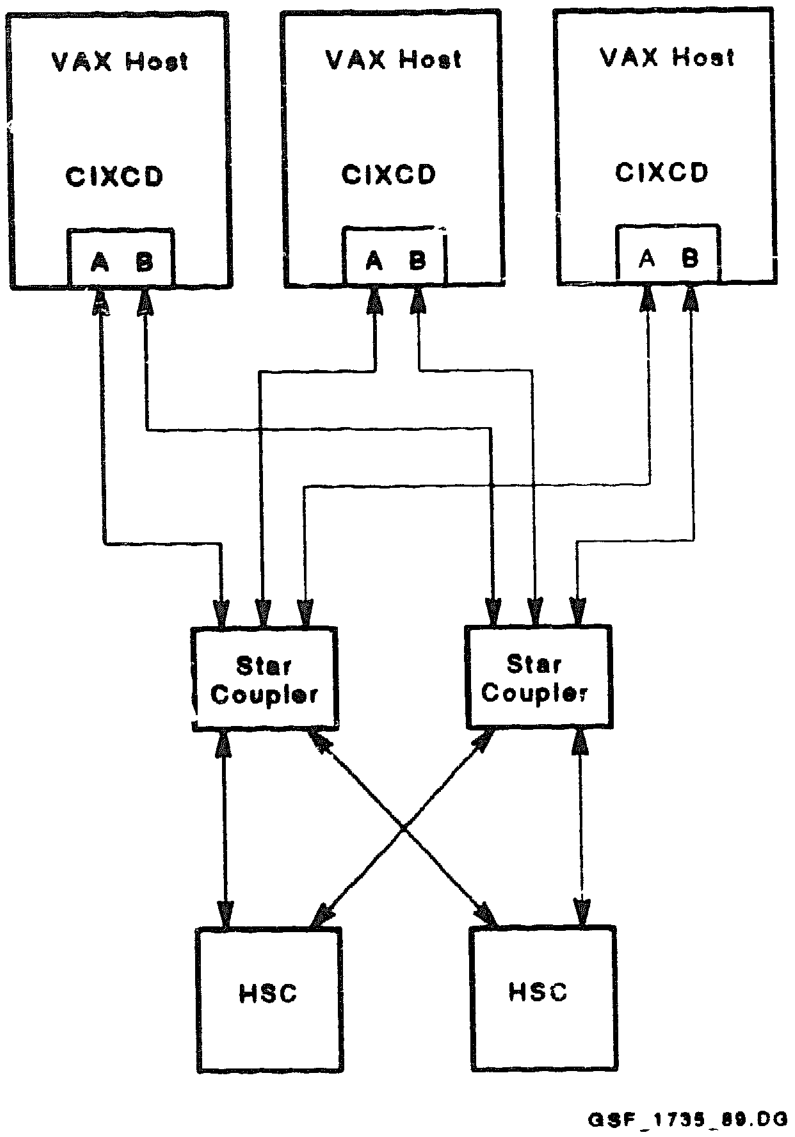


Figure 1-1 CIXCD Interface in a VAXcluster

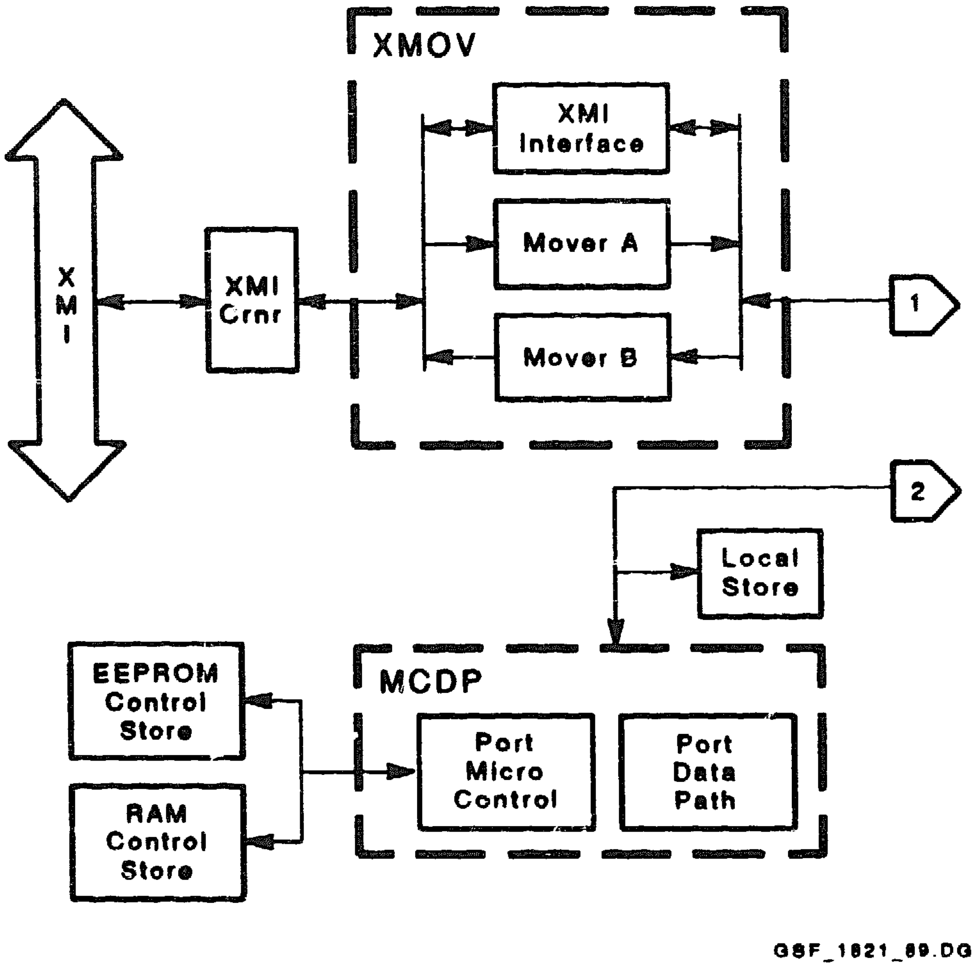


Figure 1–2 (Continued, next page)   CIXCD Functional Block Diagram

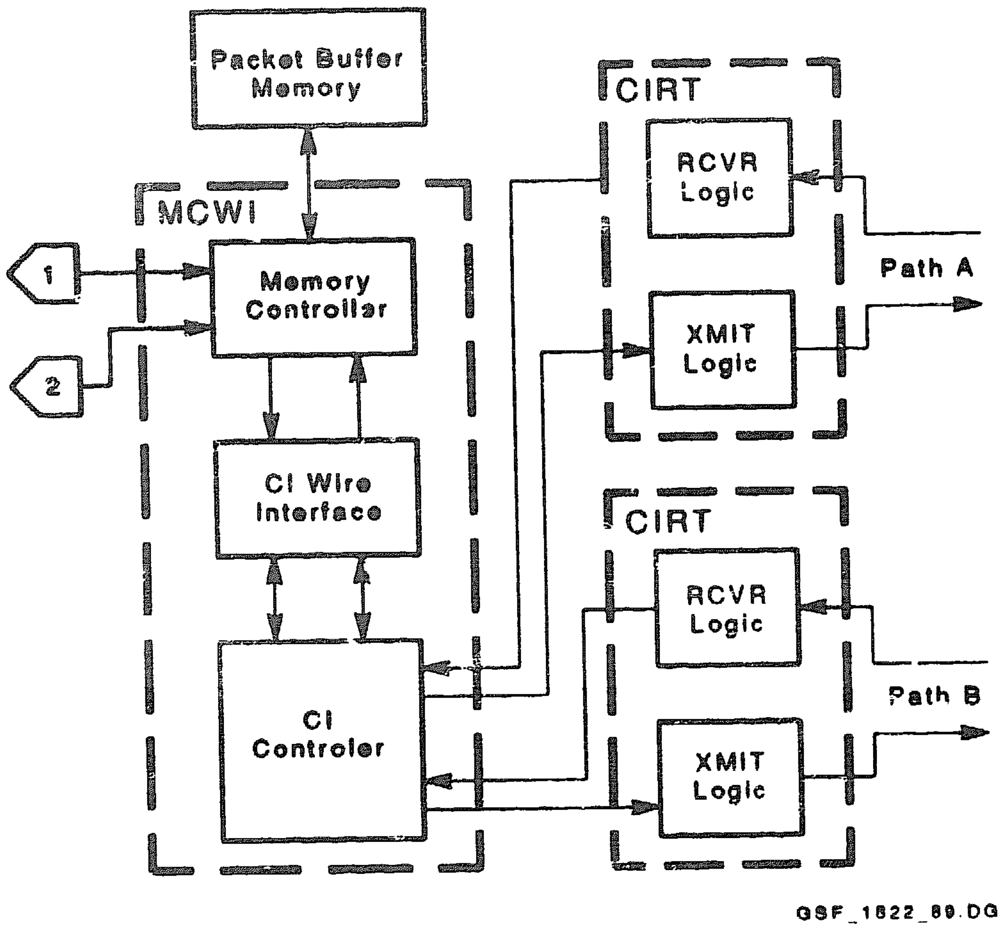


Figure 1–2 CIXCD Functional Block Diagram

Table 1–1 CIXCD Logic Element Descriptions

Element	Description
XMI corner	Provides the interface path to the XMI. Consists of the required XMI XLATCH chips (7) and the XCLOCK chip.
XMOV	XMI interface and data mover gate array. Performs the XMI read (mover A) and write (mover B) transactions. Includes the CIXCD interrupt logic. The data movers are each 32 bits wide and operate at a maximum bandwidth of 20 Mbytes/s. The movers are free running once started by the port microprocessor.
MCDP	Microcontrol and data path gate array. Houses the port microprocessor (port processor) which controls all CIXCD functions. The port processor is a bit-sliced microprocessor with a microsequencer and an 8K × 86 bit control store. ALU operations are executed every 64 ns with next-address calculations every 128 ns. Data transfers occur over a 32-bit data path with parity. The port processor is supported by 32 GPRs, a 16 × 33 bit microstack, an 8K × 33 bit local store, and a 32K × 86 bit electrically erasable programmable read-only memory (EEPROM). The EEPROM contains the CIXCD self-test microdiagnostics and the functional microcode.
MCWI	Memory controller/wire interface gate array. Implements the CI protocol and controls the CI receiver/transmitters. Requests for access to packet buffer memory (8K × 32 RAM) from the CI wire, the data movers, and from the port processor are arbitrated and controlled by the memory control logic.
CIRT	CI receiver/transmitters gate array. Independent interfaces to the CI wires. Performs Manchester encoding and decoding, clock/data separation, and byte framing and synchronization.

1.3 FEATURES

- **Resequencing dual path protocol**
- **Parity on all internal buses and control stores**
- **Writable control store**
- **Internal and external diagnostic loopback capability**
- **Data integrity with cyclic redundancy checking (CRC)**
- **Round-robin arbitration at heavy loading for each path**
- **Contention arbitration at light loading for each path**
- **Packet-orientated data transmission**
- **Immediate acknowledgment of packet reception**

1.4 SPECIFICATIONS

Table 1–2 CIXCD Specifications

Components	
CIXCD-AA — VAX 9000 VAXcluster Interface	
T2080	Module
17-02894-01	Bulkhead cable assembly
12-14314-01	Backplane jumpers (30)
54-20225-01	Header card
EK-CIXCD-UG	CIXCD Users Guide
CIXCD-AB — VAX 6000 VAXcluster Interface	
T2080	Module
17-02894-02	Bulkhead cable assembly
12-14314-01	Backplane jumpers (30)
54-20225-01	Header card
EK-CIXCD-UG	CIXCD Users Guide
Recommended Spares	
F6-T2080-00	
T2080-00	CIXCD service spare
37-01183-01	Package assembly
EK-CIXCD-01	T2080 module insert document
TK50-K	Blank TK50 tape
F5-20225-01	
54-20225-01	Header card spare
37-00813-04	Header card packing
EK-CIXCD-02	Header card insert document

Table 1–2 (Cont.) CIXCD Specifications

Environmental	
Temperature	
Operating	**10°C to 40°C (50°F to 104°F) ambient temperature with a gradient of 10°C (18°F)/h**
Storage/shipping	**-40°C to 70°C (-40°F to 158°F) ambient temperature with a gradient of 20°C (36°F)/h**
Relative humidity	
Operating	**10% to 90% with a maximum wet bulb temperature of 28°C (82°F), a minimum dew point of 2°C (36°F), and no condensation**
Storage/shipping	**5% to 95% with no condensation**
Altitude	
Operating	**Sea level to 2.4 km (8000 ft)** **Maximum operating temperatures decrease by a factor of 1°C/1000 ft (1.8°F/1000 ft) for operation above sea level**
Shipping/storage	**Up to 9.1 km (30,000 ft) above sea level (actual or effective by means of cabin pressurization)**
Shock	**5 Gs peak at 7 to 13 ms duration in three axes mutually perpendicular (maximum)**
Power	
Voltage/Current (nominal)	**Maximum ripple**
+5.0 Vdc at 5.9 A	**300 mV**
-5.2 Vdc at 1.8 A	**150 mV**
-2.0 Vdc at 0.5 A	**150 mV**

Table 1-2 (Cont.) CIXCD Specifications

Data Transfer	
Data format	Manchester encoded serial packet
Data integrity	Cyclic redundancy check
Arbitration	Light loading — contention Heavy loading — round-robin

Table 1-3 CI Bus Specifications

Bus width	Serial
External length	45 m (147.64 ft) maximum
Transfer rate	140 Mbits/s (maximum)
Bus loading	32 nodes (maximum)
Cable type	Double shielded coaxial, BNCIA-XX
Cable impedance	50 ohms

1.5 RELATED DOCUMENTATION

Order Number	Title
EK-CIXCD-UG	*CIXCD User's Guide*
EK-CIXCD-TM	*CIXCD Technical Manual*
EK-VXDSU-UG	*VAX Diagnostic User's Guide*
EK-VXDSU-U1	*VAX Diagnostic User's Guide Update*
EK-VX11D-UG	*VAX Diagnostic System User's Guide*
AA-F152A-TE	*VAX Diagnostic Software Handbook*
EK-SC008-UG	*SC008 Star Coupler User's Guide*
EK-CISCE-UG	*CISCE-AA Installation Guide*
EK-VCSRM-PK	*VAXcluster Service Reference Manual*
EK-VSCIT-RM	*Introduction to VAXcluster Troubleshooting*
EK-VCSFP-RM	*VAXcluster System Troubleshooting Flow Procedures*

CHAPTER 2

2 CIXCD CONFIGURATIONS

2.1 INSTALLATION REQUIREMENTS

The CIXCD option requires one XMI I/O slot[1] for the T2080 module, and one I/O connector panel opening for the CI bulkhead cable connector panel (Figures 2–1 and 2–2).

The CIXCD also includes a header assembly which is a circuit board that converts received CI signals to ECL logic levels. The CI bulkhead cables connect to the header assembly which plugs into the CI backplane (Figure 2–3).

2.2 CONFIGURATION JUMPERS

The CIXCD configuration jumpers are installed in zones D2 and E2 of the XMI backplane slot. The jumpers are denoted as W1 through W30, with W9 being reserved.

Figure 2–4 shows the CIXCD jumpers. Tables 2–1 to 2–5 list the jumper configurations.

[1] In VAX 9000 systems, all slots except slots 7 and 8 are I/O slots. In VAX 6000 systems, the CIXCD can be installed in slots 1 to 4 and B to E.

Note that a system with no jumpers is configured as follows:

- CI node address: 0
- Boot time: 1500 seconds
- Normal CI arbitration
- Normal header
- Delta time: 7 (see following note)
- Cluster size: 16
- Normal ACK timeout

NOTE
The CIXCD, and all devices in a cluster which contain a CIXCD, must be configured for a delta time of 10. If the delta time jumpers (Table 2-4) are configured to any other value, they must be changed to reflect a delta time of 10. The value of 7 in the preceding list is given only to indicate the delta time if no jumpers are installed.

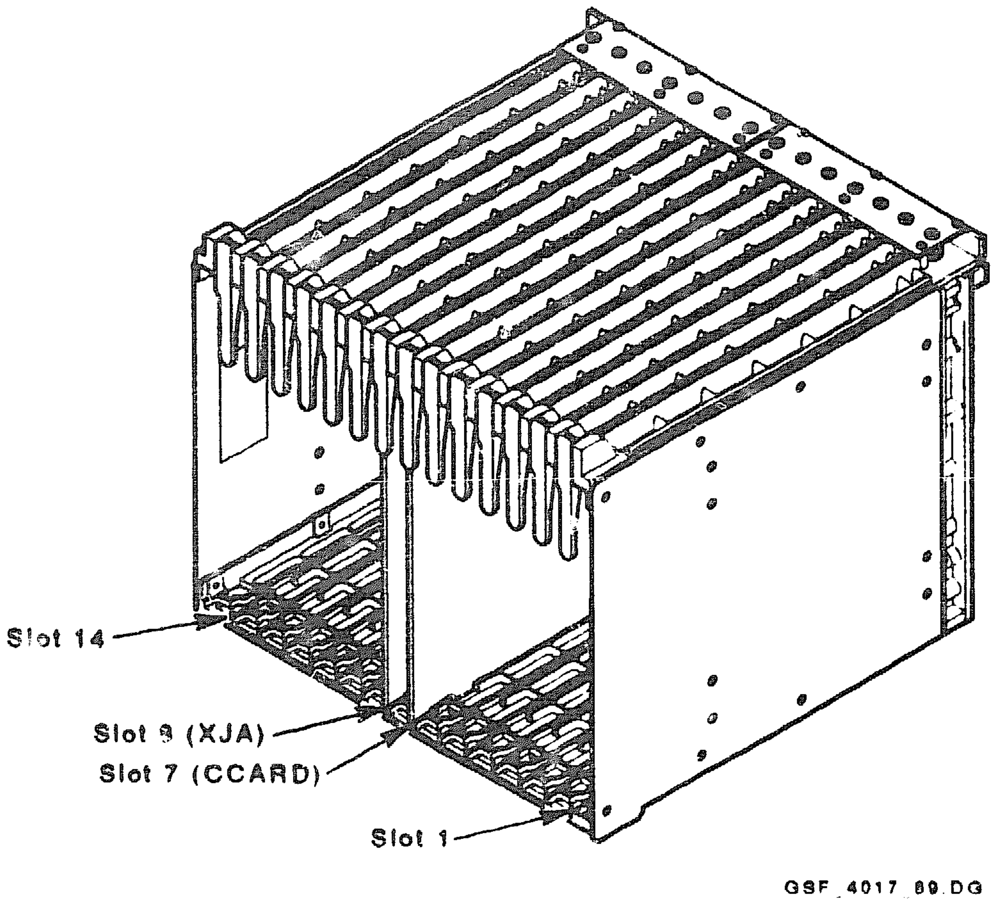


Figure 2–1 XMI Cardcage — VAX 9000 System Implementation

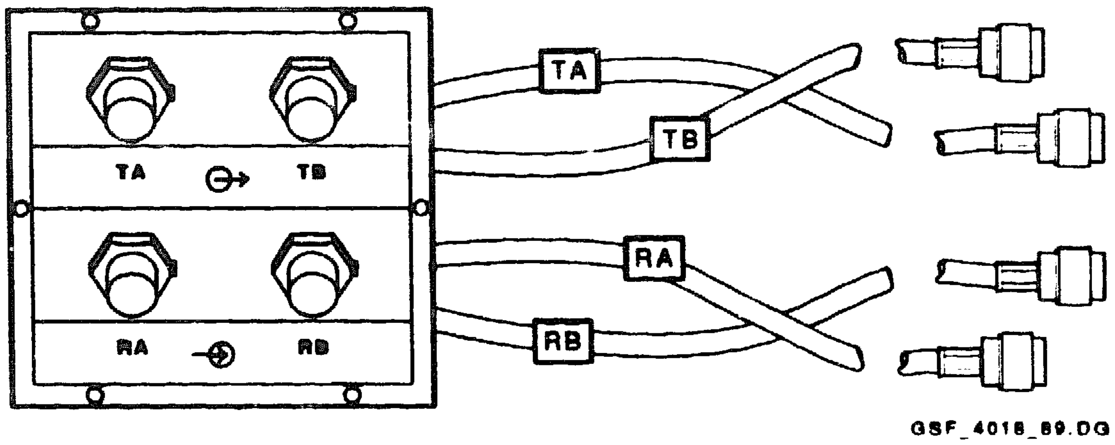


Figure 2–2 CIXCD-AA Bulkhead Cable

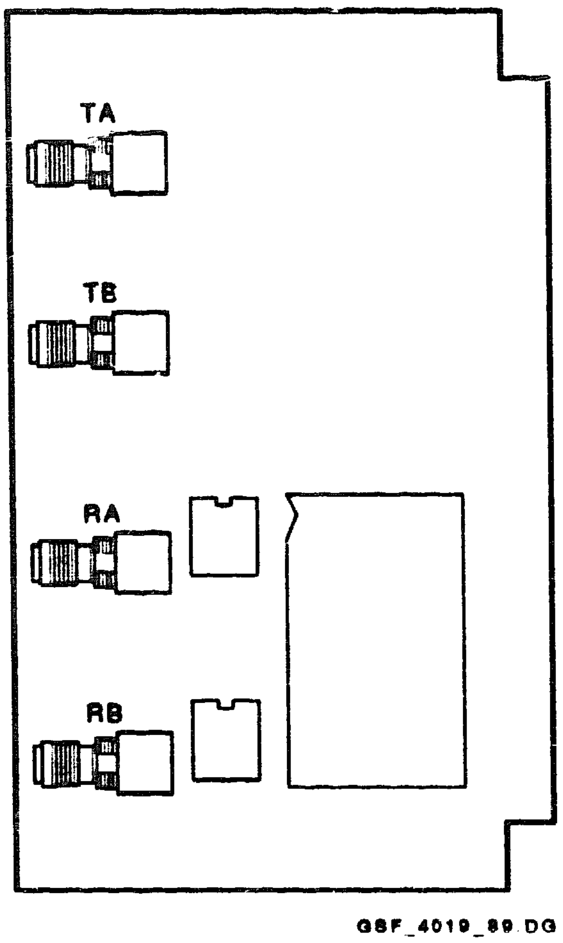


Figure 2–3 CIXCD Header Assembly

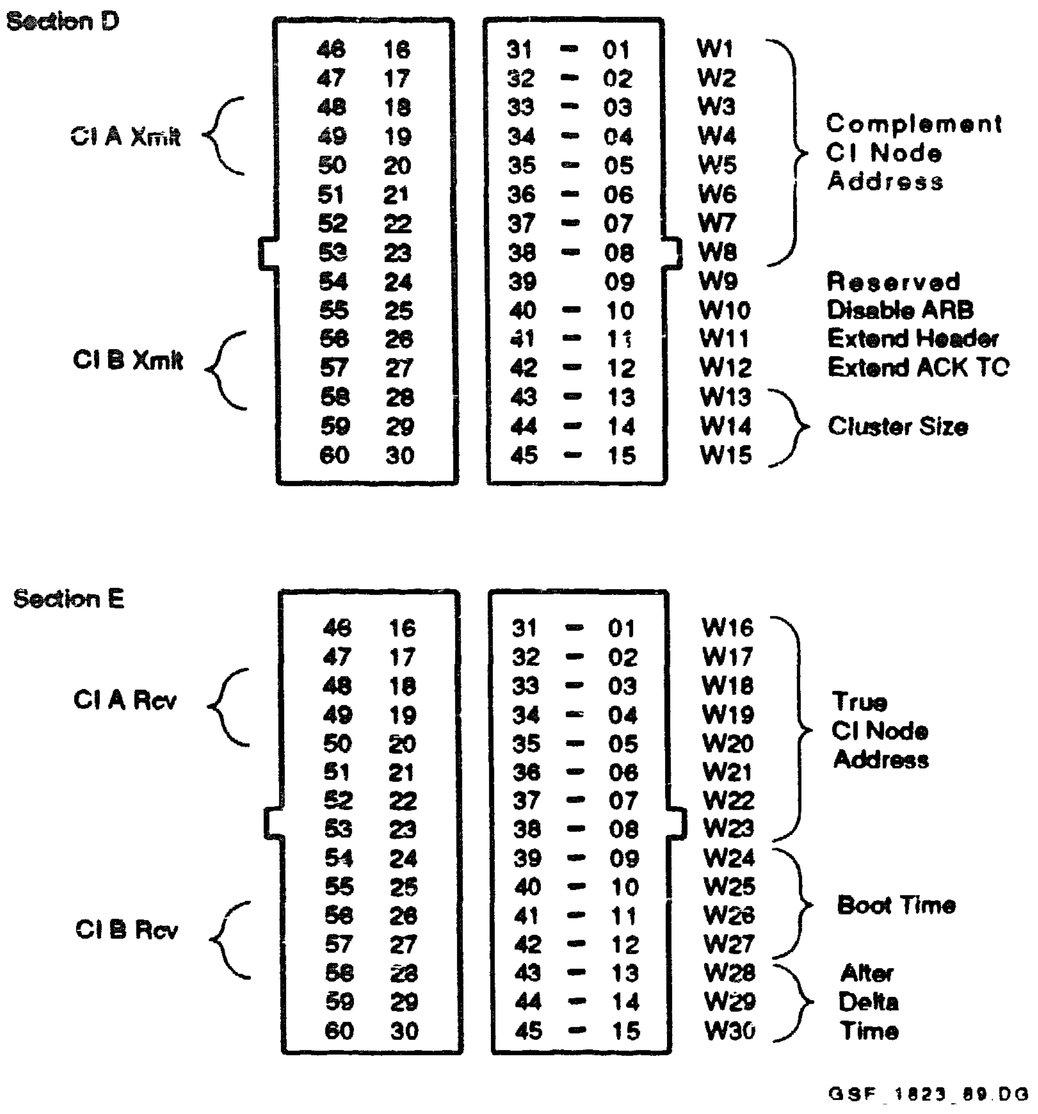


Figure 2-4 CIXCD Configuration Jumpers

Table 2–1 CI Node Address Jumpers

	True Address[1]							
CI Node	W16 E1/31	W17 E2/32	W18 E3/33	W19 E4/34	W20 E5/35	W21 E6/36	W22 E7/37	W23 E8/38
0	out	out	out	out	out	out	out	out
1	out	out	out	out	out	out	out	in
2	out	out	out	out	out	out	in	out
.				.				
.				.				
.				.				
223	in	in	out	in	in	in	in	in

	Complement Address[1]							
CI Node	W1 D1/31	W2 D2/32	W3 D3/33	W4 D4/34	W5 D5/35	W6 D6/36	W7 D7/37	W8 D8/38
0	out	out	out	out	out	out	out	out
1	out	out	out	out	out	out	out	in
2	out	out	out	out	out	out	in	out
.				.				
				.				
.				.				
223	in	in	out	in	in	in	in	in

[1]The true address and the complement address jumpers must be configured for the same CI node address. The node addresses are given in decimal. Addresses 224 through 255 are reserved for Digital.

Table 2–2 Boot Time Jumpers

Time[1]	W24 E9/39	W25 E10/40	W26 E11/41	W27 E12/42
1500	out	out	out	out
1400	out	out	out	in
1300	out	out	in	out
1200	out	out	in	in
1100	out	in	out	out
1000	out	in	out	in
0900	out	in	in	out
0800	out	in	in	in
0700	in	out	out	out
0600	in	out	out	in
0500	in	out	in	out
0400	in	out	in	in
0300	in	in	out	out
0200	in	in	out	in
0100	in	in	in	out
0000	in	in	in	in

[1]On CI ports which support maintenance states, the time, in seconds, that a port waits to exit the uninitialized state following power up. The boot time delay does not apply to the CIXCD since the CIXCD does not implement this feature.

Table 2–3 Disable Arbitration, Extend Header, and Extend ACK Timeout Jumpers

Jumper	In	Out
W10 D10/40	Disable normal CI arbitration[1]	Normal arbitration (default)
W11 D11/41	Extend header[2]	Normal header (default)
W12 D12/42	Extend ACK timeout[3]	Normal ACK timeout (default)

[1]Allows for initiating a transmit after waiting only one delta (quiet slot) time.

[2]Extends the number of bit sync characters in the header.

[3]Increases the timeout period for an ACK return.

Table 2–4 Alter Delta (Quiet Slot) Time Jumpers

Quiet Slot Count	W28 E13/43	W29 E14/44	W30 E15/45
7	out	out	out
10[1]	out	out	in
Reserved	out	in	out
Reserved	out	in	in
Reserved	in	out	out
Reserved	in	out	in
Reserved	in	in	out
Programmable	in	in	in

[1]The CIXCD requires a delta time of 10. If the jumpers are configured to any other value, they must be changed to reflect a delta time of 10.

Table 2–5 Cluster Size Jumpers

Size[1]	W13 D13/43	W14 D14/44	W15 D15/45
16	out	out	out
32	out	out	in
64	out	in	out
128	out	in	in
224	in	out	out
Reserved	in	out	in
Reserved	in	in	out
Reserved	in	in	in

[1]Cluster size is given in decimal. Value indicates the maximum number of nodes supported by a port. The default is 16.

CHAPTER 3

3 CIXCD SELF-TEST (XCDST) AND ROM-BASED DIAGNOSTICS (RBDs)

3.1 CIXCD SELF-TEST (XCDST)

The CIXCD self-test automatically runs on system power up or on an XMI reset. The XCDST can also be run as RBD 0 from the RBD user interface. The XCDST program is stored in the EEPROM and requires that the port microprocessor be operational.

Table 3–1 XCDST Indications After Power Up or XMI Reset

Result	Indication(s)
Pass	Yellow self-test passed (STP), LED illuminated.
Fail	LED extinguished. Self-test failed bit (STF, bit 10) in XMI bus error register (XBER) set. Error code (failing test number) written to port diagnostic control/status register (PDCSR).

NOTE
See Examples 3–3 and 3–4 for sample outputs of an XCDST run from the RBD interface.

Table 3–2 XCDST Diagnostics

Test	Title[1]
1	Port processor ALU status and branch
2	ALU arithmetic/logical functions
3	General purpose registers
4	Microsequencer stack
5	Internal bus loopback
6	Interval timer
7	Local store
8	Memory control and wire interface
9	Data mover A
10	Data mover B
11	XMI commander
12	XMI responder
13	Data mover loopback
14	XMI bus error register
15	XMI device register
16	XMI failing address registers
17	Port processor internal conditions
18	MCWI error detection logic
19	XMOV error detection logic
20	Interrupt control registers
21	CI internal maintenance loopback

[1]The XCDST is subject to change with new releases of the CIXCD. Refer to the XCDST listings for test numbers and titles applicable to a given CIXCD revision.

3.2 ROM-BASED DIAGNOSTICS (RBDs)

The RBDs provide more extensive testing of selected CIXCD logic functions. The RBDs are stored in the EEPROM and are accessed by invoking the RBD user interface.

Table 3–3 CIXCD RBD Sample Tests List

RBD	Title[1]
0	Power up self-test (XCDST)
1	CI internal/external maintenance loopback
2	Port local store exerciser
3	Port packet buffer exerciser
4	XMI commander exerciser
5	XMI communications register (XCOMM) exerciser
6	CIXCD soft register exerciser

[1]The RBDs are subject to change with new releases of the CIXCD. Refer to the RBD listings for test numbers and titles applicable to a given CIXCD revision.

3.2.1 RBD User Interface

The RBD user interface communicates with the host console through the XMI communications register (XCOMM). The interface is entered by issuing the console "Z" command, specifying the node to which the console is to be logically connected, followed by the TEST/RBD command (Example 3–1).

```
>>>
>>>Z C              ;Connect console to XMI node C
?43  Z connection successfully started
C>>TEST/RBD         ;Call RBD user interface
RBDC>               ;RBD prompt
```

a. VAX 6000 Systems

```
>>>
>>>Z 24             ;Connect console to XJA 2, XMI node 4
[ Use ^P to exit Z-MODE ]
TEST/RBD            ;Call RBD user interface
RBD4>              ;RBD prompt
```

b. VAX 9000 Systems

Example 3-1 Invoking the RBD User Interface

3.2.2 RBD Commands and Control Keys

The RBD command parser supports the minimum subset of commands required by the XMI RBD specification. Commands may be entered in uppercase or lowercase. The bell character and a question mark are returned on incorrect syntax.

Table 3-4 RBD Commands

Command[1]	Function
STart n	Starts the specified diagnostic (n). (See Table 3-5 for START command qualifiers.)
QUit	Returns control to the CIXCD functional microcode. After QUIT is issued, the CIXCD is initialized. TEST/RBD must be issued to resume RBD execution.
Examine x	Examines contents at address x (hex).

[1]Uppercase, bold-face characters indicate the minimum acceptable abbreviation for the command.

Table 3–5 RBD START Command Qualifiers

Qualifier	Diagnostic Action
/LE	Loop on test where the first error occurred. Continue error reporting, if enabled. Press CTRL/C, CTRL/Z, or CTRL/Y to terminate the loop and print the error summary on console terminal.
/HE	Halt on error, report error, and execute the clean-up code. The default is continue on error.
/IE	Inhibit error reporting to console terminal. The default is to enable error reporting. /IE is commonly used in combination with /LE.
/TR	Trace (display) test number at start of each test. Disabled by default.
/BE	Output bell character to terminal on error. /BE is commonly used with /IE and /LE to loop on intermittent errors.
/P=n	Run *n* (decimal) passes of each test selected. The default is one pass. Specify n=0 for infinite passes (CTRL/C, CTRL/Z, or CTRL/Y to halt).
/T=n[:m]	Run one (/T=n) test or range of tests (/T=n:m). Specify test number(s) in decimal. The default is all tests.

Table 3–6 RBD Control Keys

Key	Mode	Function
CTRL/C	Running Parser	Stop diagnostic execution, execute the clean-up code Disregard previous input
CTRL/U	Running Parser	Ignored Same as CTRL/C
CTRL/Y	Running Parser	Stop diagnostic, do NOT execute the clean-up code Same as CTRL/U
CTRL/Z	Running Parser	Same as CTRL/C Same as QUIT command

3.2.3 RBD Error Report Formats

The CIXCD follows the XMI standard for RBD error reports, supporting three levels of error reporting.

Table 3-7 RBD Error Report Levels

Level	Type	Error Report Line Fields
1	Summary	Pass/fail indicator XMI node number CIXCD identifier Decimal pass count
2	Error class/device type	Error class — HE (hard error), FE (fatal error) Device under test Unit number (if applicable) Diagnostic test number
3	Error specific	Two-digit subtest number Expected data Actual data Failing address (if non-zero field) Unused (zero filled) Error PC

```
❶ ;  F            4        0C05   00000005
❷ ; HE          XCD          xx       T01
❸ ; 07   55555555   55555554   00000800   00000000   00000000
```

❶ Failed, XMI node 4, CIXCD (device type 0C05), 5th pass

❷ Hard error, CIXCD, xx (not used), test 1

❸ Subtest 7, expected 55555555 (hex), actual 55555554 (hex), failing address 800 (hex), all zeros field, error PC 0

Example 3-2 Sample RBD Error Report

3.2.4 Sample RBD Run

```
① >>> Z C
② ?43  Z connection successfully started
③ C>> TEST/RBD
④ RBDC>ST 0/TR/HE

⑤ ;XCD_ST  1.00

⑥ ; T01  T02  T03

⑦ ;  F            C       0C05   00000001
⑧ ; HE          XCD         xx        T03
⑨ ; 23     55555555  55545555  0000064C  00000000  00000000
⑩ RBDC> QUIT
  >>>
```

① Connect console to XMI node C

② Z connection message

③ Call RBD user interface (can abbreviate to T/R)

④ Start RBD 0 (XCDST), set TRACE, HALT ON ERROR flags

⑤ Test header line — test name, revision

⑥ Test tracing (test numbers displayed at start of test)

⑦ Fail indicator, node C, CIXCD (device type 0C05), 1st pass

⑧ Hard error, device is CIXCD, xx (not used), test 03

⑨ Subtest 23, expected 55555555, actual 55545555, failing address 64C, zeros field (not used), error PC 0

⑩ Exit RBD mode

Example 3–3 Sample RBD Run With Failure (VAX 6000 System)

```
❶ >>> Z 24
❷ [ Use ^P to exit Z-MODE ]
❸ TEST/RBD
❹ RBD4>ST 0/TR/HE

❺ ;XCD_ST   1.00

❻ ; T01  T02  T03

❼ ;  F            4        0C05   00000001
❽ ; HE          XCD         xx        T03
❾ ; 23     55555555  55545555  0000064C  00000000  00000000
❿ RBDC> QUIT
  >>>
```

❶ Connect console to XJA2, XMI 4

❷ Z connection message

❸ Call RBD user interface (can abbreviate to T/R)

❹ Start RBD 0 (XCDST), set TRACE, HALT ON ERROR flags

❺ Test header line — test name, revision

❻ Test tracing (test numbers displayed at start of test)

❼ Fail indicator, node 4, CIXCD (device type 0C05), 1st pass

❽ Hard error, device is CIXCD, xx (not used), test 03

❾ Subtest 23, expected 55555555, actual 55545555, failing address 64C, zeros field (not used), error PC 0

❿ Exit RBD mode

Example 3–4 Sample RBD Run With Failure (VAX 9000 System)

CHAPTER 4

4 CIXCD MACRODIAGNOSTICS AND SUPPORT PROGRAMS

4.1 INTRODUCTION

This chapter provides an overview of the macrodiagnostics and support programs available for the CIXCD. The chapter includes:

- Brief descriptions of the diagnostics
- Diagnostic set-up procedures
- Sample diagnostic runs
- Event flag descriptions

4.2 DIAGNOSTIC PROGRAMS

The CIXCD is supported by five macrodiagnostics and one utility program.

Table 4–1 CIXCD Macrodiagnostics

Name	Level	Description
EVGAA EVGAB	3	CI functional diagnostics, parts 1 and 2. Standard CI bus interface diagnostics upgraded to include the CIXCD. Diagnostics run with memory management on and provide isolation to the failing command. Both diagnostics require the CI loopback connectors and use the CIXCD functional microcode. Microcode is assumed to be loaded (from the EEPROM) unless event flag 1 is set (loads microcode from file CIXCD.BIN prior to diagnostic execution).
EVGAC	3	Cluster functional diagnostic Verifies local to remote node communication and data integrity. Must be run under VAX/DS on an inactive CI cluster. Functional microcode must be on the same medium as the diagnostic. Assumes successful runs of EVGAA and EVGAB.

Table 4–1 (Cont.) CIXCD Macrodiagnostics

Name	Level	Description
EVGEA	3	CIXCD repair level Performs extensive testing of the CIXCD at the functional level and at the logic level. Tests include: • Scan chain and data path to EEPROM and RAM • Data integrity and addressability of EEPROM and RAM • Verifying ability to invoke XCDST and read results • Computing checksum of EEPROM code • Control store read/write capability • Functional testing of RAM memory EVGEA also includes three sections of the EEPROM update/verification utility.
EVGEB	3	CIXCD microcode update utility Contains the code to initialize and update the functional and diagnostic microcode.
EVXCI	2R	CI cluster exercizer Provides for local CI interface functional testing and tests the ability of VAXcluster nodes to communicate over the CI bus.

4.3 RUNNING EVGAA AND EVGAB

Diagnostic Setup

1. Connect CI cables to loopback connectors (Figure 4–1):

   Connect transmit A to receive A and transmit B to receive B on the CI bulkhead connector panel using one attenuator pad (P/N 12-19907-01) and two modularity cables (P/N 70-18530-00) for each connection.

2. Ensure that the VAX diagnostic supervisor (VAX/DS) and diagnostics are accessible through the default load path (may require changing media in the current load-path device).

3. Load and run VAX/DS

   Refer to the applicable system installation guide for system specific VAX/DS load and run procedures.

4. Attach and select CIXCD:

   - Using the auto-sizer:

     ```
     DS> RUN EVSBA
     DS> SHOW DEVICES
     DS> SELECT PAA-
     ```

   - Using ATTACH and SELECT commands (Example 4–1)

5. Load the diagnostic program

6. Set the desired VAX/DS execution control flags (for example: TRACE, HALT) and any desired diagnostic event flags (note that the LOAD command clears event flags)

7. Start the diagnostic

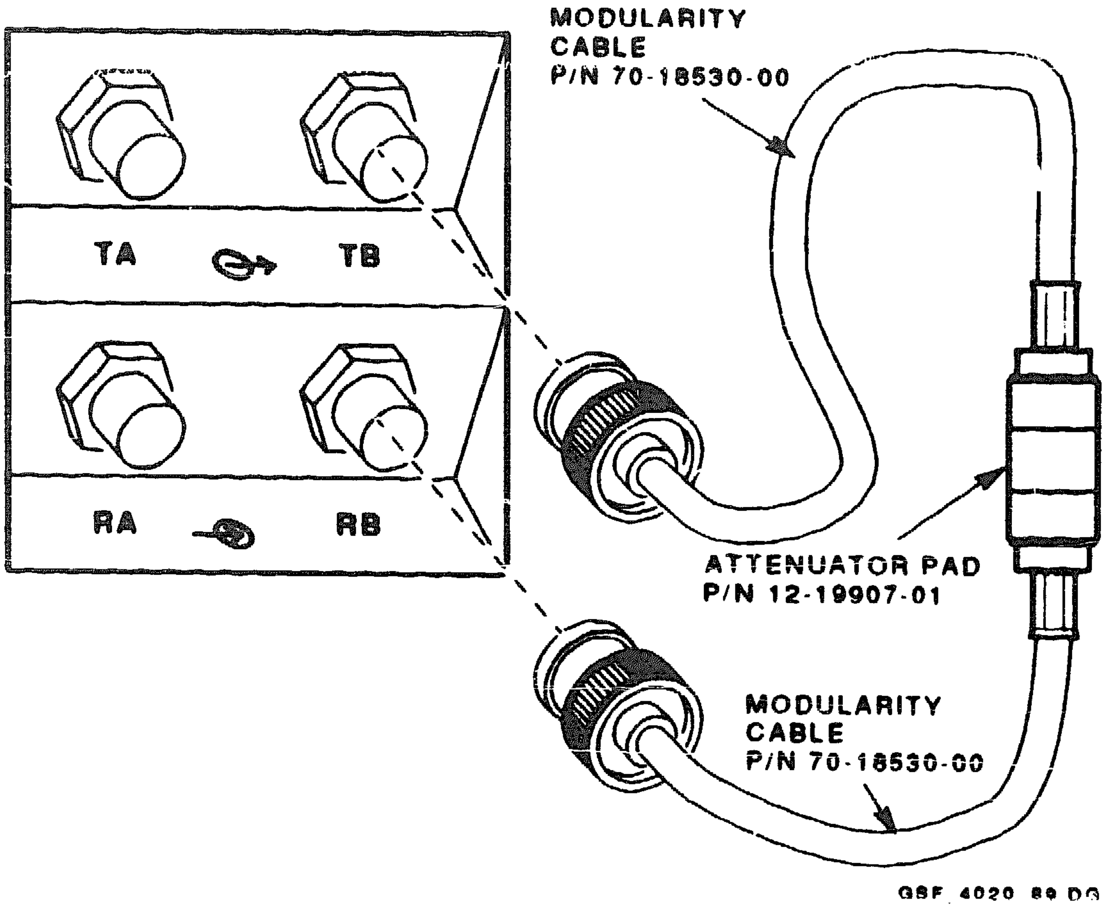


Figure 4–1 Diagnostic Loopback Cable Connections

```
DS> ATTACH CIXCD HUB PAA0 C 3    !C = XMI node number,
                                 !3 = CI node number
DS> SELECT PAA0
```

a. VAX 6000 Systems

```
DS> ATTACH XJA HUB XJA0 0 8        !0 = XMI number,
                                   !8 = XMI node number
DS> ATTACH CIXCD XJA0 PAA0 2 4 3   !2 = XMI node number,
                                   !4 = BR level,
                                   !3 = CI node number
DS> SELECT PAA0,XJA0
```

b. VAX 9000 Systems

Example 4-1 VAX/DS Attach and Select for CIXCD

4.3.1 Sample EVGAA Run

```
DS> LOAD EVGAA
DS> SET HALT,TRACE
DS> SET EVENT 1,2,3
DS> START

.. Program: EVGAA - CI Functional Part I. Level 3, revision 5.3, 17 tests
   at 11:58:12.80.

Testing _PAA0

Event Flag 1 SET - Load CI Microcode
Event Flag 2 SET - Print Queue Entries
Event Flag 3 SET - REQID Loop Function in Test 1

Testing Device _PAA0

EEprom Revision = xxxx   Functional Revision = yyyy

Test 1:  Cluster Configuration
Contents of the PORT PARAMETER REGISTER is:

PPR:[abcdefgh (X)]:
        CLUSTER_SIZE=XX,
        IBUF_LEN=XXXX(X),
        MBZ=0(X),
        DISABLE_ARB=0(X),
        EXTENDED_HEADER=0(X),
        SLOT_COUNT=7,
        PORT_NUMBER=06(X)

                Cluster Configuration for Path A
                ********************************
You CANNOT Differentiate between a CI780, CI750, or a CIBCI remotely.
(PS = Path Select,  TP = Transmit Path,  RP = Receive Path)

 Node    Device     Hard  Soft   Extended Port    Path     P T R
 Number   Type      Rev.  Rev.   Functionality   Status    S P P
 ------  --------  -----------  -------------   --------   - - -
   02    HSC50        022B       00000000(X)       OK      A A A
   06    CIXCD     xxxx  zzzz    xxxxxxxx(X)       OK      A A A

                Cluster Configuration for Path B
                ********************************
You CANNOT Differentiate between a CI780, CI750, or a CIBCI remotely.
(PS = Path Select,  TP = Transmit Path,  RP = Receive Path)

 Node    Device     Hard  Soft   Extended Port    Path     P T R
 Number   Type      Rev.  Rev.   Functionality   Status    S P P
 ------  --------  -----------  -------------   --------   - - -
   02    HSC50        022B       00000000(X)       OK      B B B
   06    CIXCD     xxxx  zzzz    xxxxxxxx(X)       OK      B B B

                 Nodes NOT Listed do not exist on Cluster
```

```
Test 2:  SETCKT with Various Masks and M_Values
Test 3:  SETCKT for Each Valid Port
Test 4:  SETCKT for Invalid Port
Test 5:  REQID Basic
Test 6:  REQID With 6 Packets on DGFQ
Test 7:  Datagram Discard
Test 8:  Response Queue Available Interrupt
Test 9:  Send Datagram
Test 10:  SNDMSG With No Virtual Circuit Open
Test 11:  Send Message Crossing Page Boundary
Test 12:  Message Length Test
Test 13:  Packet Size Violation
Test 14:  Send Loopback (SNDLB)
Test 15:  SNDLB Full Buffer on Path A
Test 16:  SNDLB Full Buffer on Path B
Test 17:  SNDLB Automatic Path Selection
.. End of run,  0 errors detected,  pass count is 1,
   time is 31-AUG-1989 11:58:36.40

DS>
```

Table 4-2 EVGAA Event Flags

Flag	Function If Flag Set
1	Load CI functional microcode (file CIXCD.BIN) prior to testing. (EVGAA assumes microcode to have been loaded from EEPROM unless event flag 1 is set.)
2	Display port queue entries prior to enabling the port and after execution of port commands.
3	Force test 1 to loop on only the specified node number and path. User prompted for node and path.

4.3.2 Sample EVGAB Run

```
DS> LOAD EVGAB
DS> SET HALT,TRACE
DS> SET EVENT 1,2
DS> START

.. Program: EVGAB - CI Functional Part II, Level 3, revision 5.3, 12 tests
   at 12:05:24.83

Testing _PAA0

Event Flag 1 SET - Load CI Microcode
Event Flag 2 SET - Print Queue Entries

Testing Device _PAA0

EEprom Revision = xxxx  Functional Revision = zzzz

Test 1:  Send Data with Offset Combinations
Test 2:  Request Data with Offset Combinations
Test 3:  Invalidate Translation Cache
Test 4:  SNDMDAT in Enabled/Maintenance State
Test 5:  SNDMDAT in Enabled State
Test 6:  REQMDAT in Enabled/Maintenance State
Test 7:  REQMDAT in Enabled State
Test 8:  Send RESET in Enabled State
Test 9:  Queue Protocol
Test 10:  Buffer Read Access
Test 11:  Buffer Write Access
Test 12:  Write to Global Buffer
.. End of run,  0 errors detected,  pass count is 1.
   time is 31-AUG-1989 12:05:47.34
DS>
```

Table 4-3 EVGAB Event Flags

Flag	Function
1	Load CI functional microcode (file CIXCD.BIN) prior to testing. (EVGAB assumes microcode to have been loaded from EEPROM unless event flag 1 is set.)
2	Display port queue entries prior to enabling port and after execution of port commands.

4.4 RUNNING EVGAC

Diagnostic Setup

Follow the same set-up procedures as for EVGAA and EVGAB, except connect the CI cables to the star coupler (refer to the *SC008 Star Coupler User's Guide*).

4.4.1 Sample EVGAC Run

```
DS> LOAD EVGAC
DS> SET HALT,TRACE
DS> SET EVENT 1,2,3
DS> START

.. Program: EVGAC - CI Functional Exerciser, revision 1.0, 8 tests
   at 11:42:21.26.

Testing _PAA0

Event Flag 1 Microcode Loading
Event Flag 2 Miscellaneous Status Messages
Event Flag 3 Datrec and Cnfrec

-*- Datchk Config VC_Disable Counters -*- PIC PDC MFQE RQA -*-

Program Parameter Register. > [(00000000), 00000000-000000FF (X)]
Use the Pattern File? > [(No), Yes]
Use the Parameter File? > [(No), Yes]
Modify Parameters? > [(No], Yes]

Test 1:  Local Configuration
Test 2:  Local Adapter Test
Test 3:  Datagram Test
Test 4:  Virtual Circuits Test
Test 5:  Message Test
Test 6:  Multiple Message Test
Test 7:  Write/Read Buffer Test
Test 8:  Activity Test
********  EVGAC - CI FUNCTIONAL EXERCISER - 1.1  ********
Pass 1, test 8, subtest 0, error 6, 31-AUG-1989 11:46:59.31
Soft error while testing PAA0: Buffered Data Error.

Port Number:      00000006
Offset:           00000E00
Expected:         305A3159
Received:         AAAAAAAA

********  End of Soft error number 6  ********

.. Halt on error at PC 0000A36B (X)
DS> cont
..Continuing from 0000A36B

.. End of run.  0 errors detected,  pass count is 1,
   time is 31-AUG-1989 11:47:17.98
DS>
```

Table 4–4 EVGAC Event Flags

Flag	Diagnostic Action If Flag Set
1	Load CI functional microcode (file CIXCD.BIN) prior to testing. (EVGAC assumes microcode to have been loaded from EEPROM unless event flag 1 is set.)
2	Display the following: • CI configuration (test 1) • Total number of usable pages in memory • Changes in virtual circuit state • Port to which traffic is being sent (tests 3:8)
3	Display confirmation received (CNFREC) and data received (DATREC) packets.

4.4.2 EVGAC Program Parameter Register

EVGAC includes a program parameter register which allows for tracing specific events and for enabling or disabling specific program routines. EVGAC prompts for input into the PPR, with the default being all bits clear. If the VAX/DS OPERATOR flag is cleared, EVGAC will not prompt for input, and will use the PPR with all bits clear. Note that if trace bits are set, interrupt-driven print routines may interfere with other common print routines.

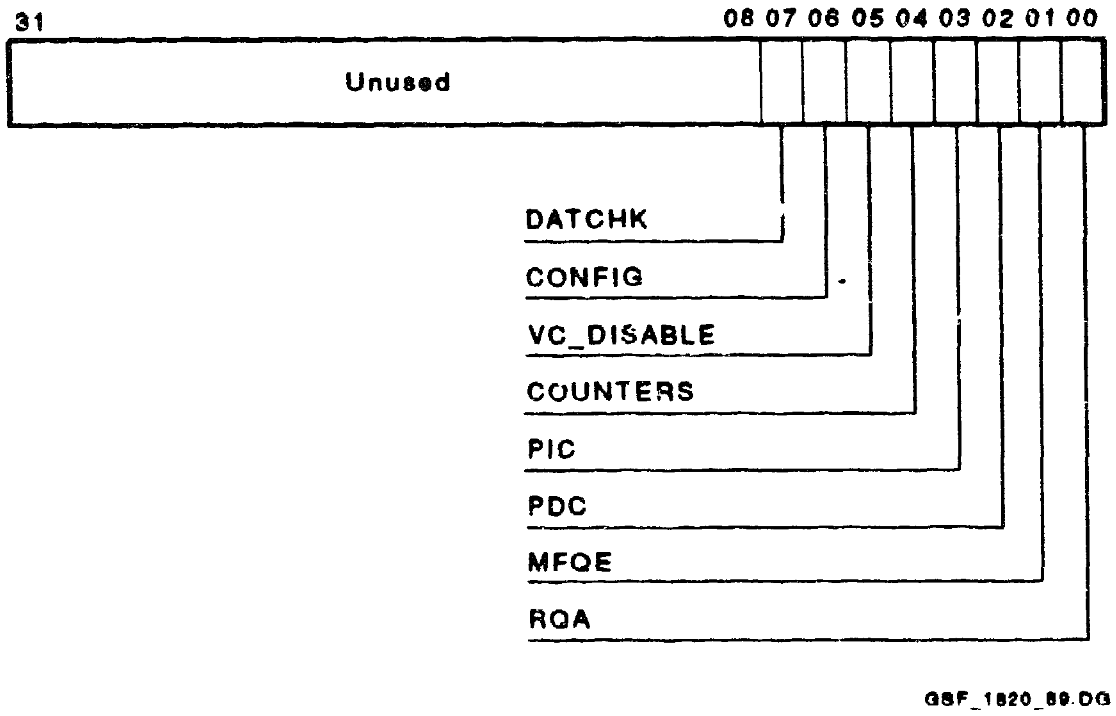


Figure 4-2 EVGAC Program Parameter Register

Table 4–5 EVGAC Program Parameter Register Bit Descriptions

Bit(s)	Name	Function
31:08	--	Unused
07	DATCHK	If set, disables data checking routines.
06	CONFIG	If set, disables running the configuration routine at the start of each test except test one (the configuration routine always runs in test one).
05	VC_ DISABLE	If clear, allows the program to re-establish virtual circuits between local and remote nodes when a virtual circuit is dropped. If set, inhibits re-establishing virtual circuits for that test pass.
04	COUNTERS	If set, reads and displays counters of the local adapter (specific to test 8).
03	PIC	If set, displays a message when the port initialization complete interrupt occurs.
02	PDC	If set, displays a message when the port disable complete interrupt occurs.
01	MFQE	If set, displays a message when the message free queue empty interrupt occurs.
00	RQA	If set, displays a message when the response queue available interrupt occurs.

4.4.3 EVGAC Program Parameters

The user can control EVGAC by modifying program parameters in one of three ways:

1. Setting program default values
2. Entering values by way of the console
3. Specifying a parameter file

Table 4–6 describes the EVGAC parameters and indicates the default value for each paramter.

Table 4–6 EVGAC Program Parameters

Parameter	Default[1]	Function
minport	CI port number of the CIXCD	Minimum port number to which the diagnostic will send test packets. The limit is the maximum cluster size found in port parameter register (PPR). Range: 0 to PPR
maxport	CI port number of the CIXCD	Maximum port number to which diagnostic will send test packets. The limit is the maximum cluster size found in the PPR. Range: minport to PPR
sanity	0	Sanity timer value. Range: 0:99.
maxcmd	47	Number of commands the program sends to each node per pass of activity test (test 8). Range: 0 to 100.
dgfq[2]	50	Number of datagram free queue entries. Range: 0:2048.

[1]All default values are in decimal.

[2]The creation of a less than acceptable number of datagram free queues will, in effect, inhibit the port from receiving necessary packets from remote ports. Try increasing the number if tests are failing due to nonreceipt of datagram type packets.

Table 4–6 (Cont.) EVGAC Program Parameters

Parameter	Default[1]	Function
msgfq	50	Initial number of message free queue entries. Range: 0:2048. This parameter can be considered dynamic; when the EVGAC receives an MFQE interrupt from the CIXCD, it tries to allocate five queue entries to the message free queue. EVGAC aborts if unsuccessful in allocating the buffers.
entrysize	Internal buffer length	Maximum datagram and message queue size. Used by EVGAC if less than internal buffer length in PPR. If greater than the internal buffer length, defaults to value in PPR.
nbuffmin[3]	512	Minimum size of named buffers. Range: 1:819200. Value may be dynamically changed if insufficient host memory is available.
nbuffmax[3]	13739	Maximum size of named buffers. Range: 1-819200. Value may be dynamically changed if insufficient host memory is available.
pm	PPR value	Packet multiple. PM value used if less than or equal to value calculated from PPR. PPR value used if greater then PM value.

[1]All default values are in decimal.

[3]Any dynamic changes will be displayed on the console.

4.4.4 EVGAC Support Files

EVGAC is supported by two ASCII files which the user can modify to pass parameters and patterns to the diagnostic. These files are created by the user and copied to the load media. Note that if the VAX/DS operator flag is clear, the parameter and pattern files are not used, no prompt is issued, and default values are used.

Table 4–7 EVGAC Support Files

File	Description
PARAMETER.PAR	Program parameters Allows for loading parameters from a file instead of using program default values, or requiring the user to input values from the console. Each file entry is eight characters long, representing one (hexadecimal) parameter (note that Table 4–6 listed defaults in decimal). Parameters must be placed in exact order (Table 4–8) and are used only if the value does not exceed the maximum value allowed for the parameter. Otherwise, default values are used (Table 4–6).
PATTERN.PTN	Program patterns Allows the user to specify text to be used in all message, datagram, and named buffer transfers. Must be created in specific format (Example 4–3). File entries are each eight characters long. The file may be any size greater than one 8-character line, up to 1024 bytes. File entries are read and stored in a data area. If an end-of-file condition is detected before the data area is full, EVGAC will close the pattern file and fill the rest of the data area with characters previously read. The pattern data area to be filled is 1024 bytes long. If the pattern file cannot be accessed, or the file is improperly formated, a message is generated and EVGAC uses a default pattern. The user is then prompted to either enter the parameters or use default values. If the parameter file contains an invalid parameter, the user is prompted to either use default values or abort the program.

Table 4–8 PARAMETER.PAR File Structure

Line	Parameter
1	minport
2	maxport
3	sanity
4	maxcmd
5	dgfq
6	msgfq
7	entrysize
8	nbuffmin
9	nbuffmax
10	pm

```
00000000
00000010
00000000
0000005D
00000064
00000064
000003F8
00000200
00007C1B
00000000
```

Example 4–2 Sample PARAMETER.PAR File

```
!1!1!1!1
2@2@2@2@
#3#3#3#3
4$4$4$4$
%5%5%5%5
6^6^6^6^
&7&7&7&7
8*8*8*8*
(9(9(9(9
0)0)0)0)
_-_-_-_-
=+=+=+=+
QqQqQqQq
wWwWwWwW
EeZeEeEe
rRrRrRrR
TtTtTtTt
yYyYyYyY
UuUuUuUu
iIiIiIiI
OoOoOoOo
pPpPpPpP
{[{[{[{[
]}]}]}]}
AaAaAaAa
sSsSsSsS
DdDdDdDd
fFfFfFfF
GgGgGgGg
hHhHhHhH
JjJjJjJj
kKkKkKkK
LlLlLlLl
;:;:;:;:
"'"'"'"'
\|\|\|\|
><><><><
zZzZzZzZ
XxXxXxXx
cCcCcCcC
VvVvVvVv
bBbBbBbB
NnNnNnNn
mMmMmMmM
```

Example 4–3 Sample PATTERN.PTN File

4.5 RUNNING EVGEA

Diagnostic Setup

Follow the same set-up procedures as for EVGAA and EVGAB, except connect the CI cables to the star coupler (refer to the *SC008 Star Coupler User's Guide*).

4.5.1 Sample EVGEA Run

```
DS> START/PASS:1

.. Program: CIXCD Functional Diag  -  ZZ-EVGEA, revision 1.0, 10 tests,
   at 14:23:06.45

Testing at _PAA0

Test -- 1  Scan Data Path Verification
      Subtest 1:  Port Scan Data Register Loopback
      Subtest 2:  Port Scan Shift Register
      Subtest 3:  PMCS EEPROM Data Path
      Subtest 4:  PMCS RAM Data Path
      Subtest 5:  Scan Visibility Bus Control Store Address
      Subtest 6:  Scan Visibility Bus GPR[0] Data Field
      Subtest 7:  Scan Visibility Bus Top of Stack and Stack
      Subtest 8:  Scan Visibility Bus Top of Stack and Stack
      Subtest 9:  Scan Visibility Bus Control Store Parity
      Subtest 10: Scan Visibility Bus X Register Parity Error
      Subtest 11: Scan Visibility Bus Y Register Parity Error
      Subtest 12: Scan Visibility Bus Internal Bus Parity
      Subtest 13: Scan Visibility Bus XMOV or MCWI Error
      Subtest 14: Scan Visibility Bus Micro Status Register

.. End of run,  0 errors detected, pass count is 1
   time is 18-SEP-1989 14:23:17.32
DS>
```

4.5.2 Sample EVGEA Error Message

```
******  CIXCD Functional Diag  --  ZZ-EVGEA  --  1.0  ******

Pass 1,  Test 1,  Subtest 2,  error 1,  18-SEP-1989 14:36:47.83
Hard error while testing PAA0:  CIXCD Port Scan Shift Register Error!

 Address            Expected           Received           XOR
00000000 (X)       FFFFFC00 (X)       FFFFFB00 (X)       00000100 (X)

******   End of Hard Error number 1   ******
```

Table 4-9 EVGEA Program Sections

Section	Function
ALL	Runs all tests in the DEFAULT section.
DEFAULT	Tests CIXCD with functional and self-test microcode loaded in EEPROM. This section contains all of the CIXCD tests. It is run if /SECTION is omitted from the command line, or if /SECTION=ALL or /SECTION=DEFAULT is specified.
ERRORLOG	Examines ERRORLOG header information stored in EEPROM.
EXAM	Examines diagnostic control block and ERRORLOG data entry stored in EEPROM.
INIT_DCB	Used by manufacturing to initialize the diagnostic control block (DCB) and, optionally, clear the error history buffer. NOTE **The error history buffer may contain valuable information for future diagnosis of the CIXCD and should only be cleared if absolutely necessary.**
RVERIFY	Verifies the contents of the primary EEPROM region against the backup EEPROM region. No load media file is used.
REPLACE	Replaces the contents of the backup EEPROM regions with copy from primary regions. No load media file is used.
RESTORE	Restores functional and diagnostic microcode in primary EEPROM regions with copy from backup regions. No load media file is used.
MFG	Tests CIXCD without microcode loaded in EEPROM. NOTE **This section destroys the data in the EEPROM. The user must run INIT_DCB or BAR_DCB sections prior to executing the UPDATE section.**
RBD	Enter RBD mode to test CIXCD.
LOCK	Enables software data protection of microcode in EEPROM. Data protection supported only on hardware revision "E" or later.

Table 4–9 (Cont.) EVGEA Program Sections

Section	Function
UNLOCK	Disables software data protection of microcode in EEPROM. Data protection supported only on hardware revision "E" or later.
UPDATE	Loads EEPROM from a microcode binary file on load media. Microcode is loaded to both primary and backup EEPROM regions.
VERIFY	Verifies the contents of EEPROM against a microcode binary file on load media.

4.6 EEPROM UPDATE/VERIFICATION UTILITY

The EEPROM update and verification utility consists of an update program resident in EVGEB and three sections of EVGEA (RESTORE, INIT_DCB, and ERRORLOG; see Table 4–9).

EVGEB provides a means for loading or updating the firmware in the EEPROM (see Figure 4–3). A CRC is generated for both the functional and the diagnostic microcode and is stored in the DCB. Microcode is loaded to both the primary regions and the backup regions of the EEPROM. After each region is loaded, it is re-read and verified. For each CIXCD, EVGEB saves the error history stored in the EEPROM, runs the diagnostic self-test, and compares the results to those saved.

To run EVGEB, follow the same procedure as for EVGEA.

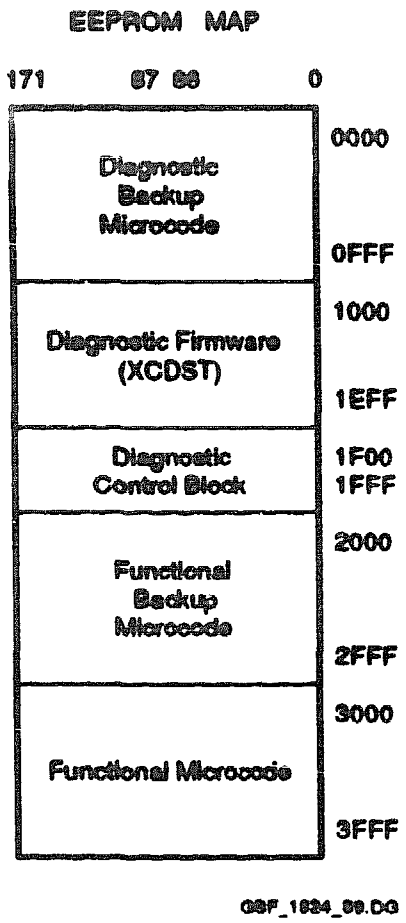


Figure 4–3 EEPROM Memory Map

4.7 MAINTENANCE SUPPORT TOOLS

Table 4–10 VAXcluster System Maintenance Tools

Tool	Function
EVXCI	Level 2R multipurpose cluster exerciser Provides local CI interface functional testing and a means to determine the ability of VAXcluster nodes to communicate over the CI bus.
ERF[1]	Errorlog report formatter Allows the user to create reports with system errors catalogued in various ways.
VAXsim[2]	VAX system intergrity monitor Monitors and filters errors as they are logged by the VMS operating system. Provides the user with a means for quickly identifing an option that has either failed or has degraded operationally.
SHOW CLUSTER[3]	Displays a wide variety of VAXcluster information related to system configuration and operation.
SET HOST/HSC[4]	Allows a terminal on a host system to effectively become an HSC50/70 terminal. The user may then issue any HSC50/70 commands and examine or control the HSC50/70 as if it were a terminal connected directly to one of the HSC50/70 terminal ports.

[1]See the *VAX/VMS Error Log Utility Reference Manual.*

[2]See the *VAX System Integrity Monitor Maunal.*

[3]See the *VAX/VMS Show Cluster Utility Manual.*

[4]See the *VAX/VMS DCL Dictionary* under SET HOST/HSC.

CHAPTER 5

5
CIXCD REGISTERS

5.1 INTRODUCTION

This chapter overviews the CIXCD register structure. Included in the chapter are:

- Lists of the CIXCD registers:
  - XMI and CI required
  - Port specific, XMI visible
  - Port specific, microcode visible only
- Register bit maps
- Descriptions of selected registers

The chapter is intended as a quick reference to register information. Refer to the *CIXCD User's Guide* and the *CIXCD Technical Manual* for detailed descriptions of all CIXCD registers.

5.2 CIXCD REGISTER TYPES

Table 5-1 CIXCD Register Types

Type	Description
Hardware	XMI and CI registers which must be present in the node. These registers reside in the XMOV gate array and are always available to the console and to the CIXCD port driver, regardless of the state of the CIXCD microcode. The hardware registers are also available to the CIXCD microprocessor over the port internal bus (IB).
Software	CIXCD specific registers which are visible over the XMI. The functional microcode must be operating to access these registers. (The XMI logic validates the register's XMI address; microcode performs the register read or write operation.)
Internal	Microcode support registers which the microcode uses to manage data transfers and to control communications between gate arrays. These registers are accessed over the IB and are available only to the functional microcode and to the self-test diagnostics. The hardware and software registers are also part of the internal register structure. However, the internal representation of these registers is not always the same as the external representation.

5.3 ADDRESSING XMI VISIBLE REGISTERS

Each XMI node is allocated a 512 Kbyte region in I/O space for node control and status registers. The address of a register in XMI node space is based on the XMI node ID and an assigned offset value.

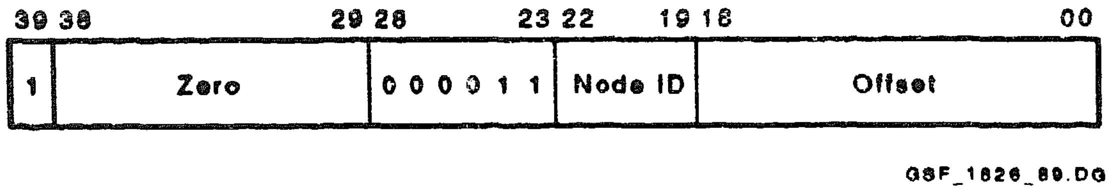


Figure 5–1 XMI Node Space Addressing

Bit(s)	Value	Description
39	1	Identifies the address as being in I/O space
38:29	0	These bits must be zero
28:23	03	XMI nodespace
22:19	n	XMI node ID (determined by the position of the node in the backplane)
18:00	aaaaa	Offset from base address

NOTE
The register addressing scheme shown in Figure 5–1 is as viewed from any other XMI node. The addresses may be different for processors which remap the XMI registers into their own physical I/O space to support multiple XMIs.

The CIXCD hardware and software registers each have a unique address on the XMI in the node space allocated to the CIXCD. The CIXCD XMI registers are listed in Table 5–2 with address offsets given in hexadecimal.

Table 5–2 CIXCD XMI Visible Registers

Mnemonic	Offset[1]	Name
Hardware Registers[2]		
XDEV	00000	XMI device register
XBER	00004	XMI bus error register
XFADR	00008	XMI failing address register
XCOMM	00010	XMI communications register
PSCR	00014	Port scan control register
PSDR	00018	Port scan data register
PMCSR	0001C	Port maintenance control/status register
PDCSR	00020	Port diagnostic control/status register
PSR	00024	Port status register
XFAER	0002C	XMI failing address extension register
Software Registers		
PQBBR	01000	Port queue block base register
PESR	01008	Port error status register
PFAR	0100C	Port failing address register
PPR	01010	Port parameter register
PSNR	01014	Port serial number register
PIDR	01018	Port interrupt destination register
PIVR	01020	Port interrupt vector register
–	01024	Reserved
PCQ0CR	01028	Port command queue 0 control register
PCQ1CR	0102C	Port command queue 1 control register
PCQ2CR	01030	Port command queue 2 control register
PCQ3CR	01034	Port command queue 3 control register

[1]Address offset (in hex) from the node's base address

[2]The term "XMI" in a hardware register name denotes an XMI architecture register. The term "Port" denotes a CI architecture register.

Table 5–2 (Cont.) CIXCD XMI Visible Registers

Mnemonic	Offset[1]	Name
Software Registers		
PSRCR	01038	Port status release control register
PECR	0103C	Port enable control register
PDCR	01040	Port disable control register
PICR	01044	Port initialize control register
PDFQCR	01048	Port datagram free queue control register
PMFQCR	0104C	Port message free queue control register
PMTCR	01050	Port maintenance/sanity timer control register
PMTECR	01054	Port maintenance/sanity timer expiration control register
PPER	01058	Port parameter extension register

[1]Address offset (in hex) from the node's base address

5.4 INTERNAL BUS (IB) REGISTER ADDRESSING

The port processor accesses registers on the port internal bus through the use of the six-bit literal field of the microword. Bits <05:04> of this field indicate the gate array and bits <03:00> indicate the register in the gate array.

NOTE
The internal registers are not covered in this document. Refer to the *CIXCD Technical Manual* for bit maps and descriptions of the internal registers.

Table 5-3 IB Accessible Register Locations

Addresses	Gate Array
00—1F	XMOV
20—2F	MCWI (CMEM)
30—3F	MCWI (CIC)
40—60	MCDP

Table 5–4 Internal Registers

Mnemonic	Addr	Register
XMOV Gate Array		
MVACSR	[00]	Mover A control and status register
MVABCR	[01]	Mover A byte count register
MVAADR	[02]	Mover A XMI address register
MVANPR	[03]	Mover A XMI next page register
MVBCSR	[04]	Mover B control and status register
MVBBCR	[05]	Mover B byte count register
MVBADR	[06]	Mover B XMI address register
MVBNPR	[07]	Mover B XMI next page register
JUMPENR	[08]	Port jumper register
PSR	[09]	Port status register
CMDRAAR	[0A]	Commander XMI address A register
CMDRABR	[0B]	Commander XMI address B register
CDAT1LR	[0C]	Commander XMI data 1 low register
CDAT1HR	[0D]	Commander XMI data 1 high register
CDAT2LR	[0E]	Commander XMI data 2 low register
CDAT2HR	[0F]	Commander XMI data 2 high register
RESPCSR	[10]	Responder control/status register
PSCR	[11]	Port scan control register
RESPDTR	[12]	Responder data register
XDEV	[13]	XMI device register
XBER	[14]	XMI bus error register
XFADR	[15]	XMI failing address register LW0
XFAER	[16]	XMI failing address register LW1
PDCSR	[17]	Port diagnostic control/status register
CMDRCSR	[18]	Commander control/status register

Table 5–4 (Cont.) Internal Registers

Mnemonic	Addr	Register
XMOV Gate Array		
PSCR	[19]	XMI communications register
PMCSR	[1A]	Port maintenance control/status register
IVIR	[1D]	Interrupt vector and IPL register
INTDMR	[1E]	Interrupt destination mask register
INTCR	[1F]	Interrupt control register
MCWI Gate Array		
MVAPBAR	[20]	Mover A packet buffer address register
MVBPBAR	[21]	Mover B packet buffer address register
PRTPBAR	[22]	Port packet buffer address register
MCERSR	[23]	MCWI error status register
ENLKCR	[30]	Enable link control register
DSLKCR	[31]	Disable link control register
LDCIR	[32]	Load configuration information register
XMABOR	[33]	Transmission abort register
STIMR	[34]	Slot time register
Z0HP	[35]	Zone zero head pointer
Z1HP	[36]	Zone one head pointer
Z0TP	[37]	Zone zero tail pointer
Z1TP	[38]	Zone one tail pointer
LDNDADR	[39]	Load true node address register
	[39]	Load complement node address register
XMITA	[3A]	Transmit register A

Table 5-4 (Cont.) Internal Registers

Mnemonic	Addr	Register
MCWI Gate Array		
XMITB	[3B]	Transmit register B
XMITSR	[3C]	Transmit status register
RCVSR	[3D]	Receive status register
MCWIDR	[3F]	MCWI diagnostic register
MCDP Gate Array		
MCDBR	[40]	MCDP diagnostic bit register
MSTATR	[41]	Micro status register
INTTR	[42]	Interval timer register
CLRICR	[43]	Clear internal conditions register
IER	[44]	Interrupt enable register
RDICR	[60]	Read internal conditions register

5.5 REGISTER DESCRIPTION CONVENTIONS

In the register description tables that follow, the access type of the bit(s) being described is denoted by the mnemonic enclosed in parentheses after the bit field name. The mnemonic indicates access to the bit by the port driver (software) and the CIXCD (hardware or microcode). The code for the port driver precedes the ":" character, and the code for the CIXCD follows. For example:

```
(RO:R/W)
 |  |
 |  +--- CIXCD has read/write capability
 +------ Port driver has read only capability
```

The register bit access codes are as follows:

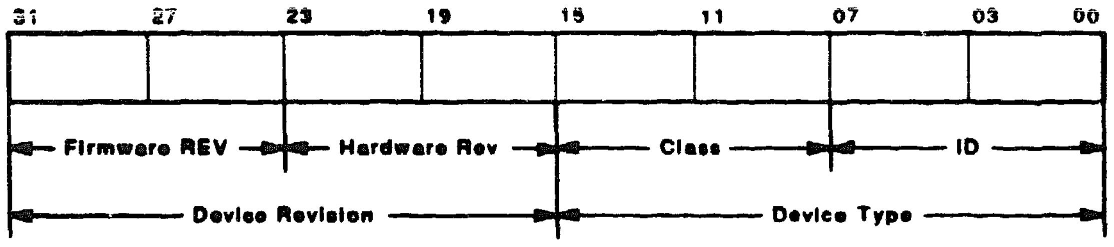
Code	Indication
DCLOC	Bit cleared following deassertion of DC LOW
DCLOS	Bit set following deassertion of DC LOW
MINC	Bit cleared when PMCSR MIN bit written with a "1"
MINS	Bit set when PMCSR MIN bit written with a "1"
OSL	Operating system must setup register or bit
R/W	Normal read/write
R/W1C	Read/write-1-to-clear. User interface cannot set bit.
RO	Read-only
ROZ	Read-only as Zero
SC	Special case, operation defined in detailed description
STC	Cleared following successful self-test, initiated on deassertion of DC LOW
STS	Set following successful self-test, initiated on deassertion of DC LOW
WO	Write-only

Bits desiginated as "zero" or as "0s" in register bit maps are read-only (RO) bits that always return "0". If the ":" character is absent, the code for the driver and the CIXCD are identical.

5.6 HARDWARE REGISTERS — XMI ARCHITECTURE

The following registers are XMI architecture registers which must be present in the node. These registers are always available to the console and to the CIXCD port driver, reguardless of the state of the CIXCD microcode.

5.6.1 XMI Device Register (XDEV, bb+00000)



Bit(s)	Name/Description
31:16	**Device revision (RO)** Identifies hardware and firmware revision of CIXCD. Loaded by port microprocessor at end of a successful seft-test. A zero value indicates an uninitialized node.
31:24	**Firmware revision (RO)** Indicates firmware revision of CIXCD.

Firmware Revision

Major	Minor	Description
000	00000	V0.0
001	00000	V1.0
001	00001	V1.1
...	...	...

Bit(s)	Name/Description
23:16	Hardware revision (RO)
	Indicates hardware revision of CIXCD.

Alpha field		Numeric field	
<23:20>	Code	<19:16>	Code
0000	—	0000	—
0001	A	0001	1
0010	B	0010	2
0011	C	0011	3
0100	D	0100	4
0101	E	0101	5
0110	F	0110	6
0111	H	0111	7
1000	J	1000	8
1001	K	1001	9
1010	L	1010	10
1011	M	1011	11
1100	N	1100	12
1101	P	1101	13
1110	R	1110	14
1111	S	1111	15

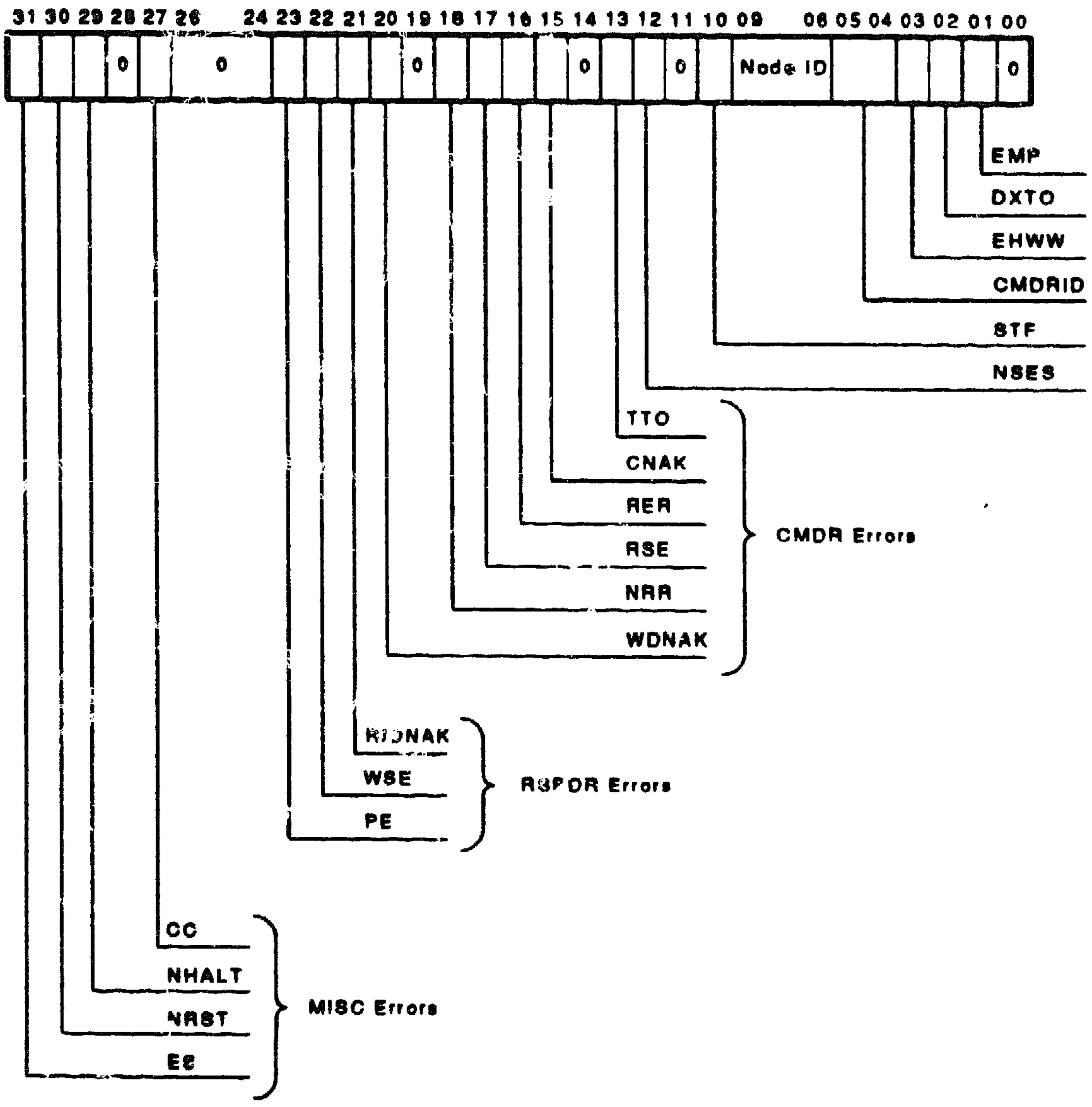
A value of zero in either field is invalid. Note that the letters "G", "I", "O", and "Q" are not used.

Bit(s)	Name/Description
15:00	Device type (RO) Identifies device type and XMI device ID of CIXCD. Loaded by port microprocessor at end of a successful self-test. A zero value indicates an uninitialized node. The DTYPE field is divided into two subfields:

Field	Bit Descriptions
Class	Indicates category in which node falls: <15>—CPU device <14>—Memory device <13>—Bus window (I/O) <12>—Bus window (memory) <11>—I/O device <10>—XCOMM register present
ID	Uniquely identifies particular device within a specified class.

The CIXCD device type is 0C05.

5.6.2 XMI Bus Error Register (XBER, bb+00004)



GSF_1738_89R.DG

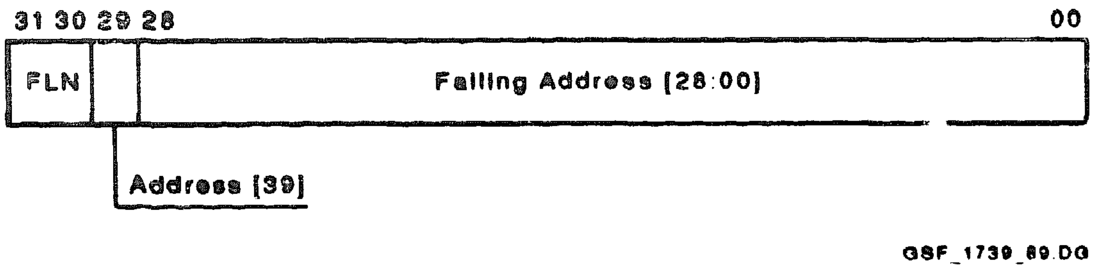
Bit(s)	Name/Description
Miscellaneous Errors	
31	Error summary (RO) Logical "OR" of bits: <27,23:20,18:15,13:12>. When ES is set, an XMI interrupt is generated, using IVIR and INTDMR for IPL, destination mask, and vector if PMCSR MIE is set.
30	Node reset (R/W:ROZ, DCLOC) Setting NRST initiates a power-up reset, similar in response to the assertion and deassertion of XMI DC LO L (see note below). On a NRST, the CIXCD executes the self-test, asserting XMI BAD until successful completion of the self-test. Other nodes are inhibited from accessing the CIXCD from the time NODE RESET is set until self-test completion (or the maximum self-test time is exceeded). In response to a power-up sequence caused by XMI DC LO L, NRST is reset. Following a NODE RESET sequence, NRST remains set. NOTE **While responding to NODE RESET, the CIXCD will not access remote nodes on the XMI. In response to a power-up sequence caused by XMI DC LO L, NRST is reset. Following a NODE RESET sequence, NRST remains set, indicating to the XMI processor that it should not attempt to perform the normal boot process.**
29	Node halt (R/W:ROZ, DCLOC) Setting NHALT forces the CIXCD into a "quiet state", retaining as much state as possible. When NHALT is set, CIXCD commander transactions are disabled; responder transactions complete normally.
28	Must be zero
27	Corrected confirmation (R/W1C:RO, DCLOC) Set by CIXCD when it detects a single-bit CNF error (single bit CNF errors are automatically corrected by the XCLOCK chip in the XMI corner). Also sets bit <31>.
26:24	Must be zero

Bit(s)	Name/Description
Responder Errors	
23	Parity error (R/W1C:RO, DCLOC) When set, indicates that CIXCD has detected a parity error on an XMI cycle. The cycle need not have been directed to the CIXCD. Also sets bit <31>.
22	Write sequence error (R/W1C:RO, DCLOC) When set, indicates that CIXCD aborted a write transaction due to a missing data cycle. Also sets bit <31>.
21	Read/ident data NOACK (R/W1C:RO, DCLOC) When set, indicates that a READ data cycle transmitted by CIXCD has received a NOACK confirmation. Also sets bit <31>.
Commander Errors	
20	Write data NOACK (R/W1C:RO, DCLOC) When set, indicates that a WRITE data cycle transmitted by CIXCD has received repeated NOACK confirmations for the duration of the timeout period. Upon receipt of a NOACK confirmation code on a write data cycle, CIXCD will retry entire transaction until it either completes successfully or a TTO (bit <13>) is encountered; in which case WDNAK is also set. Also sets bit <31>.
19	Must be zero
18	No read response (R/W1C:RO, DCLOC) When set, indicates that a READ or IDENT transaction initiated by CIXCD failed to receive all of its requested data within the timeout period. Also sets bits <31> and <13>.
17	Read sequence error (R/W1C:RO, DCLOC) When set, indicates that a READ transaction initiated by CIXCD received its read data out of sequence. The offending command/address is available in XFADR and XFAER. Also sets bit <31>.
16	Read error response (R/W1C:RO, DCLOC) When set, indicates that CIXCD has received a read error response. The offending command/address is available in XFADR and XFAER. Also sets bit <31>.

Bit(s)	**Name/Description**
Commander Errors	
15	Command NOACK (R/W1C:RO, DCLOC) When set, indicates that a command cycle transmitted by CIXCD has received repeated NOACK Confirmations for the duration of the timeout period. This can result from a reference to a non-existent memory location or a command cycle parity error. The CIXCD sets this bit when it repeatedly receives a NOACK confirmation for a given command/address which has been retried for the timeout period. Also sets bits <31> and <13>.
14	Must be zero
13	Transaction timeout (R/W1C:RO, DCLOC) When set, indicates that a transaction initiated by CIXCD has not completed within the timeout period. The offending command/address is available in XFADR and XFAER. TTO may be set along with bits <20>, <18>, or <15>. If none of these bits is set, the CIXCD either: 1. Failed to win bus arbitration within the timeout period 2. Attempted to execute an IREAD command but XMI lockout ~~remained asserted~~ for the timeout period Also sets bit <31>.
Node Specific Errors	
12	Node specific error summary (RO) Set when a node specific error condition has been detected. NSES is set when any of the error bits <30:21,18:04> is set in the PMCSR. The PMCSR error bit must be cleared to clear NSES. Also sets bit <31>.
11	Must be zero
10	Self-test fail (R/W1C:STS, WO) While set, STF indicates that CIXCD has not yet passed self-test. The port processor must clear STF when the CIXCD passes self-test.

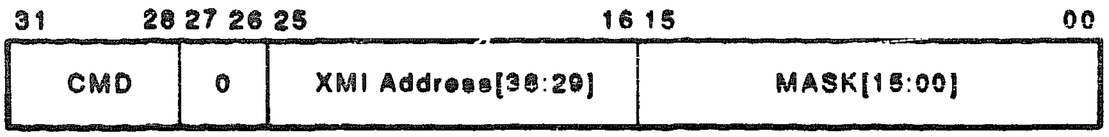
Bit(s)	Name/Description
Node Specific Errors	
09:06	Node ID [3:0] (RO) Represents the CIXCD's position in the XMI backplane and therefore its XMI node ID.
05:04	Commander ID [1:0] (RO:R/W, DCLOC) Logs the commander ID of a failing transaction. When a CMDR, MOVA, MOVB, or INTR XMI fatal error occurs, the microcode loads the code of the failing commander in this field: 0—Port transmit mover (mover A) 1—Port receiver mover (mover B) 2—Microcode CMDR 3—INTR
03	Enable hexword writes (R/W:R/W, DCLOC) Written by the host during initialization to enable mover B. If cleared, the maximum write data length is octaword.
02	Disable XMI timeout (R/W:ROZ, DCLOC) Controls reporting of all XMI timeouts by CIXCD. Setting DXTO disables the internal timeout counter, preventing any TTO errors. If the CIXCD has a current outstanding XMI transaction when DXTO transitions from 0 to 1 (TTO counters counting), the given timeout is disabled and the CIXCD will retry the transaction indefinitely. If the CIXCD has a current outstanding XMI transaction when DXTO transitions from 1 to 0 (TTO counters not counting), the given timeouts are continued from where they were prior to DXTO being set.
01	Enable more protocol (R/W:R/W, DCLOC) When set, enables XMOV's data movers to generate READ MORE and WRITE MORE transactions. MORE is only used on hexword transfers.
00	Zero

5.6.3 XMI Failing Address Register (XFADR, bb+00008)



Bit(s)	Name/Description
31:30	**Failing length** Logs the value of XMI D[31:30] during command cycle of a failed transaction.
29	**Failing address [39]** Logs the value of XMI D[29].
28:00	**Failing address [28:00]** Logs the value of XMI D[28:00].

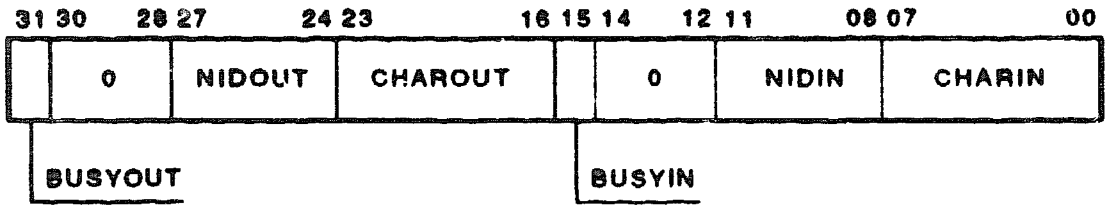
5.6.4 XMI Failing Address Extension Register (XFAER, bb+0002C)



GSF_1740_89.DG

Bit(s)	Name/Description
31:28	XMI failing command [03:00]
	Logs the command code of a failed transaction.
27:26	Zero
25:16	XMI address[38:29]
	Logs the value of XMI_D[57:48].
15:00	MASK [15:00]
	Logs the value of XMI_D[47:32].

5.6.5 XMI Communications Register (XCOMM, bb+00010)



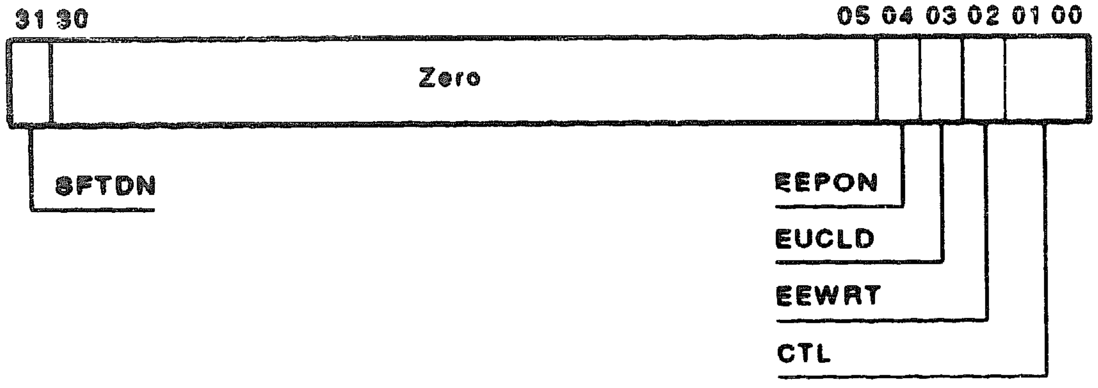
GSF_1741_89.DG

Bit(s)	Name/Description
31	Busy out (R/W)
	When set, indicates that the CHAROUT field contains a character that has not yet been read by the host CPU. BUSYOUT must be cleared by the host CPU before the CHAROUT field is available for another character.
30:28	Zero
27:24	Node ID out (RO:WO)
	Contains the XMI node ID of the slot in which the CIXCD is plugged.
23:16	Character out (RO:WO)
	Contains the message being sent from the local XMI node (this node) to the host processor.
15	Busy in (R/W)
	When set, indicates that the CHARIN field contains a character that has not yet been read by the local XMI node (this node). BUSYIN must be cleared by this node before the CHARIN field is available for another character.
14:12	Zero
11:08	Node ID in (WO:RO)
	Contains the XMI node ID of the remote XMI node (host CPU) that put a character in the CHARIN field.
07:00	Character in (WO:RO)
	Contains the console command character or console message being sent from the remote XMI node (host CPU) to the local XMI node (this node).

5.7 HARDWARE REGISTERS — CI ARCHITECTURE

The following registers must be present in the node to meet CI Port architecture requirements. These registers are always available to the console and to the CIXCD port driver, regardless of the state of the CIXCD microcode.

5.7.1 Port Scan Control Register (PSCR, bb+00014)



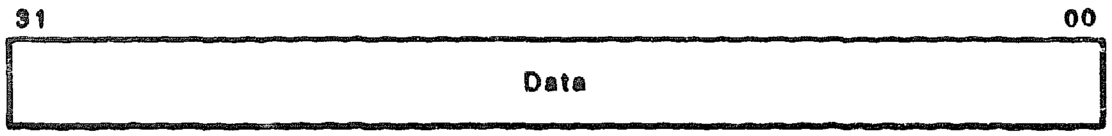
GSF_1742_89.DG

Bit(s)	Name/Description
31	**Shift done (RO:DCLOS, ROZ)** When set, indicates that scan shift has completed. Cleared by writing PSCR bits [4:0].
30:05	**Zero**
04	**EE power on (WO:R/W, DCLOS)** When set, activates chip enable to EEPROM. EEPON is set to enable loading and reading of EEPROMs, and cleared to disable EEPROMs when not in use. On power-up or node reset, EEPON is set allowing self-test microcode to be copied into RAM. After self-test, the functional microcode is loaded into RAM and EEPON is cleared. To load new microcode, the host must set EEPON to enable the EEPROMs before scanning in new data.
03	**External microcode load (WO:DCLOC, ROZ)** When set, enables designated external pins to be used as the scan path.

Bit(s)	Name/Description
02	Write EEPROM (WO:DCLOC, ROZ) Set to activate write pulse to EEPROM.
01	Scan control [1:0] (WO:DCLOC, ROZ) Control bits for scan logic:

01	00	Diagnostic Shift Register	Diagnostic Control Register
0	0	Hold	Hold
0	1	Hold	Load
1	0	Shift	Hold
1	1	Load	Hold

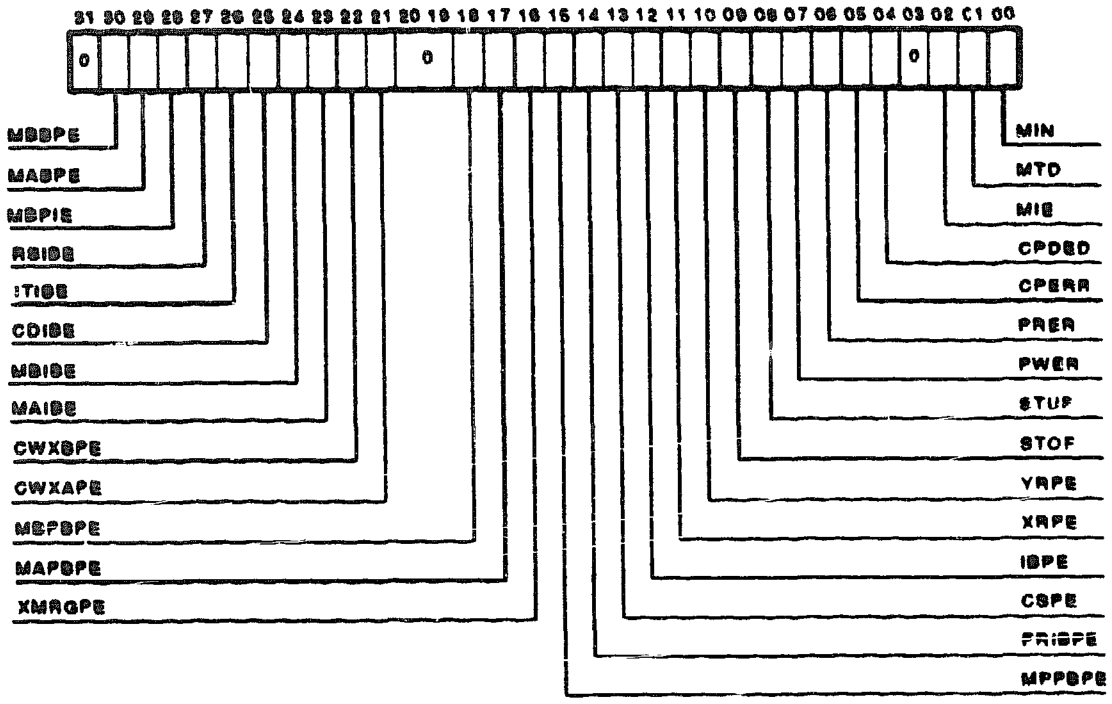
5.7.2 Port Scan Data Register (PSDR, bb+00018)



GSF_1743_89.DG

Bit(s)	Name/Description
31:00	**Scan data (R/W)** On a scan path read, contains data scanned out of the scan path. On a scan path write, loaded with data to be scanned onto the scan path. Port processor cannot access this register.

5.7.3 Port Maintenance Control/Status Register (PMCSR, bb+0001C)



GSF_1744_89.DG

Bit(s)	Name/Description
31	Zero
XMOV Parity Errors	
30	Mover B byte parity error (R/W1C:R/W, DCLOC) May be set by microcode if repeated mover B aborts occur with indication that there is a mismatch between byte parity stored in the mover's register file and parity generated on quadword to be sent to XMI. Allows microcode option of retrying packet before reporting error to XMI. Also sets XBER <12>.
29	Mover A byte parity error (R/W1C:R/W, DCLOC) Same as MBBPE except for mover A.
28	MOVB detected PB_IB parity error on PB read (R/W1C:R/W, DCLOC) May be set by microcode if repeated mover B aborts occur due to parity errors on the PB_OR_IB bus while attempting to read the packet buffer. Allows the microcode option of retrying the packet before reporting the error to XMI. Also sets XBER <12>.
27	Responder PB_IB parity error on register write (R/W1C:R/W, DCLOC) When set, indicates a parity error on the PB_OR_IB bus on register write to responder. Also sets XBER <12>.
26	Interrupt PB_IB parity error on register write (R/W1C:R/W, DCLOC) When set, indicates a parity error on the PB_OR_IB bus on register write to interrupt controller. Also sets XBER <12>.
25	Commander PB_IB parity error on register write (R/W1C:R/W, DCLOC) When set, indicates a parity error on the PB_OR_IB bus on register write to commander. Also sets XBER <12>.

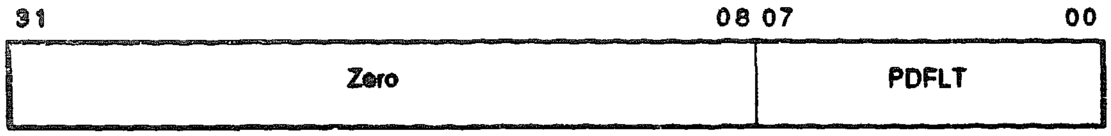
Bit(s)	Name/Description
24	MOVB PB_IB parity error on register write (R/W1C:R/W, DCLOC) When set, indicates a parity error on the PB_OR_IB bus on register write to mover B. Also sets XBER <12>.
23	MOVA PB_IB parity error on register write (R/W1C:R/W, DCLOC) When set, indicates a parity error on the PB_OR_IB bus on register write to mover A. Also sets XBER <12>.
MCWI Parity Errors	
22	CWIN transmit path B parity error (R/W1C:R/W, DCLOC) Set if CWIN logic, during transmit function on path B, detects bad parity from transmit data path between MCWI and CI CORNER logic or from conversion of longword packet buffer data to transmit data in MCWI. Also sets XBER <12>.
21	CWIN transmit path A parity error (R/W1C:R/W, DCLOC) Same as CWXBPE except for path A.
20:19	Zero
18	Mover B packet buffer read parity error (R/W1C:R/W, DCLOC) When set, indicates bad parity on mover B packet buffer read data sent from packet buffer RAMs over the MCWI_PB data bus (packet buffer memory bus) to the memory controller.
17	Mover A packet buffer write parity error (R/W1C:R/W, DCLOC) When set, indicates bad parity on mover A packet buffer write data sent by the XMOV gate arrays over the PB_OR_IB data bus to the memory controller.
16	XMOV register read parity error (R/W1C:R/W, DCLOC) When set, indicates bad parity on the XMOV register read data received by the memory controller from the XMOV gate array over the PB_OR_IB data bus. The destination of XMOV register read data is MCDP gate array.

Bit(s)	Name/Description
MCWI Parity Errors	
15	MCDP packet buffer read parity error (R/W1C:R/W, DCLOC) When set, indicates bad parity on the MCDP packet buffer read data received by the memory controller from the packet buffer RAMs over the MCWI_PB data bus (packet buffer memory bus).
14	PORT_IB receive parity error (R/W1C:R/W, DCLOC) When set, indicates bad parity on data received by the memory controller from the MCDP gate array over the port internal bus.
MCDP Parity Errors	
13	Control store parity error (R/W1C:R/W, DCLOC) Set if the port processor detects a control store parity error. CSPE can only be set if microcode can recover sufficently to write the bit. Bit must be written when error occurs. Also sets XBER <12>.
12	Internal bus parity error (R/W1C:R/W, DCLOC) Set if the port processor detects an internal bus parity error. Bit can only be set if microcode can recover sufficently to write the bit. Bit must be written when error occurs. Also sets XBER <12>.
11	X register parity error (R/W1C:R/W, DCLOC) Set if a parity error is detected in the X register of port processor data path. Bit can only be set if microcode can recover sufficently to write the bit. Bit must be written when error occurs. Also sets XBER <12>.
10	Y register parity error (R/W1C:R/W, DCLOC) Set if a parity error is detected in the Y register of port processor data path. Bit can only be set if microcode can recover sufficently to write the bit. Bit must be written when error occurs. Also sets XBER <12>.

Bit(s)	Name/Description
MCDP Parity Errors	
09	Micro stack overflow (R/W1C:R/W, DCLOC) Set on attempted push to full microstack. Bit can only be set if microcode can recover sufficently to write the bit. Bit must be written when error occurs. Also sets XBER <12>.
08	Micro stack underflow (R/W1C:R/W, DCLOC) Set on attempted pop from empty microstack. Bit can only be set if microcode can recover sufficently to write bit. Bit must be written when error occurs. Also sets XBER <12>.
Port Errors	
07	Port write error response (R/W1C:R/W, DCLOC) Set if microcode set INTCTR_SWEI (send write error interrupt). Microcode may set INTCTR_SWEI to force an IVINTR type interrupt if a register write is attempted to a non-existant register in CIXCD nodespace. The WEI causes a machine check; no additional interrupt is generated.
06	Port read error response (R/W1C:R/W, DCLOC) Set if microcode set RESPCSR_SNDRER (send read error response). Microcode may set RESPCSR_SNDRER if a register read is attempted from a non-existant register in CIXCD nodespace. The RER causes a machine check; no interrupt is generated.

Bit(s)	Name/Description
Port Errors	
05	CP error status (R/W1C:RO, DCLOC) Set if CP_ERROR_STATUS signal asserted for more than 32 cycles. CP_ERROR_STATUS is set when any error bit in MCDP the internal conditions register (IB register address 60) is set. Microcode traps and executes a port shutdown routine if any MCDP internal conditions error bit is set. The shutdown routine clears CP_ERROR_STATUS if the failure was intermittent. If CPERR is set, all other MCDP error bits in PMCSR are invalid (bits <13:08>), and CPERR is the only indication that the port processor has had an unrecoverable failure. Scan data may provide additional data. Also sets XBER <12>.
04	CPU no response error (R/W1C:RO, DCLOC) Set if the port processor fails to respond to a responder interrupt within 512 cycles. The port processor is assumed to have failed. Also sets XBER <12>.
03	Zero
Control Bits	
02	Maintenance interrupt enable (R/W:RO, DCLOC) When set, enables XMI interrupts.
01	Maintenance/sanity timer disable (R/W:RO, DCLOC) If set, the maintenance/sanity timer is set to its initial value and suspended. If clear, the timer functions normally.
00	Maintenance INIT (WO:RO, DCLOC) When set, clears all hardware state, including errors, and puts the port in the uninitialized state. Does not cause microcode to be copied from EEPROM to RAM or self-tests to be executed.

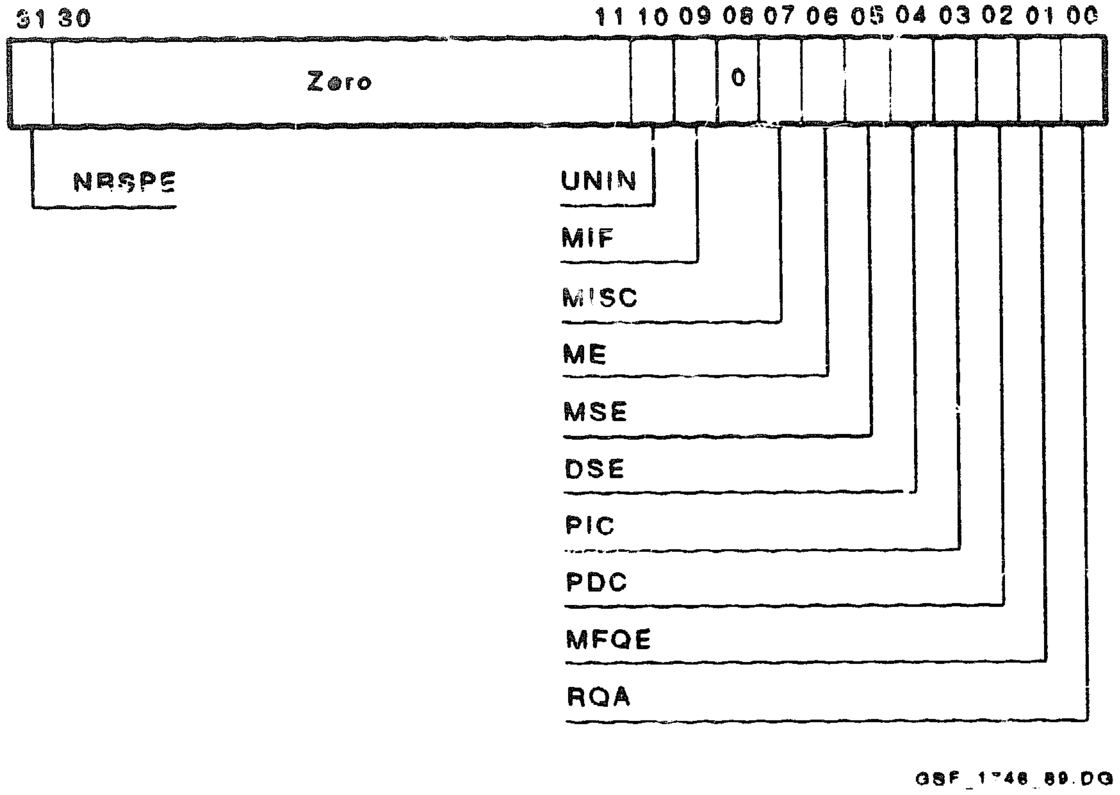
5.7.4 Port Diagnostic Control/Status Register (PDCSR, bb+00020)



GSF_1745_89.DG

Bit(s)	Name/Description
31:08	**Zero**
07:00	**Port diagnostic failing test number (RO)** Loaded with the self-test test number about to be executed. Makes the test number available to the host on XMI in case of self-test failure.

5.7.5 Port Status Register (PSR, bb+00024)



Bit(s)	Name/Description
31	No response error When set, indicates that one or more of the error bits <10,07:01> in the PSR are set. An interrupt is posted using status interrupt vector if PMCSR <02> is set.
30:11	Zero
10	Uninitialized (MINS) When set, forces the port into an uninitialized state. Port will not respond to CI traffic. Uninitialized state is exited by writing PICR or by a boot timeout. Bit is cleared on entering disabled state. Also sets bit <31>.

Bit(s)	Name/Description
09	Maintenance interrupt flag Set by microcode to indicate that an interrupt causing condition has occurred in the port. Allows a diagnostic program to operate the port with interrupts disabled. MIF indicates to the program that PSR is valid. Bit is cleared by write to PSRCR.
08	Zero
07	Miscellaneous error detected Set by microcode to indicate that port microcode has detected a miscellaneous error and has entered the disabled state. Actual error code is in PESR. Bit is reset on entering enabled state. Also sets bit <31>.
06	Maintenance/sanity timer expiration Set by microcode to indicate that the maintenance/sanity timer or boot timer has expired and the port has entered the uninitialized state with loss of processing state. Bit is cleared by microcode initialization. Also sets bit <31>.
05	Memory system error Set by microcode to indicate that the port has detected an XMI bus error (uncorrectable data or non-existant memory error) in referencing host memory. The port is in the disabled state. See PFAR for more information. Bit is cleared on entering enabled state. Also sets bit <31>.
04	Data structure error Set by microcode to indicate that the port encountered an error in a port data structure (for example, queue entry, PQB, BDT, page table, values out of range, or MBZ bits that are not zero). Port is placed in disabled state. More information about the error is contained in PESR and PFAR. Bit cleared on entering enabled state. Also sets bit <31>.

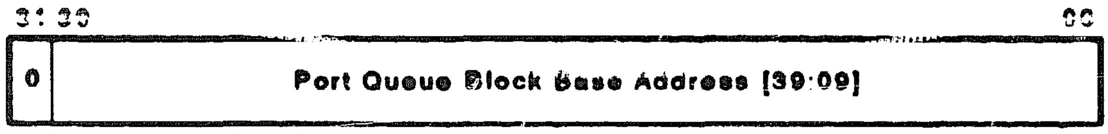
Bit(s)	Name/Description
03	**Port initialization complete** Set by microcode to indicate that the port has completed internal initialization. The port is in disabled state. Local store, virtual circuit descriptor table, and the port's internal data structures are initialized. Bit cleared on entering enabled state. Also sets bit <31>.
02	**Port disable complete** Set by microcode to indicate that the port entered disabled state from enabled state. Processing of command queues disabled and port will not respond to incoming CI transmissions. Bit cleared on exiting disabled state. Also sets bit <31>.
01	**Message free queue empty** Set by microcode to indicate that the port attempted to remove an entry from the message free queue and found the queue empty. Port processing of commands continues so that the message free queue may not be empty by the time interrupt service routine gains control. Bit cleared by write to PSRCR. Also sets bit <31>.
00	**Response queue available** Set by microcode to indicate that the port has inserted an entry on the response queue and the queue was previously empty. Bit cleared by write to PSRCR.

5.8 SOFTWARE REGISTERS

The following registers are CIXCD specific registers which are visible on the XMI. Access to these registers requires the CIXCD microcode to be operating. Any attempted access to a software register (or to local store address space) when the port is transitioning from the uninitialized state to the disabled state may result in an XMOV responder timeout and XMI error interrupt. The port driver will ignore this error, clear PMCSR CPDED, and delay XMI read access until the port has completed the transition (indicated by an interrupt and the setting of PSR_PIC in the port status register).

NOTE
This document only includes bit maps, bit names, and bit mnemonics for the software registers. Refer to the *CIXCD Technical Manual* for detailed descriptions of these registers.

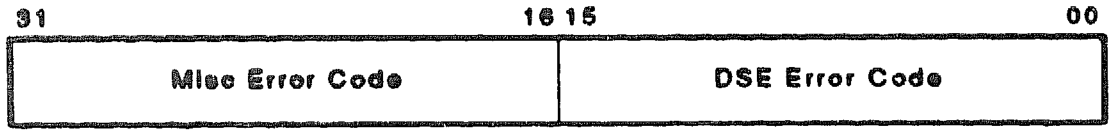
5.8.1 Port Queue Block Base Register (PQBBR, bb+01000)



GSF_1747_89.DG

Bit(s)	Name
31	Zero
30:00	Port queue block base address

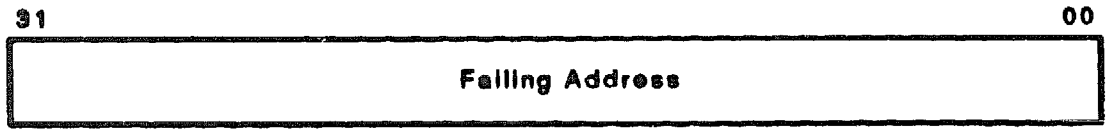
5.8.2 Port Error Status Register (PESR, bb+01008)



GSF_1748_89.DG

Bit(s)	Name
31:16	Miscellaneous error code (RO)
15:00	Data structure error code (RO)

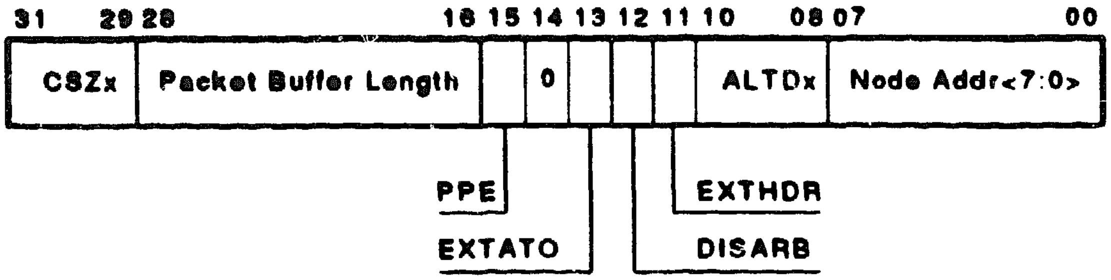
5.8.3 Port Failing Address Register (PFAR, bb+0100C)



GSF_1749_89.DG

Bit(s)	Name
31:00	Failing address (RO)

5.8.4 Port Parameter Register (PPR, bb+01010)



GSF_1750_69.DG

Bit(s)	Name
31:29	Cluster size <2:0> (RO)
	Indicates the maximum number of CI nodes supported by port:

CSZ			Cluster Size	Range
2	1	0	(decimal)	(decimal)
0	0	0	16	0-15
0	0	1	32	0-31
0	1	0	64	0-63
0	1	1	128	0-127
1	0	0	224	0-223
1	0	1	Reserved	
1	1	0	Reserved	
1	1	1	Reserved	

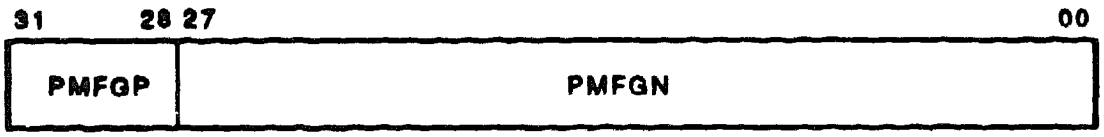
28:16	Packet buffer length (RO)
15	Port parameter extension (RO)
14	Zero
13	Extend ACK timeout (RO)

Bit(s)	Name
12	Disable arbitration (RO)
11	Extend header (RO)
10:08	Alter delta time <2:0> (RO) Indicates the specific quiet slot count the LINK hardware generates:

ALTD			
2	1	0	Quiet Slot Count (decimal)
0	0	0	7
0	0	1	10
0	1	0	Reserved
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Programmable

Bit(s)	Name
07:00	Port CI node number (RO)

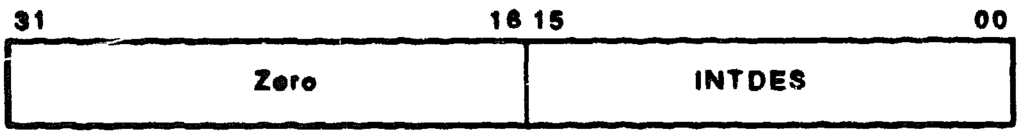
5.8.5 Port Serial Number Register (PSNR, bb+01014)



GSF_1751_89.DG

Bit(s)	Name
31:28	Port manufacturing plant
27:00	Port manufacturing number

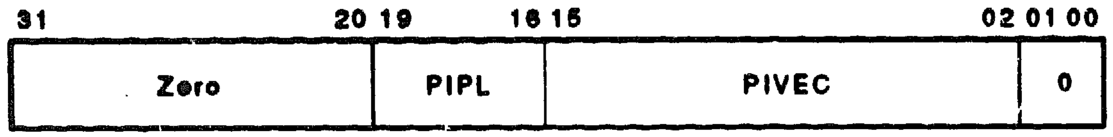
5.8.6 Port Interrupt Destination Register (PIDR, bb+01018)



GSF_1752_89.DG

Bit(s)	Name
31:16	Zero
15:00	Interrupt destination

5.8.7 Port Interrupt Vector Register (PIVR, bb+01020)



GSF_1753_89.DG

Bit(s)	Name
31:20	Zero
19:16	Port interrupt priority level
15:02	Port interrupt vector
01:00	Zero

5.8.8 PCQ0CR to PMTECR (bb+01028 to bb+01054)

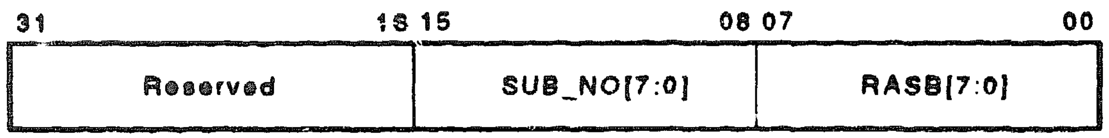
The registers listed in Table 5-5 are all write-only registers. When any of these registers is addressed for write access, the write transaction itself causes the operation to be performed; the write data are ignored. Reading any of these registers returns undefined data.

Refer to the *CIXCD Technical Manual* for descriptions of these registers.

Table 5–5 PCQ0CR (bb+01028) to PMTECR (bb+01054)

Mnem.	Offset	Name/Function When Written
PCQ0CR	01028	Port command queue 0 control register Initiate processing of entry in command queue 0.
PCQ1CR	0102C	Port command queue 1 control register Initiate processing of entry in command queue 1.
PCQ2CR	01030	Port command queue 2 control register Initiate processing of entry in command queue 2.
PCQ3CR	01034	Port command queue 3 control register Initiate processing of entry in command queue 3.
PSRCR	01038	Port status release control register Release lock on PSR (bb+00024) after interrupt service.
PECR	0103C	Port enable control register Place port in enabled state.
PDCR	01040	Port disable control register Place port in disabled state and generate interrupt request with PDC bit of PSR.
PICR	01044	Port initialize control register Initialize port and enter disabled state. Generate interrupt with PIC bit of PSR. If PICR written with port in enabled state, port will enter disabled state with loss of processing state.
PDFQCR	01048	Port datagram free queue control register Written whenever datagram free queue is empty at the time of a datagram free queue insertion.
PMFQCR	0104C	Port message free queue control register Written whenever message free queue is empty at the time of message free queue insertion.
PMTCR	01050	Port maintenance/sanity timer control register Forces a maintenance/sanity timer expiration interrupt. PMTECR has no effect unless port is in enabled or disabled states, and only when the maintenance/sanity timer is enabled.
PMTEC	01054	Port maintenance/sanity timer expiration control register Reset boot and maintenance/sanity timers to their initial values. Allows port driver to control expiration times of timers.

5.8.9 Port Parameter Extension Register (PPER, bb+01058)



GSF_1766_89.DG

Bit(s)	Name
31:16	Reserved for future use
15:08	Subnode number [7:0]
07:00	Requested adapter state block length [7:0]

DEC LANcontroller 400 (DEMNA)

Document Title: DEC LANcontroller 400 (DEMNA) HANDBOOK

Order Number: EK-DEMNA-HB-001

This handbook is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The handbook can be ordered separately or as part of the set.

The *XMI Adapters Handbook Documentation Set* is a dynamic document which will be periodically updated as new XMI adapters are announced. The first release of the set includes the following handbooks:

Order Number	Title
EK-XMIOV-HB	XMI Bus Overview Handbook
EK-CIXCD-HB	CIXCD Handbook
EK-DEMNA-HB	DEC LANcontroller 400 (DEMNA) Handbook
EK-DWMBA-HB	DWMBA Handbook

This handbook and the document set are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

DEC LANcontroller 400 (DEMNA) Handbook

Order Number EK-DEMNA-HB-001

This document is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The document can be ordered separately or as part of the set. The *DEC LANcontroller 400 (DEMNA) Handbook* and the *XMI Adapters Handbook Documentation Set* are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

Revision/Update Information: Revision 1.0

Digital Equipment Corporation

This document was prepared and published by Educational Services Development and Publishing, Digital Equipment Corporation.

Contents

6 DEMNA Adapter Registers

7 DEMNA Sequencing Flows

8 DEMNA Error Handling

Examples

Figures

About This Manual

Intended Audience

This handbook is part of a series of handbooks which comprise the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). This handbook and the handbook set are for VAX system trained Digital customer service personnel who service XMI-based systems and subsystems. Users of the handbook set should be familiar with the XMI bus architecture (either through the *XMI Bus Concepts* course or through practical experience) and have a minimum of level 1 hardware maintenance training on one or more VAX systems (for example, VAX 6000 or VAX 9000 systems).

Document Scope and Structure

Several I/O adapters have been developed to interface the XMI bus to devices which employ different bus structures and protocols. These adapters are available as stand-alone options and may be installed on a variety of systems or subsystems.

The *XMI Adapters Handbook Documentation Set* provides a single, quick reference source to the type of information most frequently required to service XMI adapters. This handbook contains information specific to the DEMNA option.

This handbook is divided into eight chapters:

Chapter 1 outlines the DEMNA's physical, functional, and operational characteristics.

Chapter 2 lists the components of the DEMNA option package and cabinet kits and overviews the configuration rules for installing the option.

Chapter 3 describes the DEMNA power-up self-tests and ROM-based diagnostics.

Chapter 4 describes the DEMNA's macro-level diagnostics and support programs.

Chapter 5 overviews the DEMNA's console monitor program.

Chapter 6 describes the XMI required and DEMNA specific registers.

Chapter 7 shows selected DEMNA sequencing flows.

Chapter 8 overviews the DEMNA's error reporting and error handling mechanisms.

Conventions

addresses	All addresses are given in hexadecimal (hex).
bits	All bit numbers are given in decimal with the bit(s) enclosed in angle brackets: for example <31>.
	Multiple individual bits or bit fields are separated by commas with bit fields indicated by two numbers separated by a colon. For example <31:24,20,18,14:10> indicates bits 31 through 24 (inclusive), bit 20, bit 18, and bits 14 through 10 (inclusive).
CTRL/x	Specifies to press and hold the Ctrl key while pressing the x key; for example, CTRL/C.
[item] . . .	Indicates the item is optional. The horizontal ellipsis indicates that additional optional items can be entered.
. . .	Vertical ellipsis in examples, tables, or figures, indicate that not all information is shown.

CHAPTER 1

1 DEMNA OVERVIEW

1.1 INTRODUCTION

The DEMNA is a high-performance I/O controller which provides a communications path between a host processor on the XMI and other nodes in an Ethernet/802 local area network. The DEMNA is compatible with both Ethernet and IEEE 802 specifications[1].

Multiple DEMNAs can be installed on the XMI, allowing a single XMI to communicate with multiple Ethernet/802 networks. The DEMNA connects to a network through a standard 15–pin Sub–D connector.

Figure 1–1 shows the DEMNA in an XMI system.

1.2 PORT OVERVIEW

The DEMNA supports one Ethernet/IEEE 802 port which provides the physical link layer and portions of the data link communication layer of the Ethernet and the 802 protocols.

The DEMNA has its own onboard CVAX processor which allows the DEMNA to control most operations independent of the host processor. Details of Ethernet transactions and XMI bus data transfer transactions are transparent to the host processor.

The DEMNA's onboard firmware is stored in an EEPROM which allows the firmware to be updated without the need for hardware modification. The EEPROM also stores various DEMNA operating parameters which can be modified in the field.

[1] IEEE 802 refers to the CSMA/CD local area network defined in the IEEE 802.2 and 802.3 specifications (physical and data link layers).

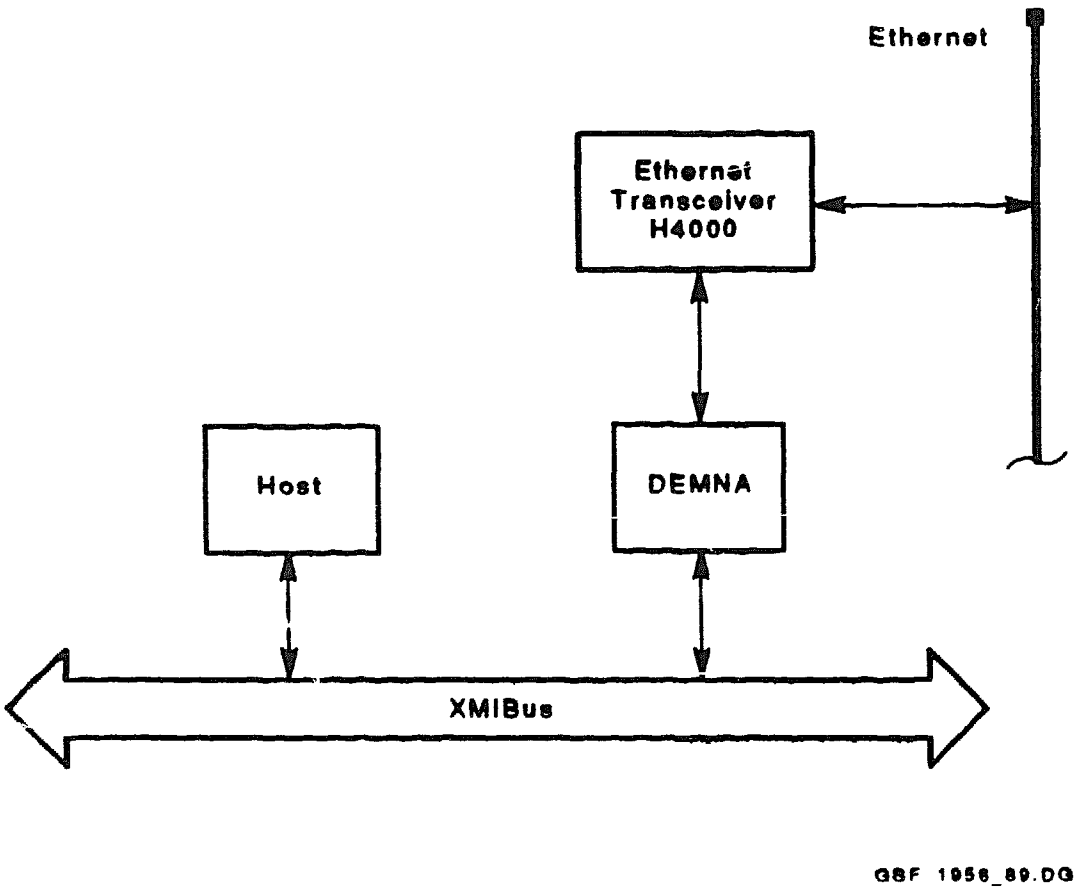


Figure 1–1 DEMNA Option in an XMI System

The DEMNA firmware includes a console monitor program which allows users at virtually any terminal on the network to monitor DEMNA operation and network traffic. The console monitor program can be accessed over the network or from a terminal attached directly to the DEMNA (the physical console).

The DEMNA has its own onboard diagnostics. On power-up or reset, the DEMNA executes self-tests and indicates the pass/fail status through LEDs on the module and through an onboard power-up diagnostic (XPUD) register. The self-tests and the onboard diagnostics can be invoked from the system console or from the DEMNA physical console.

The DEMNA may participate in network boot operations and can be specified as the boot device by its host system or enabled to involuntarily boot its host system on receiving a valid boot message over the network.

1.3 FUNCTIONAL OVERVIEW

The DEMNA logic is partitioned into four major subsystems as shown in Figure 1–2.

1.3.1 Microprocessor Subsystem

Functions:

- Stores and executes the DEMNA functional microcode, the diagnostic microcode, and the console monitor program
- Stores and supplies the module's default (MAC) Ethernet address

Components:

- CVAX

  32-bit processor dedicated to executing the DEMNA firmware (cannot be used directly by host application programs or by a user at the system console).

- System Support Chip (SSC)

  Provides control logic (for example: timers, internal registers, and address decoding) for the microprocessor and a UART for connection to the DEMNA physical console.

- EEPROM and CVAX RAM (SRAM)

  The EEPROM stores the DEMNA's functional microcode and history data of DEMNA failures and errors. The functional microcode is loaded into the SRAM during the DEMNA's power-up and node reset sequences and is then executed from the SRAM.

- MAC Address (ENET) PROM

  Stores the Medium Access Control (MAC) address, which is the DEMNA's default physical (Ethernet) address (DPA). Also stores a PROM test pattern.

  The port driver, on request from an application which starts up a protocol (such as DECnet), may assign one or more alternative addresses to the DEMNA. This type of address is called an actual physical address (APA).

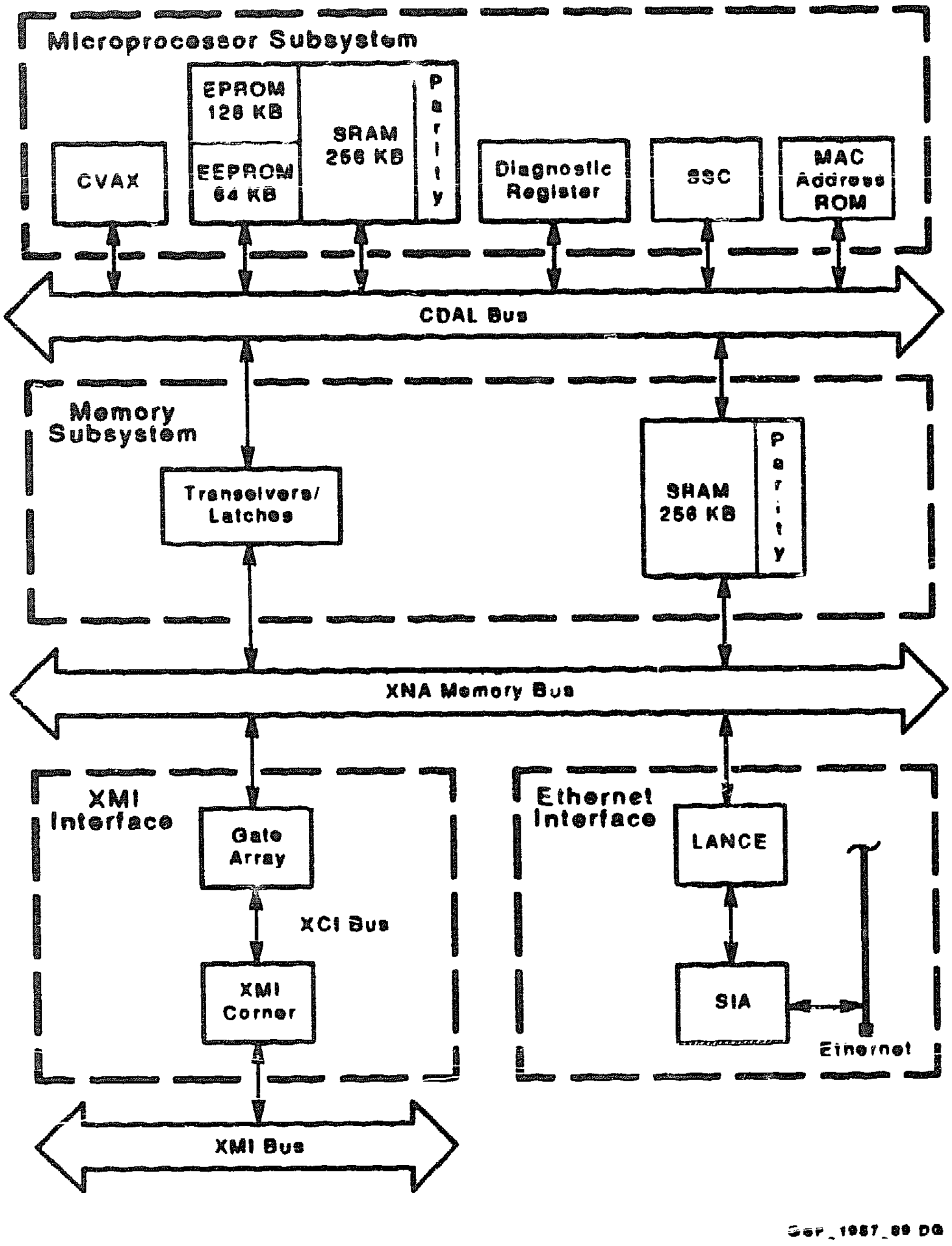


Figure 1–2 DEMNA Logic Subsystems

- EPROM

  Contains a copy of the DEMNA's operational firmware (less the console monitor program), diagnostic firmware, self-test code, and the DEMNA's boot code. The boot code is executed at the start of the power-up and node reset sequences to perform a minimal amount of module initialization. The self-test code, which is called by the boot code, tests the module components and loads the operational firmware from the EEPROM into the SRAM. Control is then transfered to the operational firmware in the SRAM on completion of the self-test.

  If the SRAM fails self-test, control is passed to EPROM diagnostic firmware to provide a means for running the ROM-based diagnostics. If the SRAM passes, but the EEPROM contents are invalid, the EPROM copy of the firmware is loaded into the SRAM.

- Diagnostic Register

  Controls certain low-level diagnostic operations, such as disabling CVAX RAM parity

1.3.2 Memory Subsystem

Functions

- Buffers packets to and from the Ethernet interface
- Buffers transfers to and from the XMI bus
- Stores shared data structures that allow communications between the CVAX and the LANCE

Components

- SRAM

  256 Kbytes, parity-protected memory. Buffers Ethernet and XMI transfers and stores data structures shared by the CVAX and LANCE.

- Bus control logic

  Controls read/write timing and read/write signals.

- DMA logic

  Controls access to the SRAM.

- **XNA timeout logic**

  Detects when a DMA grant has been outstanding longer than the timeout period.

1.3.3 XMI Interface Subsystem

Functions

- **Provides an interface between the DEMNA's shared memory and the XMI bus.**
- **Transfers Ethernet read and write data between DEMNA shared memory and host memory**
- **Performs control operations for the DEMNA CVAX (high-priority quadword XMI reads and writes to memory and longword XMI I/O reads and writes)**
- **Implements the DEMNA port registers, XMI required registers, and the XMI interrupt logic**

Components

- **Gate array**

  Implements most of the XMI interface logic.
- **XMI timeout logic**

  Detects timeouts for XMI operations.

1.3.4 Ethernet Interface Subsystem

Functions

- **Provides an interface between the memory subsystem and the Ethernet wire.**
- **Performs reads from and writes to shared memory.**

Components

- **Local Area Network Controller for Ethernet (LANCE) chip**

  Implements the microprocessor interface, performs DMA to and from DEMNA shared memory, implements the CSMA/CD network access algorithm, performs packet handling on transmits and receives, and reports errors.

- Serial Interface Adapter (SIA) chip

  Performs Manchester encoding (transmit) and decoding (receive) and TTL (LANCE) to differential (Ethernet wire) signal conversion.

- Bus interface

  Generates byte parity on transfers to, and checks byte parity on transfers from, DEMNA shared memory.

1.4 PHYSICAL DESCRIPTION

The DEMNA option consists of a T2020 module, an internal Ethernet cable, an external Ethernet cable, and an optional internal cable for a physical console if a physical console is used.

The T2020 module is a standard XMI module which plugs into the XMI backplane.

The internal Ethernet cable connects the T2020 module to a bulkhead connector for the Ethernet transceiver cable. It also provides power to the H4000 transceiver.

The external Ethernet cable connects the Ethernet transceiver bulkhead connector to the Ethernet transceiver.

The optional internal physical console cable connects the T2020 module to a bulkhead connector for a terminal cable.

The optional physical console allows the user to perform maintenance and monitoring operations (for example, running ROM-based diagnostics and examining error history logs) without requiring a host connection or a working Ethernet.

1.5 ENVIRONMENTAL SPECIFICATIONS

Parameter	Range
Temperature	
Operating	5°C to 50°C (41°F to 122°F)
Storage	-40°C to 66°C (-40°F to 151°F)
Humidity	
Operating	10% to 95% with maximum wet bulb of 32°C (89.6°F) and minimum dew point of 2°C (36°F) noncondensing
Storage	To 95% noncondensing
Altitude	
Operating	To 2.4 km (8,000 ft)
Storage	To 9.1 km (30,000 ft)

1.6 REFERENCE DOCUMENTS

Order Number	Title
EK-DEMNA-TM	*DEC LANcontroller 400 Technical Manual*
EK-DEMNA-IN	*DEC LANcontroller 400 Installation Guide*
EK-DEMNA-UG	*DEC LANcontroller 400 Console User's Guide*
EK-DEMNA-PG	*DEC LANcontroller 400 Programmers Guide*
EK-ETHER-IN	*Ethernet Installation Guide*
AA-LA50A-TE	*VMS Network Control Program Manual*

CHAPTER 2

2 DEMNA CONFIGURATIONS

2.1 INTRODUCTION

This chapter overviews the configuration requirements for installing the DEMNA. Refer to the *DEC LANcontroller 400 Installation Guide* for detailed installation instructions.

2.2 DEMNA OPTION PACKAGE AND CABINET KITS

The DEMNA option consists of a T2020 module, an I/O connector panel and internal cable for connecting to the Ethernet transceiver, and an I/O connector panel and internal cable for connecting to a physical console (if used).

The T2020 module is obtained from the DEMNA-M option package. The I/O connector panels and cables are obtained from cabinet kits applicable to the system. See Tables 2–1 and 2–2.

The DEMNA also requires an external Ethernet transceiver cable and an external terminal cable for the physical console (if used). These cables are not included in the DEMNA option package or in the cabinet kits, but must be ordered separately. Refer to the *Systems and Options Catalog*.

Table 2–1 DEMNA-M Option Package Contents

Component	Description
T2020	DEMNA module
EK-DEMNA-IN	*DEC LANcontroller 400 Installation Guide*
EK-DEMNA-UG	*DEC LANcontroller 400 Console User's Guide*
EK-DEMNA-RN	*DEC LANcontroller 400 Release Notes*

Table 2–2 Cabinet Kits

Kit [1]	Contents	
VAX 6000 Systems		
CK-DEMNA-KD	74-26407-41	Ethernet I/O connector panel
	17-01496-02	Internal Ethernet cable (8-foot)
	74-26407-01	Blank panel
	12-22196-02	Ethernet loopback connector
VAX 9000 Model 2xx Systems		
CK-DEMNA-KE	70-27894-01	Ethernet I/O connector panel
	17-01496-01	Internal Ethernet cable (3-foot)
	12-22196-02	Ethernet loopback connector
VAX 9000 Model 4xx Systems		
CK-DEMNA-KM	70-27894-01	Ethernet I/O connector panel
	17-01496-02	Internal Ethernet cable (8-foot)
	12-22196-02	Ethernet loopback connector
Internal Cable for Physical Console		
CK-DEMNA-AM	74-26407-32	I/O connector panel, VAX 6000 cabinets
	70-28010-01	I/O connector panel, VAX 9000 cabinets
	74-26407-01	Blank panel
	17-02168-01	Physical console internal cable
	EK-DEMNA-IN	*DEC LANcontroller 400 Installation Guide*
	EK-DEMNA-UG	*DEC LANcontroller 400 Console User's Guide*

[1]Cabinet kits must be ordered separately from the DEMNA-M option package. For systems not included in this table, refer to the *Systems and Options Catalog*.

2.3 T2020 MODULE PLACEMENT

The DEMNA requires one slot in the XMI backplane for the T2020 module. Table 2–3 indicates the XMI slots into which the module can be installed and the maximum number of DEMNAs allowed in each XMI cardcage.

Table 2–3 T2020 Module Placement in XMI Cardcage

System Type	XMI Slots	Maximum[1]
VAX 6000 Models 200/300/400	1 to 4; B to E[2]	Six
VAX 6000 Model 500	1 to 5; A to E[2]	Six
VAX 9000	Any slot except 7 or 8	Four

[1]Maximum number of DEMNAs supported by VMS Operating System.

[2]In VAX 6000 systems, DEMNAs are usually placed in the higher numbered slots available within the indicated ranges (CPUs are usually placed in lower numbered slots).

2.4 INTERNAL ETHERNET CABLE

This cable connects the DEMNA to the bulkhead connector for the external Ethernet transceiver cable. See Figures 2–1 to 2–4 and Tables 2–4 to 2–6.

2.5 INTERNAL CABLE FOR PHYSICAL CONSOLE

This cable connects the DEMNA to the bulkhead connector for a terminal to be used as the physical console (optional). See Figures 2–5 to 2–7 and Tables 2–7 to 2–9.

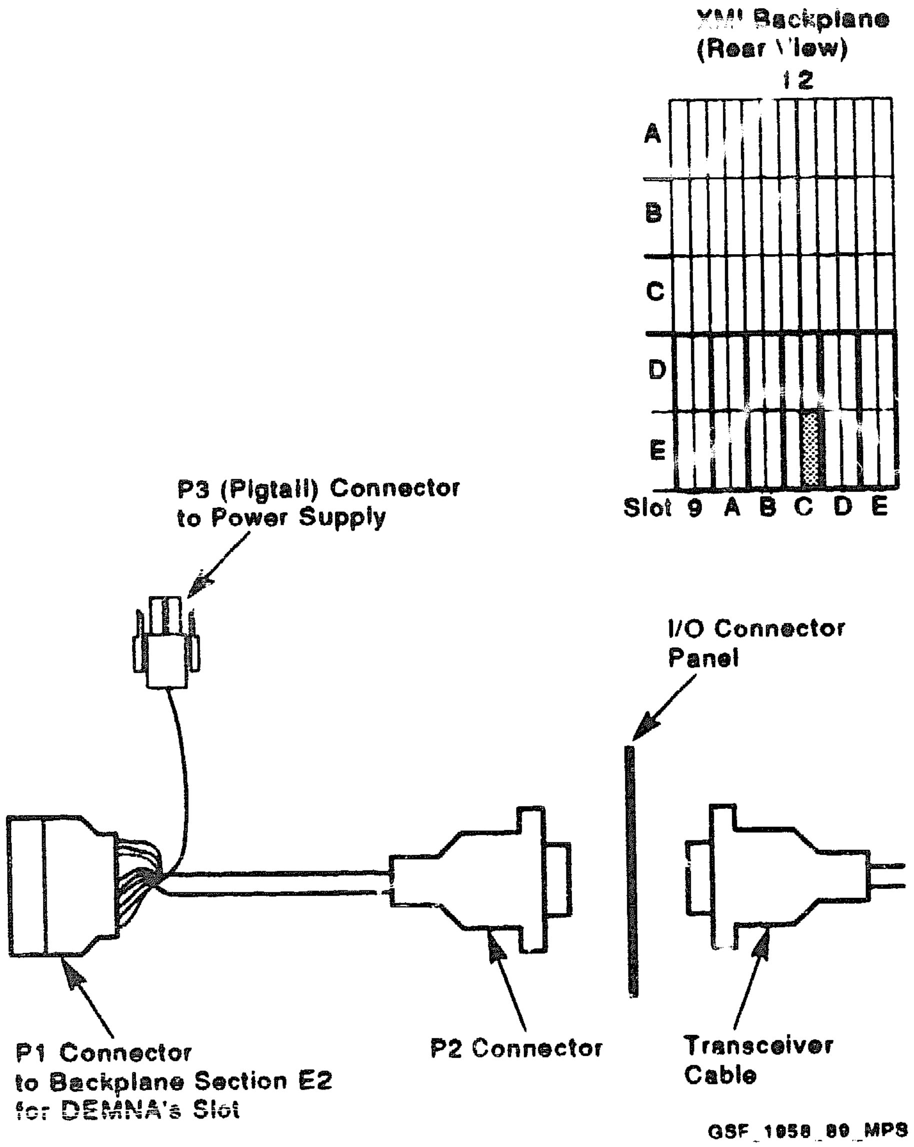


Figure 2–1 Internal Ethernet Cable Connections

Table 2–4 Internal Ethernet Cable Connectors

P1	References: Figure 2–2 and Table 2–5 P1 plugs into segment E2 of the XMI slot and is properly installed when the key is on the right as viewed from the backplane. The connector is not specifically keyed for backplane segment E2 (it is possible to install the connector in the wrong segment).
P2	References: Figure 2–3, Figure 2–4 and Table 2–6 P2 plugs into the I/O connector on the bulkhead for the external Ethernet cable. On VAX 6000 systems, connector P2 of the first DEMNA plugs into the Ethernet port on the system interconnect panel. Figure 2–3 shows the location of the Ethernet port.
P3	P3 is a two-prong connector which plugs into a +15 Vdc connector on a system power supply (H7214 on VAX 6000 systems; power distribution adapter on VAX 9000 systems). P3 supplies power for an Ethernet device which does not have its own power supply, such as an H4000, DESTA, or DECOM. If all +15 Vdc connectors are in use, the external Ethernet transceiver cable can not be connected directly to one of these transceiver types, but may be connected to one of the following: • DELNI • DEMPR • DEBET Each of these devices has its own power supply and can be cabled to an H4000. Connector P3 shold be installed regardless of the transceiver type. See the *DEC LANcontroller 400 Installation Guide* for more information.

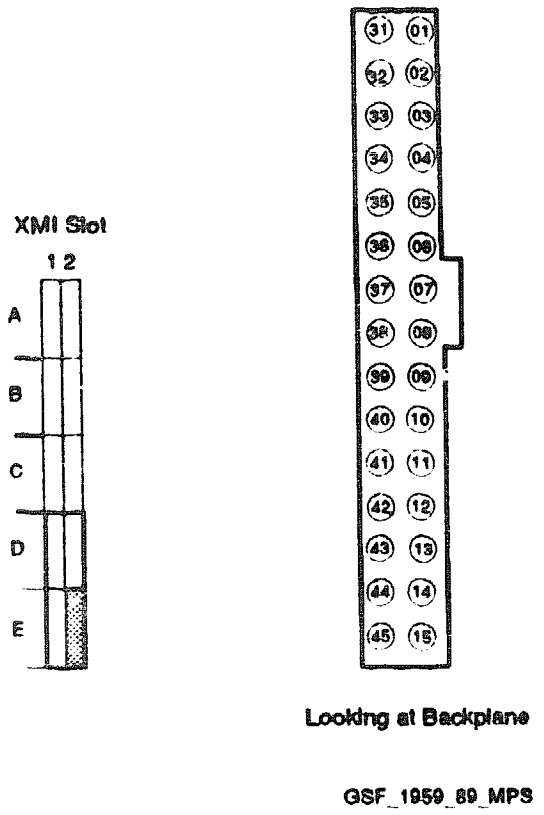


Figure 2–2 Internal Ethernet Cable, P1 Connector Pinouts

Table 2–5 Internal Ethernet Cable, P1 Connector Signals

Pin	Signal
E05 to E09	Logic Ground
E10	Ethernet Collision L
E11	Ethernet Collision H
E12	Ethernet Receive L
E13	Ethernet Receive H
E14	Ethernet Transmit L
E15	Ethernet Transmit H

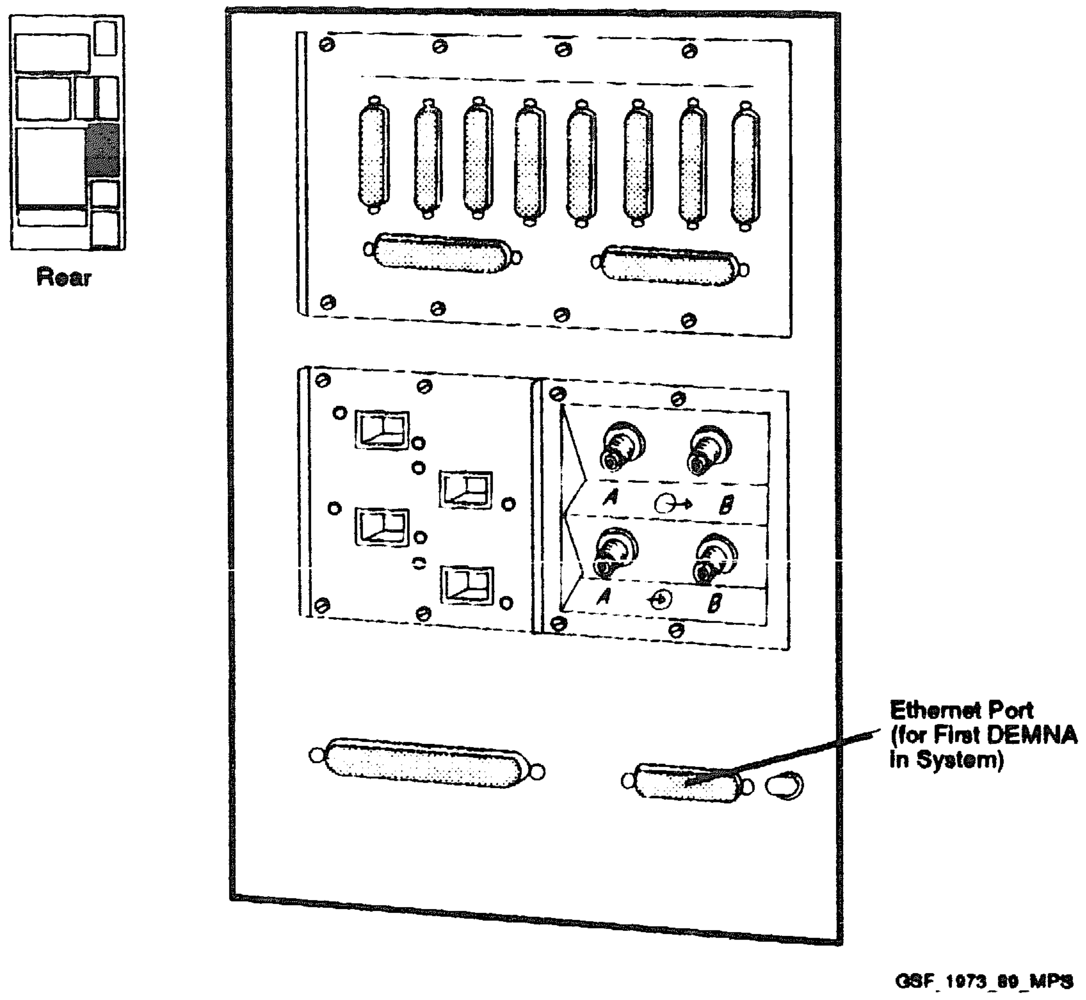
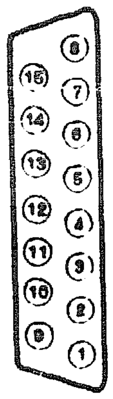


Figure 2–3 VAX 6000 Model 400 System Interconnect Panel



GSF_1960_89_MPS

Figure 2–4 Internal Ethernet Cable, P2 Connector Pinouts

Table 2–6 Internal Ethernet Cable, P2 Connector Signals

Pin	Signal
1	Shield
2	Collision Presence H
9	Collision Presence L
3	Transmit H
10	Transmit L
5	Receive H
12	Receive L
6	Power Return
13	Power

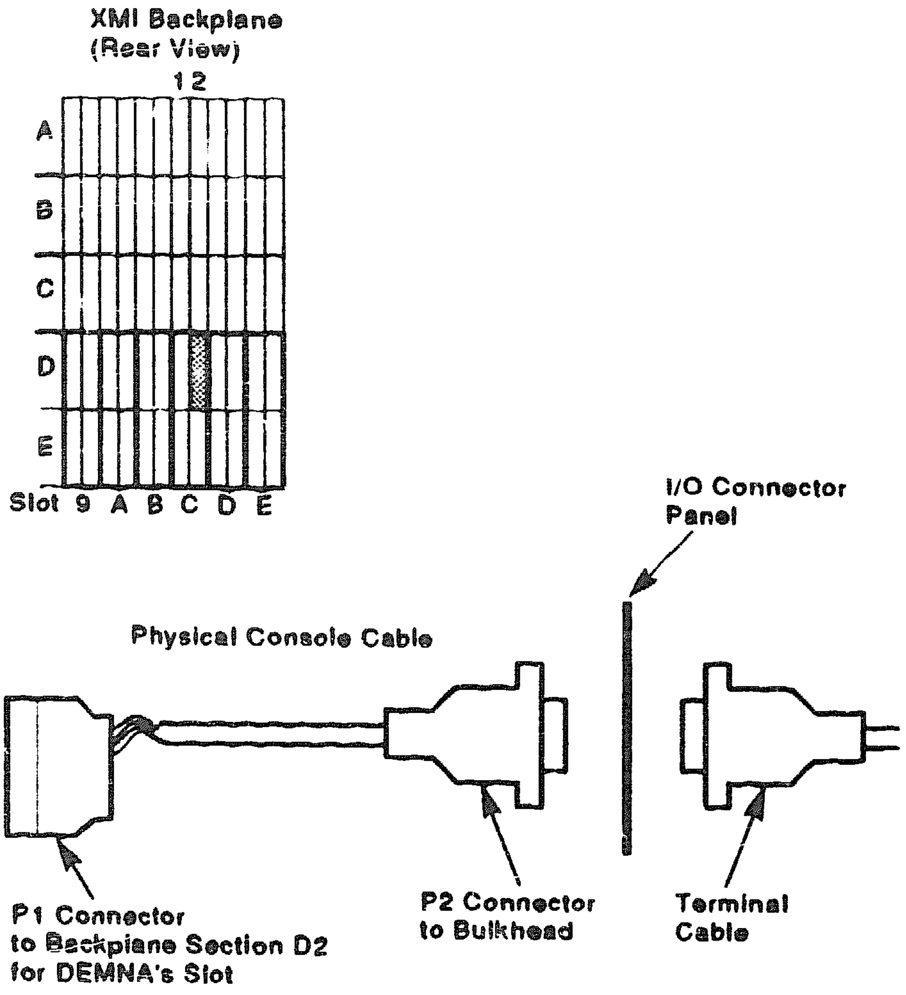


Figure 2-5 Internal Cable for Physical Console, Connections

Table 2-7 Internal Cable for Physical Console, Connectors

P1	**References: Figure 2-6 and Table 2-8** P1 plugs into segment D2 of the XMI slot and is properly installed when the key is on the right as viewed from the backplane. The connector is not specifically keyed for backplane segment D2 (it is possible to install the connector in the wrong segment).
P2	**References: Figure 2-7 and Table 2-9** P2 plugs into an I/O connector on the bulkhead for a terminal cable.

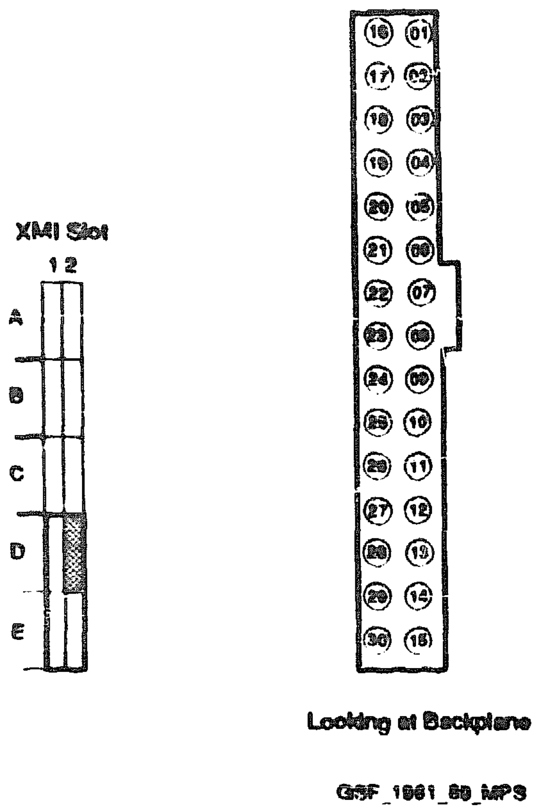


Figure 2–6 Internal Cable for Physical Console, P1 Connector Pinouts

Table 2–8 Internal Cable for Physical Console, P1 Connector Signals

Pin	Signals
D01	Transmit
D02	Receive
D03	Logic Ground

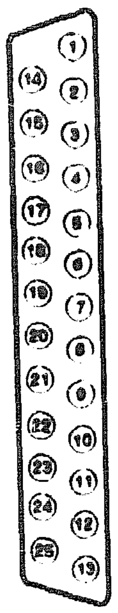


Figure 2-7 Internal Cable for Physical Console, P2 Connector Pinouts

Table 2-9 Internal Cable for Physical Console, P2 Connector Signals

Pin	Signal
2	Transmit
3	Receive
7	Logic Ground

CHAPTER 3

3
DEMNA POWER-UP SELF-TESTS AND ROM-BASED DIAGNOSTICS (RBDs)

3.1 POWER-UP SELF-TESTS

The power-up self-tests are ROM-resident diagnostics that verify the DEMNA's basic operation and ability to transmit and receive loopback packets over the network. The self-tests are automatically run on system power-up or XMI reset (see Table 3–1 and Figure 3–1) and can be run as an RBD from the system console or DEMNA physical console (Example 3–1).

Table 3–1 DEMNA Self-test Indications After Power-up or XMI Reset

Result	Indication(s)
Pass	Both DEMNA LEDs illuminated (Figure 3–1)
	Value of FFFFC007 or FFFFC027 recorded in XPUD register. (The XPUD records the status of each test. Bit <05> is set if the EEPROM contains error history entries.)
Fail	One or both LEDs extinguished
	Pass/fail status of each self-test recorded in XPUD register. Self-test complete bit (bit <31>) in register cleared.
	Self-test fail bit (bit <10>) in XBER register set.

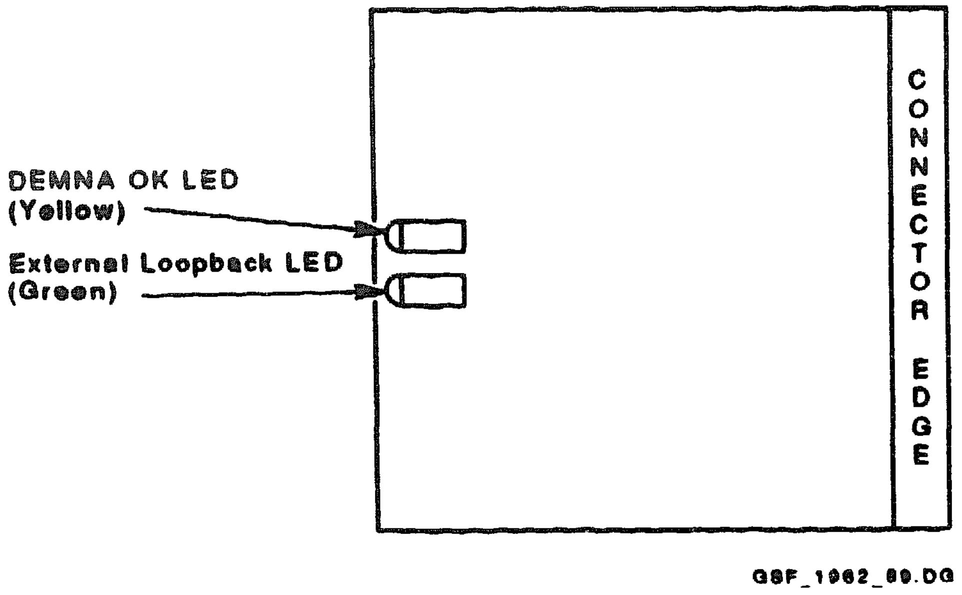


Figure 3-1 DEMNA LED Locations

Notes:

1. If the External Loopback LED is off, the fault is not necessarily due to a failed DEMNA logic component, but could be due to one or more of the following:
   - Defective or improperly seated cable
   - Defective Ethernet transceiver connector
   - DEMNA disconnected from transceiver
   - One of the other self-tests failed (the external loopback test is not executed in this case and the LED will stay off)
2. If the XPUD register indicates that all tests failed, the problem may be due to the CVAX, ROM, or bus transceivers.
3. A self-test failure can also be caused by a systemwide fault: for example, a faulty power supply or missing XMI bus terminator.
4. If the cause of a self-test fault is corrected, the LED(s) will light only if the self-test is rerun. If the self-test is not rerun, the DEMNA will still function properly with the LED(s) off.)

```
> CTRL/P
>>>Z 3                          ! Connect console to XMI node 3
?33 Z connection successfully started
T/R                             ! Enter RBD monitor
RBD3>ST 0                       ! Run RBD 0 (DEMNA self-tests)

;Selftest     1.00
;        P          3      0C03         1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD3> CTRL/Z CTRL/P
?31 Z connection terminated by ^P
>>>
```

a. VAX 6000 Systems

```
> CTRL/P
>>> Z 2E                        ! Connect console to XJA 2, XMI node E
?33 Z connection successfully started
T/R                             ! Enter RBD monitor
RBDE>ST 0                       ! Run RBD 0 (DEMNA self-tests)

;Selftest     1.00
;        P          E      0C03         1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBDE> CTRL/Z CTRL/P
?xx Z connection terminated by ^P
>>>
```

b. VAX 9000 Systems

Example 3-1 Running DEMNA Self-Tests from the System Console

3.2 ROM-BASED DIAGNOSTICS (RBDs)

The RBDs provide more extensive testing of selected DEMNA logic functions. The RBDs are resident in the EPROM and are accessed from the system console or physical console through the RBD user interface.

Self-test and RBD errors are reported to the system console and, if appropriate flags are set, are logged in the EEPROM. The first eight errors are logged, after which the error history must be cleared to allow logging of additional errors.

Table 3–2 DEMNA RBDs

RBD	Title/Description
0	Self-test RBD Same as the power-up self-test except that test results are displayed on the console. When run as RBD 0, the self-test does not affect the state of the LEDs, the XBER register, or the XPUD register. Table 3–3 lists the self-tests. If no test number is given with the START command all tests are executed.
1	NI RBD Verifies the Ethernet link between the DEMNA's Ethernet interface logic and the Ethernet transceiver. Consists of three tests: 1. External Loopback on Live Ethernet Test Requires the DEMNA to be connected to a live network. 2 MOP Loopback Test Requires at least one node that implements a MOP (maintenance operations protocol) loop server to be present on the local network. 3. External Loopback on Closed Ethernet Test Requires a loopback connector to be installed on the Ethernet connector at the system bulkhead or on the transceiver end of the transceiver cable. If no test number is given with the START command, test 1 is executed.
2	XMI RBD Verifies the DEMNA's ability to transfer data to and from host memory. The /C qualifier must be specified when invoking the test since it performs writes to host memory.
3	XNA RBD Verifies the DEMNA's ability to simultaneously perform external loopbacks to the Ethernet and datamoves to and from host memory. The XNA RBD is effectively a combination of NI RBD test 1 and the XMI RBD. The /C qualifier must be specified when invoking the test since it performs writes to host memory.

Table 3–3 DEMNA Self-Test (RBD 0)

Test	Unit or Function Tested
1	Boot ROM (EPROM)
2	CVAX IRQ Lines
3	Diagnostic Register
4	SSC Chip
5	Console UART Driver
6	CVAX RAM
7	CVAX Parity RAM
8	CVAX Chip
9	ENET PROM
10	EEPROM
11	XNADAL Readback
12	XNADAL Timeout Logic
13	Shared RAM
14	Shared Parity RAM
15	LANCE Chip
16	Ethernet Subsystem Parity
17	LANCE External Loopback
18	DEMNA Gate Array

3.2.1 RBD COMMANDS

Table 3–4 describes the RBD commands. Uppercase, bold-face characters indicate the minimum acceptable abbreviation for the command.

Commands may be entered in uppercase or lowercase. The bell character and a question mark are returned on incorrect syntax.

Table 3–4 DEMNA RBD Commands

STart	Syntax: `ST[art] RBD_number [/qual.../qual] [p1 [p2]]` Starts a test, or group of tests, of a specified RBD. **RBD_number:** 0 - Self-test RBD 1 - NI RBD 2 - XMI RBD 3 - XNA RBD **/qual** Command qualifier: See Table 3–5 **p1, p2** Command parameters: See Table 3–6
Deposit	Syntax: `D[eposit] [/qual.../qual] [address] [data]` Deposits data into XMI registers and memory locations resident on the DEMNA. **/qual** Command qualifiers: See Table 3–7 **address** A one to eight digit hex value or a special addressing character (Table 3–8). When the RBD moitor is entered, the default address is 0 and the default address type is physical. **data** Byte, word, or longword of data to be deposited. When the RBD moitor is entered, the default data size is longword.
Examine	Syntax: `E[xamine] [/qual.../qual] [address]` Displays contents of XMI registers and memory locations resident on the DEMNA. **/qual** Command qualifiers: See Table 3–7 **address** A one to eight digit hex value or a special addressing character (Table 3–8). When the RBD moitor is entered, the default address is 0 and the default address type is physical.

Table 3–4 (Cont.) DEMNA RBD Commands

QUit	Syntax: `QU[it]`
	
	Sets the Node Reset bit in the XBER, initializing the DEMNA, causing it to execute a power-up self-test.
	
	If the RBD monitor was accessed through the host's system console CTRL/P must be entered after the QUIT command to return to the console prompt:
	
	`RBD3> QUIT CTRL/P`
	`?xx Z connection terminated by ^P`
	`>>>`
	
	CTRL/Z performs the same function as the QUIT command.

SUmmary	Syntax: `SU[mmary]` Displays a summary report of the last diagnostic executed. If no diagnostic was run since the RBD monitor was invoked, the "?" character is returned.
XFC	Syntax: `XFC` Forces a jump to the address loaded into register XPD1. XFC is used to invoke the diagnostic error log reader. See Example 3–3.

Table 3–5 DEMNA RBD START Command Qualifiers

/BE	Ouput bell character to console on error. Default: No bell
/C	Confirm execution of tests which perform writes to host memory (XMI and XNA RBDs). Default: No confirmation
/DS	Disable displaying of status reports. Default: Status reports
/HE	Halt on hard error, print error and summary reports, execute cleanup code, and return to RBD prompt. A hard error is a repeatable fault (for example, ROM checksum error) from which the diagnostic can recover. In contrast, a fatal error (for example, unexpected interrupt) causes the program to abort, regardless of the state of the /HE or /LE qualifiers. Default: Continue on hard error
/HS	Halt on soft error, print error and summary reports, execute cleanup code, and return to RBD prompt. A soft error is a non-repeatable fault (error not present on retry) from which the diagnostic can recover. The only soft error detected is a missing heartbeat. /HS is applicable only for NI RBD tests 1 and 2. If /HS and /LS are both specified, the monitor indicates an error. Default: Continue on soft error
/IE	Inhibit error reports during diagnostic execution. Error reports displayed on diagnostic completion. Default: Error reports
/IS	Inhibit display of summary report after completion of diagnostic. Default: Summary reports
/LE	Loop on hard error (even if error is intermittent). Press CTRL/C, CTRL/Y, or CTRL/Z to terminate loop and return to RBD prompt. Error reports are displayed while looping unless /IE was specified. A summary report is displayed on loop termination unless /IS was specified. Default: Continue

Table 3–5 (Cont.) DEMNA RBD START Command Qualifiers

/LS	Loop on soft error (even if error is intermittent). Press CTRL/C, CTRL/Y, or CTRL/Z to terminate loop and return to RBD prompt. Error reports are displayed while looping unless /IE was specified. A summary report is displayed on loop termination unless /IS was specified. /LS is applicable only for NI RBD tests 1 and 2. The only soft error detected is a missing heartbeat. Default: Continue
/P=n	Run *n* passes of each selected test. Specify n=0 for infinite passes (press CTRL/C, CTRL/Y, or CTRL/Z to halt). Default: One pass
/T=n[:m]	Run single test (/T=n) or group of tests (/T=n:m). Specify test number(s) in decimal. Default: RBD dependant
/TR	Trace (display) test number at start of each test. Default: Disabled

Table 3–6 DEMNA RBD START Command Parameters

For the NI RBD	
P1	8-digit decimal number specifing the number of packets to be transmitted and received for each test pass. Default is 100. A value of zero specifies to transmit packets indefinitely until CTRL/C is pressed. P1 is supported by all NI RBD tests.
P2	Decimal number in the range of 64 to 1518 which specifies the transmit packet size (in bytes). If p2 is not specified, the test varies the packet size. P2 is supported by the MOP loopback test only. To use the parameter, p1 must also be specified.
For the XMI and XNA RBDs	
P1	Integer value which represents the number of datamove and peek operations performed per test pass. A value of *n* specifies *n* * 256 datamoves and *n* * 512 peeks. Default is 1 (256 datamoves, 512 peeks). A value of zero specifies to execute datamoves and peeks until CTRL/C is pressed.
P2	Specifies the starting address in host memory to be used by the test. The starting address must be page aligned. If a nonpage-aligned address is specified, the diagnostic zeros the nine least significant bits. Default is 200 (hex). To use the p2 parameter, p1 must also be specified.

Table 3–7 DEMNA RBD Deposit/Examine Command Qualifiers

/B	Defines data size as byte.
/G	Defines address space as CVAX GPRs 0 to B. Valid only for EXAMINE command. When /G is specified the address field must be a hex digit in the range 0 to B.
/L	Defines data size as longword (default).
/N:*n*	Deposits data to, or examines data from, the specified address and the next *n* addresses. If the starting address is specified with "-", the next *n* higher addresses are still used (the "-" character specifies the starting address, not the direction).
/P	Defines the address space as physical memory (default).
/W	Defines data size as a word.

Table 3–8 DEMNA RBD Deposit/Examine Commnad Special Addressing Characters

+	Increment address last referenced by DEPOSIT or EXAMINE by current data size
-	Decrement address last referenced by DEPOSIT or EXAMINE by current data size
*	Use address last referenced by DEPOSIT or EXAMINE

3.2.2 RBD CONTROL KEYS

Table 3–9 DEMNA RBD Control Keys

Key	Mode [1]	Function
CTRL/C	Diag	Stop diagnostic execution, execute cleanup code, return to RBD prompt. Enabled messages for aborted test are displayed on console.
	Mntr	Echo "^C", reissue RBD prompt.
CTRL/P	Diag	Exit console mode, return to system prompt (>>>). If the RBD monitor is reentered on the same node, enabled test messages of the aborted test are displayed. CTRL/P is disabled when RBDs are run from the console monitor program.
	Mntr	Exit console mode, return to system prompt (>>>). CTRL/Z or QUIT must be used before CTRL/P to force DEMNA into a known state. CTRL/P is disabled when RBDs are run from the console monitor program.
CTRL/R	Diag	Ignored
	Mntr	Display current command line.
CTRL/U	Diag	Ignored
	Mntr	Echo "^U", abort command line, reissue RBD prompt.
CTRL/Y	Diag	Stop diagnostic, do not execute cleanup code, return to RBD prompt. Does not display test messages on console.
	Mntr	Echo "^Y", reissue RBD prompt.
CTRL/Z	Diag	Same as CTRL/C
	Mntr	Same as QUIT command.

[1]Diag = RBD diagnostic running; Mntr = RBD monitor running

3.2.3 Running the DEMNA RBDs

From the VAX 6000 System Console

```
>>> Z 3                      ! Connect console to XMI node 3
?33 Z connection successfully started
    T/R                      ! Enter RBD monitor
RBD3>ST0/P=1/TR/HE           ! Run one pass of all self-tests with halt
                             ! on hard error and trace tests enabled.

    .
    .
    .
RBD3> QUIT [CTRL/P]
?31 Z connection terminated by ^P
>>>
```

From the VAX 9000 System Console

```
>>>Z 2E                      ! Connect console to XJA 2, XMI node E
?xx Z connection successfully started
   T/R                       ! Enter RBD monitor
RBDE>ST1/T=3 200             ! Run one pass of NI RBD test 3, with
                             ! 200 packets per transfer.

    .
    .
    .
RBDE> QUIT [CTRL/P]
?xx Z connection terminated by ^P
>>>
```

From the DEMNA Physical Console

```
XNA>T/R                      ! Enter RBD monitor
RBD3>ST2/P=3 2 1000 /C       ! Run three passes of XMI RBD; 512
                             ! datamoves and 1024 peeks per pass.
                             ! Starting address is 1000.

    .
    .
    .
RBD3> QUIT
XNA>
```

3.2.4 RBD Error Report Formats

The DEMNA follows the XMI standard for RBD error reports, supporting three levels of error reporting. Table 3-10 lists the fields in each error report level, and Example 3-2 shows a sample error report.

Table 3–10 DEMNA RBD Error Report Levels

Level	Type	Error Report Line Fields
1	Summary	Pass/Fail status XMI node number Device type (0C03 = DEMNA) Pass count
2	Error class	Error type—HE (hard), FE (fatal), SE (soft) Logic under test Unit under test (always 0) Test number (0 if test initialization failed)
3	Error specific	Subtest number Expected data Received data System control block (SCB) offset Failing memory or register address PC value in ROM at time of failure Specific error number

```
❶ ;  F          3       0C03   00000018
❷ ;  HE   CVAX_RAM        00       T0006
❸ ;  00   00000000   00800000   00000000   20150004   20051D97  03
```

❶ Failed, XMI node 3, DEMNA (device type 0C03), pass 18

❷ Hard error, CVAX RAM under test, unit 0 (always), test 6

❸ Subtest 0, expected 00000000 (hex), received 00800000 (hex), SCB 0, failing address 20150004 (hex), error PC 20051D97, error number 03

Example 3–2 Sample DEMNA RBD Error Report

3.2.5 Diagnostic Error Log Reader

The diagnostic error log reader is an EPROM-resident program that displays the contents of the diagnostic error log (part of the EEPROM error history).

The error log reader is run by depositing it's starting address into the XDP1 register and then issuing the XFC command. Example 3–3 shows a sample run of the program.

```
>>> Z 3
?33 Z connection successfully started
    T/R
RBD3> D 20150100 2004C010    !Starting address in XDP1
RBD3> XFC                    !Jump to address in XPD1


************ DEMNA EEPROM ERROR FRAME READER V1.00 ***********

EEPROM revision and date:  0600   ( 14-FEB-1990)

Module serial number:      *SG915Y8879*

Logging is currently enabled for: Selftest  NIRBD XMIRBD XNARBD

There are 2 error frames stored.






Type <CR> to continue, <CTRL/C> to abort...

------------------- Error frame number 1 -------------------

Sequence number:                1

Diagnostic number:              1
Diagnostic revision:         3.00
Operating mode:               RBD

XMI node number:                3
Test number:                    3
Error code:
Error number:                  65
Expected data:          00000003(X)
Received data:          00000007(X)
SCB offset:             00000000(X)
Failing address:        201004A3(X)
PC at failure:          0001A762(X)

Number of times logged:         1


Type <CR> to continue, <CTRL/C> to abort...
```

Example 3–3 (Continued, next page) DEMNA Error Log Reader

```
------------------- Error frame number 2 -------------------

Sequence number:              2

Diagnostic number:            0
Diagnostic revision:       3.00
Operating mode:        Power-up

XMI node number:              3
Test number:                 18
Error code:                   0
Error number:                 3
Expected data:         00000000(X)
Received data:         00800000(X)
SCB offset:            00000000(X)
Failing address:       20150004(X)
PC at failure:         20051D97(X)

Number of times logged:       2


Type <CR> to continue, <CTRL/C> to abort...

*** No more errors logged ***
```

Example 3–3 DEMNA Error Log Reader

3.2.6 Isolating Faults With the RBDs

Figure 3–2 shows the suggested order in which to run the RBDs to isolate suspected DEMNA faults.

Notes on running NI RBD test 3

To run this test, perform the following:

1. Disconnect external Ethernet transceiver cable (BNE3) at transceiver end and install loopback connector on cable
2. Run test 3:

Pass	Transceiver bad: Replace transceiver, reconnect cable and rerun test to verify operation. No further action required.
Fail	Suspect transceiver cable, internal Ethernet cable, backplane, or DEMNA module. Go to next step.

3. Disconnect external transceiver cable at system bulkhead and install loopback connector
4. Rerun test 3

Pass	Transceiver cable bad: Replace cable and rerun test to verify operation. No further action required.
Fail	Suspect internal Ethernet cable, backplane, or DEMNA module. Go to next step.

5. Replace internal Ethernet cable and install loopback connector on new cable
6. Rerun test 3

Pass	Internal Ethernet cable bad. Replace cable and rerun test to verify operation. No further action required.
Fail	Suspect DEMNA module or XMI backplane. Go to next step.

7. Replace DEMNA
8. Rerun test 3

Pass	DEMNA bad. Rerun test to verify operation. No further action required.
Fail	XMI backplane bad. Install DEMNA in different slot. Rerun test to verify proper operation. Consider replacing XMI card cage.

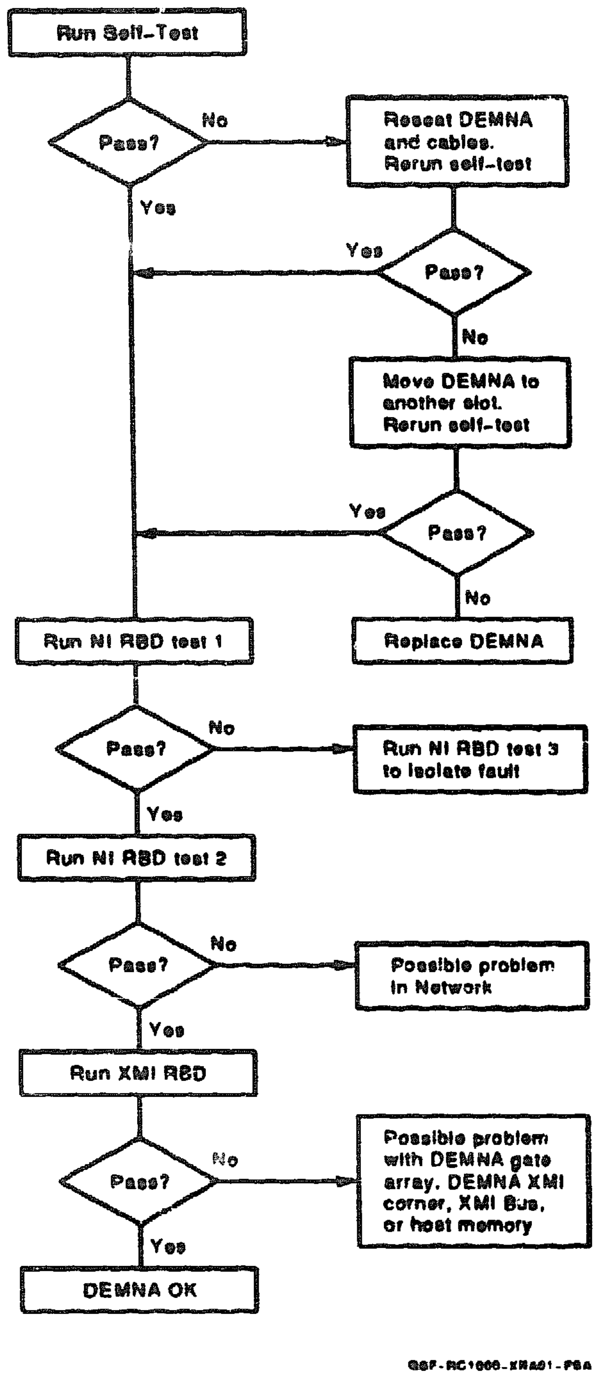

Run Self-Test
Pass?
No
Reseat DEMNA and cables. Rerun self-test
Yes
Yes
Pass?
No
Move DEMNA to another slot. Rerun self-test
Yes
Pass?
No
Run NI RBD test 1
Replace DEMNA
Pass?
No
Run NI RBD test 3 to isolate fault
Yes
Run NI RBD test 2
Pass?
No
Possible problem in Network
Yes
Run XMI RBD
Pass?
No
Possible problem with DEMNA gate array, DEMNA XMI corner, XMI Bus, or host memory
Yes
DEMNA OK
MSF-RC1000-XRA01-FSA

Figure 3–2 DEMNA RBD Troubleshooting Flowchart

CHAPTER 4

4 DEMNA MACRODIAGNOSTICS AND SUPPORT PROGRAMS

4.1 INTRODUCTION

This chapter provides an overview of the macrodiagnostics and support programs available for the DEMNA.

4.2 DIAGNOSTICS AND SUPPORT PROGRAMS

The DEMNA is supported by two macrodiagnostics and one utility program:

Table 4–1 DEMNA Macrodiagnostics and Support Programs

Name	Level	Description
EVDWC	2R	NI Exerciser Verifies the installation of the host Ethernet node and connectivity to all other nodes on the local network that support maintenance operations protocol (MOP).
EVDYE	2R	DEMNA NI Functional Diagnostic Verifies the functional operation of the DEMNA Ethernet/802 port and that the DEMNA can perform all of the functions required by the VAX/VMS Ethernet port driver (EXDRIVER).
EVGDB	2	EEPROM Update Utility Enables the user to update the firmware in the EEPROM, modify EEPROM flags and parameters, and patch the RBD code resident in the EPROM.

4.3 RUNNING EVDWC AND EVDYE

1. Log into the customer service account or SET DEFAULT to the SYS$MAINTENANCE directory.
2. Run the VAX Diagnostic Supervisor (VAX/VDS):

   ELSAA on VAX 6000 Model 2xx/3xx systems
   ERSAA on VAX 6000 Model 4xx systems
   EWSAA on VAX 9000 systems

3. Load the diagnostic:

```
DS>LOAD EVDWC    !or EVDYE
```

4. Attach and select the DEMNA:

   VAX 6000 Systems

```
DS>ATTACH DEMNA HUB EXm0 n
DS>SELECT EXm0
```

   Where:

m	Unit designator. The DEMNA with the lowest XMI node number is unit A; the one with the second lowest number is unit B, and so on.
n	DEMNA's XMI node ID

   VAX 9000 System

```
DS>ATTACH XJA HUB XJAx x
DS>ATTACH DEMNA XJAx0 EXm0 n
DS>SELECT EXm0
```

   Where:

x	XJA unit number (0 to 3)
m	Same as for VAX 6000 systems
n	Same as for VAX 6000 systems

5. Set the desired VAX/DS control flags (for example: TRACE, HALT) and any desired diagnostic event flags
6. Start the diagnostic

NOTE

Tests 2 through 10 of EVDYE can use either internal or external loopbacks during test execution. To use external loopbacks, perform the following:

- **Install a loopback connector at the system bulkhead or on the Ethernet transceiver cable.**
- **Issue the following after the SELECT EXm0 command:**

```
  DS> SET EVENT FLAG 1
```

4.4 EVGDB

EVGDB is a level 2 diagnostic and can be run with the VAX/DS running in either stand-alone mode or on-line (interfaced with the VAX/VMS operating system). When run in stand-alone, EVGDB runs the DEMNA self-test to verify the module operation. If self-test fails, EVGDB displays an error message and continues.

EVGDB is distributed with the DEMNA firmware image (EVGDBQ.BIN) to the field as part of the system tape media kits:

Table 4–2 EVGDB Distribution Media

Part Number	Name
AQ-FJ77*-ME	VAX 6000-200 Console TK50
AQ-FK60*-ME	VAX 6000-300 Console TK50
AQ-FK87*-ME	VAX 6000-400 Console TK50
AQ-PAKJ*-ME	VAX 9000 CNSL UTIL + UCODE Tape

Table 4–3 EVGDB Sections

Section	Functions Available to User
PARAM	Examine and modify user-settable EEPROM flags and parameters.
UPDATE	Load a new firmware image into the EEPROM and examine and modify EEPROM flags and parameters.
VERIFY	Load a new image into the system's main memory and compare it to the image in the EEPROM.
MFG	Load a new image into the EEPROM and examine EEPROM flags and parameters. Also clears the error log in the EEPROM and initializes the EEPROM flags and parameters.
DEFAULT	Identical to the UPDATE section. The DEFAULT section is run if no section is specified.
INVAL	Invalidate and initialize the EEPROM contents, forcing the DEMNA to run from the EPROM. The section is used to enable EEPROM updates when the normal procedure (UPDATE or DEFAULT sections) does not work.

Table 4–4 EVGDB Event Flags

Event Flag	EEPROM Flags or Parameter Made Accessible
1	Enable Local DEMNA Console Flag Enable DEMNA Monitor Facility Flag Enable Diagnostic Logging Flag Enable Self-Test Logging Flag Enable NI RBD Logging Flag Enable XMI RBD Logging Flag Enable XNA RBD Logging Flag
3	DEMNA Console Password

If no event flags are specified, only the following EEPROM flags can be modified:

Enable Remote Boot
Enable Remote DEMNA Console
Enable Promiscuous Mode

Table 4-5 DEMNA EEPROM User-Modifiable Flags and Parameters

Parameter	Description
Console Password	An 8-character ASCII field that indicates the password that must be used to connect to the DEMNA console monitor program. The default password is *XNABOARD*.

Flag	Operation if Flag Enabled
Enable Remote Boot	DEMNA allowed to participatate in remote booting over the network.
Enable Remote DEMNA Console Flag	DEMNA console monitor program made accessible from a remote network node.
Enable Local DEMNA Console Flag	DEMNA console monitor program made accessible from the local network node and from the DEMNA physical console.
Enable DEMNA Monitor Facility	Allows DEMNA to monitor network activity.
Enable Promiscuous Mode	Allows DEMNA to receive all packets on the network, regardless of the destination. If the flag is disabled, an application can override the flag by starting up a promiscuous user.
Enable Diagnostic Logging	Allows logging of self-test and RBD errors to EEPROM.
Enable Self-Test Logging	Log self-test errors to EEPROM. Diagnostic error logging must also be enabled.
Enable NI RBD Logging	Log NI RBD errors to EEPROM. Diagnostic error logging must also be enabled.
Enable XMI RBD Logging	Log XMI RBD errors to EEPROM. Diagnostic error logging must also be enabled.
Enable XNA RBD Logging	Log XNA RBD errors to EEPROM. Diagnostic error logging must also be enabled.

4.4.1 Modifying EVGDB Flags

```
   DS>LOAD EVGDB
   DS>ATTACH DEMNA HUB EXA0 3   !example shown for a VAX 6000 system
   DS>SELECT ALL
   DS>SET EVENT 1,3
   DS>START/SECTION=PARAM

   Program: EVGDB -
DEMNA EEPROM Update Utility, revision 1.1, 6 tests

❶ Testing: _EXA0

❷ Please insure that Front Panel Switch is in Update position.
   Ready [(Yes), No]

❸ Do you wish to clear the EEPROM error log?  [(No), Yes] No

   Reading parameters from EEPROM...

   EEPROM firmware rev:      0601 04-APR-1990

   DEMNA Serial Number:      *SG909T1488*

   Enable Remote Boot?             (Default =  No)  N
   Enable Remote DEMNA console?    (Default = Yes)  Y
   Enable Local DEMNA console?     (Default = Yes)  Y
   Enable DEMNA monitor facility?  (Default = No)   N
   Enable Promiscuous mode?        (Default = Yes)  Y
   Enable Diagnostic Logging?      (Default = Yes)  Y
   Enable Self-test Logging?       (Default = Yes)  Y
   Enable NI RBD Logging?          (Default = Yes)  Y
   Enable XMI RBD Logging?         (Default = Yes)  Y
   Enable XNA RBD Logging?         (Default = Yes)  Y

❹ Do you wish to modify any of these parameters?  [(No), Yes] Yes

❺ Enable Remote Boot?             (Default =  No)    [(No), Yes]
   Enable Remote DEMNA console?    (Default = Yes)    [(Yes), No]
   Enable Local DEMNA console?     (Default = Yes)    [(Yes), No]
   Enable DEMNA monitor facility?  (Default = No)     [(No), Yes]
   Enable Promiscuous Mode?        (Default = Yes)    [(Yes), No]
   Enable Diagnostic Logging?      (Default = Yes)    [(Yes), No]
   Enable Self-test Logging?       (Default = Yes)    [(Yes), No]
   Enable NI RBD Logging?          (Default = Yes)    [(Yes), No]
   Enable XMI RBD Logging?         (Default = Yes)    [(Yes), No]
   Enable XNA RBD Logging?         (Default = Yes)    [(Yes), No]

❻ Enter remote DEMNA console password (up to 8 alpha-
numeric characters):

   OK to modify EEPROM parameters?  [(No), Yes] Yes

   Are you sure?  [(No), Yes] Yes
```

```
  Writing new parameters to EEPROM...
  .. End of run, 0 errors detected, pass count is 1,
     time is  20-FEB-1990 11:14:17.08
⑦ DS>EXIT
```

① If run in stand-alone mode, EVGDB will execute the DEMNA self-test. If the self-test fails, EVGDB will display an error message and continue.

② On VAX 6000 systems, ensure that the key switch is in the update position before responding "Yes".

On VAX 9000 systems, ensure that the SPU access switch is set to LOCAL/SPU or REMOTE/SPU and then issue the following command to enable EEPROM updating:

```
SET XMI_UPDATE/XMI:n ON     !Where n = XMI card cage number
```

③ Answer "No" in most cases

④ If the reply is "No", EVGDB will display the following and then exit to VAX/DS:

```
No parameter changes made.
.. End of run, 0 errors detected, pass count is 1,
   time is  20-FEB-1990 11:14:58.77
DS>
```

⑤ Current flag settings are enclosed in parentheses.

⑥ If more than eight characters are entered, the console password prompt is redisplayed. If fewer than eight characters are specified, the password is null filled. If a password is not specified, (RETURN pressed), the default password *XNABOARD* is used.

⑦ On VAX 6000 systems, return the key switch to its former position (Halt or Auto Start). On VAX 9000 systems, issue the following console command to disable EEPROM updating and then set the SPU access switch to the appropriate position:

```
SET XMI_UPDATE/XMI:n OFF    !Where n = XMI card cage number
```

4.4.2 Updating the EEPROM Firmware (VAX 9000 System)

```
DS>LOAD EVGDB
DS>ATTACH XJA HUB XJA0 0
DS>ATTACH DEMNA XJA0 EXA0 3
DS>SELECT ALL
DS>START/SECTION=UPDATE

Program: EVGDB -
DEMNA EEPROM Update Utility, revision 1.1, 6 tests

Testing: _EXA0

Please insure that Front Panel Switch is in Update position.
Ready [(Yes), No]

Data Image file to be loaded? <EVGDBQ.BIN>

Searching...
Load complete.

Data Image firmware rev: 0601 04-APR-1990

Do you wish to clear the EEPROM error log?  [(No), Yes] No

Reading parameters from EEPROM...

EEPROM firmware rev:     0601 04-APR-1990

DEMNA Serial Number:     *SG909T1488*

Enable Remote Boot?             (Default =  No)  N
Enable Remote DEMNA console?    (Default = Yes)  Y
Enable Promiscuous Mode?        (Default = Yes)  Y

Do you wish to modify any of these parameters?  [(No), Yes] No

No parameter changes made.

Reading parameters from EEPROM...

Writing new image to RAM...
Reading image from RAM...
Writing RAM image to EEPROM...

Reading parameters from EEPROM...

Data Image firmware rev: 0601 04-APR-1990

EEPROM firmware rev:     0601 04-APR-1990

DEMNA Serial Number:     *SG909T1488*

Enable Remote Boot?             (Default =  No)  Y
Enable Remote DEMNA console?    (Default = Yes)  Y
Enable Promiscuous Mode?        (Default = Yes)  Y

Reading EEPROM image...
Verification complete.
```

```
.. End of run, 0 errors detected, pass count is 1,
   time is  9-APR-1990 13:30:18.50
DS>EXIT
```

4.5 DIAGNOSTIC PATCH MECHANISM

The DEMNA EEPROM contains a 2-Kbyte region for holding patches made to the diagnostic code in the EPROM. When diagnostic code is executed from the EPROM, checks are made at strategic points to determine if the EEPROM contains patch code. If the patch code is present, the code is executed from the EEPROM instead of the corresponding segment in the EPROM.

When the DEMNA self-test or any RBD is run, a checksum is calculated for the diagnostic patch area in EEPROM and compared with a valid checksum. If the checksum test fails, the diagnostic patch area is declared invalid. In this case, no diagnostic patches are executed. If the checksum failure occurs during self-test, the self-test fails, and the Bad Diagnostic Patch Table bit in the XPUD register is set. If the checksum failure occurs when an RBD is invoked, the following message is displayed on the console before the RBD is executed:

```
BAD_PATCH
```

CHAPTER 5

5 DEMNA Console Monitor Program

5.1 OVERVIEW

The console monitor program is a EEPROM resident program which allows users on the network to monitor DEMNA operations and network traffic.

The console monitor program consists primarily of 12 interactively invoked screens (displays) that indicate current operating parameters and errors. The console monitors over 100 parameters. These parameters are updated every 3 seconds on-screen while being displayed.

In addition to displaying and updating key operational and diagnostic parameters, the console monitor program allows a user to examine the contents of DEMNA memory locations and registers.

The console monitor includes an online help facility.

Security Features

- Password protected
- Access allowed by only one user at a time
- System manager can disable program entirely, or deny access from a remote network node

Parameters that control access to the monitor can be changed with the EEPROM Update Utility, EVGDB.

5.2 CONNECTING TO THE CONSOLE MONITOR PROGRAM

A user can access the DEMNA console monitor program from a terminal:

- Attached directly to the DEMNA (physical console)
- On the local node (DEMNA's node)
- On a remote node

Accessing the monitor program from a terminal other than physical console requires that one of the following be used to make the connection:

- Network Control Program (NCP)
- A console connection program

5.2.1 Using the Physical Console

The only setup required to access the console monitor program from the physical console is to connect the terminal cable to the DEMNA's XMI slot and set the terminal baud rate to 19.2K baud. The console monitor prompt (XNA>) is displayed when the terminal is powered on.

5.2.2 Using the Network Control Program (NCP)

The following examples show how to setup NCP parameters to allow access to the console monitor program from a terminal connected to the local node and from a terminal connected to a remote node.

Note that DECnet must be running for NCP fields to be valid.

Terminal on the Local Node (DEMNA's node)

```
  $MCR NCP
① NCP> SHOW NODE node_name

② Node Volatile Summary as of 12-SEP-1990 13:02:52
  %NCP-W-UNRCMP, Unrecognized component, Node

③ NCP> SHOW NODE DECnet_address

④ Node Volatile Summary as of 13-JUL-1990

  Node      State      Active      Delay      Circuit      Next
                       Link                                Node

⑤ NCP> SET NODE DECnet_address NAME node_name
  NCP> DEF NODE DECnet_address NAME node_name
  NCP> SET NODE node_name HARD ADDR address
  NCP> DEF NODE node_name HARD ADDR address
  NCP> SET NODE node_name SERVICE PASSWORD 584E41424F415244   ! Default
  NCP> DEF NODE node_name SERVICE PASSWORD 584E41424F415244
  NCP> SET NODE node_name SERVICE CIRCUIT circuit_name
  NCP> DEF NODE node_name SERVICE CIRCUIT circuit_name
```

① **Verify that the node name to be created is unique**

② **Message displayed if the node name is unique**

③ **Verify that the DECnet address to be assigned is unique**

④ **Message displayed if the DECnet address is unique**

⑤ **Commands to create or modify parameters in the volatile (SET) database or the permanent (DEFINE) database:**

Parameter	**Description**
DECnet_address	assigned DECnet address
node_name	Ethernet node name of DEMNA
address	DEMNA default physical address (DPA) on the Ethernet
circuit_name	service circuit for the system

Terminal on a Remote Node

```
$MCR NCP
NCP> SET NODE node_name HARD ADDR address
NCP> DEF NODE node_name HARD ADDR address
NCP> SET NODE node_name SERVICE PASSWORD password
NCP> DEF NODE node_name SERVICE PASSWORD password
NCP> SET NODE node_name CIRCUIT NAME circuit_name
NCP> DEF NODE node_name CIRCUIT NAME circuit_name
```

Parameter	Description
node_name	Ethernet node name of DEMNA
address	DEMNA default physical address (DPA) on the Ethernet
circuit_name	service circuit for the system
password	password for the DEMNA console monitor program. Default: 584E41424F415244

5.2.3 Using the Console Connection Program

The console connection program is only used if NCP is not available.

```
$ MACRO CONSOLE              !compile and link the program
$ LINK CONSOLE
$ ASSIGN Ethernet_device CONSOLE$DEVICE
$ RUN CONSOLE
XNA>                         !DEMNA console prompt displayed
                             !if the connection is successful
```

Refer to the *DEC LANcontroller 400 Technical Manual* for a listing of CONSOLE.MAR, an assembly language program which can be used to access the console monitor program if NCP is not available.

Parameter	Description
Ethernet_device	device number for the user's Ethernet node

5.3 INVOKING AND EXITING THE CONSOLE

Using NCP

Refer to Section 5.2 for information on connecting to the Console Monitor Program, then invoke the console as follows:

```
$MCR NCP
NCP> CONNECT NODE node_name      !Ethernet node_name of DEMNA
Console connected (press CTRL/D when finished)
XNA>
```

Note that if the service password was not supplied when the console was set up (Section 5.2), the user must supply the service password:

```
NCP> CONNECT NODE node_name SERVICE PASSWORD password
```

If NCP cannot connect to the console, it will return an error message. For more information refer to the *VMS Network Control Program Manual*.

Using the Console Connection Program

Before using the console connection program, the user must compile and link the program as follows:

```
$ MACRO CONSOLE
$ LINK CONSOLE
```

Refer to the *DEC LANcontroller 400 Technical Manual* for a listing of CONSOLE.MAR.

Issue the following commands to invoke the console:

```
$ASSIGN Ethernet_device CONSOLE$DEVICE   ! device number for user's
                                         ! Ethernet node
$RUN CONSOLE
XNA>
```

Exiting the Console

To exit the console, enter CTRL/D

5.4 CONSOLE COMMANDS

Table 5-1 DEMNA Console Commands

BLANK	**Syntax:** `BLANK` Clears the screen and displays the console prompt (XNA>).
EXAMINE	**Syntax:** `EXAMINE [/qual] [parameter]` Displays the contents of the specified location in DEMNA I/O or memory space. **/qual** Command Qualifiers: */NUMBER=n* — Displays the next *n* longwords. */REGISTER* — Displays the gate array registers. **parameters** Command Parameters: *. (period)* — Displays the contents of the current location. *address* — Displays the contents of a longword location.
HELP	**Syntax:** `HELP [parameter]` Displays information on the EXAMINE and SHOW console commands, as well as the console command language control characters. **parameters** Command Parameters: *command* — Displays help information for the EXAMINE command or the SHOW command. *controlchar* — Displays help information for the console command language control characters.
SHOW	**Syntax:** `SHOW parameter` **parameters** Command Parameters: See Table 5-2

Table 5–1 (Cont.) DEMNA Console Commands

T/R	Syntax: `T/R` Invokes the DEMNA diagnostic monitor from which the DEMNA Rom-based diagnostics (RBDs) are run. **Restriction:** **The T/R command is valid only when entered from the physical console attached directly to the DEMNA module or when the DEMNA is in the uninitialized state.**

Table 5–2 DEMNA Console SHOW Command Parameters

BUS	Displays the configuration of the XMI system containing the DEMNA.
ERROR H*n*	Displays the fatal error block specified by *n* (integer from 1 to 5).
ERROR S*n*	Displays the nonfatal error block specified by *n* (integer from 1 to 5).
HISTORY [*n*]	Displays the error summary stored in the DEMNA EEPROM. (integer from 1 to 31) If a value for *n* is supplied, the data for only that error is displayed. If a value is not supplied, a summary of all errors recorded in the EEPROM is displayed.
IMAGE	Displays the firmware revision number and date for the EEPROM image and the EPROM image.
NETWORK	Displays a continuously updated summary of network activity for the six most active Ethernet users and the seven most active Ethernet nodes.

Table 5–2 (Cont.) DEMNA Console SHOW Command Parameters

STATUS	Displays a continuously updated screen that includes the following: • Statistical information on the DEMNA's use of the network • Data link counters • Percentage of DEMNA CPU time used • Error summary counters • Number of DEMNA-internal buffers in use • Percentage of XMI traffic generated • Statistical information on the use of the entire network **Qualifiers:** */ERROR* — Displays a continuously updated screen that includes the following: • Transmit error counters • Receive error counters • LANCE counters • Date and time of error */INTERVAL* — Displays the same screen as the SHOW STATUS command. The only difference between the two screens is the time interval for which the NI counters and the Error Summary counters record events.
USER	Displays the setup parameters for users defined to the DEMNA port.
XPUD	Displays the DEMNA Power-up Diagnostic (XPUD) Register.

5.5 CONSOLE CONTROL KEYS

Table 5–3 DEMNA Console Control Keys

Key	Function
CTRL/A	Alternates between the Status screen and the Status/Error screen or between the Network screen and the Accumulated Network screen.
CTRL/D	Disconnects the console and exits to the system prompt. Has no effect on the DEMNA's physical console.
CTRL/E	Alternates between the Status screen and the Status/Interval screen or between the Interval Status/Error screen and the Accumulated Status/Error screen.
	If none of these screens are displayed, entering the control character invokes the Status screen.
CTRL/L	Retrieves the last console command line entered.
CTRL/U	Clears the current command line.
CTRL/W	Refreshes the screen when the Status, Status/Error, Status/Interval, or Network screen is displayed.
	If none of these screen are displayed, entering the control character clears the screen and invokes the Status screen.

5.6 DEMNA STATUS SCREENS

Status and Status/Interval Screens

```
-- 08-00-2B-00-00-01 -- Status -- 01-AUG-1989 19:01:19 -- Uptime:   01:43:35

-- NI Statistics -------   -- NI Counters ---------   -- Process --     --XMI---
Bytes/Pk ............. 64   BytesSnt .... 6327255447   Null      88.0%   0    0.0%
Bytes/Xmt ............ 64   BytesRcv .... 6327084034   Port       0.7%   1    0.0%
Bytes/Rcv ............ 64   MbytesSnt ......... 20470   Xmt-Ln     2.8%   2    0.0%
Pk/Sec .............. 510   MbytesRcv ............ 0   Xmt-Hs     2.0%   3    0.0%
Xmt/Sec ............. 255   PkSnt .......... 17464886   Rcv-Ln     1.6%   4    0.0%
Rcv/Sec ............. 255   PkRcv .......... 17462507   Rcv-Hs     3.0%   5    0.0%
MBaudRate ..... 0.274471   MPkSnt .............. 230   Com-Hs     0.0%   6    0.0%
Interrupts ... 104599634   MPkRcv ................ 0   Mon        1.0%   7    0.0%
Interrupts/Sec ....  255                               Cons       0.4%   8    0.0%
                                                                        9    0.0%
-- Total NI Traffic ----   -- Error Summary -------   --Buffers--      A    0.0%
Bytes/Pk ............ 237   Xmt/Wire ............  0   Rcv ....   0    B    0.0%
Pk/Sec ............. 1418   Rcv/Wire ............  0   Xmt ....   1    C   11.7%
ThisNI + Other = TotBaud   Rcv/Validation ......  1                    D    0.0%
  3.5% + 26.0% =   29.5%   Rcv/NoBuffers .......  0   --XNA Bus---     E   89.3%
                                                       LANCE    9.3%    F    0.0%
                                                       XMAGA    0.0%
```

Status/Error Screen

```
-- 08-00-2B-00-00-01 -- Status -- 01-AUG-1989 19:40:45 -- Uptime:    2:23:01

-- Rcv Counters --------   -- Xmt Counters --------   -- Lance Counters ------
BytesRcv .... 6327084034   BytesSnt .... 6327280690   Lan/Restart .........  0
PkRcv ......... 17462507   PkSnt ......... 17465275   Lan/UOflo ...........  0
Rcv/MCAUrfd .........  0   Xmt/Def ............. 769   Lan/TRxoff ..........  0
Rcv/SizeFilter ......  0   Xmt/One ............. 123   Lan/Merr ............  0
Rcv/SrcMCA ..........  0   Xmt/Mul ............. 132   Lan Tx/Rx ...........  0
Misc/Cntl ...........  0   Xmt/Rtry ............   0   Rcv/Buffer ..........  0
Rcv/Invalid .........  0   Xmt/LCar ............   0   Rcv/NoSTP ...........  0
Rcv/Short802 ........  0   Xmt/LCol ............   0   -- Misc Counters ------
Rcv/Long802 .........  0   Xmt/MLen ............   0   Err/HostXfer ........  0
Rcv/Missed ..........  0   Xmt/CTest ...........   0   RX/NoRxBuf ..........  0
Rcv/Der .............  0   Xmt/Timeout .........   0   RX/XmtRngFull . .....  0
Rcv/NoRcvBuf ........  0   ---------------- Saved Error Data ------------------
Rcv/Stale ...........  0   Rtry at ........... None   LCol at ........... None
Rcv/Ubus ............  0   LCar at ........... None   CTst at ........... None
Rcv/Sbus ............  0   Sbus at ........... None
Rcv/Crc+Frame .......  0   Crc  at ........... None
Rcv/Mlen ............  0   Mlen at ........... None
Rcv/Urfd ............  1   01-AUG-1989 08:02:05  60-02 MopRC 11.111
```

GSF_1883_89_MPS

Figure 5–1 DEMNA Status Screens

Table 5–4 DEMNA Status and Status/Interval Screens—Parameter Definitions

Parameter	Description
Ethernet address	DEMNA's actual physical address (APA)
Date and time	current date and time
Uptime	time since the DEMNA was last reset
NI Statistics	
Bytes/Pk	average number of bytes per packet (transmit or receive) during the last 3 seconds
Bytes/Xmt	average number of bytes per transmit packet during the last 3 seconds
Bytes/Rcv	average number of bytes per receive packet during the last 3 seconds
Pk/Sec	number of packets transmitted and received per second during the last 3 seconds
Xmt/Sec	number of packets transmitted per second during the last 3 seconds
Rcv/Sec	number of packets received per second during the last 3 seconds
MBaudRate	megabaud rate for the DEMNA (transmit plus receive) during the last 3 seconds
Interrupts	number of DEMNA-generated interrupts (both error and port interrupts)
Interrupts/sec	number of DEMNA-generated interrupt that occurred during the last 3 seconds

Table 5–4 (Cont.) DEMNA Status and Status/Interval Screens—Parameter Definitions

Parameter	Description
Total NI Traffic	
Bytes/Pk	average number of bytes per packet on the network during the last 3 seconds
Pk/Sec	average number of packets per second on the network during the last 3 seconds
ThisNI	percentage of network bandwidth consumed by DEMNA-related traffic during the last 3 seconds
Other	percentage of network bandwidth consumed by traffic related to other nodes during the last 3 seconds
TotBaud	percentage of network bandwidth consumed by all nodes during the last 3 seconds (sum of ThisNI and Other)

Table 5-4 (Cont.) DEMNA Status and Status/Interval Screens—Parameter Definitions

Parameter	Description
NI Counters	
BytesRcv	number of user data bytes received without error Does not include header or CRC bytes.
BytesSnt	number of user data bytes transmitted without error Does not include header or CRC bytes.
MBytesRcv	number of user data bytes in multicast packets received without error Does not include header or CRC bytes.
MBytesSnt	number of user data bytes in multicast packets transmitted without error Does not include header or CRC bytes.

Table 5–4 (Cont.) DEMNA Status and Status/Interval Screens—Parameter Definitions

Parameter	Description
NI Counters	
PkSnt	number of packets transmitted without error This number includes: • Xmt/Def—packets successfully sent after transmission was deferred because of Ethernet traffic • Xmt/One—packets transmitted without error after a single collision-and-backoff sequence • Xmt/Mul—packets transmitted on the third or subsequent attempt
PkRcv	number of packets received without error
MPkSnt	number of multicast packets transmitted without error This number includes: • Xmt/Def—packets successfully sent after transmission was deferred because of Ethernet traffic • Xmt/One—packets transmitted without error after a single collision-and-backoff sequence • Xmt/Mul—packets transmitted on the third or subsequent attempt
MPkRcv	number of multicast packets received without error

Table 5–4 (Cont.) DEMNA Status and Status/Interval Screens—Parameter Definitions

Parameter	Description
Error Summary	
Xmt/Wire	sum of the following transmit errors: • Maximum number of retries exceeded (Rtry) • Lost carrier (LCar) • Late collision (LCol) • Maximum length exceeded (MLen) • Collision check test (CTest) • Transmit timeout (Timeout)
Rcv/Wire	sum of the following receive errors: • CRC error (Crc) • Framing error (Frame) • Maximum length exceeded (MLen) • Invalid (Invalid)
Rcv/Validation	number of receive packets that had one or more filtering/validation errors
Rcv/NoBuffers	number of receive packets discarded due to one or more resource errors

Table 5–4 (Cont.) DEMNA Status and Status/Interval Screens—Parameter Definitions

Parameter	Description
Process Statistics	
Null	percentage of CVAX time used by the kernel or scheduler, or both in the last 3 seconds
Port	percentage of CVAX time used by the Port firmware process in the last 3 seconds
Xmt-Ln	percentage of CVAX time used by the LanceXmt firmware process in the last 3 seconds
Xmt-Hs	percentage of CVAX time used by the HostXmt firmware process in the last 3 seconds
Rcv-Ln	percentage of CVAX time used by the LanceRcv firmware process in the last 3 seconds
Rcv-Hs	percentage of CVAX time used by the HostRcv firmware process in the last 3 seconds
Cmd-Hs	percentage of CVAX time used by the Command firmware process in the last 3 seconds
Mon	percentage of CVAX time used by the Monitor firmware process in the last 3 seconds
Cons	percentage of CVAX time used the Console firmware process in the last 3 seconds
Buffers in Use	
Rcv	number of DEMNA-internal receive buffers in use during the last 3 seconds
Xmt	number of DEMNA-internal transmit buffers in use during the last 3 seconds

Table 5–4 (Cont.) DEMNA Status and Status/Interval Screens—Parameter Definitions

Parameter	Description
XNA Bus	
LANCE	percentage of total XNA memory bus traffic generated by the LANCE in the last 3 seconds
XNAGA	percentage of total XNA memory bus traffic generated by the DEMNA gate array in the last 3 seconds
XMI	
0 . . . F	percentage of existing XMI bus traffic generated by the XMI node (0–F) in the last 3 seconds

Table 5–5 DEMNA Status/Error Screen—Parameter Definitions

Parameter	Description
Ethernet Address	DEMNA's actual physical address (APA)
Date and Time	current date and time
Uptime	time since the DEMNA was last reset

Table 5–5 (Cont.) DEMNA Status/Error Screen—Parameter Definitions

Parameter	Description
Rcv Counters	
BytesRcv	number of user data bytes received without error Does not include header or CRC bytes.
PkRcv	number of packets received without error
Rcv/MCAUrfd	number of multicast packets discarded because the packet's user designator was not enabled for any of the users defined to the port
SizeFilter	number of receive packets longer than the maximum size requested by the destination user
Rcv/SrcMCA	number of packets received with multicast source addresses
Misc/Cnt1	Miscellaneous counter 1 (reserved for future use)
Rcv/Invalid	number of 802 receive packets that were too short to determine anything from
Rcv/Short802	number of 802 packets whose length was shorter than what was stated in the Length field
Rcv/Long802	number of 802 packets whose length was longer than what was stated in the Length field
Rcv/Missed	number of times the LANCE reported a missed error
Rcv/Dor	number of receive packets discarded by the firmware because the DEMNA was unable to keep up with the data rate
NoRcvBuf	number of times the port looked for, but did not obtain, a system buffer
Rcv/Stale	number of receive packets discarded because a system buffer was unavailable
Rcv/Ubua	number of receive packets discarded because a user buffer was unavailable

Table 5–5 (Cont.) DEMNA Status/Error Screen—Parameter Definitions

Parameter	Description
Rcv Counters	
Rcv/Sbua	number of receive packets discarded by the firmware because a system buffer was unavailable
Rcv/Crc+Frame	number of receive packets that had either a CRC error or a framing error
Rcv/MLen	number of Ethernet receive packets whose length is longer than 1518 bytes
Rcv/Urfd	number of nonmulticast receive packets discarded because the user designator was not recognized by the port

Table 5–5 (Cont.) DEMNA Status/Error Screen—Parameter Definitions

Parameter	Description
Xmt Counters	
BytesSnt	number of user data bytes transmitted without error Does not include header of CRC bytes.
PkSnt	number of packets transmitted without error This number includes: • Xmt/Def • Xmt/One • Xmt/Mul
Xmt/Def	number of packets transmitted without error after transmission is delayed once
Xmt/One	number of packets transmitted without error after a single collision-and-backoff sequence
Xmt/Mul	number of packets transmitted without error after more than one collision-and-backoff sequence
Xmt/Rtry	number of packets not transmitted because the maximum (16) transmission retries was exceeded
Xmt/LCar	number of packets that failed transmission because the LANCE did not detect the carrier during transmission
Xmt/LCol	number of packets that failed transmission because of a late collision
Xmt/MLen	number of packets that failed transmission because the total packet length was long than the maximum allowable size
Xmt/CTest	number of times the Collision Detect signal was not detected by the LANCE
Xmt/Timeout	number of times the LANCE failed to complete transmission of a packet with 800 milliseconds

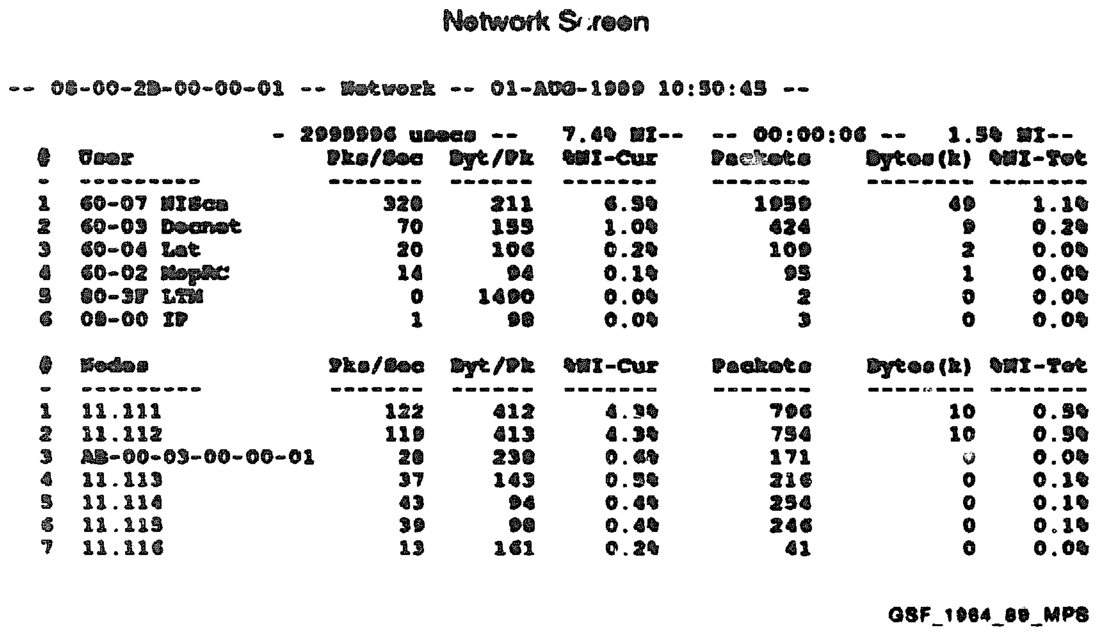
Table 5–5 (Cont.) DEMNA Status/Error Screen—Parameter Definitions

Parameter	Description
LANCE Counters	
Lan/Restart	number of times the DEMNA firmware restarted the LANCE
LAN/UOflo	number of transmit underflow error pule the number of receive overflow error detected by the LANCE
Lan/TRxoff	number of times the firmware noticed the LANCE transceiver or receiver was turned off when it should have been turned on
Lan/Merr	number of memory errors detected by the LANCE
Lan/TxRx	number of nonloopback receive packets whose source address is the same as the DEMNA's actual physical address (APA)
Rcv/Buffer	number of times the LANCE reports a buffer error in a receive buffer descriptor
Lan/NoSTP	number of buffer descriptors that did not have a start-of-packet indicator
Miscellaneous Counters	
Err/HostXfer	number of transfer errors that occurred during a transfer to or a transfer from host memory
RX/NoRxBuf	number of packets not transmitted in response to a MOP or loopback message because no LANCE transmit buffers were available
RX/XmtRngFull	number of packets not transmitted in response to a MOP or loopback message because no LANCE transmit ring entries were available

Table 5–5 (Cont.) DEMNA Status/Error Screen—Parameter Definitions

Parameter	Description
Saved Error Data	
Rtry at	date and time at which the last Xmt/Rtry error occurred
LCar at	date and time at which the last Xmt/LCar error occurred
Sbus at	date and time at which the last Rcv/Sbus error occurred
Crc at	date and time at which the last Rcv/Crc error occurred The "Crc at" field records all bad-CRC packets even those not addressed to the DEMNA. The Rcv/Crc+Frame counter records only packets addressed to the DEMNA.
Mlen at	date and time at which the last Rcv/Mlen error occurred
Urfd at	date and time at which the last Rcv/Urfd error occurred
LCol at	date and time at which the last Xmt/LCol error occurred
CTst at	date and time at which the last Xmt/CTst error occurred

5.7 DEMNA NETWORK SCREEN



```
                              Network Screen

-- 08-00-2B-00-00-01 -- Network -- 01-AUG-1989 10:50:45 --

                         - 2999996 usecs --   7.4% NI--  -- 00:00:06 --   1.5% NI--
#  User                  Pks/Sec  Byt/Pk  %NI-Cur   Packets    Bytes(k) %NI-Tot
-  ---------             -------  ------  -------   -------    -------- -------
1  60-07 NISca               320     211     6.5%      1959          49    1.1%
2  60-03 Decnet               70     155     1.0%       424           9    0.2%
3  60-04 Lat                  20     106     0.2%       109           2    0.0%
4  60-02 MopRC                14      94     0.1%        95           1    0.0%
5  80-3F LTM                   0    1490     0.0%         2           0    0.0%
6  08-00 IP                    1      98     0.0%         3           0    0.0%

#  Nodes                 Pks/Sec  Byt/Pk  %NI-Cur   Packets    Bytes(k) %NI-Tot
-  ---------             -------  ------  -------   -------    -------- -------
1  11.111                    122     412     4.3%       796          10    0.5%
2  11.112                    119     413     4.3%       754          10    0.5%
3  AB-00-03-00-00-01          28     238     0.6%       171           0    0.0%
4  11.113                     37     143     0.5%       216           0    0.1%
5  11.114                     43      94     0.4%       254           0    0.1%
6  11.115                     39      98     0.4%       246           0    0.1%
7  11.116                     13     161     0.2%        41           0    0.0%

                                                                GSF_1964_89_MPS
```

Figure 5–2 DEMNA Network Screen

Table 5–6 DEMNA Network Screen—Parameter Definitions

Parameter	Description
Ethernet address	DEMNA's actual physical address (APA)
Date and time	current date and time
usecs	length of the last interval for which the following network parameter were recorded: • Pks/Sec • Byt/Pk • %NI-Cur
% NI	percentage of maximum Ethernet bandwidth consumed by all nodes on the network during the last interval

Table 5–6 (Cont.) DEMNA Network Screen—Parameter Definitions

Parameter	Description
Time	cumulative time (in seconds) for which the following network parameters were recorded • Packets • Bytes (k) • %NI-Tot
#	User column: six network users that generated the most network traffic during the last recorded interval Nodes column: seven nodes that generated the most network traffic during the last recorded interval
User	user designator for the six most active network users
Nodes	DECnet address or Ethernet address for the seven most active network nodes
Pks/Sec	average number of packets transmitted or received per second (per user or per node)
Byt/Pk	average number of bytes transmitted or received per user or per node
%NI-Cur	percentage of maximum Ethernet bandwidth consumed by each user or by each node on the network (timing interval determined by the usecs field)
Packets	cumulative number of packets transmitted or received per user or per node
Bytes (k)	cumulative number of kilobytes transmitted or received per user or per node
%NI-Tot	percentage of maximum Ethernet bandwidth consumed by each user or each node (timing interval indicated by the Time field)

CHAPTER 6

6 DEMNA Adapter Registers

6.1 INTRODUCTION

This chapter overviews the DEMNA register structure. Included in the chapter are:

- Lists of the DEMNA registers:
  - XMI architecture
  - Port specific, XMI visible
  - Port specific, node-private
- Register bit maps
- Descriptions of selected registers

This chapter is a quick reference to DEMNA register information. Refer to the *DEC LANcontroller 400 Technical Manual* for detailed descriptions of all registers.

6.2 REGISTER TYPES

Table 6–1 DEMNA XMI Visible Registers

Mnemonic	Offset[1]	Name
XMI Architecture		
XDEV	00000	XMI device type register
XBER	00004	XMI bus error register
XFADR	00008	XMI failing address register
XCOMM	00010	XMI communications register
XFAER	0002C	XMI failing address extension register
DEMNA Port Specific		
XDP1	00100	Port data register 1
XDP2	00104	Port data register 2
XDPST	00108	Port status register
XPUD	0010C	Port power-up diagnostic register
XPCI	00110	Port control initialization register
XPCP	00114	Port control poll register
XPCS	00118	Port control shutdown register

[1]Address offset (in hex) from the node's XMI base address

Table 6–2 DEMNA Node-Private Registers

Mnemonic	Name
Gate Array Registers	
GACSR	Gate array control and status register
GAHIR	Gate array host interrupt register
GAIVR	Gate array IDENT vector register
GATMR	Gate array timer register
Datamove Registers	
DMPORn	Datamove port address register (n=0 to 3)
DMCSRn	Datamove control and status register (n=0 to 3)
DMXMIn	Datamove XMI address register (n=0 to 3)
DMNPAn	Datamove next page address register (n=0 to 3)
Peek Registers	
PKXMILn	Peek XMI low address register (n=0 or 1)
PKXMIHn	Peek XMI high address register (n=0 or 1)
PKDATAn	Peek data A register (n=0 or 1)
PKDATBn	Peek data B register (n=0 or 1)

NOTE
Table 6–2 only lists the node-private registers which are included in the DEMNA non-fatal error blocks. Refer to the *DEC LANcontroller 400 Technical Manual* for information on error reporting and the non-fatal error blocks.

6.3 REGISTER BIT DESCRIPTION CONVENTIONS

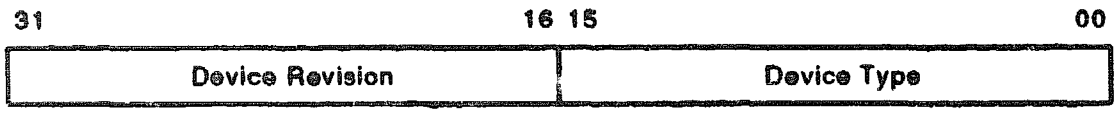
In the register description tables that follow, the access type of the bit(s) being described is denoted by a mnemonic enclosed in parentheses after the bit field name. The bit access codes are as follows:

Code	Indication
0	Bit(s) initialized to logic 0
1	Bit(s) initialized to logic 1
RO	Read-only
R/W	Read/write
R/W1C	Read/Write-1-to-clear
U	Undefined

6.4 XMI ARCHITECTURE REGISTERS

The following registers must be present in the node to support the XMI bus architecture. These registers all reside in the DEMNA gate array.

6.4.1 XMI Device Register (XDEV, bb+0000)



QSF-RC1000-XNA02-PSA

Bit(s)	Name/Description
31:16	Device revision (RO, 0)
	Identifies the DEMNA hardware and EEPROM firmware revision levels. A zero value indicates an uninitialized node.
	The high-order byte of the field is the hardware revision. The low-order byte is the EEPROM firmware revision.
31:23	Hardware revision (RO, 0)
	Encoded with a value which represents the letter code of the hardware revision level. The encoding for the first ten revisions are as follows. Note that letter codes "G", "I", and "O" are not used:

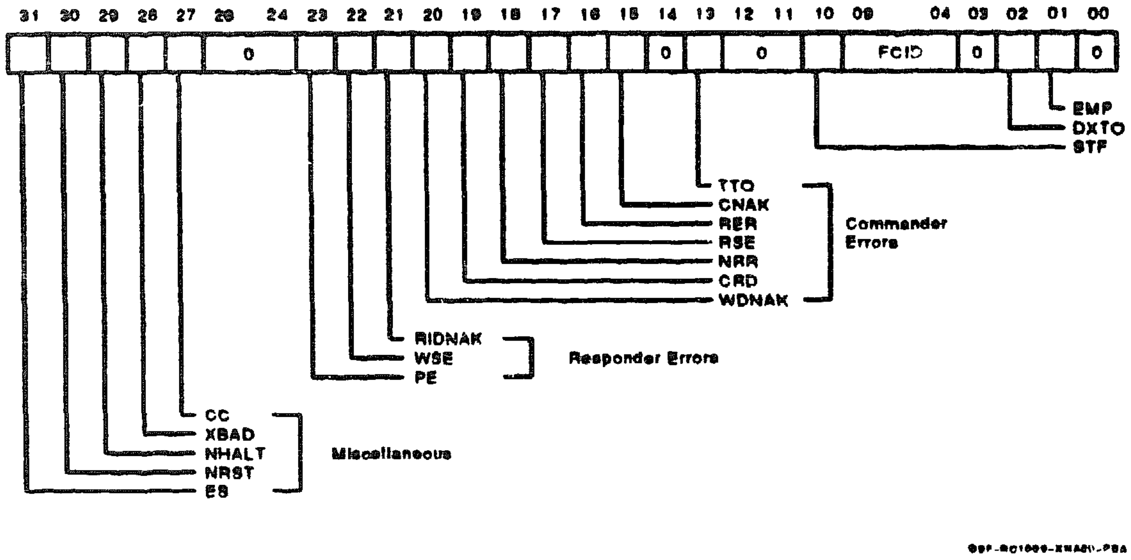
Value	Revision
01	A
02	B
03	C
04	D
05	E
06	F
08	H
0A	J
0B	K
0C	L

Bit(s)	Name/Description
24:16	EEPROM firmware revision level (RO, 0)

Value	Revision
01	01
02	02
03	03
.	
.	
.	
08	08
09	09
0A	10

15:00	Device type (RO, 0) The DEMNA device type is 0C03. A zero value indicates an uninitialized node.

6.4.2 XMI Bus Error Register (XBER, bb+0004)

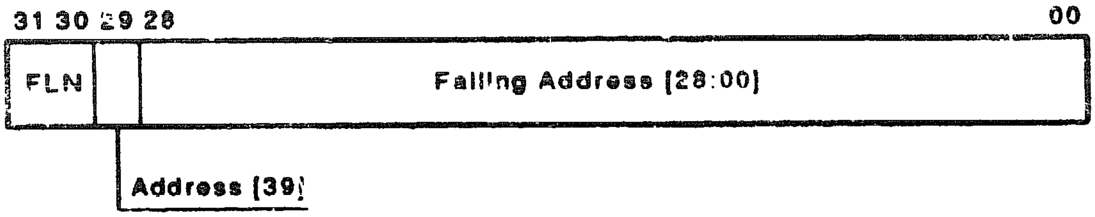


Bit(s)	Name/Description
31	Error summary (RO, 0) Logical "OR" of the error bits in the register.
30	Node reset (R/W, 0) When set by the host, initiates a power-up reset which is similar to a power-up caused by XMI DC LO. When NRST is set, the DEMNA executes its self-tests and asserts XMI BAD until the self-tests pass. Other nodes cannot access the DEMNA, and the DEMNA will not access other nodes, until self-tests pass or the maximum self-test time is exceeded. NRST is cleared on a power-up caused by XMI DC LO, but remains set if the host issued a node reset. The bit remains set in this case to indicate to the DEMNA CVAX that the host issued a node reset. DEMNA firmware clears the bit once the node reset has completed.

Bit(s)	Name/Description
29	Node halt (R/W, 0) Set by the host to force the DEMNA to execute its halt sequence and enter a quiet state. When the host clears NHALT, the DEMNA executes its restart sequence, which is similar to a power-up except that self-tests are not performed.
28	XMI bad (R/W, 1) Reflects the state of self-test fail (bit <10>) and drives the XMI BAD line. When STF is set, indicating that the DEMNA has not passed self-test, the DEMNA sets XBAD and asserts XMI BAD. When STF is cleared, the DEMNA clears XBAD and deasserts XMI BAD.
27	Corrected confirmation (R/W1C, 0) Set if the DEMNA detected a single-bit CNF error. Single-bit CNF errors are automatically corrected by the XCLOCK chip.
26:24	Not implemented, reads of these bits return a zero.
23	Parity error (R/W1C, 0) Set if the DEMNA detected a parity error on an XMI cycle. The cycle need not have been directed to the DEMNA.
22	Write sequence error (R/W1C, 0) Set if the DEMNA detected missing data cycles on a write transaction to the DEMNA.
21	Read/IDENT data NoAcK (R/W1C, 0) Set if a DEMNA initiated Read or IDENT data cycle received a NoAck confirmation.
20	Write data NoAck (R/W1C, 0) Set if a DEMNA initiated Write data cycle received a NoAck confirmation.
19	Corrected read data (R/W1C, 0) Set if the DEMNA received a CRDn read response.
18	No read response (R/W1C, 0) Set if a DEMNA initiated transaction failed due to a read response timeout.

Bit(s)	Name/Description
17	Read sequence error (R/W1C, 0) Set if a DEMNA initiated transaction failed due to a read sequence error.
16	Read error response (R/W1C, 0) Set if the DEMNA received a read error response.
15	Command NoAck (R/W1C, 0) Set if a DEMNA initiated C/A cycle received repeated NoAck confirmations for the duration of the timeout period. CNAK can result from a reference to nonexistent memory or a C/A cycle parity error and is set only if repeated attempts fail.
14	Reserved, must be zero.
13	Transaction timeout (R/W1C, 0) Set if a DEMNA initiated transaction did not complete within the timeout period.
12:11	Not implemented. Reads of these bits return a zero.
10	Seft-test fail (R/W1C, 1) Set during a power-up or node rest until the DEMNA passes self-test. Cleared when the DEMNA passes self-test.
9:4	Failing commander ID (RO, 0) Logs the commander ID of a failed transaction. FCID is logged if any of the bits <20,18:13> is set.
3	Not implemented. Reads of this bit return a zero.
2	Disable XMI timeout (RW, 0) When set, disables the DEMNA's reporting of NRR or TTO if retries are disabled.
1	Enable MORE protocol (RW, 0) When set, allows the DEMNA to set the MORE bit (XMI D <59>) during the C/A cycle of a data transfer transaction. When clear, inhibits the DEMNA from setting MORE.
0	Reserved, must be zero.

6.4.3 XMI Failing Address Register (XFADR, bb+0008)



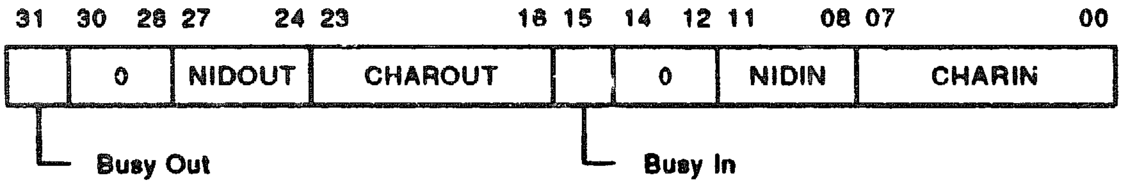
MSF_1739_89.DG

Bit(s)	Name/Description
31:30	Failing length (RO, 0) Logs the value of XMI D <31:30> (length field) during the C/A cycle of a failed transaction. FLN is loaded on every C/A cycle issued by the DEMNA and is locked if all retries of the transaction fail. The field is unlocked when the error that caused the lock is cleared.
29:0	Failing address (RO, 0) Logs the value of XMI D <29:00> (address field) during the C/A cycle of a failing transaction. The failing address is loaded on every C/A cycle issued by the DEMNA and is locked if all retries of the transaction fail. The field is unlocked when the error that caused the lock is cleared. On systems with 30-bit addressing, XMI D <29:00> are equal to addess bits <29:00>. On systems with 40-bit addressing, the XMI D bits are encoded as follows:

XMI D Bit(s)	Address Bits
57:48	38:29
29	39
28:00	28:00

The XFADR register bit-map is shown for systems with 40-bit addressing.

6.4.4 XMI Communication Register (XCOMM, bb+00010)



GSF-RC1000-XNA03-PSA

Bit(s)	Name/Description
31	Busy out (R/W) When set, indicates that the CHAROUT field contains a character that has not yet been read by the host. The host clears the bit after reading the CHAROUT field.
30:28	Reserved, bits must be zeros.
27:24	Node ID out (R/W) Written with the XMI node ID of the slot in which the DEMNA is plugged. Indicates the CHAROUT field is from the DEMNA.
23:16	Character out (R/W) Contains the character sent by the DEMNA to the host.
15	Busy in (R/W) When set, indicates that the CHARIN field contains a character that has not yet been read by the DEMNA. The DEMNA clears this bit after reading the CHARIN field.
14:12	Reserved, bits must be zeros.
11:08	Node ID in (R/W) Contains the XMI node ID of the node that wrote the data in the CHARIN field.
07:00	Character in (R/W) Contains the character sent by the host to the DEMNA.

Using XCOMM to Read the DEMNA Default Physical Address

1. Deposit FFFFFFFF into XCOMM. Examine register to obtain bytes 0 to 3 of the DPA.
2. Deposit FFFFFFFE into XCOMM. Examine register to obtain bytes 4 and 5 of the DPA.

Default Physical Address

	31 24	23 16	15 08	07 00
1st Register Read	Byte 3	Byte 2	Byte 1	Byte 0
2nd Register Read	0	0	Byte 5	Byte 4

GSF-RC1000-XNA21-PSA

Using XCOMM to Read the DEMNA Module Serial Number

1. Deposit FFFFFFFD into XCOMM. Examine register to obtain serial number bytes 0 to 3.
2. Deposit FFFFFFFC into XCOMM. Examine register to obtain serial number bytes 4 to 7.
3. Deposit FFFFFFFB into XCOMM. Examine register to obtain serial number bytes 8 to 11.

Module Serial Number Bytes

	31 24	23 16	15 08	07 00
1st Register Read	Byte 3	Byte 2	Byte 1	Byte 0
2nd Register Read	Byte 7	Byte 6	Byte 5	Byte 4
3rd Register Read	Byte 11	Byte 10	Byte 9	Byte 8

GSF-RC1000-XNA22-PSA

Using XCOMM to Invalidate the EEPROM

1. Deposit FFFFFFFA into XCOMM, then read register.
2. If XCOMM contains all zeros, EEPROM is invalidated.

Using XCOMM to Clear the EEPROM History Data

1. Deposit FFFFFFF9 into XCOMM, then read register.
2. If XCOMM contains all zeros, error history is erased.

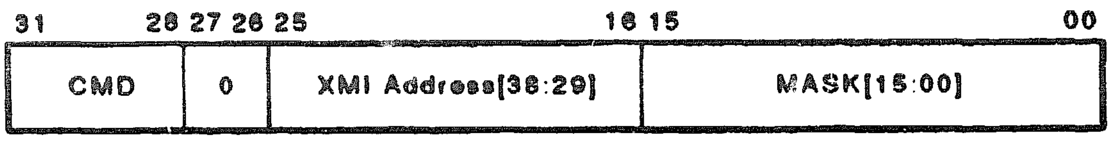
Using XCOMM to Read EEPROM History Data

History data (256 longwords) can be read from the EEPROM, one longword at a time. The offsets of history data longwords 0 through 255 are encoded with hexidecimal numbers -8 through -107 (FFFFFFF8 to FFFFFEF9), respectively. To read a longword of history data:

1. Deposit encoded offset value into XCOMM
2. Examine XCOMM

For example, to read history data longword 0, deposit FFFFFFF8 (-8 in hex) into the XCOMM; to read longword 255, deposit FFFFFEF9 (-107 in hex).

6.4.5 XMI Failing Address Extension Register (XFAER, bb+002C)



GSF_1740_89.DG

Bit(s)	Name/Description
31:28	**Failing command (RO, 0)** Logs XMI D<63:60> (command field) during the C/A cycle of a failed transaction. The field is loaded on every C/A cycle issued by the DEMNA and is locked if all retries of the transaction fail. The field is unlocked when the error that caused the lock is cleared.
27:26	Reserved, bits must be zero.
25:16	**Failing address extension (RO, 0)** Logs XMI D <57:48> (extended address field) during the C/A cycle of a failed XMI transaction or bits <38:29> of the address specified in a DMA read or write transaction. The failing address extension is loaded on every C/A cycle issued by the DEMNA and is locked if all retries of the transaction fail. The field is unlocked when the error that caused the lock is cleared. On systems with 40-bit addressing, XMI D <57:48> are the extended XMI address bits. On these systems, the XMI D bits are encoded as follows:

XMI D <57:00>	Address Bits
57:48	38:29
29	39
28:00	28:00

Bit(s)	Name/Description
15:0	Failing mask (RO, 0) Logs XMI D <47:32> (mask field) during the C/A cycle of a failed transaction or the write mask for DMA writes. The field is undefined for other transactions. Failing Mask is loaded on every C/A cycle issued by the DEMNA and is locked if all retries of the transaction fail. The field is unlocked when the error that caused the lock is cleared.

6.5 PORT SPECIFIC, XMI VISIBLE REGISTERS

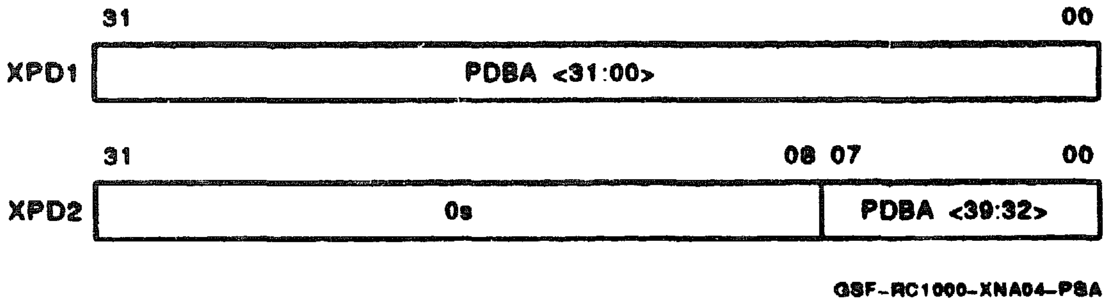
The following registers are required to communicate with the port driver. These registers all reside in the DEMNA gate array.

6.5.1 Port Data Registers (XPD1, bb+00100; XPD2, bb+00104)

XPD1 and XPD2 are accessed by the port and the port driver during the following information transfers:

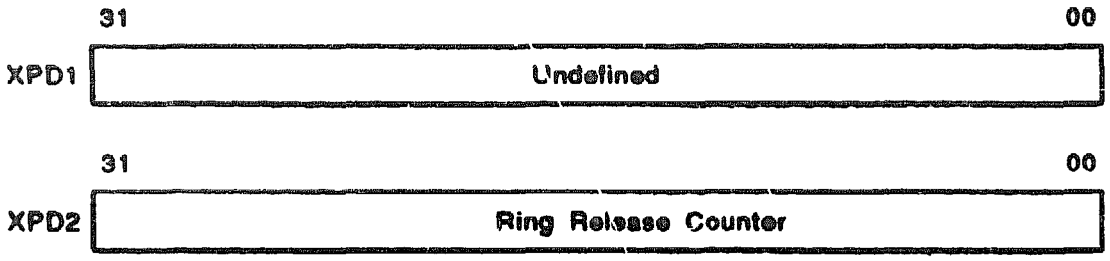
- Port Data Block (PDB) Base Address
- Ring Release Counter
- Default Ethernet Address
- Port Error Data

Port Data Block Base Address Transfer



Register	Contents
XPD1	Port data block physical base address bits <31:00>
XPD2	Port data block physical base address bits <39:32>

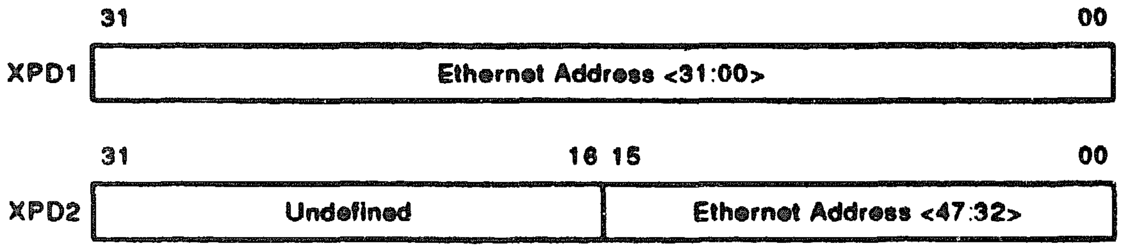
Ring Release Counter Transfer



GSF-RC1000-XNA05-PSA

Register	Contents
XPD1	Undefined
XPD2	Value which indicates the total number of commands and receive ring entries processed by the port driver since the port was last initialized. (The ring release count is always one less than the actual number of ring entries processed.)

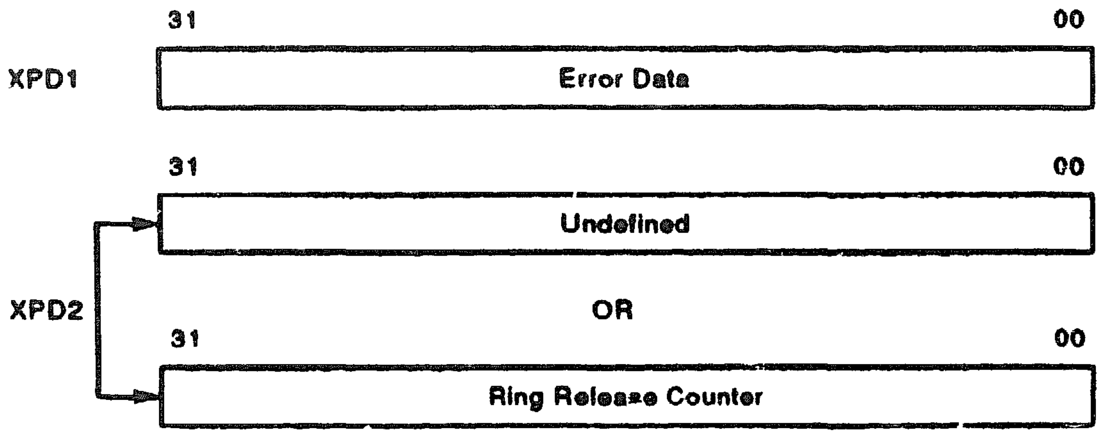
Default Ethernet Address Transfer



GSF-RC1000-XNA06-PSA

Register	Contents
XPD1	Bits <31:00> of the DEMNA's default (MAC) Ethernet address. Written by the port after power-up, node reset, or node halt/restart.
XPD2	Bits <47:32> of the DEMNA's default (MAC) Ethernet address. Writen by the port after power-up, node reset, or node halt/restart.

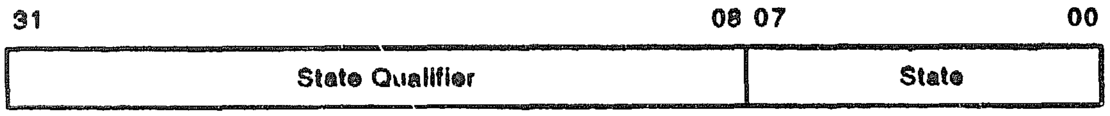
Port Error Data Transfer



GSF-RC1000-XNA07-PSA

Register	Contents
XPD1	Written with one of three values depending on the error type when a fatal port error is detected: • Invalid Port Data Block field address • Firmware PC • Current ring offset For some errors (XPST state qualifier field equal to 2, 3, 5, 6, or 7), XPD1 may contain information from a previous write.
XPD2	Value which indicates the total number of command and receive ring entries the port driver processed since the port was last initialized. When XPD1 contains an invalid PDB field address, XPD2 is undefined. When XPD1 contains the firmware PC or the current ring offset, XPD2 contains the ring release counter.

6.5.2 Port Status Register (XPST, bb+00108)



GSF-RC1000-XNA08-PSA

Bit(s)	Name/Description
31:08	**State Qualifier** Indicates the reason the port is in its current state. See the entries following this table.
07:00	**State** Indicates the current port state.

Code	State
0	**Resetting** DEMNA is executing its power-up or node halt/restart sequence.
1	**Uninitialized** DEMNA has completed its reset, power-up, node halt/restart, or shutdown sequence.
2	**Initialized** DEMNA is in its normal operating mode.

The following tables list the state qualifier codes and the contents of the XPD1 and XPD2 registers corresponding to each code.

After Power-up or Node Reset

Code	Meaning	Port State	XPST	XPD1/XPD2
0	No error	Uninitialized	00000001	MAC address
1	Self-test failed	Uninitialized	00000101	MAC address

After Node Halt

Code	Meaning	Port State	XPST	XPD1/XPD2
0	Halt/Restart complete	Uninitialized	00000001	MAC address
13	Halt/Restart in progress	Resetting	00000D00	Unchanged

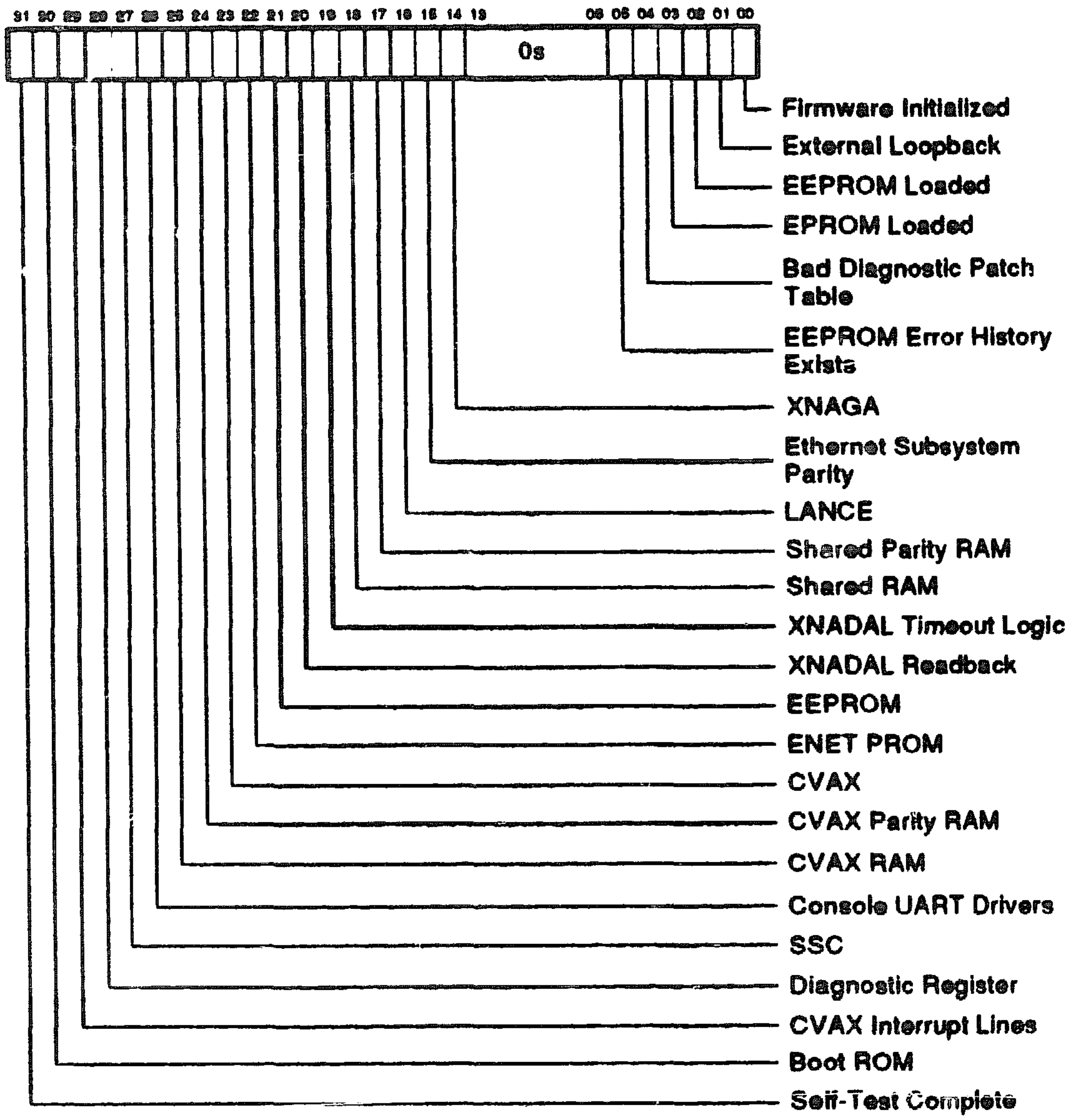
After Port Node Initialization

Code	Meaning	Port State	XPST	XPD1/XPD2
0	Initialization succeeded	Initialized	00000002	Unchanged
2	Initialization failed: failed self-test	Uninitialized	00000201	Unchanged
3	Initialization failed: invalid base address of Port Data Block (PDB) in XPD1 and XPD2	Uninitialized	00000301	Invalid base address of PDB
4	Initialization failed: contents of a PDB field not valid	Uninitialized	00000401	Address of invalid PDB field
5	Initialization succeeded but EEPROM contents invalid.	Uninitialized	00000502	Unchanged

After Port Shutdown

Code	Meaning	Port State	XPST	XPD1/XPD2
7	Initialization attempted when port not in uninitialized state.	Uninitialized	00000701	Unchanged
8	Invalid command ring	Uninitialized	00000801	Current ring offset (in bytes) in XPD1
9	Invalid receive ring	Uninitialized	00000901	Current ring offset (in bytes) in XPD1
10	Power failure	Uninitialized	00000A01	Firmware PC in XPD1
11	Unexpected firmware exception	Uninitialized	00000B01	Firmware PC in XPD1
12	Unrecoverable XMI failure, including memory error	Uninitialized	00000C01	Firmware PC in XPD1
14	Fatal firmware internal error occurred	Uninitialized	00000E01	Firmware PC in XPD1
15	Fatal firmware internal error - keep-alive counter error (firmware was in an infinite loop)	Uninitialized	00000F01	Firmware PC in XPD1
16	Firmware update completed	Uninitialized	00001001	Unchanged

6.5.3 Power-Up Diagnostic Register (XPUD, bb+010C)



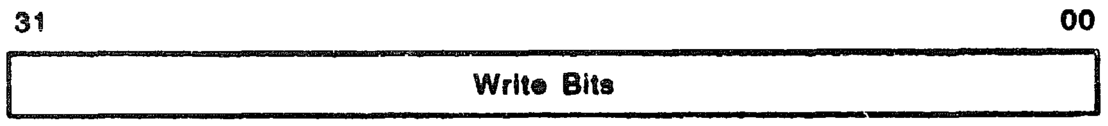
GSF_1977_89.DG

Bit(s)	Name/Description
31	Self-test complete (RO, 0) When set, indicates that the DEMNA self-tests completed and the contents of the XPUD are valid. The register contents are invalid when STC is cleared.
30	Boot ROM (RO, 0) When set, indicates that the contents of the Boot ROM (also called the EPROM) are valid.
29	CVAX interrupt lines (RO, 0) When set, indicates that the CVAX interrupt lines are not stuck (always asserted) or being driven by onboard logic.
28	Diagnostic register (RO, 0) When set, indicates that all bits in the Diagnostic Register powered-up to the correct state and can be read and written.
27	SSC (RO, 0) When set, indicates that the SSC chip is operational.
26	Console UART drivers (RO, 0) When set, indicates that the console UART drivers are operational.
25	CVAX RAM (RO, 0) When set, indicates that the CVAX RAM is operational (passed the CVAX RAM march test).
24	CVAX parity RAM (RO, 0) When set, indicates that the CVAX parity RAM is operational.
23	CVAX (RO, 0) When set, indicates that the CVAX chip is operational.
22	ENET PROM (RO, 0) When set, indicates that the contents of the ENET PROM are valid.
21	EEPROM (RO, 0) When set, indicates that the contents of the EEPROM are valid.
20	XNADAL readback (RO, 0) When set, indicates that the address latches and bus transceivers for the gate array/XNA memory bus interface are operational.

Bit(s)	Name/Description
19	XNADAL timeout logic (RO, 0) When set, indicates that the timeout logic for the gate array/XNA memory bus interface is operational.
18	Shared RAM (RO, 0) When set, indicates that the shared RAM is operational (passed the RAM march test).
17	Shared parity RAM (RO, 0) When set, indicates that the shared parity RAM is operational.
16	LANCE (RO, 0) When set, indicates that the LANCE chip is operational.
15	Ethernet subsystem parity (RO, 0) When set, indicates that the parity circuit in the Ethernet subsystem is operational.
14	XNAGA (RO, 0) When set, indicates that the gate array is operational.
13:06	Reserved, bits must be zeros.
5	EEPROM error history exists (RO, 0) When set, indicates that the EEPROM error history has one or more entries.
4	Bad diagnostic patch table (RO, 0) When set, indicates that the diagnostic patch table in EEPROM is invalid.
3	EPROM loaded (RO, 0) When set, indicates that the contents of the EPROM have been loaded into the CVAX RAM. The EPROM contains a subset of the EEPROM code. If the EEPROM fails self-test, the contents of the EPROM are loaded into the CVAX RAM. The EPROM code provides enough functionality for the CVAX to run diagnostics, update the EEPROM, and perform transmit and receive operations.

Bit(s)	Name/Description
2	EEPROM loaded (RO, 0) When set, indicates that the contents of the EEPROM have been loaded into CVAX RAM. The EEPROM contains the DEMNA operational firmware.
1	External loopback (RO, 0) When set, indicates that the DEMNA is connected to a live Ethernet or to a loopback connector and that the external loopback test passed.
0	Firmware initialized (RO, 0) When set, indicates that the DEMNA firmware is initialized.

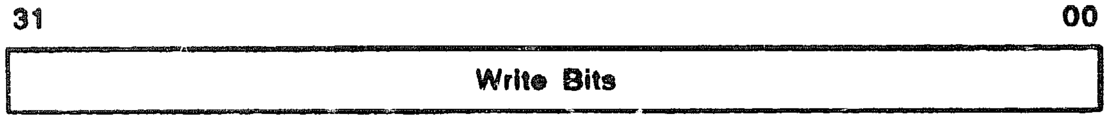
6.5.4 Port Control Initialization Register (XPCI, bb+00110)



GSF-RC1000-XNA09-PSA

Bit(s)	Name/Description
31:0	Write bits (WO to port driver, U to port) The port driver writes this register to initialize the port. The write transaction itself causes the operation to be performed; the write data are ignored.

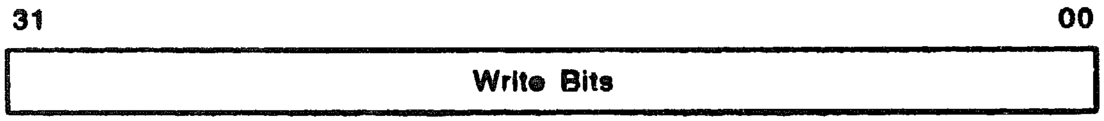
6.5.5 Port Control Poll Register (XPCP, bb+00114)



GSF-RC1000-XNA09-PSA

Bit(s)	Name/Description
31:00	**Write bits (WO to port driver, U to port)** The port driver writes this register to command the port to poll the command ring for a new entry. The write transaction itself causes the operation to be performed; the write data are ignored.

6.5.6 Port Control Shutdown Register (XPCS, bb+00118)



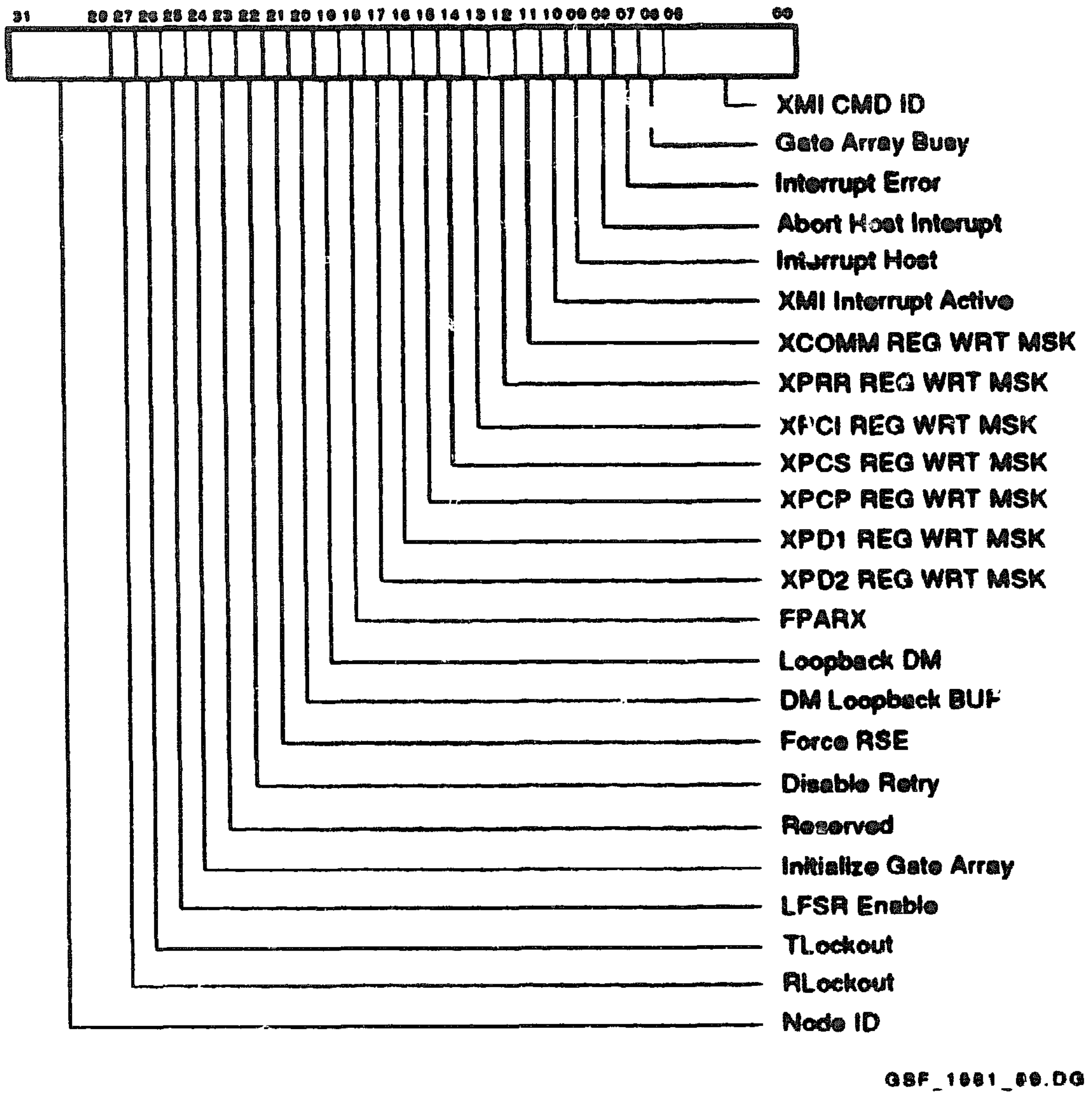
GSF-RC1000-XNA09-PSA

Bit(s)	Name/Description
31:00	**Write bits (WO to port driver, U to port)** The port driver writes this register to shut down the port. The write transaction itself causes the operation to be performed; the write data are ignored.

6.6 NODE-PRIVATE REGISTERS

The following registers are not visible on the XMI, but are included in the DEMNA non-fatal error blocks (see the *DEC LANcontroller 400 Technical Manual*). Only the GACSR is described.

6.6.1 Gate Array Control and Status Register (GACSR)



Bit(s)	Name/Description
31:28	Node ID (RO) Physical node ID (XMI Node ID<3:0>) of the DEMNA.
27	RLockout (RO) Indicates the status of the XCI Receive Lockout line (1 = line asserted).
26	TLockout (RO) Indicates the status of the XCI Transmit Lockout line (1 = line asserted).
25	Not implemented, reads of this bit return a zero.
24	Initialize gate array (RW, 0 after reset, unaffected by INIT) Provides a means for resetting the DEMNA gate array without losing error information. When this bit is set, the gate array: • Clears its control logic • Transfers the ownership bits in the Peek and Datamove registers back to firmware ownership • Clears its internal registers except for XMI and port registers The bit is cleared when initialization is finished.
23	Reserved, must be zero
22	Disable retry on NoAcks (RW, 0 after reset, unaffected by INIT) When set, causes the gate array to record an error on the first NOACK received from the XMI.
21	Force read sequence error (RW, 0 after reset, unaffected by INIT) When set, forces GRD1 onto the XCI function lines when the gate array is the responder and returning read data. This function can be used in Loopback Peek or Datamove Read operations.

Bit(s)	Name/Description
20	Datamove loopback buffer (RW, 0 after reset, unaffected by INIT) Used in conjunction with bit <19> (Loopback Datamove) to control the action of the gate array on loopback datamove operations:

Loopback datamove transmits	
Bit clear	Gate array uses the first or second quadword in the internal memory buffer depending on the byte offset of the datamove loopback address.
Bit set	Gate array uses the third or fourth quadword in the internal memory buffer depending on the byte offset of the datamove loopback address.

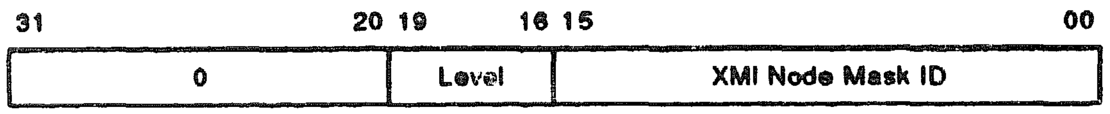
Loopback datamove receives	
Bit clear	Gate array uses the first two quadword locations in the internal memory buffer.
Bit set	Gate array uses the last two quadword locations in the buffer.

Bit(s)	Name/Description
19	Loopback datamove (RW, 0 after reset, unaffected by INIT) When set, enables XMI loopback datamove transactions. When cleared, disables these transactions.
18	Force bad XMI receive parity (RW, 0 after reset, unaffected by INIT) When set, disables XMI parity checking and forces bad receive parity on XMI P <2>. The parity error bit in XBER is set one cycle after FPARX is set.
17	XPD2 register written mask (R/W1C, 0 after reset, unaffected by INIT) Set when Port Data Register 2 (XPD2) is written by the host.
16	XPD1 register written mask (R/W1C, 0 after reset, unaffected by INIT) Set when Port Data Register 1 (XPD1) is written by the host.

Bit(s)	Name/Description
15	XPCP register written mask (R/W1C, 0 after reset, unaffected by INIT) Set when the Port Control Poll Register (XPCP) is written by the host.
14	XPCS register written mask (R/W1C, 0 after reset, unaffected by INIT) Set when the Port Control Shutdown Register (XPCS) is written by the host.
13	XPCI register written mask (R/W1C, 0 after reset, unaffected by INIT) Set when the Port Control Initialize (XPCI) Register is written by the host.
12	XPRR register written mask (R/W1C, 0 after reset, unaffected by INIT) Set when the Port Ring Release (XPRR) Register is written by the host.
11	XCOMM register written mask (R/W1C, 0 after reset, unaffected by INIT) Set when the XMI Communications (XCOMM) Register is written by the host.
10	XMI interrupt active (R0, 0 after reset and INIT) Set when the DEMNA receives an ACK for an interrupt sent to the host. Bit is cleared when the DEMNA receives an IDENT.
9	Interrupt host (R/W, 0 after reset and INIT) Set by the firmware to initiate a host interrupt. The firmware clears the bit when interrupt processing completes by resetting the gate array.
8	Abort host interrupt (R0, 0 after reset and INIT) Set by the firmware to abort a host interrupt in progress. The gate array will not issue another host interrupt until the abort is completed. The firmware clear the bit after the interrupt is successfully aborted by resetting the gate array.

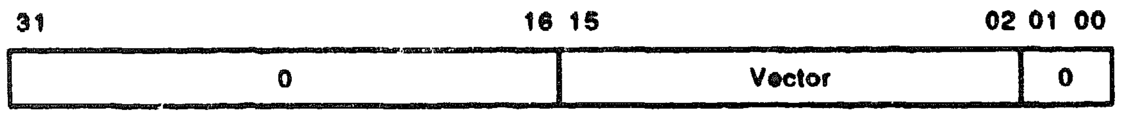
Bit(s)	Name/Description
7	Interrupt error (R/W1C, 0 after reset, unaffected by INIT) Set if a DEMNA issued interrupt resulted in an error condition. An interrupt error can be caused by one of the following: • Transaction timeout • INTERRUPT command NOACKed • IDENT returned with wrong IPL • IDENT response NOACKed
6	Gate array busy (RO, 0 after reset, unaffected by INIT) Set when the gate array is busy processing a datamove or a peek operation. This bit is an OR of all the Ownership bits in the Datamove and Peek registers.
5:0	XMI command ID (RO) Logs the XMI command ID last on the XMI bus.

6.6.2 Gate Array Host Interrupt Register (GAHIR)



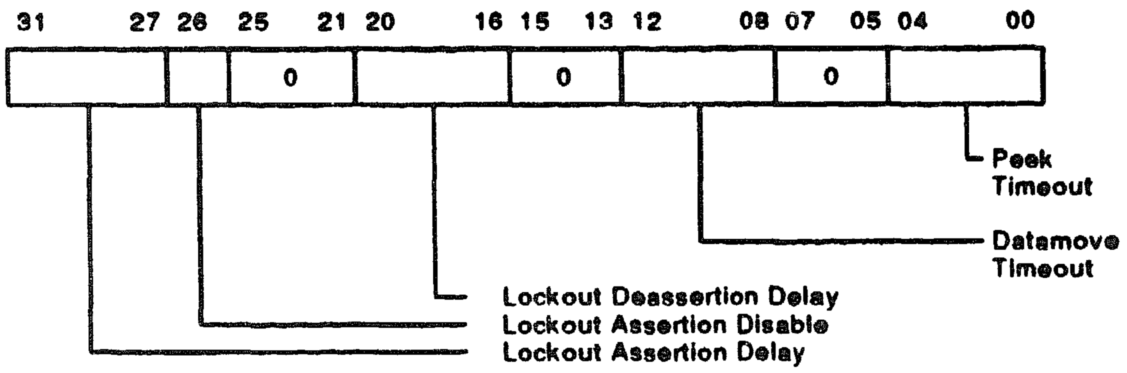
GSF-RC1000-XNA10-PSA

6.6.3 Gate Array IDENT Vector Register (GAIVR)



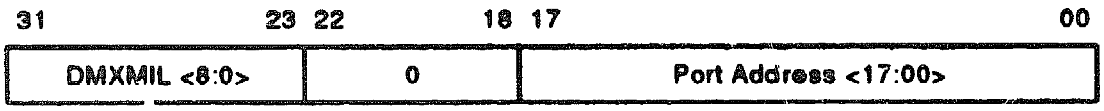
GSF-RC1000-XNA11-PSA

6.6.4 Gate Array Timer Register (GATMR)



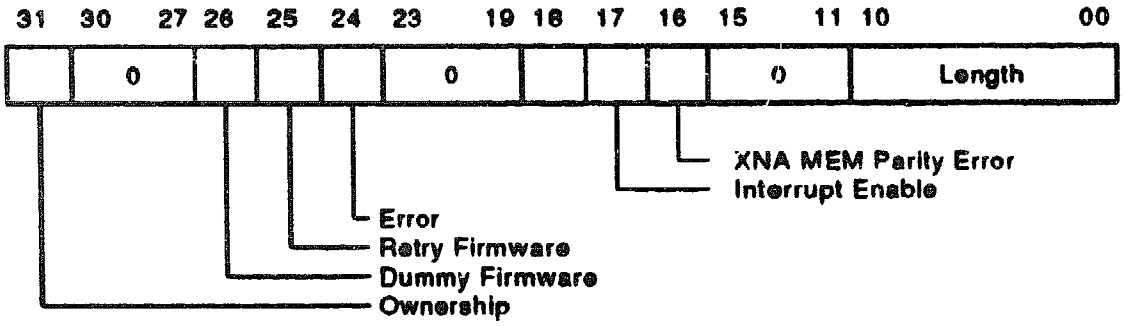
GSF-RC1000-XNA12-PSA

6.6.5 Datamove Port Address Registers (DMPORn)



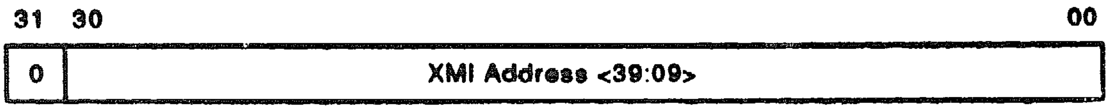
GSF-RC1000-XNA13-PSA

6.6.6 Datamove Control and Status Registers (DMCSRn)



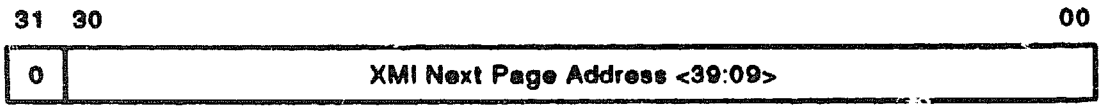
GSF-RC1000-XNA14-PSA

6.6.7 Datamove XMI Address Register (DMXMIn)



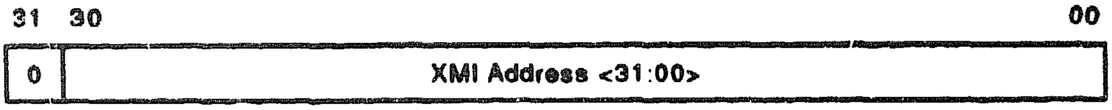
GSF-RC1000-XNA15-PSA

6.6.8 Datamove Next Page Address Register



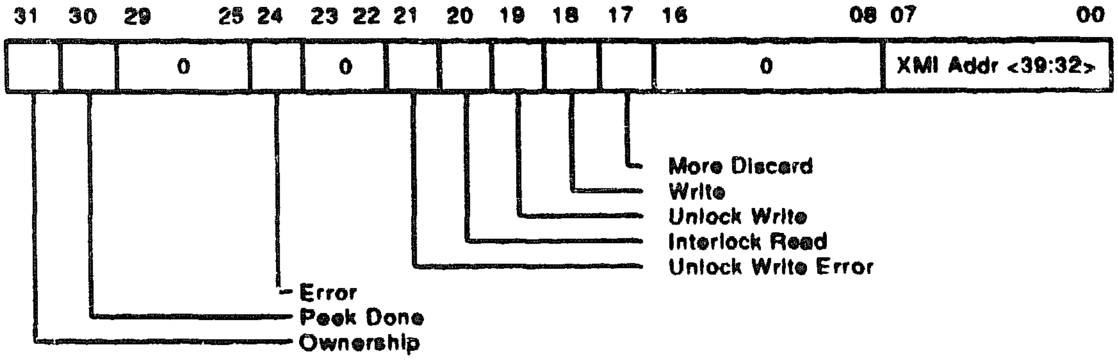
GSF-RC1000-XNA16-PSA

6.6.9 Peek XMI Low Address Register (PKXMILn)



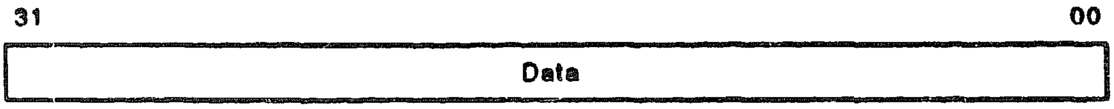
GSF-RC1000-XNA17-PSA

6.6.10 Peek XMI High Address Register (PKXMIHn)



GSF-RC1000-XNA18-PSA

6.6.11 Peek Data A and Peek Data B Registers (PKDATAn, PKDATBn)



GSF-RC1000-XNA19-PSA

CHAPTER 7

7 DEMNA Sequencing Flows

7.1 POWER-UP/RESET SEQUENCE

The DEMNA executes its power-up/reset sequence in response to any of three events:

- System power-up
- System reset
- Node reset

The system wide events are signified by the transitioning of the XMI DC LO and XMI AC LO signals. Node reset is initiated when the host sets the NRST bit in the XBE register. Port state information is not saved across a power-up/reset.

Figure 7–1 shows the power-up/reset sequence.

7.2 NODE HALT/RESTART SEQUENCE

The port driver initiates a node halt/restart by setting then clearing the NHALT bit in the XBE register. Setting the bit halts the DEMNA; clearing the bit restarts the port.

The following port state information is saved across a node halt/restart:

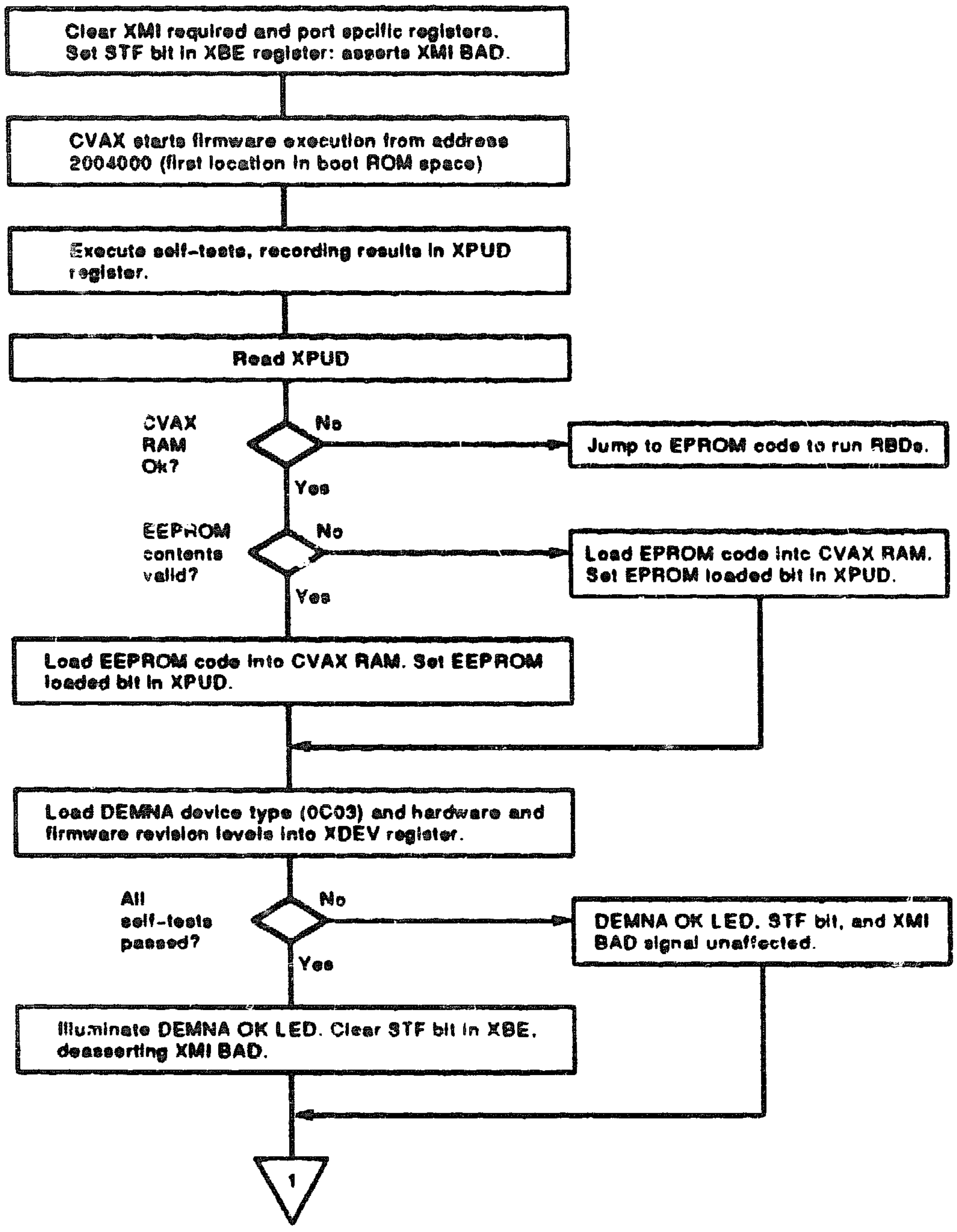
- Node halt fatal error block. Written to the PDB if the port was in the initialized state when NHALT was set.
- Port-internal error data. Visible by the console monitor program or with the READ$ERROR port driver command.
- Port-internal counters, including the data link counters. Visible by the console monitor program or with the port driver commands: RCCNTR/RDCNTR, READ$ERROR, READ$SNAPSHOT, and READ$STATUS.

Figure 7–2 shows the node halt/restart sequence. Note that the DEMNA self-tests are not executed as in the power-up/reset sequence.

7.3 PORT SHUTDOWN

The port initiates a shutdown if it is in the initialized state and either a fatal error occurs or the port driver issues a shutdown command by writing the XPCS register. The shutdown sequence is also invoked in response to a power fail, indicated by the assertion of the XMI AC LO signal. When this signal is asserted, a power-fail trap occurs in the CVAX, causing the port to execute the shutdown sequence.

Figure 7–3 shows the shutdown sequence.


Clear XMI required and port spcific registers. Set STF bit in XBE register: asserts XMI BAD.
CVAX starts firmware execution from address 2004000 (first location in boot ROM space)
Execute self-tests, recording results in XPUD register.
Read XPUD
CVAX RAM Ok?
No
Jump to EPROM code to run RBDs.
Yes
EEPROM contents valid?
No
Load EPROM code into CVAX RAM. Set EPROM loaded bit in XPUD.
Yes
Load EEPROM code into CVAX RAM. Set EEPROM loaded bit in XPUD.
Load DEMNA device type (0C03) and hardware and firmware revision levels into XDEV register.
All self-tests passed?
No
DEMNA OK LED, STF bit, and XMI BAD signal unaffected.
Yes
Illuminate DEMNA OK LED. Clear STF bit in XBE, deasserting XMI BAD.
1

MSF-RC1090-XMA30-PSA

Figure 7–1 (Continued, next page) DEMNA Power-Up and Node Reset

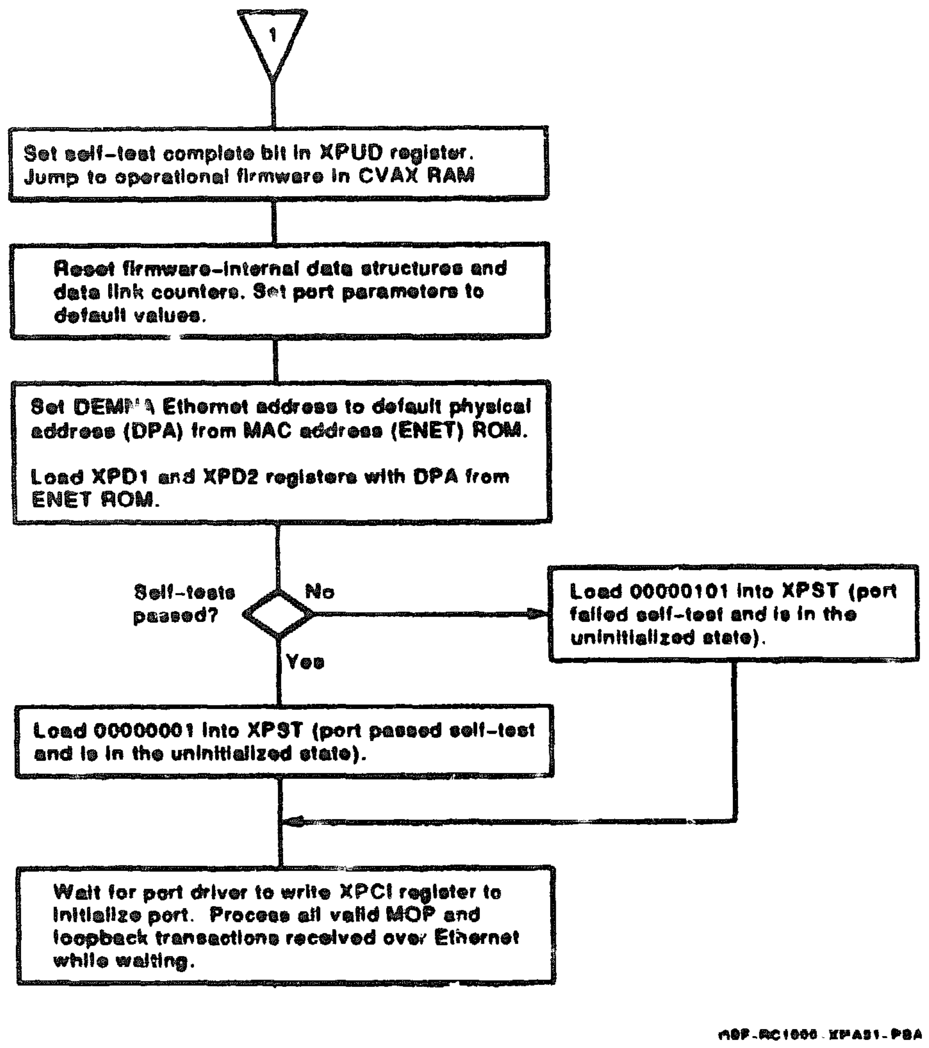

1
Set self-test complete bit in XPUD register. Jump to operational firmware in CVAX RAM
Reset firmware-internal data structures and data link counters. Set port parameters to default values.
Set DEMNA Ethernet address to default physical address (DPA) from MAC address (ENET) ROM.
Load XPD1 and XPD2 registers with DPA from ENET ROM.
Self-tests passed?
No
Yes
Load 00000101 into XPST (port failed self-test and is in the uninitialized state).
Load 00000001 into XPST (port passed self-test and is in the uninitialized state).
Wait for port driver to write XPCI register to initialize port. Process all valid MOP and loopback transactions received over Ethernet while waiting.
msf-RC1000-XPA51-PBA

Figure 7–1 DEMNA Power-Up and Node Reset

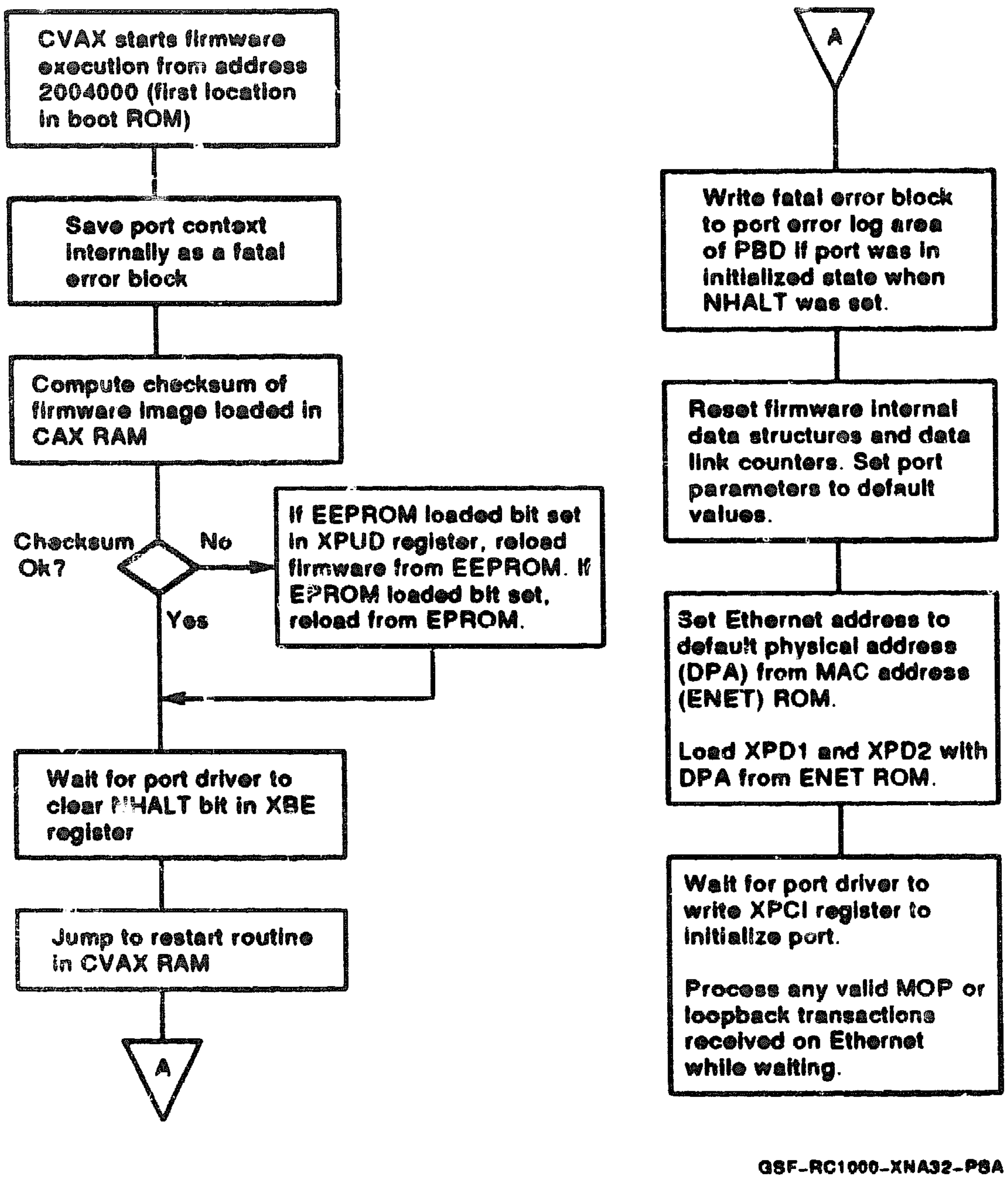
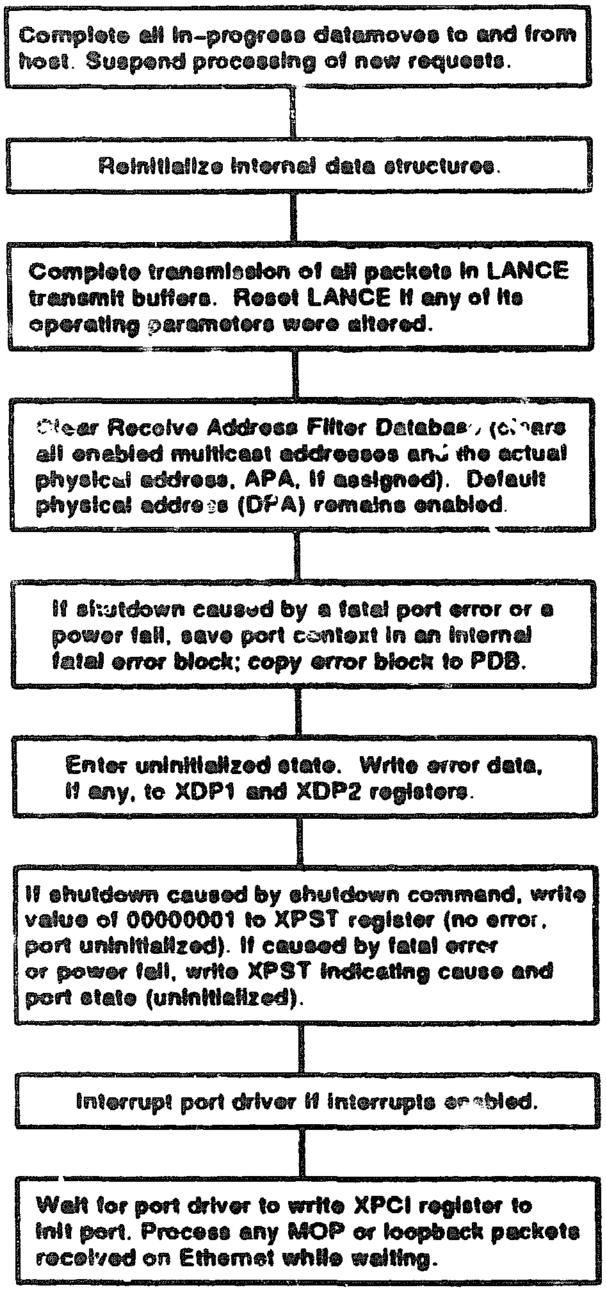

CVAX starts firmware execution from address 2004000 (first location in boot ROM)
Save port context internally as a fatal error block
Compute checksum of firmware image loaded in CAX RAM
Checksum Ok?
No
Yes
If EEPROM loaded bit set in XPUD register, reload firmware from EEPROM. If EPROM loaded bit set, reload from EPROM.
Wait for port driver to clear NHALT bit in XBE register
Jump to restart routine in CVAX RAM
A
A
Write fatal error block to port error log area of PBD if port was in initialized state when NHALT was set.
Reset firmware internal data structures and data link counters. Set port parameters to default values.
Set Ethernet address to default physical address (DPA) from MAC address (ENET) ROM.
Load XPD1 and XPD2 with DPA from ENET ROM.
Wait for port driver to write XPCI register to initialize port.
Process any valid MOP or loopback transactions received on Ethernet while waiting.
GSF-RC1000-XNA32-PBA

Figure 7–2 DEMNA Node Halt/Restart


Complete all in-progress datamoves to and from host. Suspend processing of new requests.

Reinitialize internal data structures.

Complete transmission of all packets in LANCE transmit buffers. Reset LANCE if any of its operating parameters were altered.

Clear Receive Address Filter Database (clears all enabled multicast addresses and the actual physical address, APA, if assigned). Default physical address (DPA) remains enabled.

If shutdown caused by a fatal port error or a power fail, save port context in an internal fatal error block; copy error block to PDB.

Enter uninitialized state. Write error data, if any, to XDP1 and XDP2 registers.

If shutdown caused by shutdown command, write value of 00000001 to XPST register (no error, port uninitialized). If caused by fatal error or power fail, write XPST indicating cause and port state (uninitialized).

Interrupt port driver if interrupts enabled.

Wait for port driver to write XPCI register to init port. Process any MOP or loopback packets received on Ethernet while waiting.

Figure 7-3 DEMNA Shutdown

CHAPTER 8

8 DEMNA Error Handling

8.1 Introduction

This chapter overviews the error handling features of the DEMNA. It describes the types of errors reported by the port, the error blocks maintained by the port, error logging, and the port's response to errors.

8.2 Error Types

Fatal Errors

Fatal errors result in a port shutdown. For several fatal error types, the port writes a fatal error block to the port error log area of the port data block (PDB). Fatal errors include the following:

- DEMNA CVAX machine check and exceptions
- Node halt/restart while port is in the initialized state
- Port initialization failures
- Port driver protocol errors and port command failures
- Specifying more than one buffer for a port command
- Specifying an invalid number of transmit buffers
- Other firmware-related errors (such as keep-alive timeouts)
- Unrecoverable failed access to the command ring or receive ring
- Firmware updates

Nonfatal Errors

These errors do not result in a port shutdown. The error is retried once. If the retry fails, a fatal error occurs. On a nonfatal error the port writes a nonfatal error block in the PDB. Nonfatal errors include the following:

- Datamove and peek errors which are recovered on the first retry or that did not directly access the command ring or receive ring
- Buffer transfer failures
- Address translation errors
- Port command errors (for example, a command length error)

Ethernet Errors

These errors are nonfatal errors which occur in the normal course of Ethernet activity. Ethernet errors include the following:

- Transmit and receive errors caused by activity on the Ethernet wire

  These errors are recorded in the port's internal data link counters. A user can read the counters by issuing the Network Control Program (NCP) command SHOW KNOWN LINE COUNTERS. Software can read the counters by issuing a RCCNTR/RDCNTR or READ$STATUS command. The counters are displayed in the Status and Status/Error screens of the console monitor program.

- Receive errors caused by insufficient allocation of system buffers by the host

  These errors are recorded in the port's SBUA counter which is displayed in the Status/Error screen of the console monitor program. Software can read the counter by requesting that the port driver issue a RCCNTR/RDCNTR, READ$STATUS, or READ$SNAPSHOT command. A copy of the SBUA is located in the PDB. The PDB copy is updated as often as once per second.

- Receive errors caused by insufficient allocation of user buffers by the host

  These errors are recorded in the port's UBUA counter which is located in host memory and is displayed in the Status/Error screen of the console monitor program. Software can read the counter by issuing a RCCNTR/RDCNTR, READ$STATUS, or READ$SNAPSHOT command.

8.3 Error Blocks

The DEMNA maintains two types of internal error blocks: one for fatal errors and one for nonfatal errors. A user can examine the error blocks by issuing the console monitor commands SHOW ERROR Hn and SHOW ERROR Sn. Software can read the error blocks by issuing the READ$ERROR command. The port writes the fatal error block to the port error log area of the port data block (PDB) when a fatal error occurs or when a node halt/restart is issued while the port is in the initialized state.

Figures 8–1 and 8–2 show the error block formats.

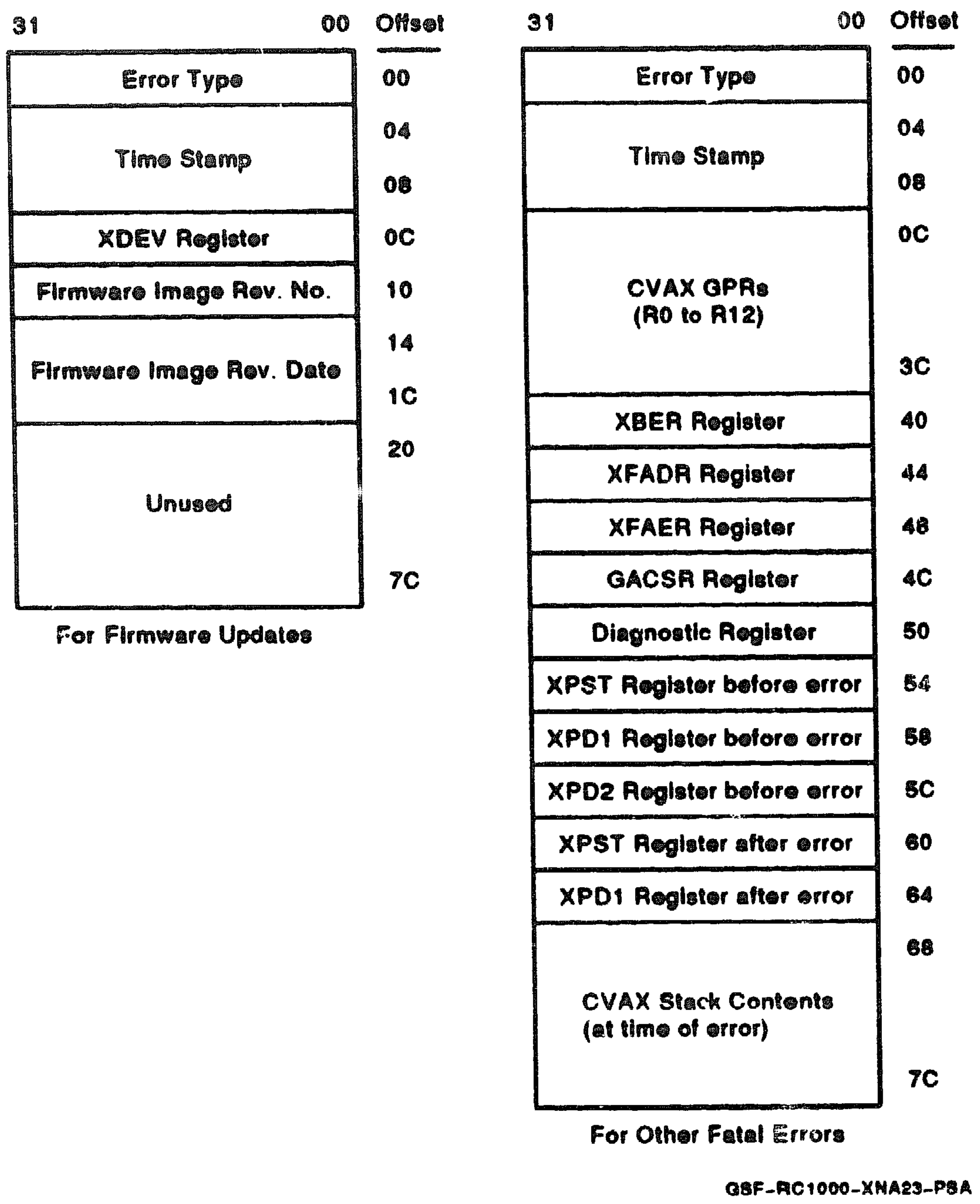


Figure 8–1 Fatal Error Block Formats

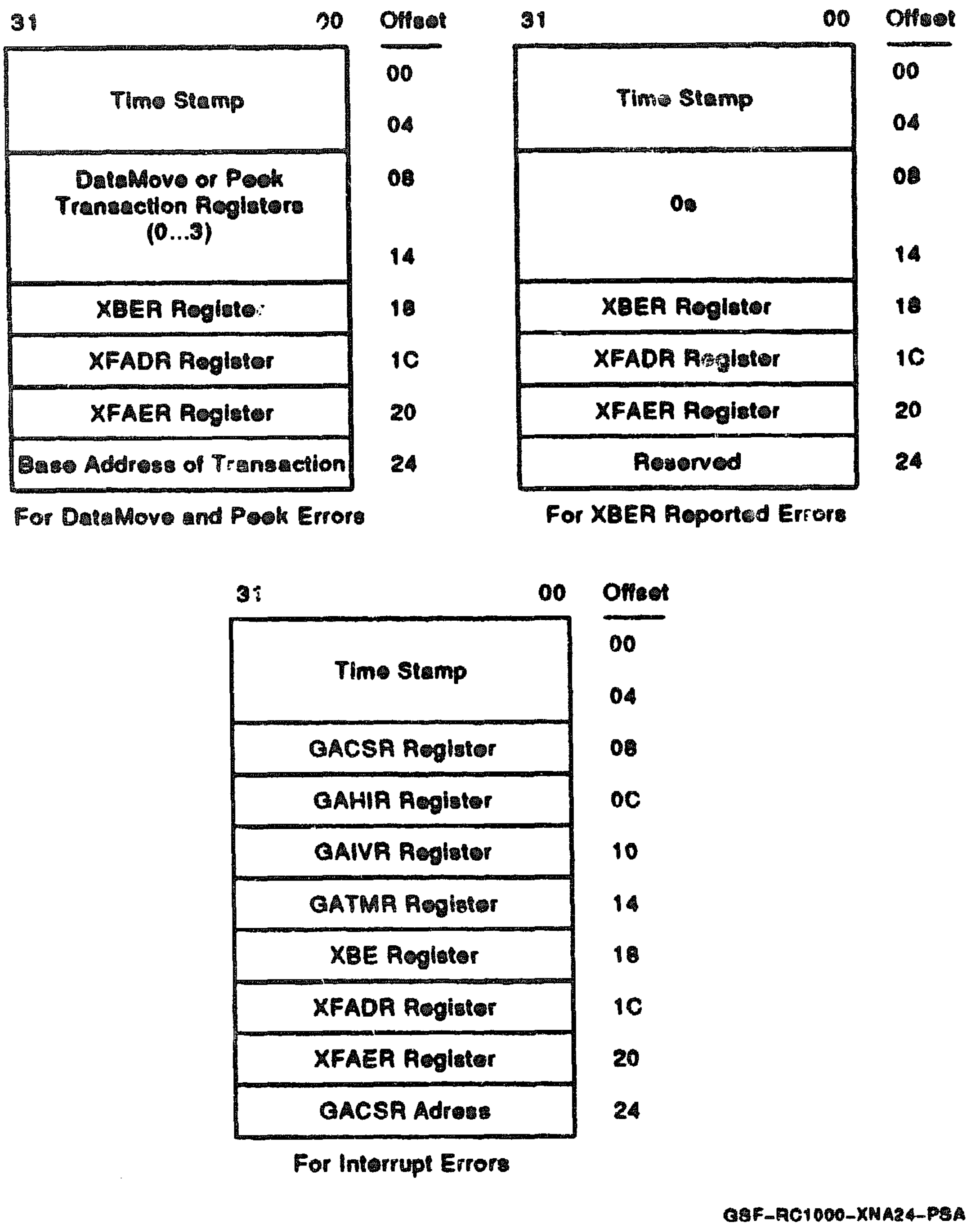


Figure 8–2 Nonfatal Error Block Formats

Notes for Figure 8–1

- Error type:

0	No error
1	DEMNA machine check or exception
2	Node halt/restart issued while the port was in the initialized state. (When the port receives a node halt/restart in the initialized state, it assumes that it is being restarted from an error.)
3	Fatal error other than machine check or exception (for example, port initialization failure or a keep-alive timeout)
4	Firmware updates
6	Driver error. The port driver attempted to initialize the port but the port was already in the initialized state.

- Time Stamp:

  Date and time of error expressed in binary absolute format. A value of zero indicates that no error occured. The entry is the sum of the base time specified by the host in the PARAM command, and the DEMNA uptime until the error occured. If the system base time was not specified in the PARAM command, the base time defaults to 01-Jan-88.

- Firmware image revision number and date refer to the EEPROM firmware image. The fields are in ASCII. The revision number is the ASCII code of the low-order byte of the XDEV register. The revision date is of the form *dd-mmm-yyyy*. For example, 12-JUL-1989.

Notes for Figure 8–2

- Datamove/Peek transaction registers:

  For datamove transaction errors, these are the four datamove registers: DMPOR*n*, DMCSR*n*, DMXMI*n*, and DMNPA*n*.

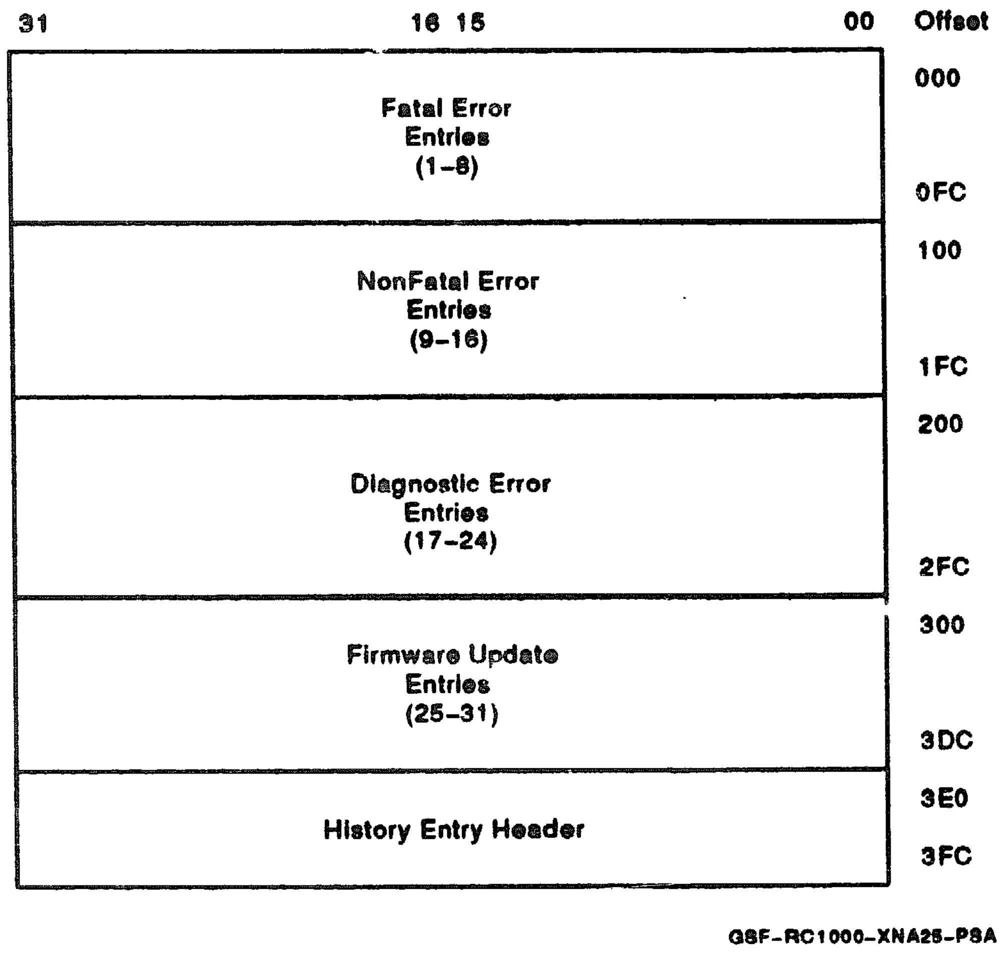
  For peek transaction errors, these are the four peek registers: PKXMIL*n*, PCKMIH*n*, PKDATA*n*, and PKDATB*n*.

- Base address of transaction is the starting address of the failed datamove or peek operation.

8.4 ERROR LOGGING

A 1-KByte area in the EEPROM is reserved for the logging of error history data. This area contains 31 error history entries of 32 bytes each and a history entry header which is also 32 bytes long. The user can read the history data with the console monitor command SHOW HISTORY or by depositing and examining the XCOMM register. Software can read the history entries by issuing a READ$HISTORY command.

The history data area is partitioned into five segments: four for the logging of error history events and one for the header.



31
16 15
00
Offset
Fatal Error Entries (1–8)
000
0FC
NonFatal Error Entries (9–16)
100
1FC
Diagnostic Error Entries (17–24)
200
2FC
Firmware Update Entries (25–31)
300
3DC
History Entry Header
3E0
3FC
GSF-RC1000-XNA25-PSA

Figure 8–3 EEPROM History Data Area

Table 8–1 EEPROM History Data Entries

Entries	Error Type	Logging Information
1 to 8	Fatal	Logged immediately after error occurs. Entries overwritten if more errors occur than can be recorded. Logging stops after 32 fatal errors have been recorded.
9 to 16	Nonfatal	For datamove, peek, or host interrupt errors: logged immediately after error occurs. For XBER-reported errors: logged after firmware polls XBER register and discovers error. Logging stops after 16 nonfatal errors have been recorded.
17 to 24	Diagnostic	Logged immediately after each self-test or RBD error provided that error logging is enabled by the corresponding EEPROM flag (Figure 8–5). Entries not overwritten if more errors occur than can be recorded. Logging stops after eight diagnostic errors have been recorded.
25 to 31	Firmware updates	Logged after each firmware update. Entries overwritten if more firmware updates occur than can be recorded. No limit on number of firmware updates logged.
Header	n/a	Provides module-specific information, including: number of history entries written to EEPROM since the last DEMNA power-up or reset, the console password, the module serial number, and the module runtime.

8.4.1 History Entry Header

The history entry header contains the DEMNA's current operational parameters which can be modified with the EEPROM Update Program (EVGDB), firmware update data, and history data pointers. Figure 8–4 shows the header format. Table 8–2 describes the header fields. Figure 8–5 shows the EEPROM flags which can be modified by running EVGBD. Table 8–3 describes the EEPROM flags.

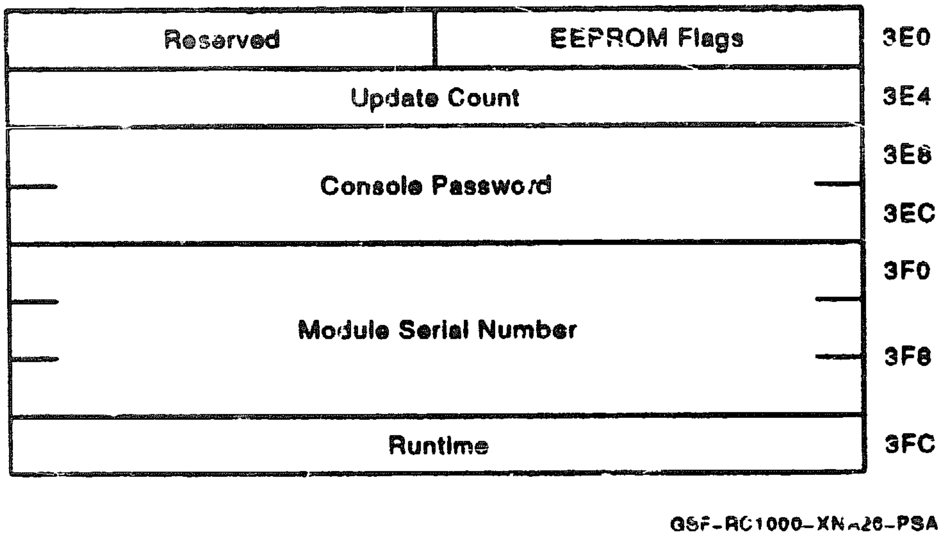


Figure 8–4 History Entry Header

Table 8–2 History Entry Header Fields

Field	Description
EEPROM Flags	Flags that control various aspects of DEMNA operation See Figure 8–5 and Table 8–3.
Update Count	Indicates the number of times history entries have been written to the EEPROM since the last DEMNA power-up or reset. If no unused history entries are available, used history entries may be overwritten (the update count can be greater than the total number of history entries in the EEPROM.
Console Password	Eight ASCII characters that indicate the password required to connect to the DEMNA console monitor program.
Module Serial Number	Twelve ASCII characters that identify the module.
Runtime	Total time, in 524,288 second (6.068 day) increments since the DEMNA EEPROM was initialized.

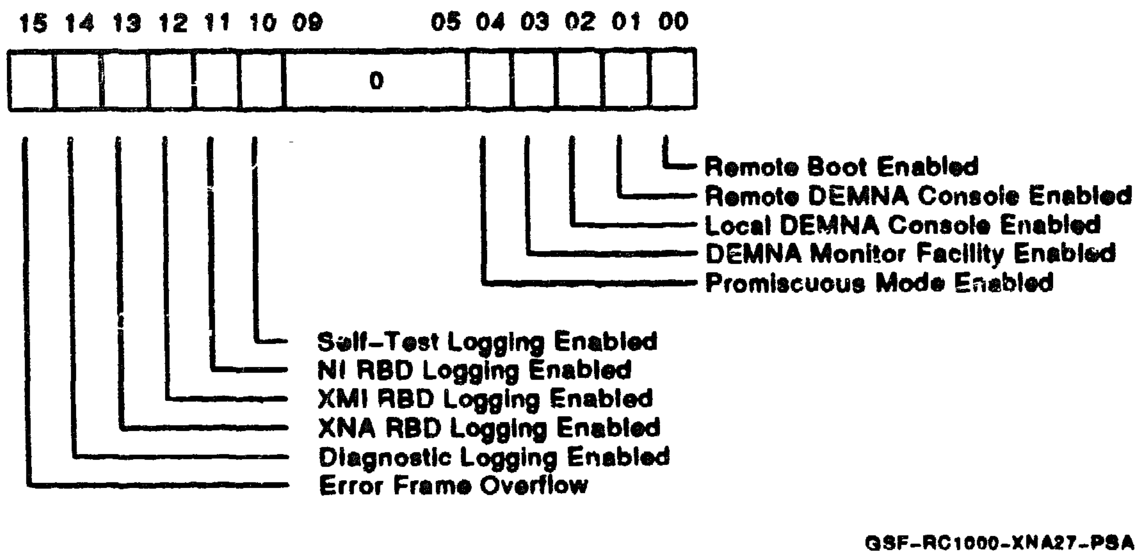


Figure 8–5 EEPROM Flags

Table 8–3 EEPROM Flags

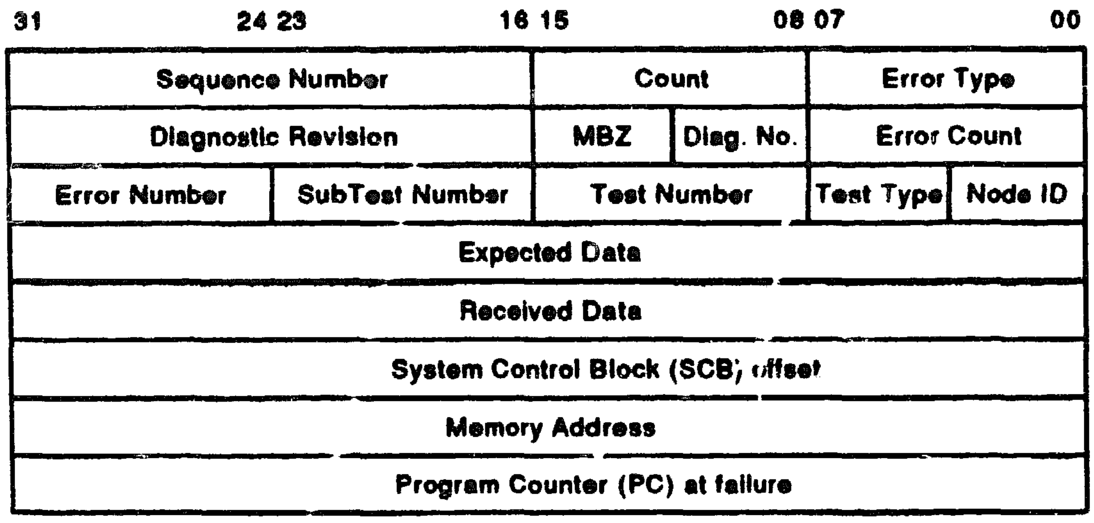
Bit	Name/Function When Set
15	Error Frame Overflow Indicates that no more diagnostic errors could be written to the EEPROM because all history entries allocated to diagnostic errors have been written.
14	Diagnostic Logging Enabled Enables the logging of self-test and RBD errors to the EEPROM.
13	XNA RBD Logging Enabled Enables logging XNA RBD errors to the EEPROM if bit <14> is also set.
12	XMI RBD Logging Enabled Enables logging XMI RBD errors to the EEPROM if bit <14> is also set.
11	NI RBD Logging Enabled Enables logging NI RBD errors to the EEPROM if bit <14> is also set.

Table 8–3 (Cont.) EEPROM Flags

Bit	Name/Function When Set
10	**Self-Test Logging Enabled** Enables logging self-test errors to the EEPROM if bit <14> is also set.
9:5	Reserved, must be zeros.
4	**Promiscuous Mode Enabled** Allows the DEMNA to operate in promiscuous mode by default. If the flag is cleared, an application can override the flag setting by starting up a promiscuous user. In promiscuous mode, the DEMNA receives all packets on the network, regardless of a packet's destination. The DEMNA console monitor program uses this information to determine characteristics of the network traffic. If no users defined to the DEMNA are enabled for promiscuous mode, the DEMNA discards the packets not addressed to a DEMNA user. Otherwise, the DEMNA delivers all received packets to each DEMNA user for whom promiscuous mode is enabled.
3	**DEMNA Monitor Facility Enabled** Enables operation of the DEMNA monitor facility, which monitors network operation.
2	**Local DEMNA Console Enabled** Enables DEMNA console monitor program to be accessed from the local network node and from the DEMNA physical console.
1	**Remote DEMNA Console Enabled** Enables DEMNA console monitor program to be accessed from a remote network node.
0	**Remote Boot Enabled** Enables DEMNA to participate in remote booting over the network.

8.4.2 History Data Entries

There are two basic formats for history entries: one for diagnostic errors, and one for all other errors. Each history entry is 32 bytes (8 longwords) in length. Figures 8–6 and 8–7 show the entry formats.



QSF–RC1000–XNA28–PSA

Figure 8–6 History Entry Format For Diagnostic Errors

Table 8–4 History Entry Fields for Diagnostic Errors

Field	Description
Error type	Error type for diagnostic errors is 5.
Count	Number of times this type of diagnostic error has been recorded.
Sequence Number	Integer from one through 255 which indicates the order in which the entry was logged with respect to other history entries. Lower-numbered entries were logged before higher-numbered entries.
Diagnostic Revision	Two ASCII characters that indicate the revision number of the diagnostic. For example, 39 33 (ASCII) = revision 3.9.
Diagnostic Number	Binary code for the test which reported the error: 0 = Self-test 1 = NI RBD 2 = XMI RBD 3 = XNA RBD
Error Count	Binary count of the number of times this type of diagnostic error has occurred.

Table 8–4 (Cont.) History Entry Fields for Diagnostic Errors

Field	Description
Error Number	Binary code for the specific error reported by the diagnostic test. Refer to the diagnostic listings for error numbers.
Subtest Number	Number of the failing subtest.
Test Number	Number of the failing test.
Test Type	1 = Power-up mode 2 = RBD mode
Node ID	DEMNA's XMI node ID

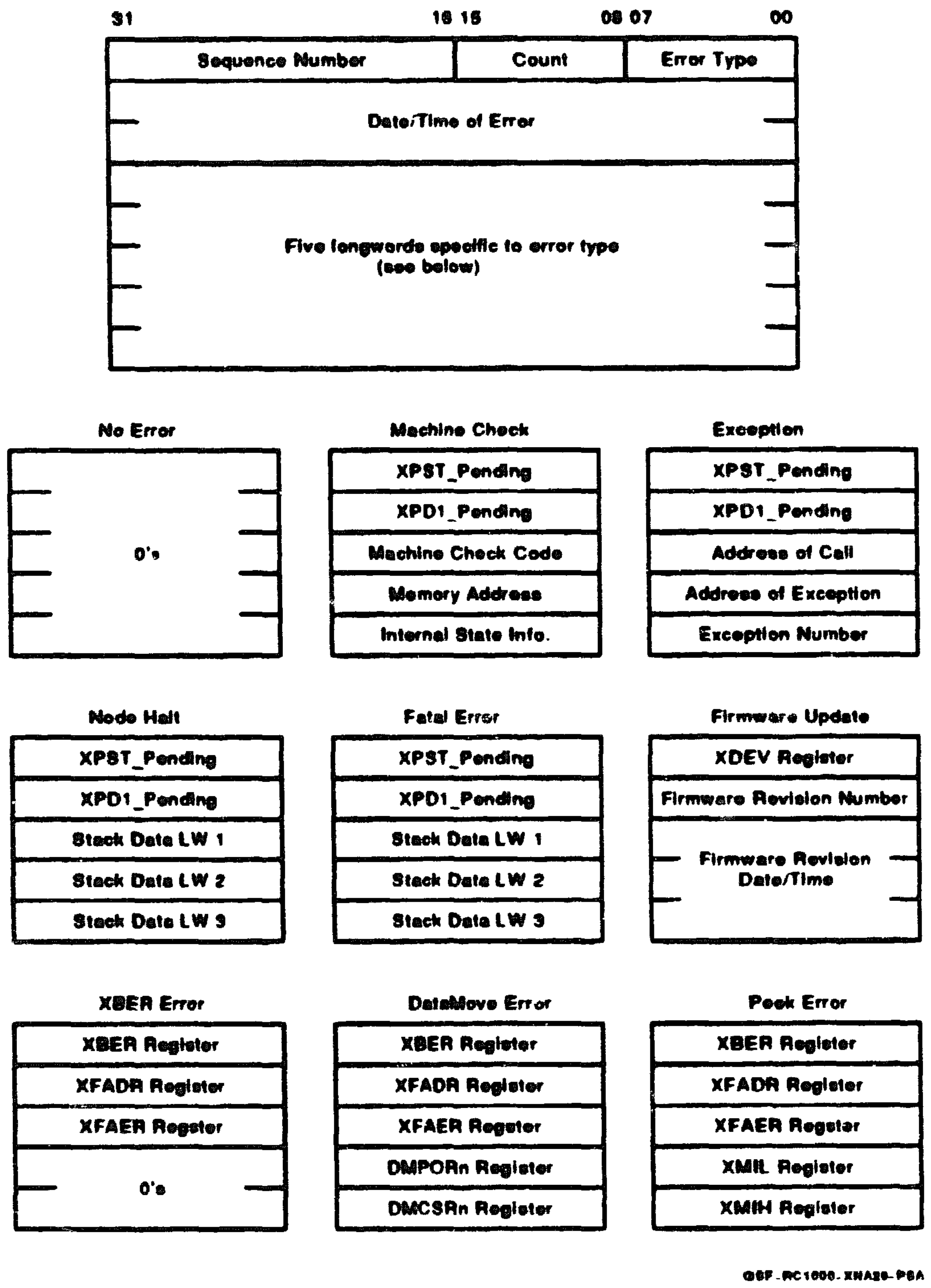


Figure 8–7 History Entry Format for All Other Errors

Table 8–5 History Entry Fields for All Other Errors

Fields Common to All Error Types	
Error type	0 - No error 1 - Machine check 2 - Node halt 3 - Fatal error 4 - Firmware update 6 - XBER-detected error 7 - Peek, datamove, or interrupt failed
Count	Number of times this specific error has been recorded.
Sequence Number	Integer from 1 to 31 which indicates the order in which the entry was logged with respect to other history entries. Lower-numbered entries were logged before higher-numbered entries.
Date/Time of Error	Date and time of error expressed in binary absolute format. The entry is the sum of the base time specified by the host in the PARAM command, and the DEMNA uptime until the error occured. If the system base time was not specified in the PARAM command, the base time defaults to 01-Jan-88.
Machine Checks	
XPST_Pending	Value that will be loaded into the XPST after the next state change (after error handling is completed).
XPD1_Pending	Value that will be loaded into the XPD1 after the next state change (after error handling is completed).
Machine Check Code	Value (usually 80 to 83 indicating an invalid address) which indicates the reason for a DEMNA CVAX machine check.
Memory Address	Most recent memory address.
Internal State Information	

Table 8-5 (Cont.) History Entry Fields for All Other Errors

Exceptions	
XPST_Pending XPD1_Pending	Same as for machine checks
Address of Call	Address of call to shutdown request.
Address of exception	
Exception number	Offset into system control block (SCB).
Node Halt and Fatal Errors	
XPST_Pending XPD1_Pending	Same as for machine checks
Stack Data LW 1-LW 3	Longwords 1 to 3 of the stack when the error occurred.
Firmware Updates	
Firmware Revision Number	Four ASCII characters that indicate the DEMNA firmware revision. For example, 30313233 (ASCII) = revision 01.23.
Firmware Revision Date/TIme	EEPROM firmware revision date and time expressed in binary absolute format.

8.5 ERROR RESPONSE

The port driver is notified of port-detected errors by one of four means:

- State qualifier field in the XPST register
- Error codes written by the port to the command and receive rings
- Port-generated interrupt when any hard error bit in the XBER register is set (and error interrupts are enabled)
- Fatal error block and counters in the PDB

When an error is detected, the port driver can issue a RCCNTR/RDCNTR command to read the port's data link counters, a READ$STATUS command to read status information, or a READ$ERROR command to read the error blocks.

Tables 8–7 to 8–10 describe the hardware and firmware response to errors. Table 8–6 describes the keys which indicate the port driver's response referenced in the tables.

Table 8–6 Error Response Keys

Key[1]	Description
Status	Driver returns transmit error status to user application and continues as normal. User application can view the error status in the I/O Status Block pointed to by the failing QIO operation (if applicable) or in the device counters which can be examined with the NCP command SHOW LINE COUNTERS or with the MOP command REQUEST COUNTERS.
Counters	Driver does not recognize error. User application can view error by issuing the NCP command SHOW LINE COUNTERS or the MOP command REQUEST COUNTERS.
Shutdown	Driver records a port shutdown error, returns outstanding transmits to users with a transmit error, and shuts down all users. User application can view error as a Circuit Down error in the device counters with the NCP command SHOW CIRCUIT COUNTERS. Subsequent and outstanding QIO requests are returned with transmit failure status.
Crash	Driver machine checks if it had initiated the transaction that experienced the error. This causes a system crash. If the driver did not initiate the transaction that experienced the error, the result cannot be characterized here. The system may crash.

[1]The keys defined in this table describe the port driver's response to errors and the indication the user's application receives following an error. The keys apply to Tables 8–7 to 8–10.

Table 8–7 Response To Ethernet Errors

Error	Response	Key[1]
Loss of Carrier	LANCE completes transmission of packet and continues to transmit and receive packets. Firmware returns Transmit Failed—Loss of Carrier error to host command ring entry, increments Send Failures—Loss of Carrier counter, and continues as normal.	Status
Late Collision	LANCE does not retry error; continues to transmit and receive packets. Firmware returns Transmit Failed—Late Collision error to host command ring entry, increments Send Failures—Late Collision counter, and continues as normal.	Status
Retry Error (excessive collisions)	LANCE aborts transmission of packet; continues to transmit and receive subsequent packets. Firmware Returns Transmit Failed—Retries Exhausted error to host command ring, increments Send Failures—Retries Exhausted counter, and continues as normal.	Status
Framing Error	LANCE continues to transmit and receive packets. Firmware returns Receive Failed—CRC Error to host receive ring, increments Receive Failures—Framing Error counter, discards packet[2], and continues as normal.	Counters
CRC Error	LANCE continues to transmit and receive packets. Firmware returns Receive Failed—CRC Error to host receive ring, increments Receive Failures—CRC Error counter, discards packet[2], and continues as normal.	Counters
Collision Error (heartbeat)	LANCE continues to transmit and receive packets. Firmware increments Send Failures—Collision Check Failure counter and continues as normal.	Counters

[1]See Table 8–6 for descriptions of the keys.

[2]If a user requested receipt of bad packets, the firmware delivers a packet with a framing or CRC error and writes appropriate error status to the host receive ring. The driver passes the received packet to the user with appropriate error status.

Table 8–8 Response To Internal Errors that Affect the LANCE

Error	Response	Key[1]
Miss Error	LANCE continues to transmit and receive packets. Firmware increments Receive Failures—Data Overrun counter and continues as normal.	Counters
Overflow (exceeded 21-microsecond bus latency on receive DMA)	Packet not received completely. LANCE continues to transmit and receive packets. Firmware increments Receive Failures—Data Overrun counter and Overflow/Underflow counter and continues as normal.	Counters
Underflow (exceeded 9 microseconds on transmit DMA)	Packet incompletely transmitted, resulting in transmission of a runt packet or packet with CRC error. LANCE transmitter shut off, but LANCE receiver continues to function. Firmware increments Overflow/Underflow counter, restarts LANCE, increments LANCE Restarts counter, and continues as normal.	Status
Memory Error	LANCE transmitter and receiver shut off. Firmware saves internal status in a fatal error block, increments memory error counter, stops command and receive ring processing, and shuts down the port.	Shutdown
Parity Error	Parity logic on DEMNA detects a parity error and causes LANCE to experience a memory error. (Parity error is detected by external logic while LANCE is bus master.) Firmware saves internal status in a fatal error block, increments Memory Error counter, stops command and receive ring processing, and shuts down the port.	Shutdown
LANCE/Gate Array Grant Timeout	DEMNA logic detects a timeout and sets the Grant Timeout bit in the DEMNA Diagnostic register. ERR is asserted to the CVAX, which causes the CVAX to machine check. Firmware resets LANCE or gate array, saves status in a fatal error block, stops command and receive ring processing, and shuts down the port.	Shutdown

[1]See Table 8–6 for descriptions of the keys.

Table 8–8 (Cont.) Response To Internal Errors that Affect the LANCE

Error	Response	Key[1]
Powerfail	Nonmaskable powerfail interrupt sent to CVAX. Firmware saves internal status in a fatal error block, stops command and receive ring processing, and shuts down the port.	Shutdown

[1]See Table 8–6 for descriptions of the keys.

Table 8–9 Response to Hardware Errors that Affect the CVAX

Error	Response	Key[1]
Nonexistent Memory Error	CVAX machine checks. Firmware saves internal status in a fatal error block, stops command and receive ring processing, and shuts down port.	Shutdown
Internal CVAX exception (unexpected exception)	Unexpected interrupt handler activated. Firmware saves internal status in a fatal error block, stops command and receive ring processing, and shuts down port.	Shutdown
XMI Node Halt (bit 29 set in XBE)	Halt asserted to the CVAX. CVAX restarts with restart code of 02 and starts execution at 20040000 (Boot ROM). Firmware executes from EPROM (Boot ROM). Firmware determines that a node halt occurred and initiates node halt processing.	Shutdown
Fatal Parity Error Detected by CVAX	CVAX machine checks. Firmware saves current status in a fatal error block, stops command and receive ring processing, and shuts down port.	Shutdown
Unexpected Interrupt from LANCE or Gate Array	Either the LANCE or the gate array initiates an unexpected interrupt to the CVAX. The reason is unknown. Firmware ignores interrupt.	None

[1]See Table 8–6 for descriptions of the keys.

Table 8–10 Response to XMI Related Errors

Error	Response[1]	Key[2]
Parity Error	Sets XBE <23>, NoAcks transaction, and continues.	Possible Crash
Write Sequence Error	Sets XBE <22>, NoAcks transaction, and continues.	Possible Crash
Read/IDENT Data NO ACK	Sets XBE <21> and continues.	Possible Crash
Write Data NO ACK	Sets XBE <20> and continues.	Shutdown[3]
No Read Response	Sets XBE <18> and writes error status to the appropriate datamove or peek register. This error is always set in conjunction with another error.	Shutdown
Read Sequence Error	Sets XBE <17> and continues.	Shutdown[3]
Read Error Response	Sets XBE <16> and writes error status to the appropriate datamove or peek register.	Shutdown[3]
Command NO ACK	Sets XBE <15> and writes error status to the appropriate datamove or peek register.	Shutdown[3]
Transaction Timeout	Sets XBE <13> and writes error status to the appropriate datamove or peek register.	Shutdown

[1]Only the hardware's response is listed. The firmware's response is the same for all XBE reported errors: Firmware periodically polls the XBE register to see if bit <31> (error summary) is set. If the bit is set, firmware saves the current status in a nonfatal error block, sends an error interrupt to the port driver (if error interrupts are enabled), and proceeds normally.

[2]See Table 8–6 for descriptions of the keys.

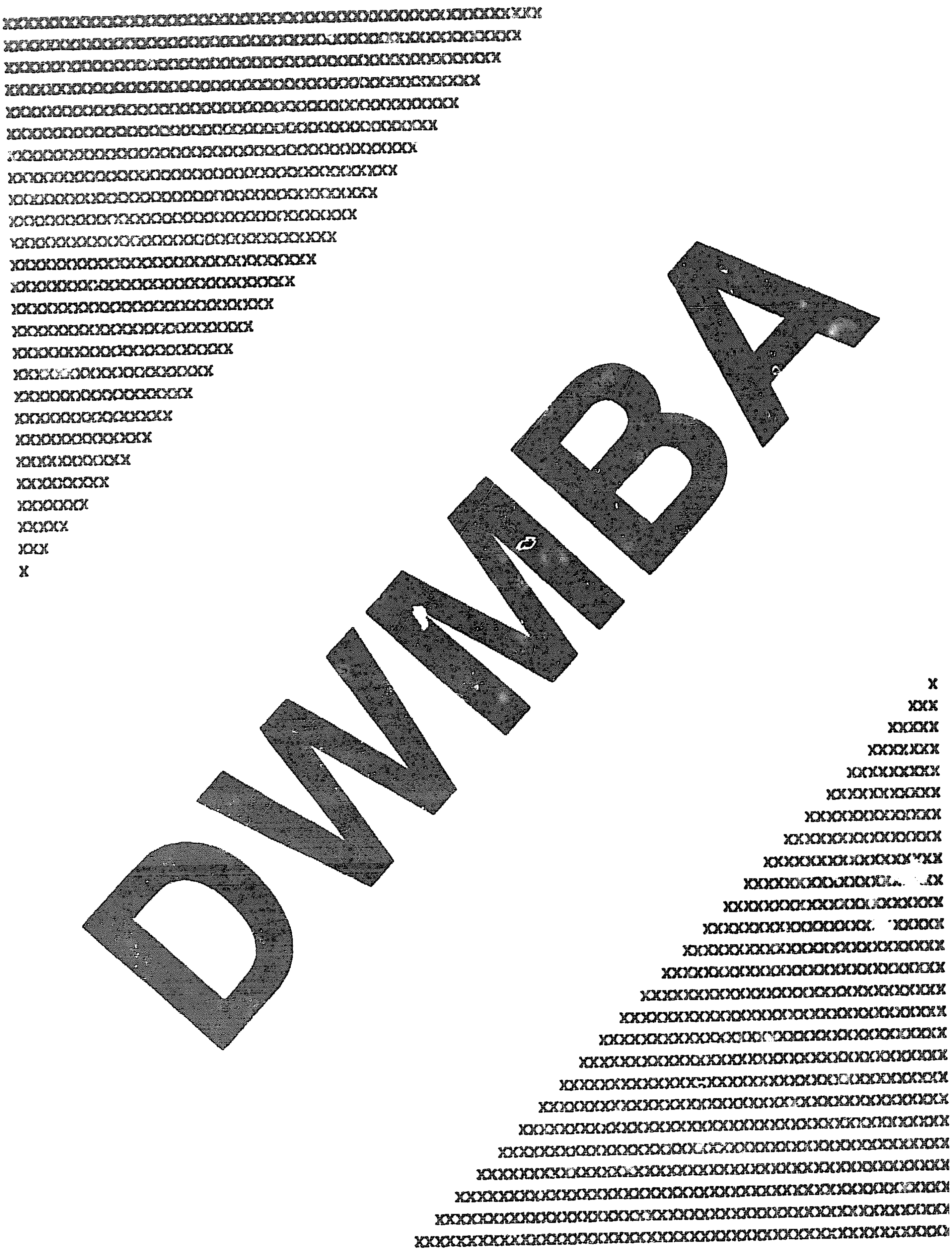
[3]Shutdown, if retry failed. None, if retry succeeded.

8.6 Restarting the Port from a Fatal Error

There are three ways to attempt to restart the port after a fatal port error (listed by order of increasing overhead and impact):

- Initialize the port by writing the XPCI register (after setting up the PDB and the XPD1 and XPD2 registers)
- Initiate a node halt/restart by setting and clearing the node halt bit in the XBE register and then initialize the port
- Initiate a node reset by setting the node reset bit in the XBE register (causing the DEMNA to execute its power-up/reset sequence) and then initialize the port

Initializing the port involves the least overhead and is the least forceful method. A node reset has the greatest impact, but involves the most overhead. If initializing the port does not restart the DEMNA, try a node halt/restart, and if that fails, a node reset.


DWMBA

Document Title: DWMBA HANDBOOK

Order Number: EK-DWMBA-HB-001

This handbook is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The handbook can be ordered separately or as part of the set.

The *XMI Adapters Handbook Documentation Set* is a dynamic document which will be periodically updated as new XMI adapters are announced. The first release of the set includes the following handbooks:

Order Number	Title
EK-XMIOV-HB	XMI Bus Overview Handbook
EK-CIXCD-HB	CIXCD Handbook
EK-DEMNA-HB	DEC LANcontroller 400 (DEMNA) Handbook
EK-DWMBA-HB	DWMBA Handbook

This handbook and the document set are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

DWMBA Handbook

Order Number EK-DWMBA-HB-001

This document is part of the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). The document can be ordered separately or as part of the set. The *DWMBA Handbook* and the *XMI Adapters Handbook Documentation Set* are for VAX system trained Digital customer service personnel who are familiar with the XMI bus architecture.

Revision/Update Information: Revision 1.0

Digital Equipment Corporation

January, 1991

This document was prepared and published by Educational Services Development and Publishing, Digital Equipment Corporation.

Contents

Examples

Figures

Tables

PAGE vi INTENTIONALLY LEFT BLANK

About This Manual

Intended Audience

This handbook is part of a series of handbooks which comprise the *XMI Adapters Handbook Documentation Set* (EK-XMIAD-HB). This handbook and the handbook set are for VAX system trained Digital customer service personnel who service XMI-based systems and subsystems. Users of the handbook set should be familiar with the XMI bus architecture (either through the *XMI Bus Concepts* course or through practical experience) and have a minimum of level 1 hardware maintenance training on one or more VAX systems (for example, VAX 6000 or VAX 9000 systems).

Document Scope and Structure

Several I/O adapters have been developed to interface the XMI bus to devices which employ different bus structures and protocols. These adapters are available as stand-alone options and may be installed on a variety of systems or subsystems.

The *XMI Adapters Handbook Documentation Set* provides a single, quick reference source to the type of information most frequently required to service XMI adapters. This handbook contains information specific to the DWMBA option.

This handbook is divided into four chapters:

Chapter 1 introduces the DWMBA and overviews its physical and functional characteristics.

Chapter 2 indicates the configuration requirements for installing the option.

Chapter 3 reviews the DWMBA's power-up self-tests and ROM-based diagnostics.

Chapter 4 describes the XMI required and DWMBA specific registers.

Conventions

addresses	All addresses are given in hexadecimal (hex).
bits	All bit numbers are given in decimal with the bit(s) enclosed in angle brackets; for example <31>.
	Multiple individual bits or bit fields are separated by commas with bit fields indicated by two numbers separated by a colon. For example <31:24,20,18,14:10> indicates bits 31 through 24 (inclusive), bit 20, bit 18, and bits 14 through 10 (inclusive).
$\boxed{\text{CTRL/x}}$	Specifies to press and hold the $\boxed{\text{Ctrl}}$ key while pressing the $\boxed{\text{x}}$ key; for example, $\boxed{\text{CTRL/C}}$.
[item] . . .	Indicates the item is optional. The horizontal ellipsis indicates that additional optional items can be entered.
. . .	Vertical ellipsis in examples, tables, or figures, indicate that not all information is shown.

CHAPTER 1

1 DWMBA ADAPTER OVERVIEW

1.1 INTRODUCTION

The DWMBA option interfaces the XMI bus to the VAXBI bus, providing a data path between host processors on the XMI and I/O devices on the VAXBI.

Figure 1–1 shows a typical system with DWMBA adapters.

1.2 PHYSICAL DESCRIPTION

The DWMBA consists of a DWMBA/A module (also called the XBIA), a DWMBA/B module (XBIB), and four, 30-conductor cables which connect the modules.

The XBIA module is a standard XMI module which plugs into the XMI backplane. This module contains the XMI corner, XMI required registers, XMI interface control sequencers, DWMBA register files, and DWMBA specific registers.

The XBIB module is a standard VAXBI module which plugs into the VAXBI backplane. This module contains the BIIC chip, interconnect drivers, data transfer control sequencers, BIIC and register file status bits, DWMBA specific registers, DMA decode logic, and the VAXBI clock generation circuitry.

The four, 30-conductor cables are collectively known as the IBus. These cables interconnect the XBIA and XBIB modules by way of the I/O connector pin segments of the XMI and the VAXBI backplanes.

Figures 1–2 and 1–3 show the DWMBA modules. Figure 1–4 shows the major functional logic elements of each module.

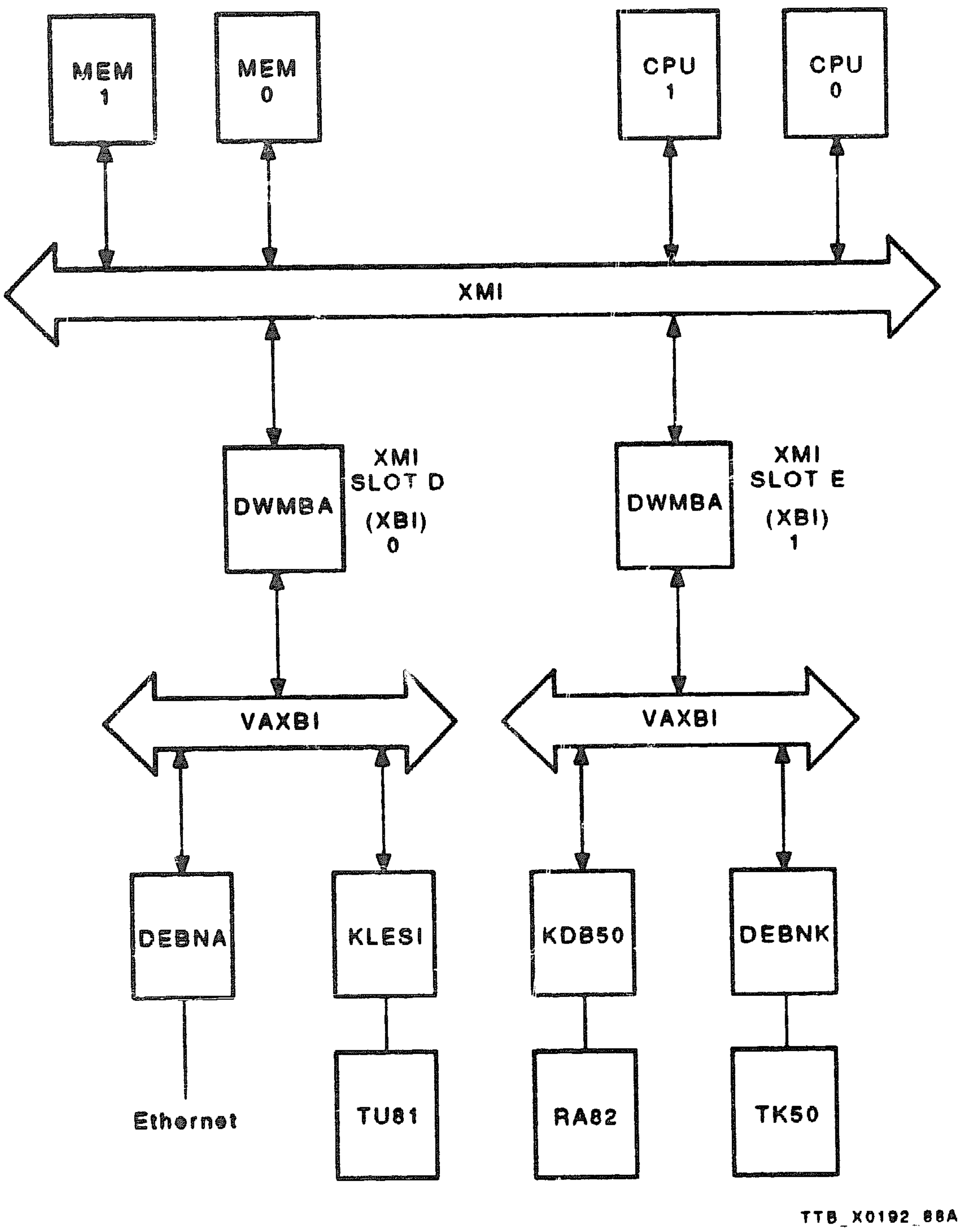


Figure 1–1 DWMBA Adapters in a Typical System

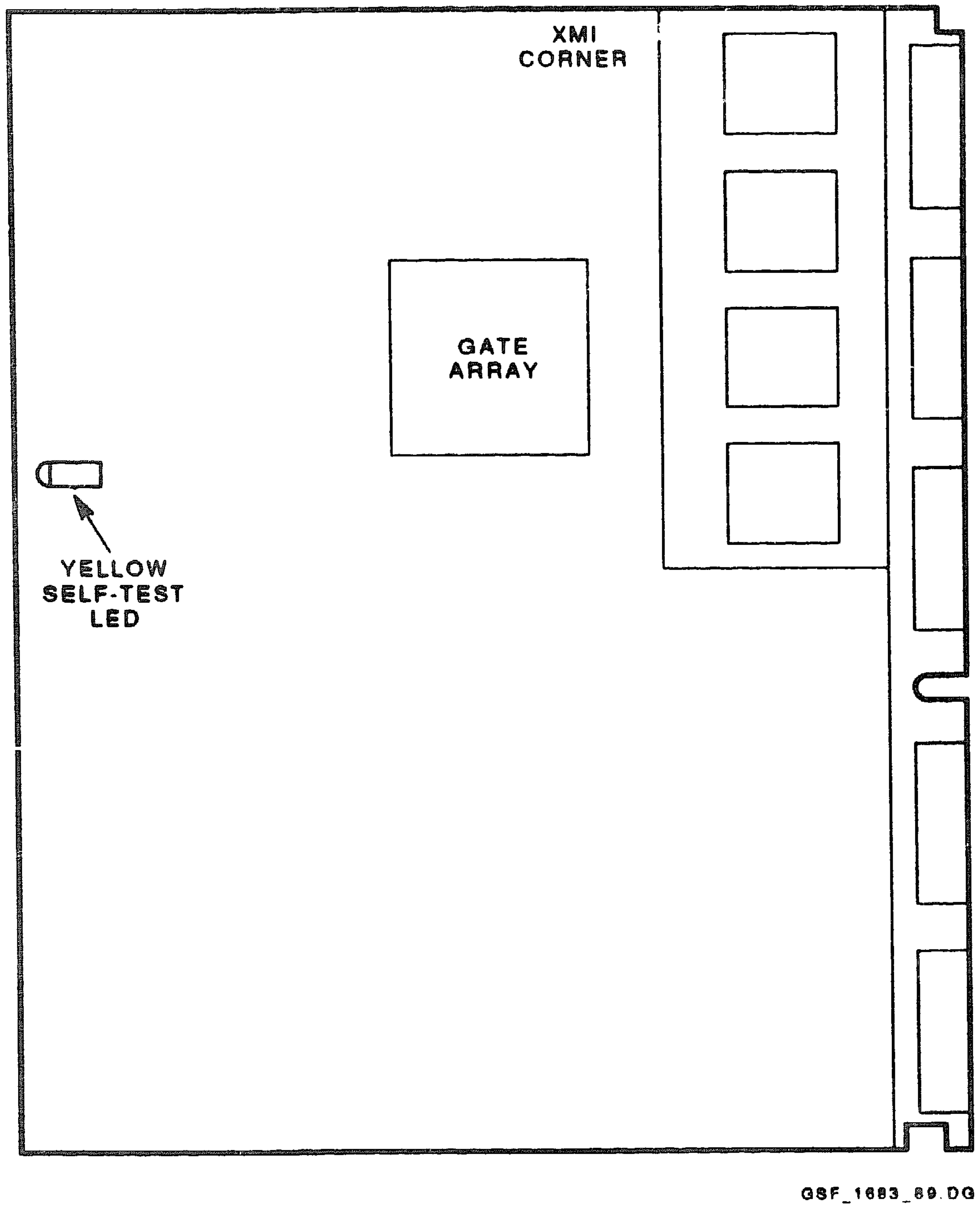


Figure 1–2 DWMBA/A (XBIA) Module (T2012)

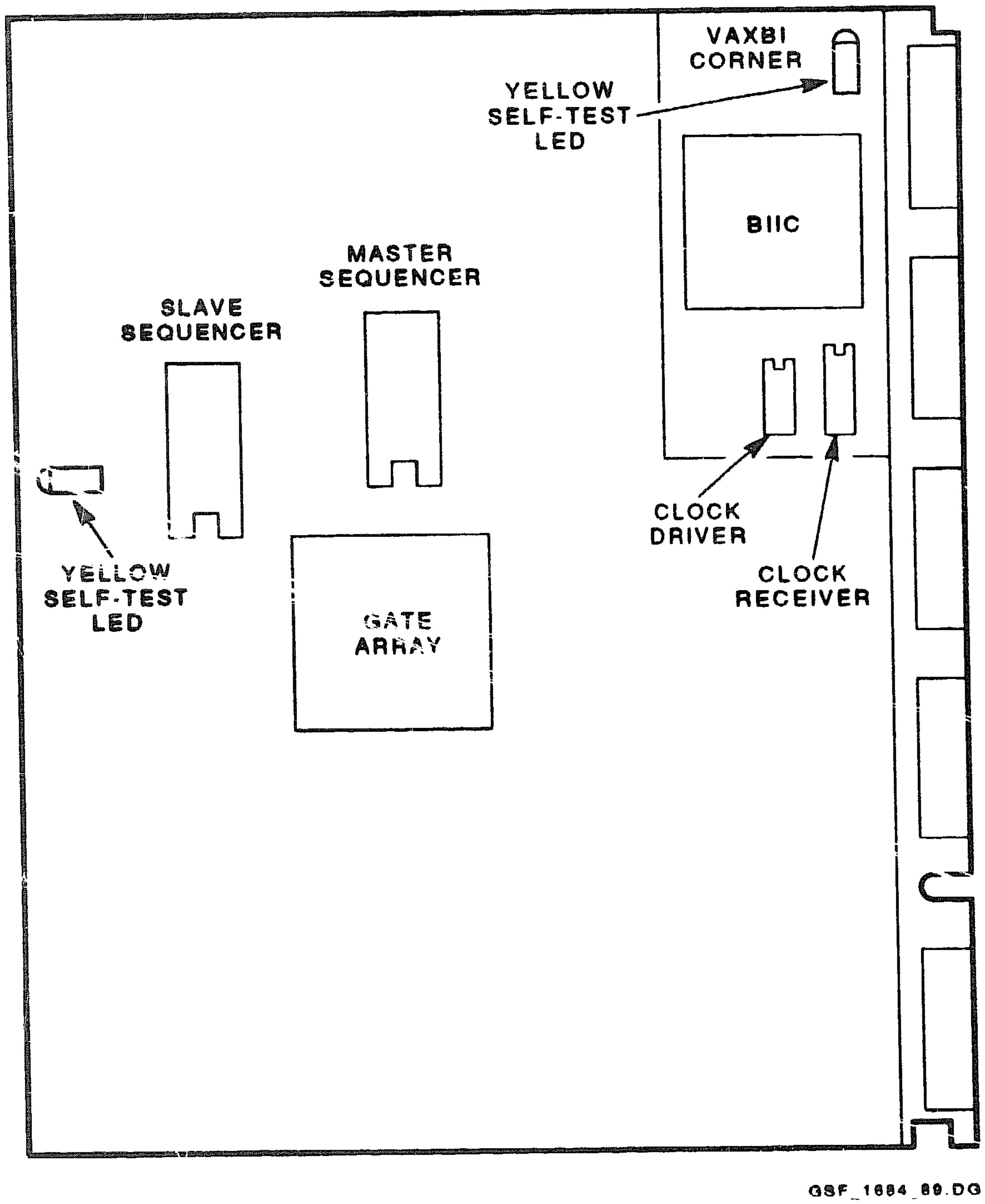


Figure 1–3 DWMBA/B (XBIB) Module (T1043)

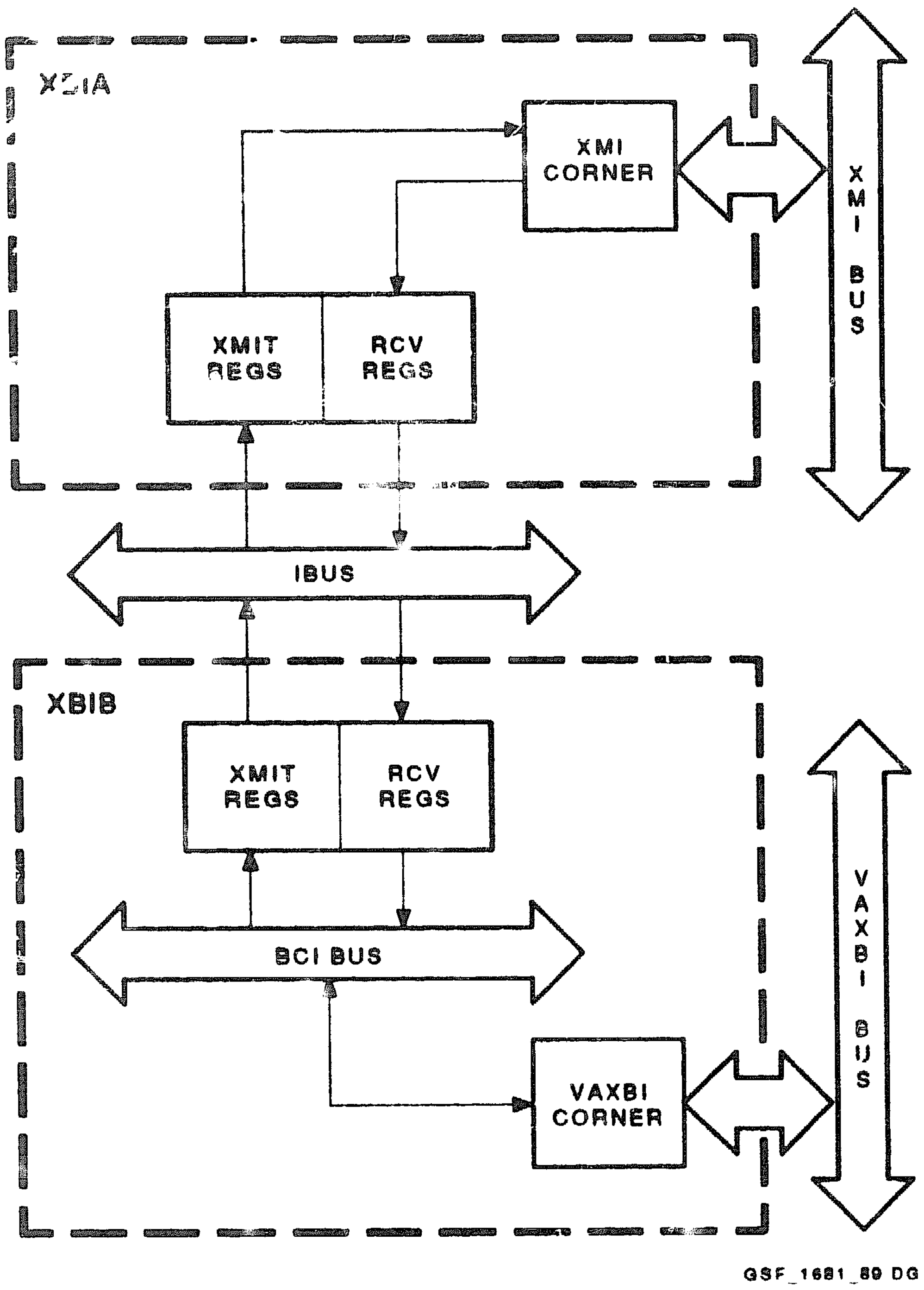


Figure 1–4 DWMBA Functional Logic Elements

1.3 FUNCTIONAL OVERVIEW

The DWMBA converts the 64-bit wide XMI bus transactions to the 32-bit wide VAXBI bus transactions and controls the timing of operations between the two buses. The DWMBA can function as either bus master or bus slave on the VAXBI, and as either commander or responder on the XMI.

The DWMBA performs two basic types of transactions: central processing unit (CPU) and direct memory access (DMA). Table 1–1 describes the transactions.

Table 1–1 DWMBA Transactions

Type	Description
CPU	Transaction originated by CPU and presented on XMI bus to DWMBA. Commands progress from the XBIA to the XBIB. The XBIA functions as XMI bus responder and the XBIB as the VAXBI bus master. Only longword transactions take place. The CPU can reference a portion of physical address space dedicated to I/O, a DWMBA specific register, or VAXBI address space, including BIIC internal registers. If the transaction does not reference a DWMBA specific register, the DWMBA initiates a VAXBI transaction to perform the required operation.
DMA	Transactions originated by a VAXBI node. Commands progress from the XBIB to the XBIA. The XBIA functions as the XMI bus commander and the XBIB as the VAXBI slave. Transactions can be longword, quadword, or octaword. When a VAXBI node selects the DWMBA for a transaction, the DWMBA translates the request into an XMI transaction which is serviced by a memory node. Data is read from or written to XMI memory. The DWMBA is considered selected when the address given in the VAXBI transaction falls between the starting and ending address registers internal to the BIIC.

1.4 REFERENCE DOCUMENTS

Order Number	Title
EK-VBISY-RM	*VAXBI System Reference Manual*
EK-DWMBA-MA	*DWMBA XMI to VAXBI Adapter Maintenance Advisory*
EK-640EA-TM	*VAX 6000-400 System Technical Users Guide*

CHAPTER 2

2 DWMBA CONFIGURATIONS

2.1 INTRODUCTION

This chapter describes the configuration requirements for installing the DWMBA. The chapter includes the following topics:

- Module placement
- Cabling

The material in this chapter is based on the VAX 6000-400 systems implementation. Refer to the appropriate user's guide or installation guide for information on installing the DWMBA in other systems.

2.2 MODULE PLACEMENT

The DWMBA requires one slot in the XMI card cage for the XBIA module and one slot in the VAXBI card cage for the XBIB module. The following configuration rules apply when installing the DWMBA (see Figure 2–1):

- The XBIB module which corresponds to the XBIA module in XMI slot D must be placed in VAXBI card cage 1, slot 1.
- The XBIB module which corresponds to the XBIA module in XMI slot E must be placed in VAXBI card cage 2, slot 1.
- Additional XBIB modules are placed in slot 1 of each additional VAXBI card cage as indicated in Table 2–1.

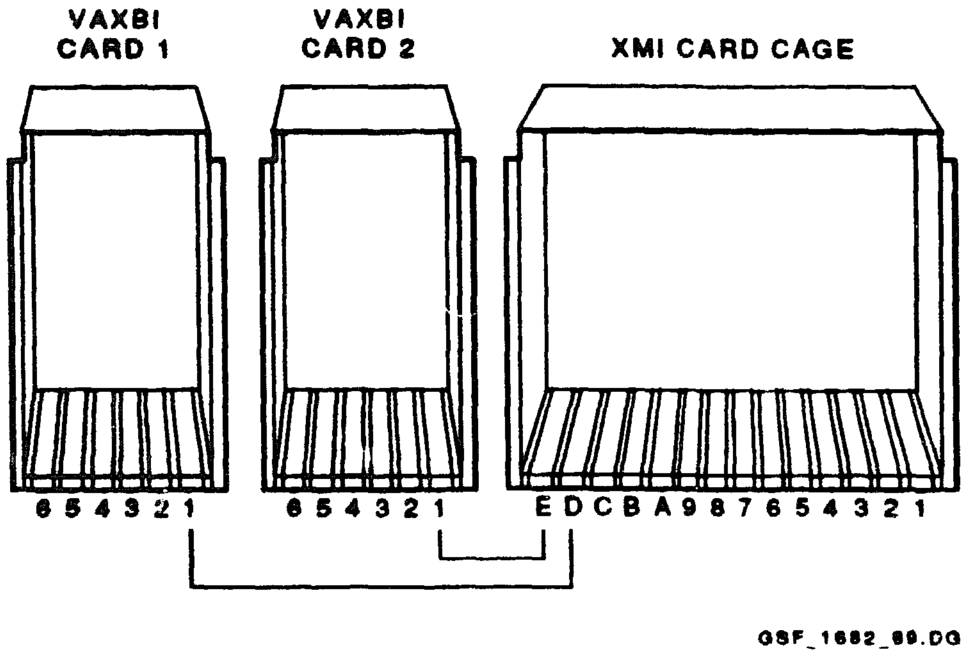


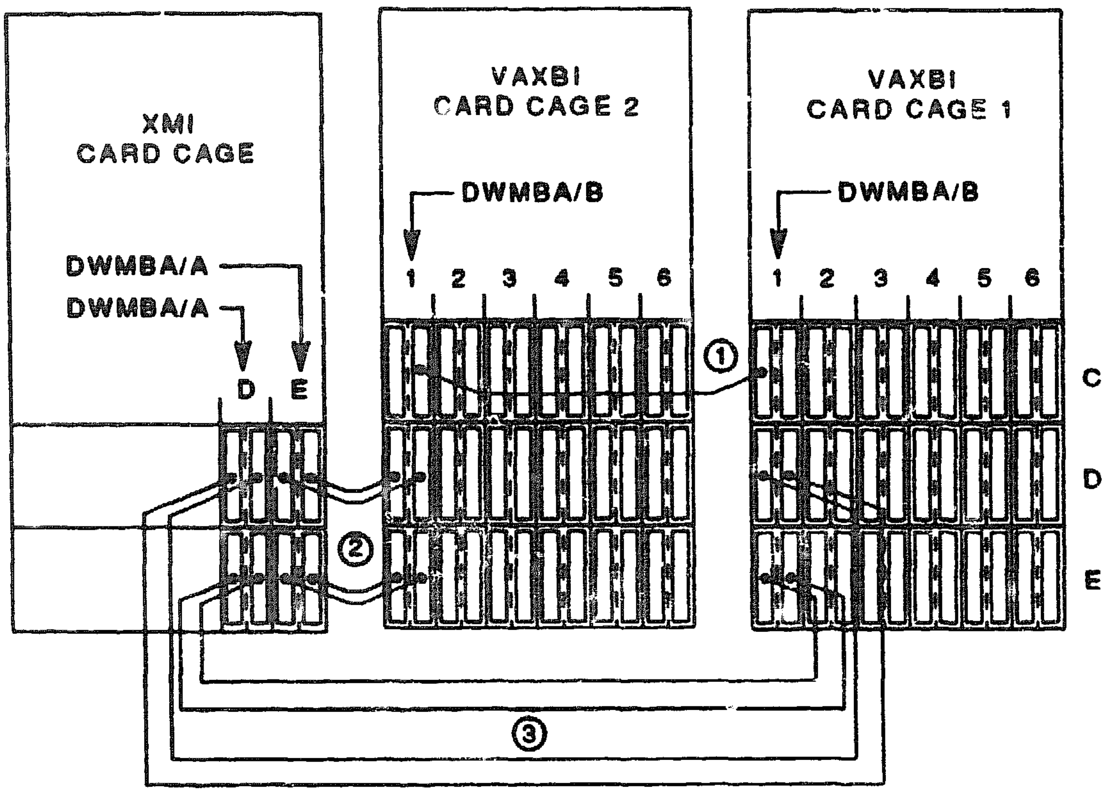
Figure 2-1 XMI and VAXBI Cardcages

Table 2-1 XMI Node/VAXBI Card Cage Configurations

XMI Node	VAXBI Card Cage
D	1
E	2
C	3
B	4
1	5
2	6

2.3 CABLING

The four IBus signal cables which connect the XBIA module to the XBIB module plug into the I/O connector pin segments of the backplane slots into which the modules are installed (segments D and E of each backplane). On systems which implement more than one VAXBI backplane, an AC/DC OK cable is also installed between the VAXBI backplanes. Figure 2-2 shows the DWMBA cabling and Table 2-2 describes each cable.



① 17-00849-08
② 17-01897-02 (2 EACH)
③ 17-01897-03 (2 EACH)

TTB_X1538_88A

Figure 2-2 DWMBA Cabling

Table 2–2 DWMBA Cables

Part Number	Length	Description/Routing
17-00849-08	45.72 cm (18.00 in)	AC/DC OK cable From: VAXBI card cage 2, slot 1, segment C2 To: VAXBI card cage 1, slot 1, segment C1
17-01897-01	4.57 m (15.00 ft)	IBus signal cables (2) to VAXBI expander cabinets (not shown) From: XMI slot C, B, 1, or 2, segments D and E To: VAXBI card cage 3, 4, 5, or 6, slot 1, segments D and E
17-01897-02	17.78 cm (7.00 in)	IBus signal cables (2) From: XMI slot E, segments D and E To: VAXBI card cage 2, slot 1, segments D and E
17-01897-03	63.50 cm (25.00 in)	IBus signal cables (2) From: XMI slot D, segments D and E To: VAXBI card cage 1, slot 1, segments D and E

CHAPTER 3

3 DWMBA DIAGNOSTICS

3.1 INTRODUCTION

This chapter describes the diagnostics available for the DWMBA. The chapter includes the following topics:

- Power-up tests
- ROM-based diagnostics
- Loop-back tests

The material in this chapter is based on the VAX 6000-400 system. Refer to the VAX 6000-400 documentation set for more information on running diagnostics.

3.2 POWER-UP SELF-TESTS

The DWMBA does not have on-board self-tests, but is tested by the boot processor during the system power-up sequence.

After the boot processor conducts its self-tests on system power-up, it automatically sizes the DWMBAs on the system and executes a subset of tests specifically designed for the DWMBA. If the DWMBA passes the tests, its self-test LED is illuminated. If the tests fail, the LED remains extinguished. In either case, power-up test results are displayed on the console. On systems with multiple DWMBAs, the DWMBAs are tested in sequence.

Example 3–1 and Example 3–2 show the console displays for power-up tests with and without errors.

Note that the DWMBA power-up tests are a subset of the boot processor's ROM-based diagnostics and may also be run from the RBD monitor.

```
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE#     ❶

   A  A  .  .  M  M  M  M  .  .  P  P  P  P     TYPE      ❷
   o  o  .  .  +  +  +  +  .  .  +  +  +  +     STF       ❸
   .  .  .  .  .  .  .  .  .  .  E  E  E  B     BPD

   .  .  .  .  .  .  .  .  .  .  +  +  +  +     ETF
   .  .  .  .  .  .  .  .  .  .  E  E  E  B     BPD

.  .  .  .  .  .  .  .  .  +  +  .  +  .  +  .  XBI D +   ❹
.  .  .  .  .  .  .  .  .  +  .  +  .  .  +  .  XBI E +   ❺

   .  .  .  .  .  A4 A3 A2 A1 .  .  .  .  .  .     ILV
   .  .  .  .  .  32 32 32 32 .  .  .  .  .  .     128Mb

ROM = 2.3   EEPROM = 2.0/0.0   SN = NI154

>>>
```

❶ XMI node numbers. Entries in columns 0 and F indicate self-test results for nodes on the VAXBI only (items ❹ and ❺).

❷ XMI node module type: A = I/O adapter; P = processor; M = memory.

❸ Self-test result:

+ = pass
- = fail
. = empty slot
o = VAXBI adapter node

❹ XMI node D: VAXBI nodes 1, 3, 5, and 6 passed self-tests (XBI D +).

❺ XMI node E: VAXBI nodes 1, 4, and 6 passed self-tests (XBI F +).

Example 3-1 Power-up Self-Test Display (No Errors)

```
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE#    ❶

   A  A  .  .  M  M  M  M  .  .  P  P  P  P     TYPE
   o  o  .  .  +  +  +  +  .  .  +  +  +  +     STF
   .  .  .  .  .  .  .  .  .  .  E  E  D  B     BPD

   .  .  .  .  .  .  .  .  .  .  +  +  +  -     ETF
   .  .  .  .  .  .  .  .  .  .  E  B  D  E     BPD

.  .  .  .  .  .  .  .  .  +  +  .  +  .  +  .  XBI D +  ❷
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  XBI E -  ❸

   .  .  .  .  B1 -  A2 A1 .  .  .  .  .  .     ILV
   .  .  .  .  32 -  32 32 .  .  .  .  .  .     96Mb

ROM = 2.3  EEPROM = 2.0/0.0  SN = NI154

>>>
```

❶ XMI node numbers. Entries in column 0 and F indicate self-test results for nodes on the VAXBI only.

❷ XMI node D: All VAXBI nodes passed self-test (XBI D +)

❸ XMI node E: One or more VAXBI nodes failed self-test (XBI E -)

Example 3–2 Power-up Self-Test Display (With Errors)

3.3 ROM-BASED DIAGNOSTICS (RBD)

The DWMBA ROM-based diagnostics are not resident on the DWMBA, but are a subset of the boot processor's RBDs. On VAX 6000-400 systems, RBD 2 is designated for the DWMBA.

Table 3–1 lists the RBD tests. The Default column indicates the tests which are run when the RBD is started and no test number is specified. To run all tests, specify /T=1:26 on the RBD command line.

3.3.1 RBD Monitor

The RBD monitor provides a means for controlling diagnostic execution (for example: test tracing, looping on error, halting on error), selecting specific tests to be run, and for depositing and examining addresses in node private space, XMI space, and VAXBI space.

The DWMBA tests are run by calling the RBD monitor with the T/R console command and issuing the RBD START command:

```
>>>
>>> T/R                    ! Enter RBD monitor
RBD1> START 2 D            ! Start RBD 2, XMI node D
```

The prompt RBD1> indicates that the console is logically connected to the CPU installed in XMI node 1.

Refer to the system user's guide, maintenance guide, or installation guide for detailed information on the RBD monitor.

Table 3–1 DWMBA RBD Tests

Test	Default	Test Title
T0001	Yes	DWMBA/A XMI module CSR test
T0002	No	XMI low longword parity error test
T0003	No	XMI high longword parity error test
T0004	No	XMI function and ID parity error test
T0005	Yes	DWMBA/B CSR test
T0006	Yes	BIIC VAXBI loopback transaction test
T0007	Yes	BIIC VAXBI transaction test
T0008	Yes	DMA test
T0009	Yes	DMA buffer test
T0010	No	XMI parity error interrupt test
T0011	No	Write sequence error interrupt test
T0012	Yes	CPU buffer C/A fetch parity error (interrupt) test
T0013	Yes	CPU buffer data fetch parity error (interrupt) test
T0014	Yes	DMA buffer data fetch parity error (interrupt) test
T0015	Yes	VAXBI interlock read error (interrupt) test
T0016	Yes	DMA-A buffer C/A load parity error (interrupt) test
T0017	Yes	DMA-A buffer data load parity error (IVINTR) test
T0018	Yes	DMA-B buffer C/A load parity error (interrupt) test
T0019	Yes	DMA-B buffer data load parity error (IVINTR) test
T0020	Yes	CPU buffer data load parity error (interrupt) test
T0021	Yes	BCI parity error test
T0022	Yes	Nonexistent memory (interrupt) test
T0023	Yes	CRD error (interrupt) test
T0024	Yes	VAXBI interrupt test
T0025	Yes	VAXBI IP interrupt test
T0026	Yes	No stall timeout test

3.3.2 Sample RBD Runs

```
>>>                            ! Console prompt
>>> T/R                        ! Enter RBD monitor
RBD1>                          ! RBD prompt (1 = XMI node number of
                               ! CPU currently receiving input)
RBD1> ST 2/TR D                ! Run RBD 2 (DWMBA), trace tests,
                               ! XMI node D, default test set

;XBI_SLF  3.0

;T0001  T0005  T0006  T0007  T0008  T0009  T0012  T0013  T0014
;T0015  T0016  T0017  T0018  T0019  T0020  T0021  T0022  T0023
;T0024  T0025  T0026

;  P         D      2001 00000001
; 00 00000000 00000000 00000000 00000000 00000000 00

RBD1>
```

Example 3–3   DWMBA RBD Run With No Errors

```
>>>                            ! Console prompt
>>> T/R                        ! Enter RBD monitor
RBD1> ST 2/TR/HE/T=1:26 D      ! RBD 2, trace, halt on error, all
                               ! tests (/T=1:26), XMI node D.

;XBI_SLF          3.4

;T0001  T0002  T0003  T0004  T0005

; F         D      2001 00000001                             ➊
; HE BCSR_REG        00       T05                            ➋
; 05 80000000 00000000 00000000 21E80058 200628F5 01         ➌

RBD1>
```

➊ Failed, XMI node D, DWMBA (device type 2001), 1st pass

➋ Hard error, BCSR_REG failing component, unit 00, test T05

➌ Error code 05, expected data 80000000, recieved data 00000000, implementation specific (00000000, 21E80058), PC 200628F5, error number 01

Example 3–4   DWMBA RBD Run With Errors

3.4 LOOPBACK TESTS

When the default set of RBD tests are run, two types of loopback tests are performed:

- VAXBI loopback
- DMA loopback

VAXBI Loopback Tests

The VAXBI loopback test verifies the data path from the CPU through the XMI, all modules of the DWMBA, and back through the XMI to the CPU. Figure 3–1 shows the VAXBI loopback data path.

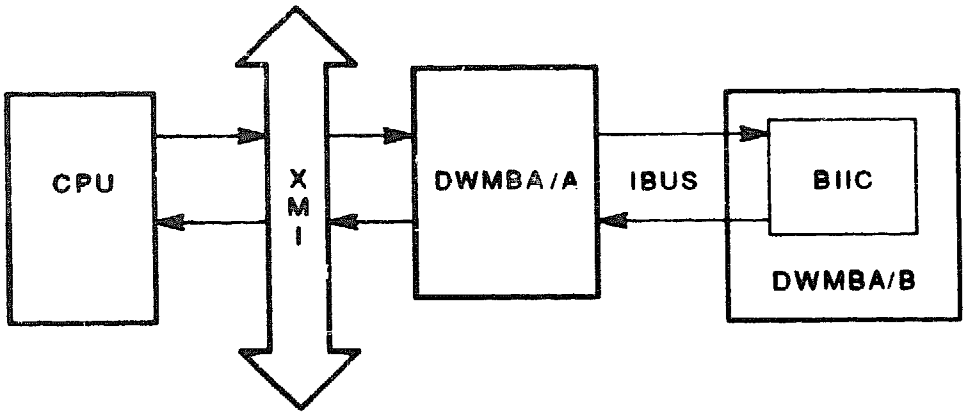


Figure 3–1 VAXBI Loopback Data Path

CPU DMA Loopback Tests

The DMA loopback tests include tests of CPU write transactions and CPU re transactions. Figure 3–2 shows the data path for CPU write transactions and Figure 3–3 shows the data path for CPU read transactions. In both cases, the data path originates at the CPU.

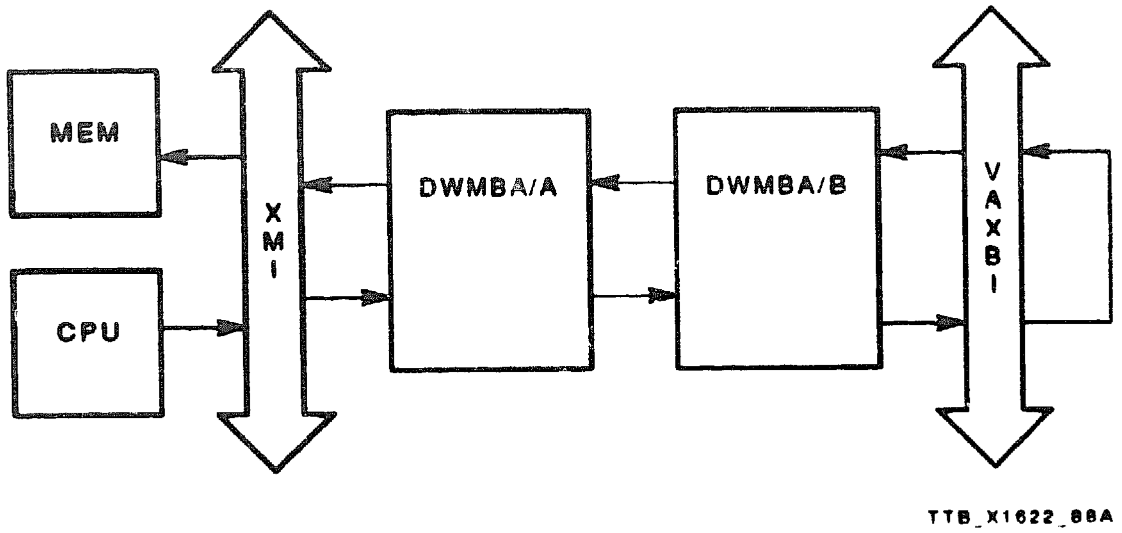


Figure 3-2 CPU Write Loopback Data Path

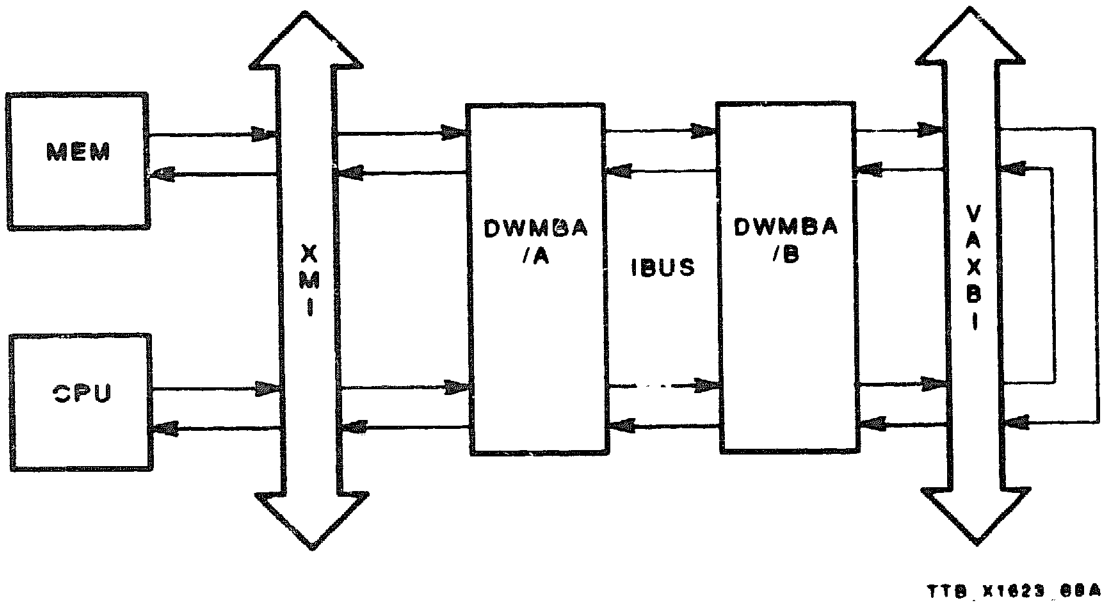


Figure 3-3 CPU Read Loopback Data Path

CHAPTER 4

4 DWMBA REGISTERS AND IBUS SIGNALS

4.1 INTRODUCTION

This chapter overviews the DWMBA register structure. Included in the chapter are:

- Lists of the DWMBA registers:
  - XMI architecture
  - VAXBI architecture
  - DWMBA specific
- Register bit maps
- Descriptions of selected registers
- List of the IBus signals

The chapter is a quick reference to DWMBA register information. Refer to the *DWMBA XMI to VAXBI Adapter Maintenance Advisory* and the *VAXBI System Reference Manual* for detailed descriptions of the registers.

4.2 DWMBA REGISTER TYPES

The DWMBA includes three types of registers:

- XMI architecture
- VAXBI architecture
- DWMBA specific

The XMI architecture registers reside in the XBIA module. The VAXBI architecture registers reside in the BIIC chip of the XBIB module. The DWMBA specific registers reside in both modules. The XMI architecture and DWMBA specific registers are addressed in the DWMBA's XMI nodespace. The VAXBI architecture registers are addressed in the DWMBA's XMI I/O adapter space.

Table 4–1 lists the XMI architecture and DWMBA specific registers. Table 4–2 lists the VAXBI architecture registers.

Table 4–1 XMI Architecture and DWMBA Specific Registers

Mnemonic	Address[1]	Register Name
XMI Architecture		
XDEV	bb+0000	Device type register
XBE	bb+0004	Bus error register
XFADR	bb+0008	Failing address register
DWMBA Specific, XBIA Module Resident		
AREAR	bb+000C	Responder error address register
AESR	bb+0010	Error summary register
AIMR	bb+0014	Interrupt mask register
AIVINTR	bb+0018	Implied vector interrupt destination/diagnostic register
ADG1	bb+001C	Diagnostic control register 1
DWMBA Specific, XBIB Module Resident		
BCSR	bb+0040	Control and status register
BESR	bb+0044	Error summary register
BIDR	bb+0048	Interrupt destination register
BTIM	bb+004C	Timeout address register
BVOR	bb+0050	Vector offset register
BVR	bb+0054	Vector register
BDCR1	bb+0058	Diagnostic control register 1
—	bb+005C	Reserved

[1]Offset (in hex) from node's XMI nodespace base address.

Table 4–2 VAXBI Architecture Registers

Mnemonic	Address[1]	Name
DTYPE	bb+00	Device register
VAXBICSR	bb+04	VAXBI control and status register
BER	bb+08	Bus error register
EINTRSCR	bb+0C	Error interrupt control register
INTRDES	bb+10	Interrupt destination register
IPINTRMSK	bb+14	IPINTR mask register
FIPSDES	bb+18	Force-bit IPINTR/STOP destination register
IPINTRSRC	bb+1C	IPINTR source register
SADR	bb+20	Starting address register
EADR	bb+24	Ending address register
BCICSR	bb+28	BCI control and status register
WSTAT	bb+2C	Write status register
FIPSCMD	bb+30	Force-bit IPINTR/STOP command register
UINTRCSR	bb+40	User interface interrupt control register
GPR0	bb+F0	General purpose register 0
GPR1	bb+F4	General purpose register 1
GPR2	bb+F8	General purpose register 2
GPR3	bb+FC	General purpose register 3
SOSR	bb+100	Slave-only status register
RXCD	bb+200	Receive console data register

[1]Offset (hex) from node's I/O adapter space base address.

4.3 REGISTER DESCRIPTION CONVENTIONS

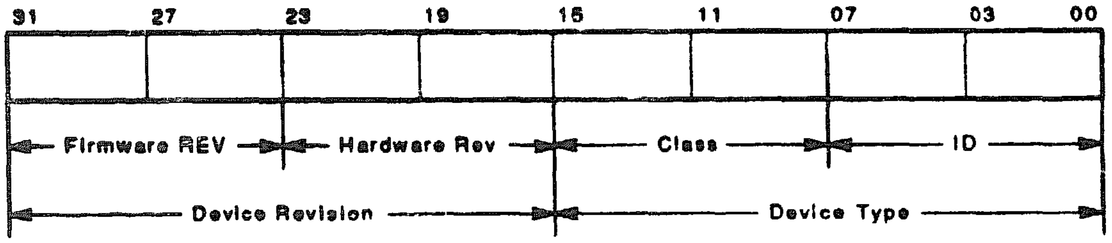
In the register description tables that follow, the access type of the bit(s) being described is denoted by a mnemonic enclosed in parentheses after the bit field name. The bit access codes are as follows:

Code	Indication
0	Bit(s) initialized to logic 0
1	Bit(s) initialized to logic 1
RO	Read-only
R/W	Read/write
R/W1C	Read/write-1-to-clear

4.4 XMI ARCHITECTURE REGISTERS

The following registers are the minimum XMI architecture registers which must be present in the node. These registers all reside on the DWMBA/A module.

4.4.1 XMI Device Type Register (XDEV, bb+00000)



GSF_1736_89 DG

NOTE
The XDEV register bit map shown is generic to XMI I/O devices. The DWMBA does not implement separate firmware and hardware revision fields.

Bit(s)	Name/Description
31:16	Device revision (RO, 0)

Identifies the revision level (letter only) of the DWMBA/A module. A zero value indicates an uninitialized node:

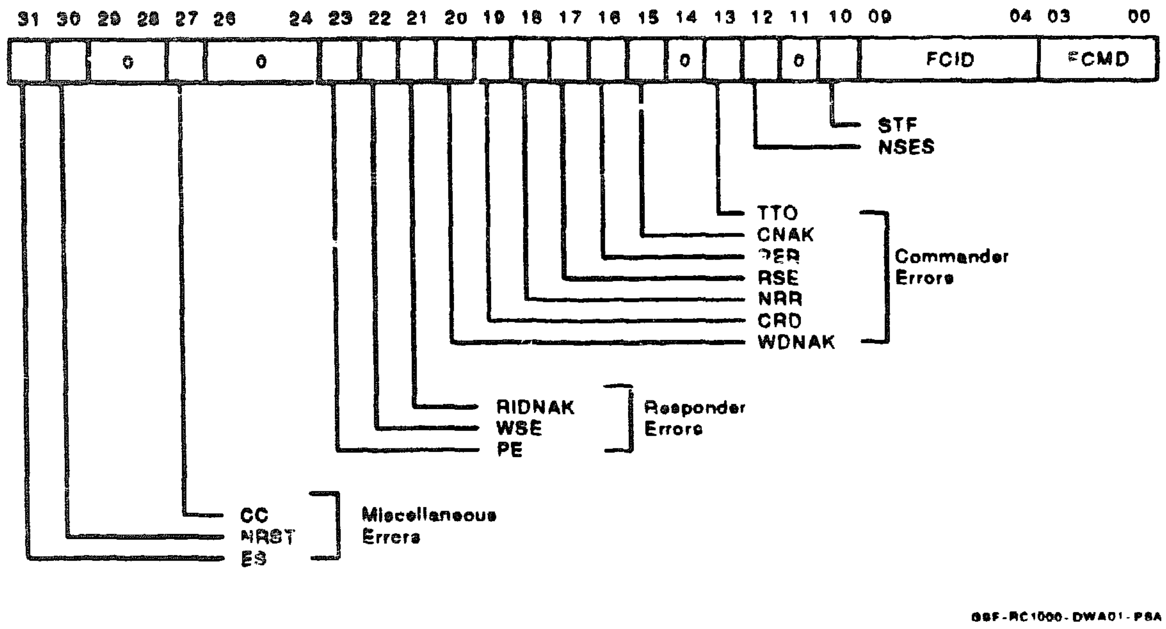
Value	Revision
0001	A0, A1, . . . An
0002	B0, B1, . . . Bn
.	
.	
.	
001A	Z0, Z1, . . . Zn

Bit(s)	Name/Description
15:00	Device type (RO, 0) Identifies the device type and XMI device ID of the DWMBA. A zero value indicates an uninitialized node. The DTYPE field is divided into two subfields:

Field	Bit Descriptions
Class	Indicates category in which node falls: <15>—CPU device <14>—Memory device <13>—Bus window (I/O) <12>—Bus window (Memory) <11>—I/O device <10>—XCOMM register present
ID	Uniquely identifies particular device within specified class.

The DWMBA device type is 2001.

4.4.2 XMI Bus Error Register (XBE, bb+00004)



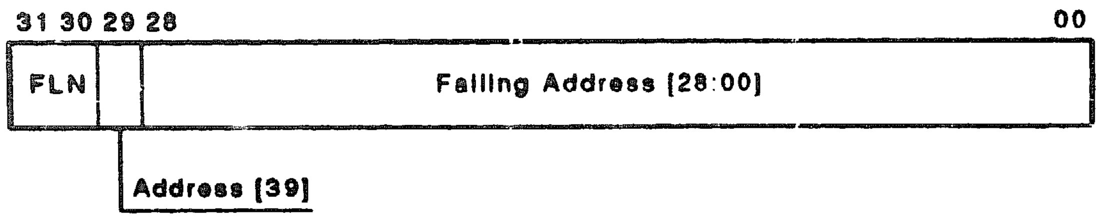
Bit(s)	Name/Description
Miscellaneous Errors	
31	Error summary (RO, 0) Logical "OR" of the error bits in this register: <27,23:20,18:15,13:12>.

Bit(s)	Name/Description
Miscellaneous Errors	
30	Node reset (R/W, 0) When set, initiates a system power-up reset. Reads of this bit return a zero. When NRST is set, the DWMBA: • Resets the XBIA module to the initialized (power-up) state. • Asserts RESET control to the XBIB module, sequencing the VAXBI power supply(s). The assertion of RESET to the XBIB causes the module to sequence BI AC LO, and BI DC LO. The assertion of BI DC LO resets the XBIB to the initialized (power-up) state. NRST remains set for six to eight XMI cycles until cleared by the XBIA. During this time, the DWMBA does not affect the operation of the XMI bus.
29:28	Not implemented. Reads of these bits return a zero.
27	Corrected confirmation (R/W1C, 0) Set by DWMBA on a single-bit CNF error (single bit CNF errors are automatically corrected by the XCLOCK chip). Also sets bit <31>.
26:24	Not implemented. Reads of these bits return a zero.
Responder Errors	
23	Parity error (R/W1C, 0) Set if the DWMBA detected bad parity on an XMI cycle. The cycle need not have been directed to the DWMBA. Also sets bit <31>.
22	Write sequence error (R/W1C, 0) Set if the DWMBA aborted a write transaction due to a missing data cycle. Also sets bit <31>.
21	Read/IDENT data NOACK (R/W1C, 0) Set if a DWMBA initiated READ or IDENT data cycle received a NOACK confirmation. Also sets bit <31>.

Bit(s)	Name/Description
Commander Errors	
20	Write data NOACK (R/W1C, 0) Set if a DWMBA initiated WRITE data cycle received repeated NOACK confirmations for the duration of the timeout period. Also sets bit <31>.
19	Corrected read data (R/W1C, 0) Set if the DWMBA received a CRDn response.
18	No read response (R/W1C, 0) Set if a DWMBA initiated READ failed due a read response timeout. Also sets bits <31> and <13>.
17	Read sequence error (R/W1C, 0) Set if a DWMBA initiated READ received read data out of sequence. The failing command/address is available in XFADR. Also sets bit <31>.
16	Read error response (R/W1C, 0) Set if the DWMBA received a read error response. The failing command/address is available in XFADR. Also sets bit <31>.
15	Command NOACK (R/W1C, 0) Set if a DWMBA initiated command cycle received repeated NOACK confirmations for the duration of the timeout period. Also sets bits <31> and <13>. CNAK can result from a reference to a non-existent memory location or a command cycle parity error. The bit is set only if repeated attempts fail.
14	Not implemented. Reads of this bit return a zero.

Bit(s)	Name/Description
Commander Errors	
13	Transaction timeout (R/W1C, 0) Set if a DWMBA initiated transaction did not complete within the timeout period. The failing command/address is available in the XFADR. Also sets bit <31>. TTO may be set along with bits <20>, <18>, or <15>. If none of these bits is set, the DWMBA either: 1. Failed to win bus arbitration within the timeout period 2. Attempted to execute an IREAD command but XMI lockout remained asserted for timeout period
Node Specific Errors	
12	Node specific error summary (RO, 0) Set if the DWMBA detects a node specific error condition. Error information is contained in DWMBA specific registers. Also sets bit <31>.
11	Not implemented. Reads of this bit return a zero.
10	Self-test fail (R/W1C, 1) Set during the power-up sequence until the DWMBA passes the power-up tests. Bit is cleared by the CPU node which initiated the tests.
09:04	Failing commander ID (RO) Logs the commander ID of a failed XMI transaction. FCID is set only if the transaction failed on retry.
03:00	Failing command (RO) Logs the command code of a failed transaction. FCMD is set only if the transaction fails on retry.

4.4.3 XMI Failing Address Register (XFADR, bb+00008)



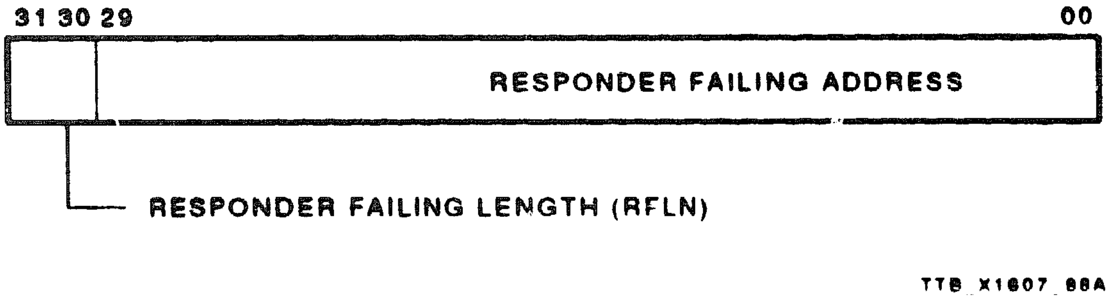
GSF_1739_89.DG

Bit(s)	Name/Description
31:30	Failing length (RO) Logs the value of XMI D <31:30> during the command cycle of a failed transaction.
29:00	Failing address (RO) Logs the value of XMI D <29:00> during the command cycle of a failed transaction. NOTE **The XFADR register bit map is shown for systems with 40-bit addressing. On these systems, XFADR bits <28:00> log address bits <28:00>, and bit <29> logs address bit <39>.**

4.5 DWMBA/A RESIDENT NODE SPECIFIC REGISTERS

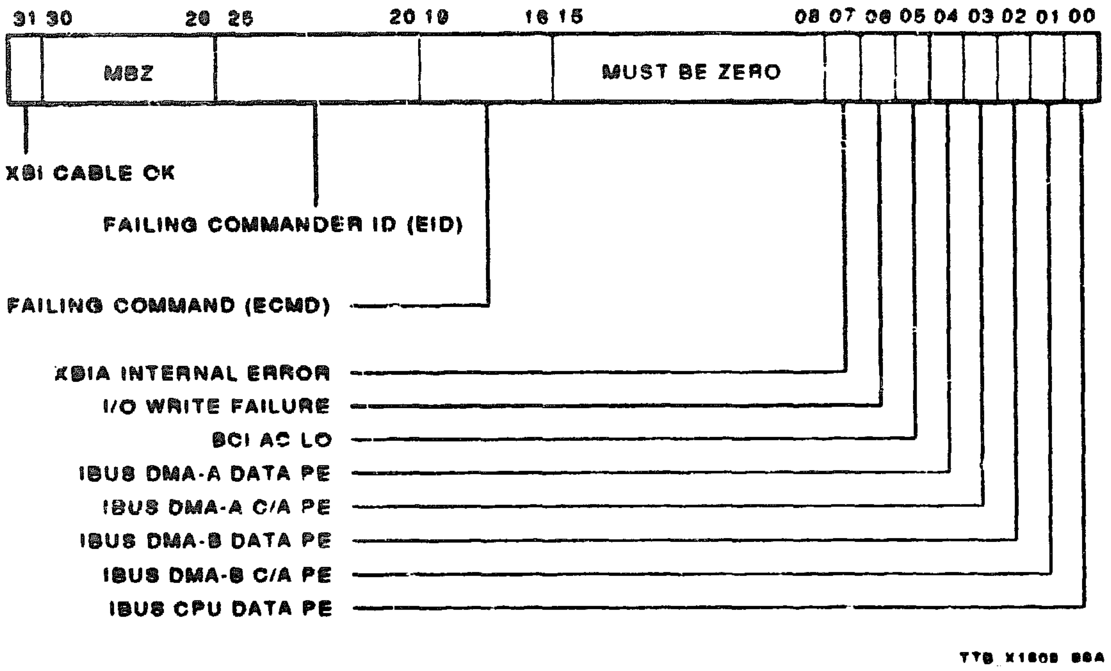
The DWMBA specific registers resident on the DWMBA/A module are primarily associated with XMI bus transactions and events. These registers are indicated by the prefix "A" in the register name.

4.5.1 Responder Error Address Register (AREAR, bb+000C)



Bit(s)	Name/Description
31:30	Responder failing length (RO) Logs the value of XMI D <31:30> of a failed XMI transaction. RFLN is loaded when the DWMBA ACKs the C/A cycle.
29:00	Responder failing address (RO) Logs the value of XMI D <29:00> of a failed XMI transaction. The address field is loaded when the DWMBA ACKs the C/A cycle.

4.5.2 Error Summary Register (AESR, bb+0010)

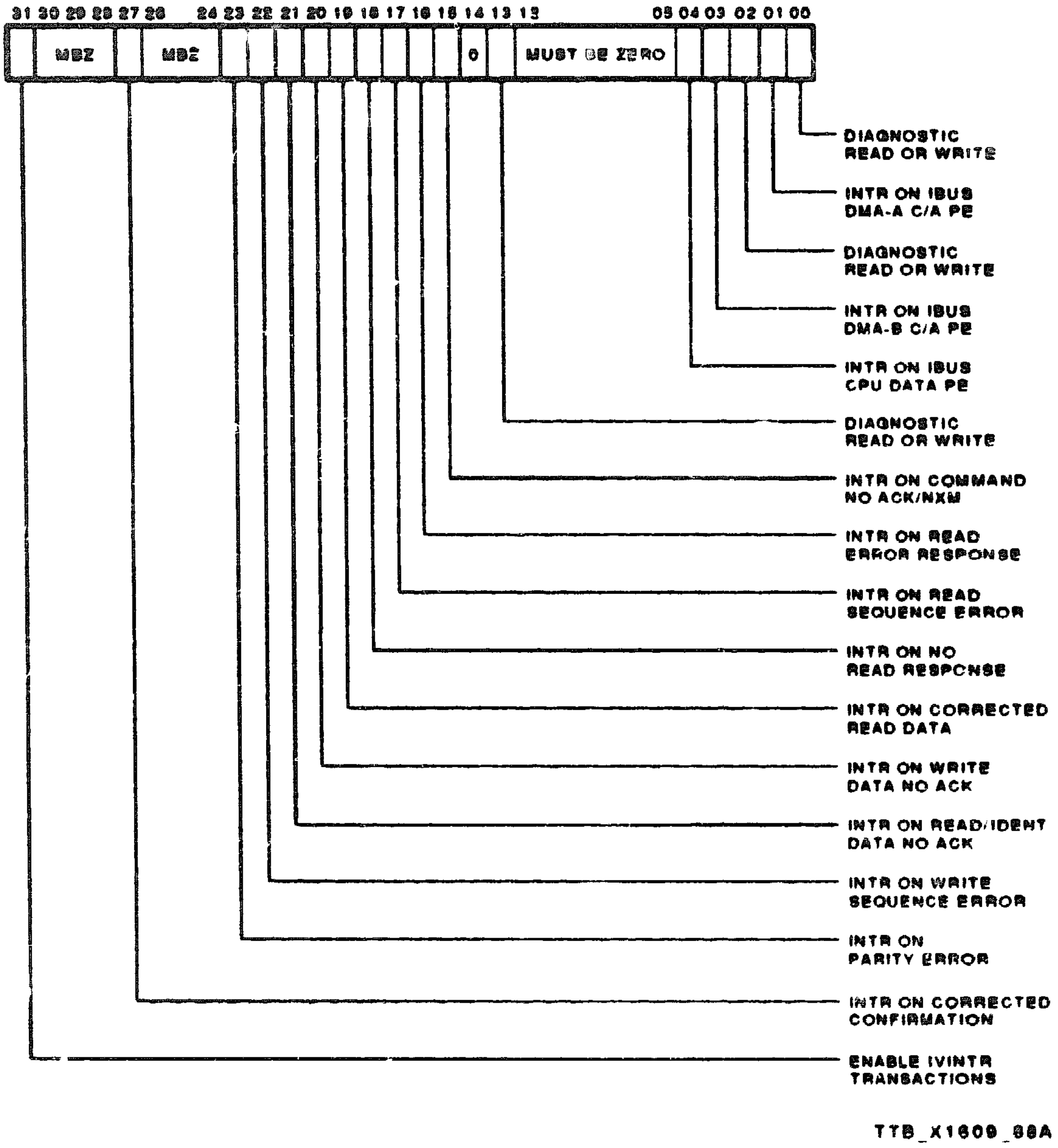


Bit(s)	Name/Description
31	**XBI cable OK (RO)** Set on initialization if the IBus cables are properly installed and the XBIB module has dc power from the VAXBI backplane. If clear, and the XBIB has dc power, indicates that one or more cables is disconnected or improperly installed.
30:26	Reserved, must be zero.

Bit(s)	Name/Description
25:20	Failing commander ID (RO) Logs the XMI commander ID of a failed I/O write, I/O read, or IDENT transaction. EID is loaded after the DWMBA ACKs the XMI commander C/A cycle. The EID is locked if the DWMBA is unable to complete the operation as follows: 1. CPU write transaction fails -- sets AESR bit <06> (I/O write failure). 2. CPU read or IDENT transaction fails — sets XBER bit <21> (RIDNAK). EID is unlocked when the locking error condition clears.
19:16	Failing command (RO) Logs the XMI command of a failed DWMBA I/O write, I/O read, or IDENT transaction. DWMBA loads the ECMD when it ACKs the XMI commander C/A cycle. The ECMD is locked and unlocked under the same conditions as for bits <25:20>.
15:08	Reserved, must be zero.
07	XBIA Internal error (R/W1C, 0) Set if an *unexplained* error internal to the XBIA module gate array is detected. Usually indicates a hardware problem (control logic encountered an undefined condition). The DWMBA issues an IVINTR with "memory write error" set in the type field.
6	I/O write failure during CPU write transaction (R/W1C, 0) Set if the XBIB module is unable to complete a CPU write transaction to one of its registers or to VAXBI address space. Setting this bit generates an IVINTR transaction with "memory write error" in the type field. When I/O write failure is set, bits <25:20> (EID), bits <19:16>, and the contents of AREAR are locked.

Bit(s)	Name/Description
5	**BCI AC LO (R/W1C, 1)** Set if VAXBI power falls below specifications. The DWMBA issues an IVINTR with "memory write error" in the type field. The interrupt service routine clears the bit. BCI AC LO is cleared by the DWMBA power-up test.
4	**IBus DMA-A data parity error (R/W1C, 0)** Set if the XBIA detects bad IBus parity while attempting to load a DMA-A data buffer location. The DWMBA issues an IVINTR with "memory write error" in the type field.
3	**IBus DMA-A C/A parity error (R/W1C, 0)** Set if the XBIA detects bad IBus parity while attempting to load a DMA-A C/A location. The DWMBA issues an IVINTR with "memory write error" in the type field if the the failing transaction is a write or interrupt. The DWMBA issues an error interrupt if this error bit is set and the appropriate mask bit is also set.
2	**IBus DMA-B data parity error (R/W1C, 0)** Set if the XBIA detects bad IBus parity while attempting to load a DMA-B data buffer location. The DWMBA issues an IVINTR with "memory write error" in the type field.
1	**IBus DMA-B C/A parity error (R/W1C, 0)** Set if the XBIA detects bad IBus parity while attempting to load a DMA-B C/A location. The DWMBA issues an IVINTR with "memory write error" in the type field if the failing DMA transaction is a write. The DWMBA issues an error interrupt if this error bit is set and the appropriate mask bit is also set.
0	**IBus CPU data parity error (R/W1C, 0)** Set if the XBIA detects bad IBus parity while attempting to load a CPU DATA location on a CPU-initiated I/O read or IDENT. The DWMBA issues a read error response (RER) to the commander and an error interrupt to the XMI if the appropriate mask bit is also set.

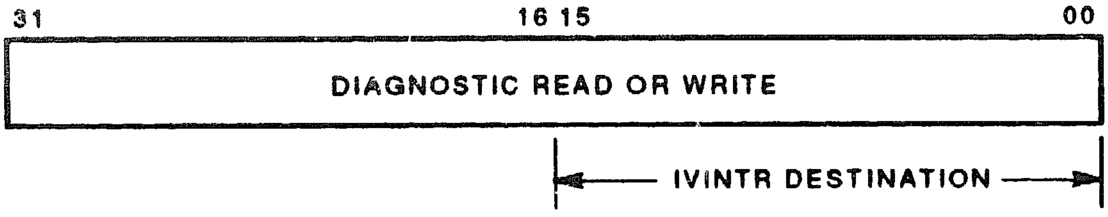
4.5.3 Interrupt Mask Register (AIMR, bb+0014)


31 30 29 28 27 26 24 23 22 21 20 19 18 17 16 15 14 13 12 05 04 03 02 01 00
MBZ
MBZ
0
MUST BE ZERO
DIAGNOSTIC READ OR WRITE
INTR ON IBUS DMA-A C/A PE
DIAGNOSTIC READ OR WRITE
INTR ON IBUS DMA-B C/A PE
INTR ON IBUS CPU DATA PE
DIAGNOSTIC READ OR WRITE
INTR ON COMMAND NO ACK/NXM
INTR ON READ ERROR RESPONSE
INTR ON READ SEQUENCE ERROR
INTR ON NO READ RESPONSE
INTR ON CORRECTED READ DATA
INTR ON WRITE DATA NO ACK
INTR ON READ/IDENT DATA NO ACK
INTR ON WRITE SEQUENCE ERROR
INTR ON PARITY ERROR
INTR ON CORRECTED CONFIRMATION
ENABLE IVINTR TRANSACTIONS
TTB_X1609_88A

Bit(s)	Name/Description
31	Enable IVINTR transactions (R/W, 0) When set, enables issuing IVINTRs on XMI if the IVINTR destination register is properly configured. NOTE **Bit <31> must be set to ensure proper error reporting of asynchronous write failures and the occurrence of a pending VAXBI power-fail not initiated by XMI AC LO, XMI DC LO, or VAXBI node reset.**
30:28	Reserved (RO, 0). Bits must be zero.
27	INTR on corrected confirmation (R/W, 0) When set, the XBIA asserts the IR XMI ERR BIT SET L line on the IBus, which generates an interrupt request if XBER <27> (corrected confirmation) is set.
26:24	Reserved (RO, 0). Bits must be zero.
23	INTR on parity error (R/W, 0) Same as bit <27> except interrupt generated if XBER <23> is set.
22	INTR on write sequence error (R/W, 0) Same as bit <27> except interrupt generated if if XBER <22> is set.
21	INTR on read/IDENT NOACK (R/W, 0) Same as bit <27> except interrupt generated if XBER <21> is set.
20	INTR on write data NOACK (R/W, 0) Same as bit <27> except interrupt generated if XBER <20> is set.
19	INTR on corrected read data (R/W, 0) Same as bit <27> except interrupt generated if XBER <19> is set.
18	INTR on no read response (R/W, 0) Same as bit <27> except interrupt generated if XBER <18> is set.
17	INTR on read sequence error (R/W, 0) Same as bit <27> except interrupt generated if XBER <17> is set.
16	INTR on read error response (R/W, 0) Same as bit <27> except interrupt generated if XBER <16> is set.

Bit(s)	Name/Description
15	INTR on command NOACK (R/W, 0) Same as bit <27> except interrupt generated if XBER <15> is set.
14	Reserved (RO, 0). Bits must be zero.
13	Diagnostic read or write (RO) Used by diagnostics.
12:5	Reserved (RO, 0). Bits must be zero.
4	Diagnostic read or write (RO) Used by diagnostics.
3	INTR on IBus DMA-A C/A PE (R/W, 0) Same as bit <27> except interrupt generated if an IBus parity error was detected while the XBIB attempted to load a DMA-A C/A location.
2	Diagnostic read or write (RO) Used by diagnostics.
1	INTR on IBus DMA-B C/A PE (R/W, 0) Same as bit <27> except interrupt generated if an IBus parity error was detected while the XBIB was attempting to load a DMA-B C/A location.
0	INTR on IBus CPU DATA PE (R/W, 0) Same as bit <27> except interrupt generated if an IBus parity error was detected while the XBIB was attempting to load the CPU data location.

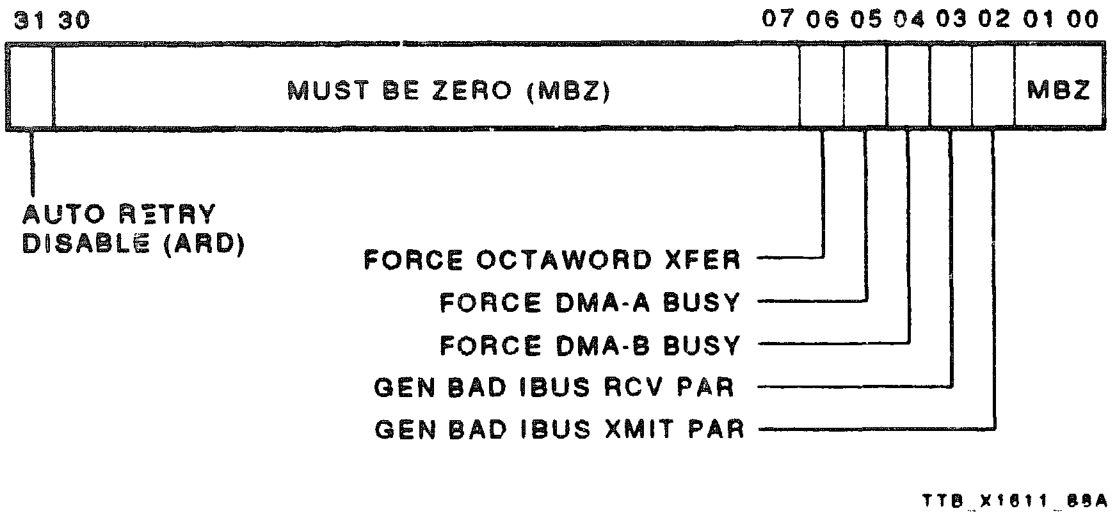
4.5.4 Implied Vector Interrupt Destination/Diagnostic Register (AIVINTR, bb+0018)



TTB_X1610_88A

Bit(s)	Name/Description
31:00	Diagnostic read or write (R/W) Used by diagnostic routines to verify the integrity of the main data path in the XBIA gate array. Diagnostics raise the processor IPL level above IPL 30 to inhibit the XBIA from issuing an IVINTR (generating an unexpected interrupt) should an error occur. On DWMBA initiated IVINTR transactions, bits<15:00> are the IVINTR destination bits.
15:00	IVINTR destination (R/W, 0) Specifies the XMI nodes targeted by the DWMBA on an implied vector interrupt transaction. Each bit corresponds to an XMI node. For example, if bit <12> is set, XMI node 12 is selected for the IVINTR transaction. Any number of bits can be set.

4.5.5 Diagnostic Control Register 1 (ADG1, bb+001C)



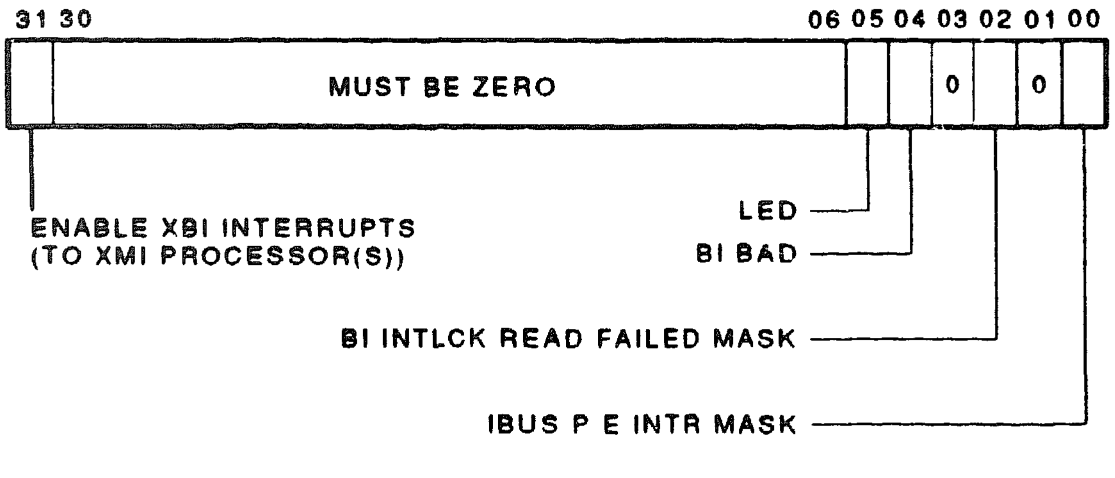
Bit(s)	Name/Description
31	**Auto retry disable (R/W, 0)** When set, disables retries of failed XMI commander transfers. XMI error indications (NOACKs) are immediately logged in the XBER and the appropriate action is taken. NOTE **Since an XMI node can issue a valid NOACK due to a busy condition, the user must ensure that either a busy NOACK cannot be issued by the targeted node or that the DWMBA can handle an incomplete transaction if auto retry disable is set.**
30:7	**Reserved (RO, 0).** Bits must be zero.

Bit(s)	Name/Description
6	Force octaword transfers (R/W, 0) When set, forces the XBIA to generate octaword DMA transactions, regardless of the length code loaded in the DMA buffer. This bit is used with ADG1 <5:4> (force DMA-A/B busy), BDCR1 <6> (flip FADDER bit 1), and BDCR1 <4> (flip bit 29) to allow diagnostics to test the XBI DMA buffer memory using loopback transactions to XMI memory. NOTE **When BDCR1 <4> is set to use the diagnostic DMA loopback feature, only legal addresses (2xxx xxx0 or 2xxx xxx4) are allowed; illegal addresses (2xxx xxx8 and 2xxx xxxC) result in undefined data.**
5	Force DMA-A buffer busy (RW, 0) When set, places the DMA-A buffer into the busy state, forcing all DMA traffic through the DMA-B buffer. NOTE **If bits <5> and <4> are both set, all DMA transactions (VAXBI transactions that select the DWMBA as the slave and whose address falls within the bounds of the starting and ending address registers) will stall.**
4	Force DMA-B buffer busy (R/W, 0) Same as bit <5> except that all DMA traffic is forced through the DMA-A buffer.
3	Generate bad IBus receiver parity (R/W, 0) When set, forces the IBus parity check bit in the XBIA to a one, regardless of the data being loaded. Diagnostic routines use this bit with specific data patterns to force IBus parity check errors in the XBIA when the XBIB loads the C/A or data buffers in the XBIA gate array.
2	Generate bad IBus transmit parity (R/W, 0) When set, forces the IBus parity bit sent to the XBIB to a one, regardless of the data being transmitted. Diagnostic routines use this bit with specific data patterns to force IBus parity errors in the XBIB when the XBIB fetches the C/A or data buffers from the XBIA gate array.
1:0	Reserved (RO, 0). Bits must be zero.

4.6 DWMBA/B RESIDENT NODE SPECIFIC REGISTERS

The DWMBA specific registers resident on the DWMBA/B module are primarily associated with VAXBI bus transactions and events. These registers are indicated by the prefix "B" in the register name.

4.6.1 Control and Status Register (BCSR, bb+0040)

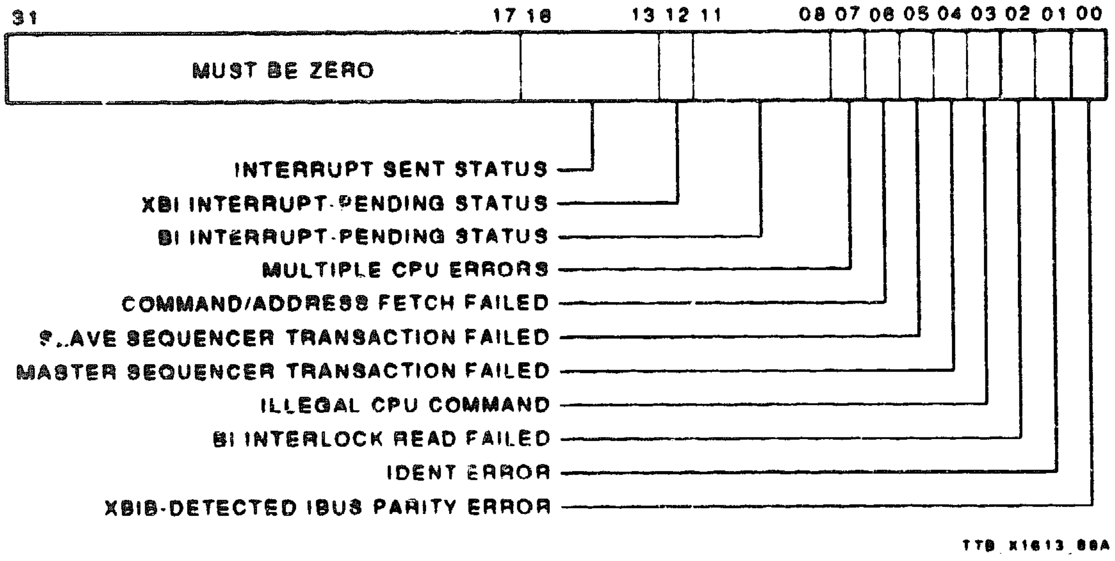

31 30
06 05 04 03 02 01 00
MUST BE ZERO
0
0
ENABLE XBI INTERRUPTS (TO XMI PROCESSOR(S))
LED
BI BAD
BI INTLCK READ FAILED MASK
IBUS P E INTR MASK

TTB_X1612_88A

Bit(s)	Name/Description
31	Enable XBI interrupts (R/W, 0) When set, enables the DWMBA to generate XMI interrupts in response to DWMBA generated or VAXBI generated interrupts. The appropriate interrupt mask bits must also be set for interrupts to be generated.
30:6	Reserved (RO, 0). Bits must be zero.
5	LED (R/W, 0) When set, illuminates LED D1. Cleared on power-up until node passes self-test.

Bit(s)	Name/Description
4	BI BAD (RO) On power-up or reset, reflects the state of BI BAD L on the VAXBI. Used by console initialization software and error handling routines to detect faulty VAXBI nodes. The assertion of BI BAD L on the VAXBI results in the assertion of XMI BAD.
3	Reserved (R/W, 0). Bit must be zero.
2	BI interlock read failed mask (R/W, 0) When set, causes the DWMBA to generate an error interrupt request if BESR <2> (BI interlock read failed) is set.
1	Reserved (RO, 0). Bit must be zero.
0	IBus parity error interrupt mask (R/W, 0) When set, causes the DWMBA to generate an error interrupt request if BESR <0> (XBIB-detected IBus parity error) is set.

4.6.2 Error Summary Register (BESR bb+0044)


31
17 16
13 12 11
08 07 06 05 04 03 02 01 00
MUST BE ZERO
INTERRUPT SENT STATUS
XBI INTERRUPT-PENDING STATUS
BI INTERRUPT-PENDING STATUS
MULTIPLE CPU ERRORS
COMMAND/ADDRESS FETCH FAILED
SLAVE SEQUENCER TRANSACTION FAILED
MASTER SEQUENCER TRANSACTION FAILED
ILLEGAL CPU COMMAND
BI INTERLOCK READ FAILED
IDENT ERROR
XBIB-DETECTED IBUS PARITY ERROR
TTB_X1613_88A

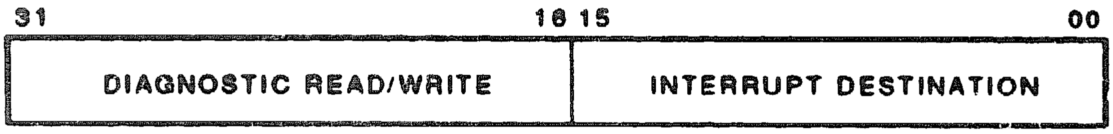
Bit(s)	Name/Description
31:17	Reserved (RO, 0). Bits must be zero.
16:13	Interrupt-sent status (RO, 0) These bits correspond to IPL <17:14>. BESR <16> corresponds to IPL <17>, BESR <15> to ILP <16>, an so on. Bits <16:13> and <12:8> determine the current interrupt-pending status.
12	XBI interrupt-pending status (RO, 0) When set, indicates that a DWMBA interrupt is pending.
11:8	BI interrupt-pending status (RO, 0) These bits indicate that one or more VAXBI generated interrupts targeting the DWMBA were received, but that a CPU IDENT at the correct IPL has not yet been received. BESR <11> corresponds to IPL <17> and BESR <8> to IPL <14>.

Bit(s)	Name/Description
7	Multiple CPU errors (R/W1C, 0) Set if BESR <4> and <0> were set due to an IBus parity error on a CPU transaction while the C/A or data is removed from the CPU buffer. Indicates that an error occurred on a subsequent CPU transaction before software acknowledged a previously failed transaction. Bit <7> is not set if a parity error occurs on write data accompanying the command/address on which an error was detected since the transaction has already been recorded as failed.
6	Command/address fetch failed (RO, 0) Set with BESR <0> to indicate that the XBIB detected an IBus parity error on the C/A fetch from the CPU C/A buffer. Bit is not set on a XBIB detected IBus parity error when write data is fetched from the CPU write data buffer.
5	Slave sequencer transaction failed (RO, 0) Set with BESR <0> to indicate that an IBus parity error occurred while the slave sequencer was in control of the IBus during a read data fetch from the DMA read buffer.
4	Master sequencer transaction failed (RO, 0) Set with BESR <0> to indicate that an IBus parity error occurred while the master sequencer was in control of the IBus during a C/A or write data fetch from the CPU buffer. The bit is not valid unless bit <0> is also set.
3	Illegal CPU command (RO) Set to indicate that an illegal CPU command was decoded by the XBIB. This error occurs only if an undetected multi-bit parity error condition exists during the time the XBIB fetches the commmand/address from the CPU buffer. The master sequencer will terminate the transaction and signal the XBIA that the transaction failed. The setting of this bit does not generate an error interrupt.

Bit(s)	Name/Description
2	BI interlock read failed (R/W1C, 0) When set, indicates that a VAXBI to XMI memory interlock read operation failed to complete on the VAXBI. When this occurs, the lock set in XMI memory will most likely not be unlocked by the VAXBI device that issued the interlock. The contents of BTIM and the setting of bit <2> can be used to determine the locked address in XMI memory. The operating system can clear the XMI memory lock by writing to a specific CSR in XMI memory. Bit <2> is set whenever a VAXBI interlock read command was decoded and the summary EV code of illegal CNF received for slave data (ICRSD) is decoded during a VAXBI interlock read transaction. The setting of bit <2> locks the timeout address register. Writing a one to the bit clears the bit and its lock on the register. When this bit and the corresponding mask bit are set, an error interrupt request is generated.
1	IDENT error (R/W1C, 0) When set, indicates that the DWMBA received an XMI IDENT transaction and no VAXBI nor DWMBA interrupt requests were pending at the IDENT IPL. This may indicate that an error condition exists on the XMI bus with multiple IDENTs being issued for the same interrupt transaction. Only one XMI IDENT is issued on the XMI if a single interrupt targets multiple CPUs. All other CPUs cancel their IDENT transactions if they detect an IDENT transaction that matches the node ID and IPL of the IDENT they are waiting to issue. IDENT error is set if a CPU IDENT command is decoded and no interrupts are pending in the XBIB gate array. The setting of the bit does not generate an XBI error interrupt.

Bit(s)	Name/Description
0	XBIB-detected IBus parity error (R/W1C, 0) Set if the XBIB detects an IBus parity error during one of the following: • C/A cycle of a CPU transaction • Write data cycle when the data is removed from the CPU buffer by the master sequencer • DMA read data cycle when the read data is removed from the DMA read buffer by the slave sequencer Bits <6:4> identify the error condition. If XBIB-detected IBus parity error is set with its corresponding mask bit, an error interrupt request is generated. If the bit is set due to an error during a DMA read data cycle, the BTIM register is locked. Writing a one to the bit clears bits <6:4> and the lock on the BTIM register.

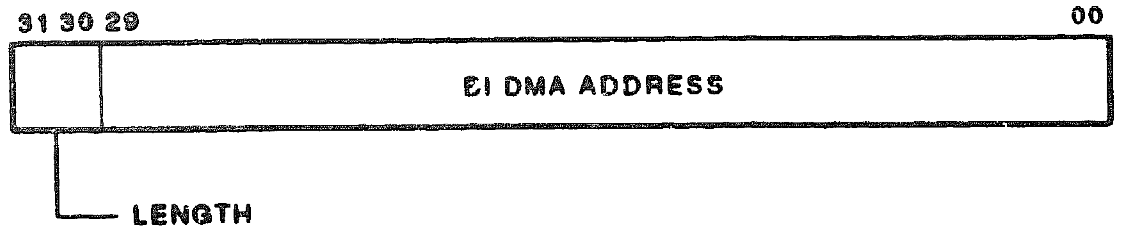
4.6.3 Interrupt Destination Register (BIDR, bb+0048)



TTB_X1614_89A

Bit(s)	Name/Description
31:00	**Diagnostic read/write (R/W)** Used by diagnostics to verify the integrity of the XBIB gate array data path.
15:00	**Interrupt destination (R/W, 0)** These bits specify the XMI nodes to be the targets of DWMBA generated interrupts. Each bit corresponds to one XMI node. Multiple bits can be set to interrupt multiple XMI nodes. During diagnostic execution, bits <15:00> are treated as diagnostic read/write bits.

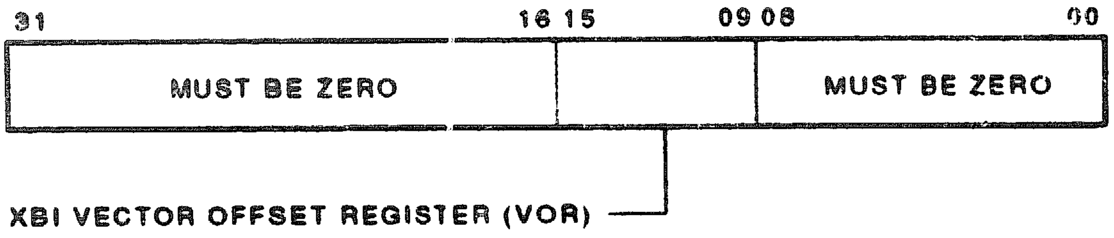
4.6.4 Timeout Address Register (BTIM, bb+004C)



TTB_X1615_89A

Bit(s)	Name/Description
31:30	Length (RO) Data size of the last VAXBI-to-XMI transaction. Loaded when the VAXBI C/A is latched from the VAXBI.
29:00	BI DMA failing address (RO) Physical address of the last VAXBI-to-XMI transaction. If no errors are detected, the BTIM register reads back the last VAXBI transaction. The register is locked on certain error conditions (see BESR 5:4,0 bit descriptions) and unlocked when the error condition is cleared.

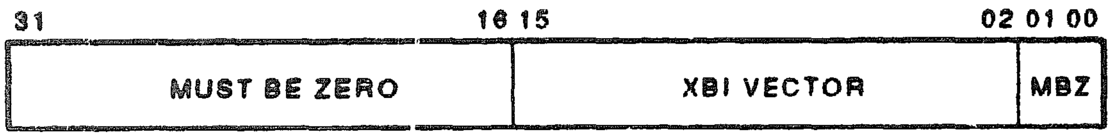
4.6.5 Vector Offset Register (BVOR, bb+0050)



TTB_X1616_89A

Bit(s)	Name/Description
31:16	Reserved, must be zero.
15:09	XBI vector offset register (R/W, C)
	Loaded by software on system initialization with a value that is concatenated with VAXBI device-supplied vectors. Ensures that multiple DWMBAs, and VAXBI buses with the same devices, have unique entry points into the SCB (provided that bits <13:09> of the VAXBI vector are equal to zero).
08:00	Reserved, must be zero.

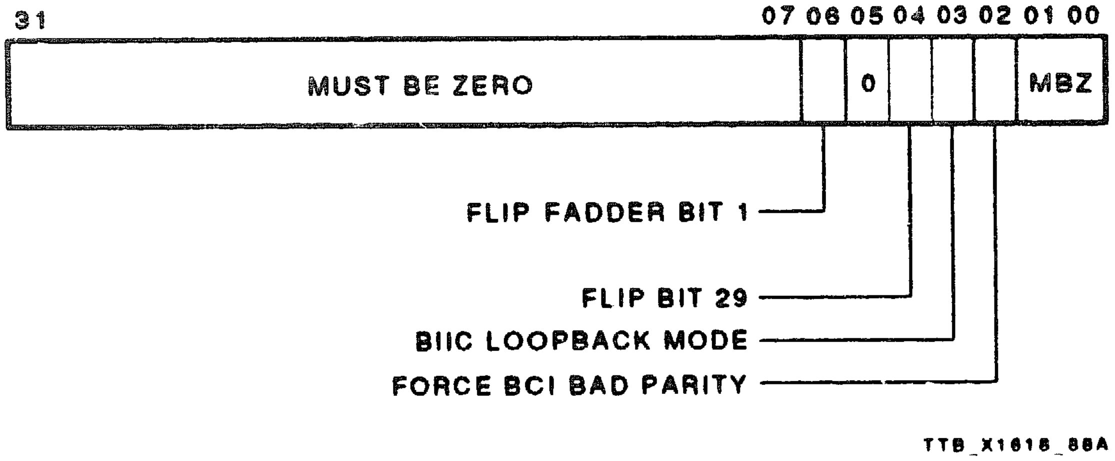
4.6.6 Vector Register (BVR, bb+0054)



TTB_X1617_89A

Bit(s)	Name/Description
31:16	**Reserved, must be zero.**
15:2	**XBI vector (R/W, 0)**
	Loaded by software on system initialization with the DWMBA XMI vector. The vector is transmitted to the IDENTing XMI node if the pending DWMBA interrupt request matches the interrupt source and IPL sent during the XMI IDENT transaction. The vector is not sent on VAXBI generated interrupts or BIIC interrupts due to error conditions.
1:0	**Reserved, must be zero.**

4.6.7 Diagnostic Control Register 1 (BDCR1, bb+0058)



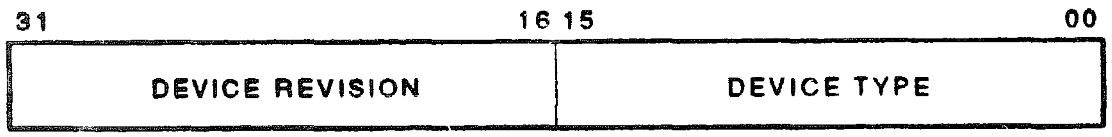
Bit(s)	Name/Description
31:7	**Reserved, must be zero.**
6	**Flip FADDR address bit 1 (R/W, 0)** Used with bit <04> (flip bit 29) and ADG1 <5:4> (force DMA-A/B busy bits) to enable diagnostics to test DMA buffer memory using CPU loopback transactions to XMI memory. When the bit is set, the inverted state of FADDR address bit 1 is used to address the data words in the buffer, allowing diagnostics to use the buffer locations that normally would only be used for transfers greater than a quadword. Setting this bit only affects FADDR address bit 1 when the XBIB logic accesses data locations in the DMA buffer. During the cycle when the C/A is addressed in the buffer, the state of the bit has no effect on the buffer address.
5	**Reserved, must be zero.**

Bit(s)	Name/Description
4	Flip bit 29 (R/W, 0) When set, inverts the state of address bit 29 after the CPU C/A was fetched and decoded by the master sequencer. The new address (now pointing to XMI memory space) is issued to the VAXBI, and the DWMBA is selected as the VAXBI slave. The DWMBA processes the transaction as it would any other VAXBI initiated DMA longword transaction, allowing diagnostic programs executing on the XMI to issue a CPU transaction to the DWMBA, which converts it into a DMA transaction.
3	BIIC loopback mode (R/W, 0) When set, forces all requests to the BIIC master port to be loopback requests. This allows the master sequencer to make loopback requests to access BIIC registers. The loopback mode prevents the BIIC from initiating VAXBI cycles to access the BIIC registers. When the BIIC is in loopback mode, it ignores the node ID portion of the address presented to it.
2	Force BCI bad parity (R/W, 0) When set, forces bad parity onto the BCI bus to the VAXBI during CPU C/A, CPU data cycles, and DMA read data cycles.
1:0	Reserved, must be zero.

4.7 VAXBI REGISTERS

The DTYPE register is the only VAXBI register described in this handbook. Refer to the *VAXBI System Reference Manual* for descriptions of all VAXBI registers.

The DTYPE register is loaded during self-test by console code with the DWMBA VAXBI device type, and by the revision select logic with the revision level. The DTYPE register is located at the base address (offset: 0000) of the DWMBA's I/O adapter address space.



TTB X1620 88A

Bit(s)	Name/Description
31:16	Device revision (R/W, 0) Loaded by hardware with the revision level of the device. For revision H, the DREV field contains 7 (hex). There is no revision I. Starting with revision J, the DREV field reflects the letter revision of the module as follows:

DREV	DWMBA/B Revision
000A	J0
000A	J1
000B	K1, K2, . . . Kn
	.
	.
	.
001A	Z0, Z1, . . . Zn

Bit(s)	Name/Description
15:0	Device type (R/W, 0) Identifies the VAXBI node type. Loaded by the console code after successful completion of self-test. The DTYPE for the DWMBA is 2107 (hex).

4.8 IBUS SIGNALS

Bidirectional

IB D <31:00>
IB I <3:0>
IB P0

XBIA to XBIB

IR DMAA BUF AVAIL L
IR DMAB BUF AVAIL L
IR CPU BUF LOADED L
IR XMI ERR BIT SET L
IR READ DATA AVAIL L
IR READ DATA FAULT L
IR LOC RESPONSE L
IR ADAPTER RESET L
IR XMI AC LO H
IR XMI DC LO H
IR XMI RESET L

XBIB to XBIA

IM FADDR <3:0>
IM FILE LOAD STROBE L
IM FILE READ ENABLE L
IM DMA READ CMD L
IM CPU XACTION DONE L
IM CPU LOC RESPONSE L
IM DMAA BUF LOADED L
IM DMAB BUF LOADED L
IM CLR READ STATUS L
IM XACTION FAULT L
IM CLR INTR L
IM XBIB POWER OK H <3:0>
IM BUF BI RESET L
IM BI AC LO L
IM BI BAD L