DEC 7000/10000

Diagnostics and Module Level Repair Student Workbook

Order Number: EY-Q755E-SG-0001

Digital Equipment Corporation

CONTENTS

3 Advanced Console

4 LASER System Bus

5 LSB Memory Operation and Error Registers

6 VAX 7000-600 and VAX 10000-600 Processor Operation and Error Registers

8 VAX 7000-600 and VAX 10000-600 Systems Fault Management

9 DEC 7000-600 and DEC 10000-600 Systems Differences

10 DEC 7000-600 and DEC 10000-600 Systems Fault Management

11 Test

A XMI I/O Modules and Error Registers

B VAX 7000-600 Errorlog Examples

C DEC 7000-600 Errorlog Examples

EXAMPLES

FIGURES

TABLES

About This Course

Introduction

This 5-day lecture-lab course is designed to help support level Digital Service Engineers gain the information and skills needed to provide maintenance support for 7000 and 10000 systems.

Support level DSEs provide maintenance service support on systems which have not been repaired by first level maintenance services. They may provide this service support over the telephone or in person.

The typical student is presently providing support level services for VAX 6000 and/or VAX 9000 systems.

Course Description

7000/10000 systems are presented to the support level Digital Service Engineer (DSE) who has completed the 7000/10000 System Level 1 Maintenance Course (CBI). Labs will supplement and reinforce concepts covered in the classroom.

A quick overview of the system is contained in the first module. The power subsystem is covered in the second module. That is followed by advanced topics of system console operation.

The LASER system bus (LSB) protocol is described. The functional operation of bus modules is presented with emphasis on error detection and error registers.

System error reporting via errorlogs is covered using VAX 7000 and DEC 7000 examples.

There are exercises at the end of each module to summarize and review the main topics of the module.

Target Audience

The target audience is DSEs with support level VAX systems training and experience who have completed the 7000/10000 Systems Level 1 Course.

They will be providing support level maintenance service on systems which have not been repaired by FRU replacement. They may provide support service over the telephone or in person.

Usually they are presently working at the support level on VAX 6000 and/or VAX 9000 systems.

Resources

- DEC 7000 AXP System and VAX 7000 Console Reference Manual
- DEC 7000 AXP System and VAX 7000 System Installation Guide
- VAX 7000 Pocket Service Guide
- DEC 7000 AXP and VAX 7000 System Service Manual

- DEC 7000 AXP System VAX 7000 Operations Manual
- VAX 7000 Advanced Troubleshooting
- DEC 7000 AXP System Pocket Service Guide
- VAX 7000/10000 KA7AA CPU Technical Manual
- MS7AA Memory Technical Manual

Goals

The Digital Service Engineer providing support level maintenance service for the 7000/10000 system **should be able to**:

- Use the manuals, guides, specifications which are job relevant.
- Use the onboard diagnostic and exerciser programs as an aid to fault detection and isolation.
- Analyze system failures using knowledge of system operation and the contents of error registers.
- Use the system errorlog as an aid to diagnose and identify system FRUs.

Nongoals

This course will not cover these topics or tasks:

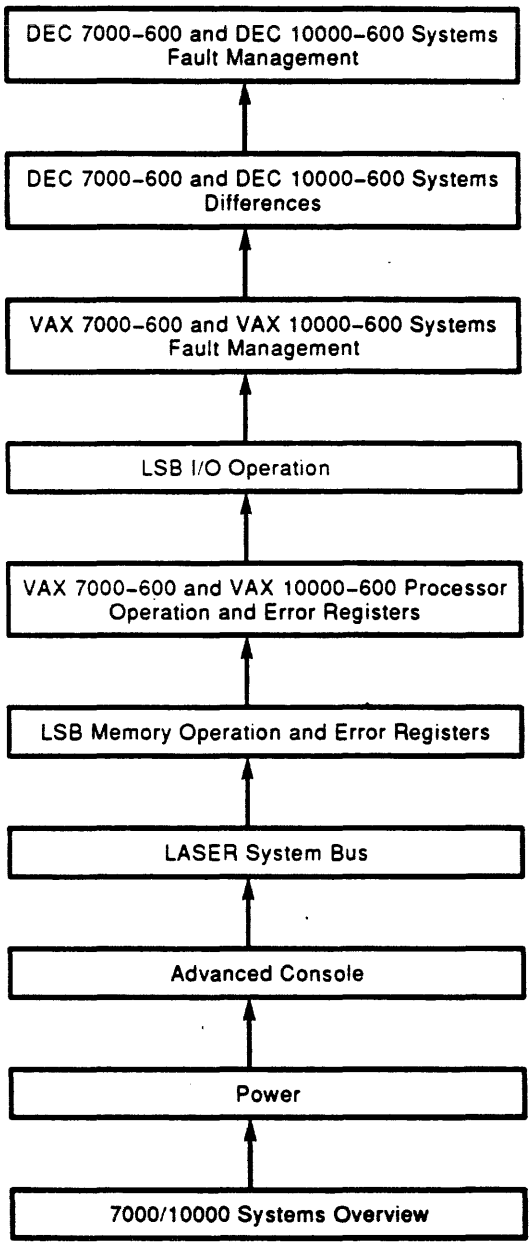
- Operating system error handling code (as found in SYSLOA).
- Print set interpretation.
- Microcode interpretation.
- XMI protocol or concepts
- VAXBI protocol or concepts

Course Organization

The course consists of ten learning modules. Lab exercises are associated with most modules. There are also troubleshooting exercises that are to be used when convenient during the course. The modules are arranged as illustrated in Figure 1 and listed on the following page.

The course map, starting at the bottom, illustrates the sequence in which the learning modules are encountered.

Figure 1 Course Map



Description of Course Modules

The course modules are describe here.

1. 7000/10000 Systems Overview—A very short overview of the 7000/10000 systems.
   a. 7000-600 system functional arrangement
   b. 7000-600 system physical characteristics
   c. system configuration rules
   d. power characteristics of the 7000/10000 systems
2. Power—The power characteristics of the 7000/10000 systems is described and illustrated in this module. The sequence of events at poweron are covered.
3. Advanced Console—Console and diagnostic features not covered in the Level 1 course will be covered here.
   a. Memory interleaving scheme
   b. Advanced and diagnostic mode console commands
   c. Aystem initialization and boot sequence
      1. Sequence of events at system poweron
      2. Initialization and self-test
      3. Boot sequence including Hardware Restart Parameter Block (HWRPB)
   d. Processor EEPROMs
4. LSB Bus—The characteristics of the LSB Bus are described and the required and module optional registers and their significance is covered.
5. LSB Memory operation and error registers— Memory module operation and error reporting scheme are covered. Significant error registers bits are identified.
6. VAX 7000-600 System Processor (LNP) operation and error registers— Processor module operation and error reporting scheme are covered. Significant error registers bits are identified. Conditional Update Writeback Cache protocol is covered here.

7. IO Subystem

   a. IOP Module Operation and Error Registers
      IOP module operation and error reporting scheme are covered. Error registers bits are identified.

   b. LAMB Module Operation and Error Registers
      IOP module operation and error reporting scheme are covered. Error registers bits are identified.

8. VAX 7000-600 System Fault Management

   The methods and sequence used by the system to react and report system errors is covered in this learning module. Sample system errorlogs will be analyzed in the classroom. VAX 7000-600 system error handling flow sequence is described. A flowchart of operating system error handling code is presented.

9. DEC 7000-600 System Differences

   The ways in which the DEC 7000-600 system (Alpha AXP/RISC) differs from the VAX 7000-600 system (VAX/CISC) will be covered in this module. DEC 7000-600 system material considered significant to support level DSEs will be included here.

10. DEC 7000-600 System Fault Management

    The methods and sequence used by the system to react and report system errors is covered in this learning module. Sample system errorlogs will be analyzed in the classroom. DEC 7000-600 system error handling flow sequence is described. A flowchart of operating system error handling code is presented.

Description of Course Appendices

These appendices are included in the student guide for student reference. They are not part of the course curriculum.

A. XMI I/O Modules and Error Registers— The XMI I/O modules and are described in this module. Significant error register bits are identified. Also the addresses scheme for XMI registers and VAXBI registers is documented.

B. VAX 7000-600 Errorlog Examples—selected examples of processor, memory, and I/O errorlogs.

C. DEC 7000-600 Errorlog Examples—selected examples of processor, memory, and I/O errorlogs.

7000 and 10000 Systems Overview

Introduction

This module provides the student with a quick look of the 7000/10000 systems from a functional and a physical point of view.

Objectives

A Digital Services Engineer (DSE) who maintains a 7000/10000 system should be able to:

- Discuss the general features of the system with customers and other DSEs.
- Identify the major components of the system.
- Identify the devices that can be connected to the system.
- Identify the maximum amount of memory and the maximum amount of I/O adapters that can be on the system.

Resources

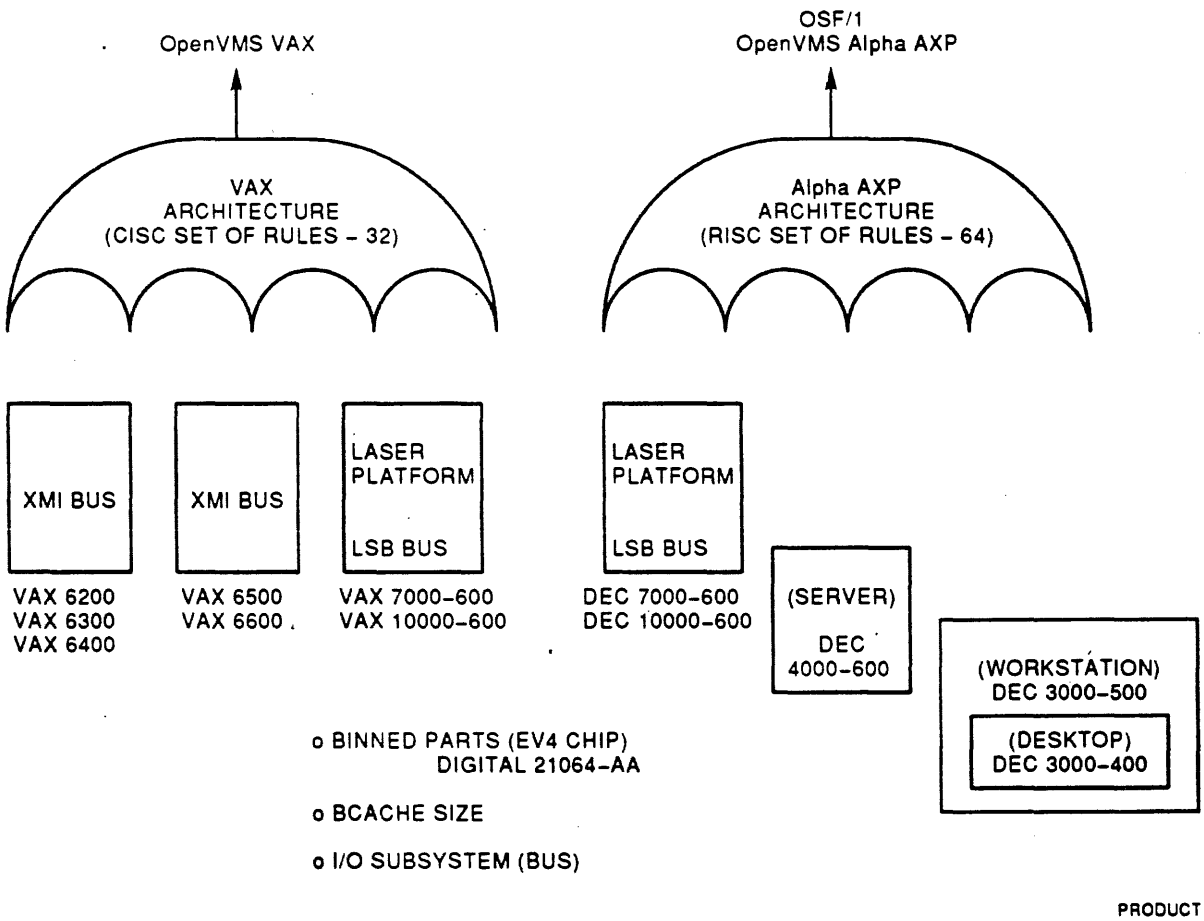
VAX 7000 Pocket Service Guide	EK-7000A-PG
DEC 7000 AXP and VAX 7000 System Service Manual	EK-7000A-SV

Product Position and System Performance

Product Position

The continued development of processors with higher performance capability has strained the ability of the XMI bus to handle the traffic. A system bus with a higher bandwidth was needed. The LASER system bus (LSB), which has a bandwidth of 640 MB, was developed for more powerful processors. The 7000/10000 platform was then developed to support systems using the LSB.

Figure 1–1 7000 and 10000 Systems Product Postion



VAX 7000-600 and VAX 10000-600 Systems Performance

The VAX 7000-600 and VAX 10000-600 systems use the LASER system bus and the NVAX+ high performance microprocessor. These systems provide a consistent upgrade path for customers as higher performance processors, and I/O devices are expected to be integrated in the future.

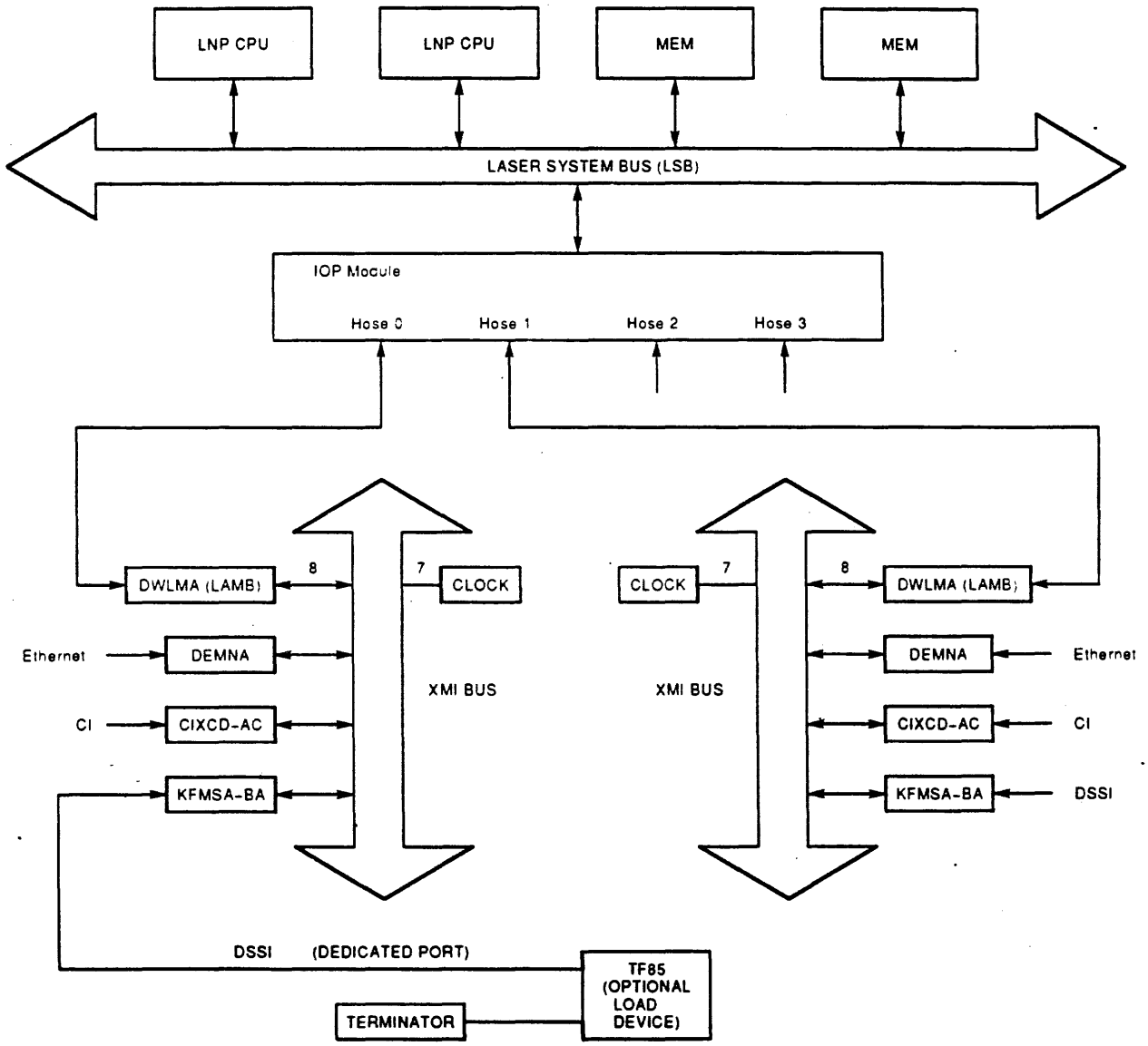
A system with a single-processor module has a performance measurement of 50 SPECmarks while a 4-processor system will perform up to 200 SPECmarks. The NVAX+ microprocessor brings proven VAX architecture and proven CMOS 4 technology to VAX 7000-600 and VAX 10000-600 systems.

Performance has increased from the VAX 6000 Model 600 processor which has a performance measurement of 40 SPECmarks. The increase in performance is partially due to the change of microprocessor (NVAX to NVAX+) and partially due to a larger backup cache (4 MB versus 512 kB).

Functional Description

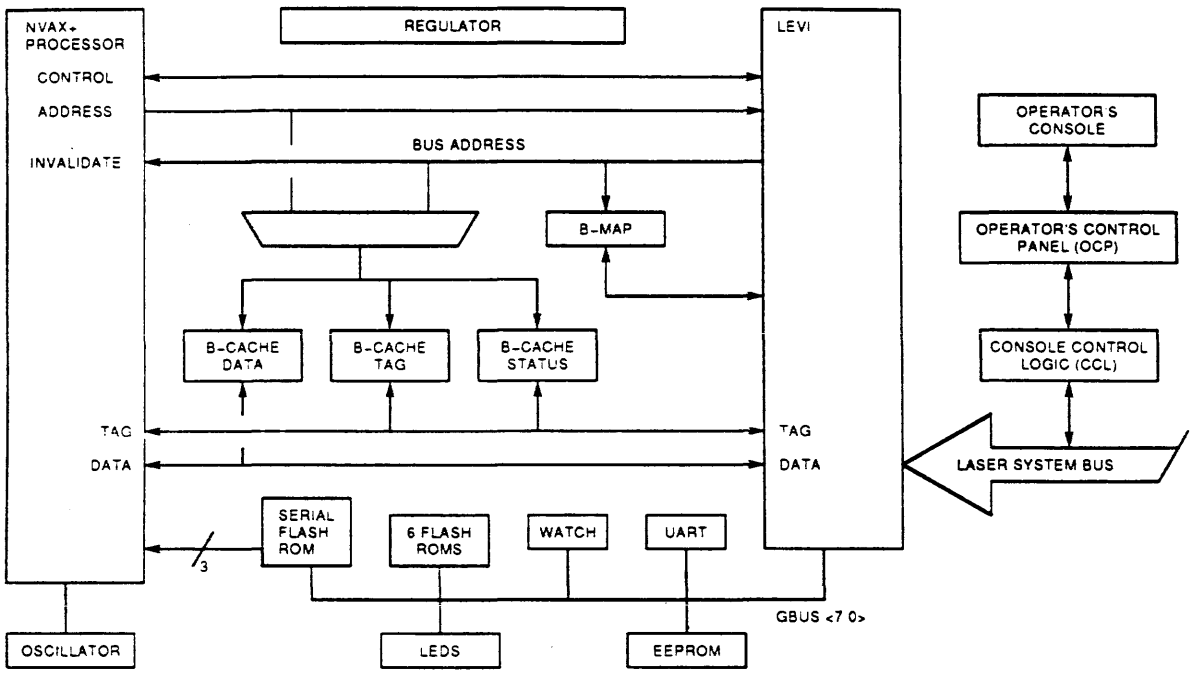
A functional diagram of a typical VAX 7000-600 system is shown in Figure 1–2.

Figure 1–2 Typical VAX 7000-600 System Functional Diagram



LNP Processor Module, KA7AA-AA, E2045-AA

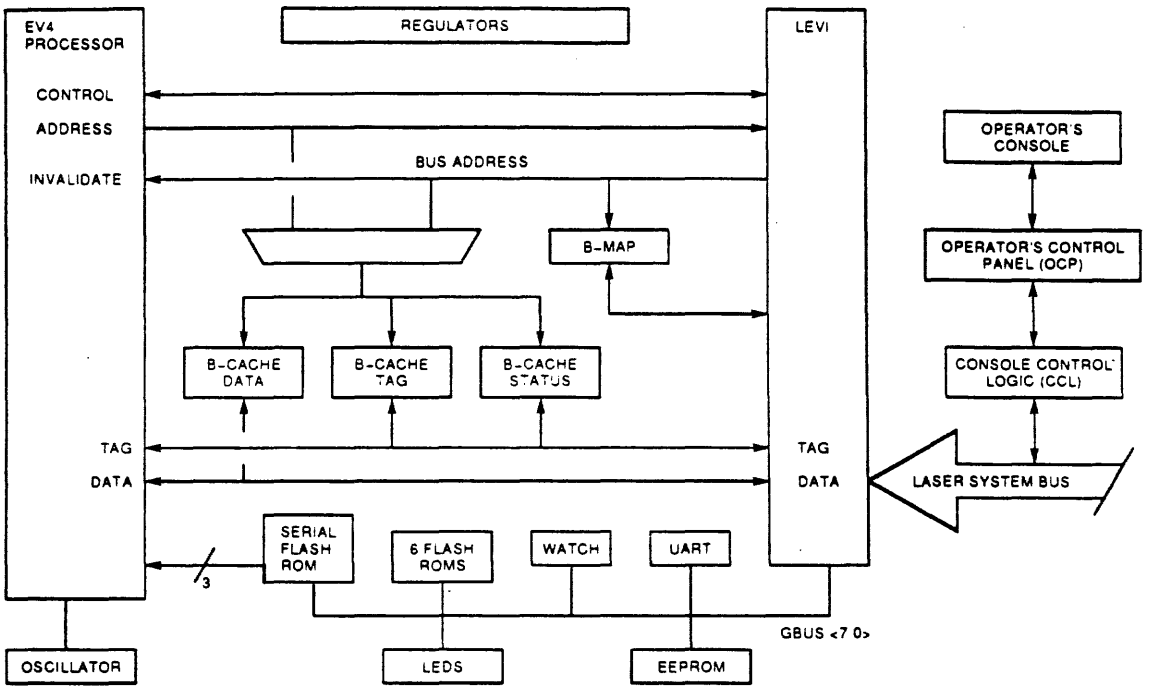
Figure 1–3 LNP Processor Module



- CMOS-4 macropipelined NVAX+ microprocessor
  - — Implements the basic 242-instruction set
  - — 8 kB primary cache, instruction and data
  - — 2 kB virtual instruction cache (VIC)
  - — 96 entry translation buffer
  - — Floating point
  - — 11 ns cycle time
- 4 MB backup cache
- LASER/EVAX interface (LEVI-A/B)
- Console support hardware
- DC-to-DC power regulators (48 Vdc to 5.0 Vdc, 3.3 Vdc, 2.0 Vdc, etc.)

LEP CPU Modules (KN7AA-AA, E2040-AA; KN7AA-YA, E2040-YA)

Figure 1–4 LEP Processor Module



- CMOS-4 macropipelined chip (about 150 Alpha AXP instructions)
  - — 8 kB data cache, 8 kB virtual instruction cache (VIC)
  - — Translation buffer: 32 entries for data, 8 entries for instructions
  - — Supports Digital F- and G-floating point data types
    Supports IEEE single- and double-floating data types
  - — Cycle time: 5.5 ns for E2040-AA, 5.0 ns for E2040-YA
- 4 MB backup cache
- LASER/EVAX interface (LEVI-A/B)
- Privileged Architecture Library code (PALcode)
- Console support hardware
- DC-to-DC power regulators (48 Vdc to 5.0 Vdc, 3.3 Vdc, 2.0 Vdc, etc.)

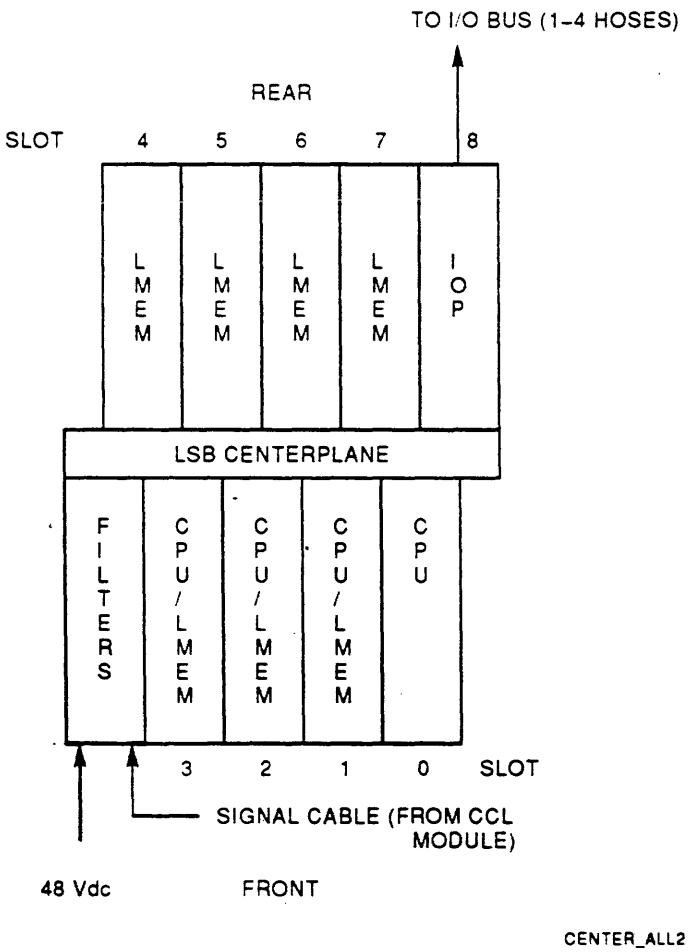
LSB Centerplane Slot Allocation

Systems with pass 2 LEVIs on the CPU modules have memory configuration restrictions.

LSB Centerplane Slot Allocation (Pass 2 LEVIs, DC7300C)

- 1 slot dedicated to CPU module
- 3 slots allotted to CPU or MEM modules
- 4 slots dedicated to MEM modules
- 1 slot dedicated to I/O module

Figure 1–5 LSB Centerplane Slot Allocation (Pass 2 LEVIs)

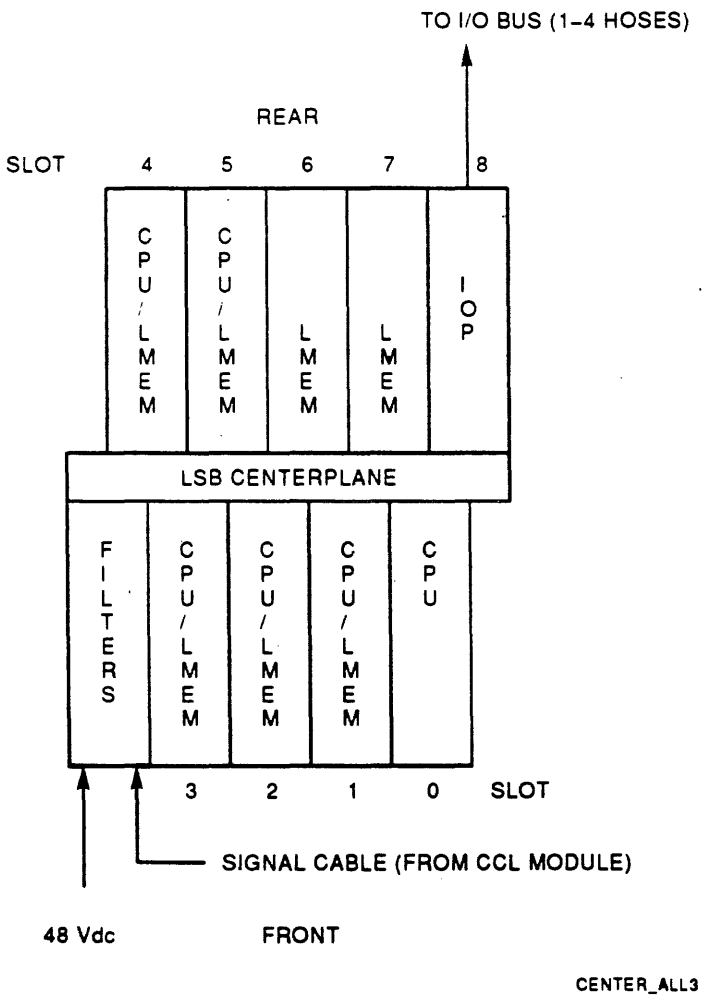


Systems with pass 3 LEVIs on the CPU modules have no memory configuration restrictions.

LSB Centerplane Slot Allocation (Pass 3 LEVIs, DC7300D)

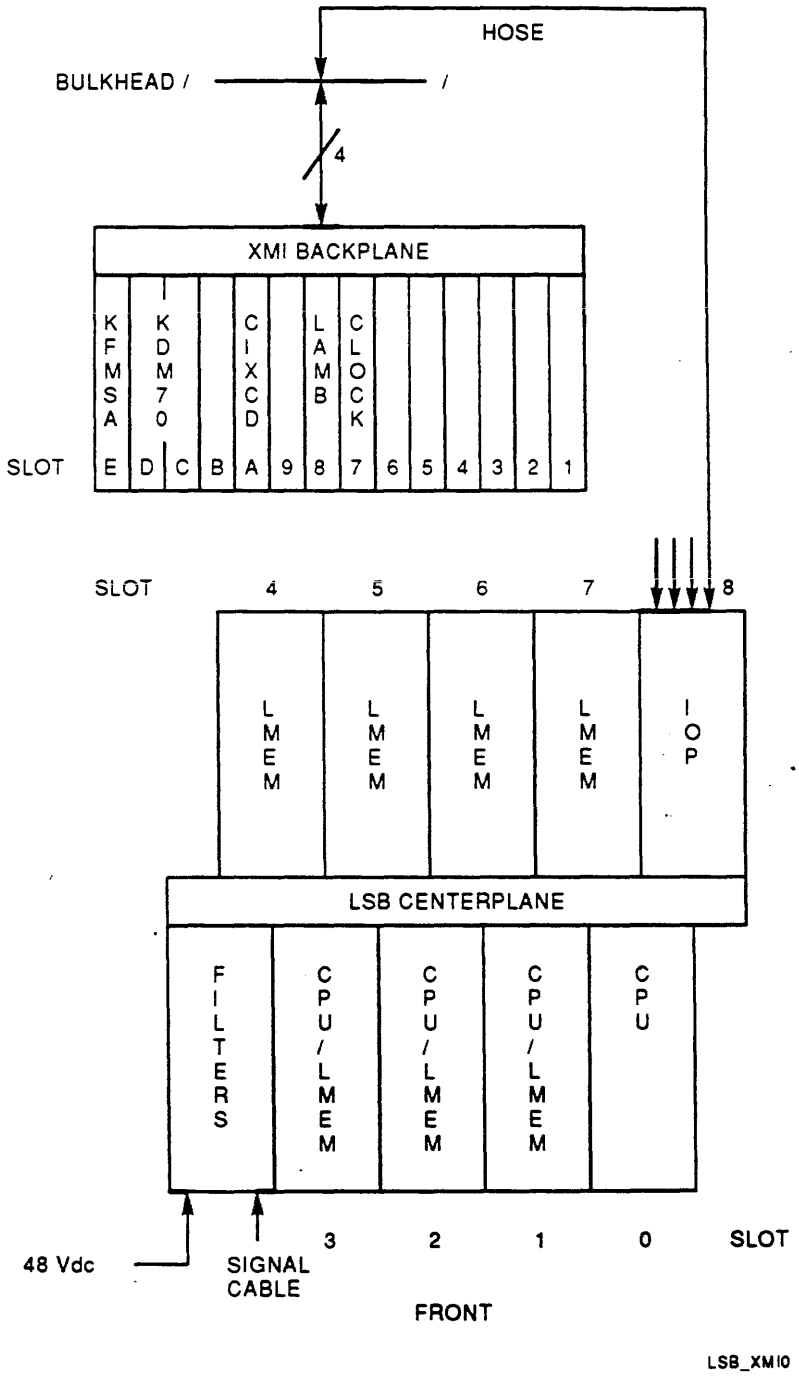
- Slot 0 recommended for CPU module
- 5 slots allotted to CPU or MEM modules
- 2 slots dedicated to MEM modules
- 1 slot dedicated to I/O module

Figure 1–6 LSB Centerplane Slot Allocation (Pass 3 LEVI-A/Bs)



Using Pass 2 LEVIs, the LSB modules are connected as shown in Figure 1–7. A hose (2-cable unit) goes from the IOP module to the XMI PIU bulkhead. Then a single plug at the bulkhead is connected by four cables to segments D and E of LAMB slot (8) on the XMI backplane.

Figure 1–7 LSB Modules Configuration



I/O Port

The IOP module is connected to the LAMB module via a hose (2 cable unit) of up to 9 feet in length. The XMI bus is in a backplane and card cage which is similar to the unit used in the VAX 9000 system. Processor and memory modules are not allowed on the XMI bus.

IOP Module (E2044)

- IOP module sits in slot 8 of the LSB centerplane at rear of system
- Conforms to LSB protocol
- Main elements of IOP are:
  - — Two PGAs which interface with the LSB
    - * I/O Port Chip A (IPC-A) control, registers, data
    - * I/O Port Chip B (IPC-B) data
  - — A "vortex" bus connection between IPCs and HICs
  - — Two hose interface chips (HICs) which interface with 1 to 4 hoses
    - * UP HIC
    - * DOWN HIC
- IOP synchronizes transfers between hoses and the LSB
- Translates between hose protocol and the LSB protocol
- DC-to-DC power regulator

Hose

A full-duplex hose connects the IOP module to the PIU bulkhead.

DWLMA Module (T2028, LAMB)

- LAMB module sits in slot 8 of the XMI bus
- Conforms to XMI bus protocol
- Main elements of LAMB module are:
  - — One LGA which provides the "brains" for the module
  - — Up and down hose memories (store commands and data)
  - — Buffers, FIFOs
- LAMB synchronizes transfers between hoses and XMI bus
- Translates between XMI bus protocol and hose protocol

LSB Memory Module (MS7AA-xA, LMEM)

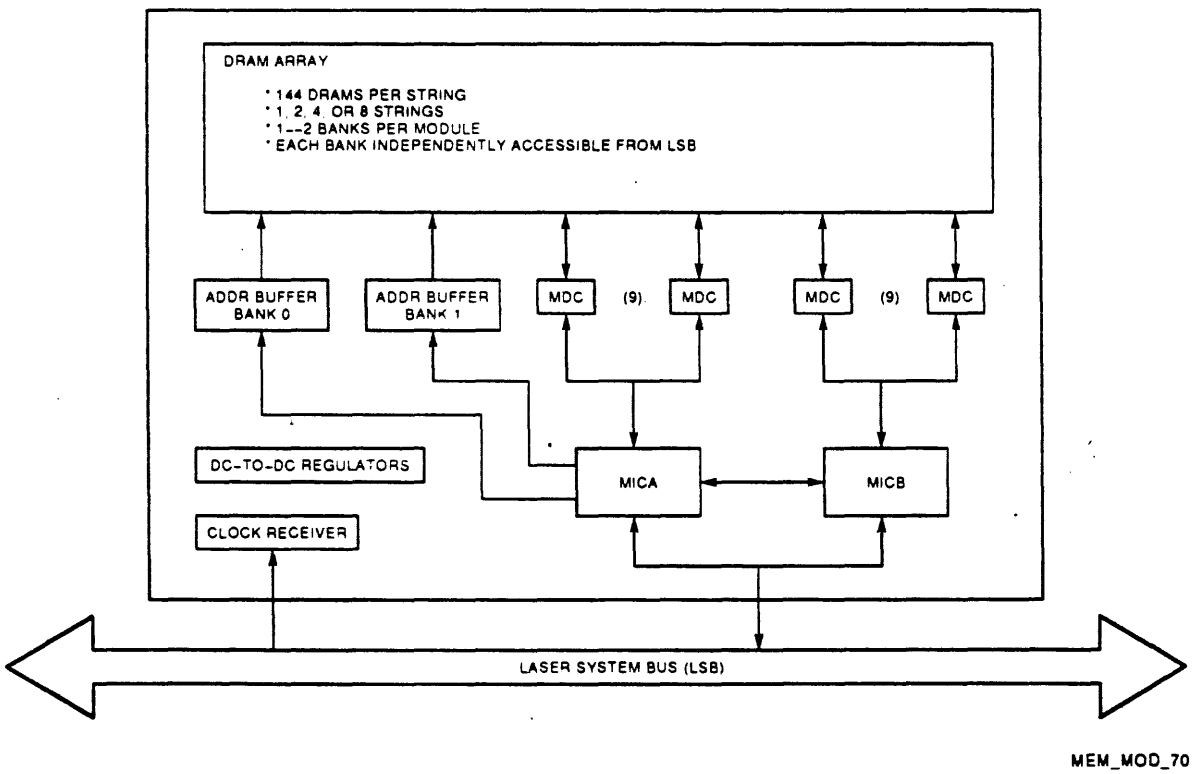
The LSB memory module, shown in Figure 1–8, is available as listed in Table 1–1.

Table 1–1 LASER Memory Modules

Option	Module	Memory Chip	Memory Size
MS7AA-AA	E2043-AA	4 Mb	64 MB
MS7AA-BA	E2043-BA	4 Mb	128 MB
MS7AA-CA	E2043-CA	4 Mb	256 MB
MS7AA-DA	E2046-AA	4 Mb (SIMMs)	512 MB

Data is stored in 64-byte blocks units at naturally aligned 64-byte addresses. Sixty-four ECC bits are stored with each 64-byte block.

Figure 1–8 LSB Memory Module Block Diagram



- Memory interface controllers A and B (MICA, MICB)
- DRAM data buffers (MDC) and DRAM arrays
- Address buffers for two banks

VAX 7000/10000-600 System XMI I/O Options

Each VAX 7000/10000-600 system and each XMI bus in the system supports the number of devices listed in Table 1–2.

Table 1–2 XMI Devices Supported on VAX 7000-600 Systems

Option	Maximum per XMI bus	Maximum per System	Description
CIXCD-AC[1]	6	10	XMI to CI adapter
KFMSA-BA[2]	5	12	XMI to DSSI adapter
DEMNA	4	16	XMI to NI adapter
DEMFA	4	7	XMI to FDDI adapter
KDM70[3]	3	12	XMI to SI adapter
DWMVE[4]	2	2	XMI to VME bus adapter
DWMBB[4]	1	1	XMI to VAXBI adapter

[1]CIXCD-AC, T2080-YA, new ucode for interlocks

[2]KFMSA-BA, new ucode

[3]KDM70, new ucode

[4]For third party and customer devices

NOTE

These configuration limits are subject to change. See the Systems and Options Catalog for the latest configuration information.

VAXBI I/O Options

The VAX 7000-600 and 10000-600 systems are expected to provide limited support to devices on the VAXBI bus.

- Only support the DMB32 and DRB32 devices
- No support for interlocks on the VAXBI

DEC 7000/10000-600 System XMI I/O Options

The DEC 7000-600 and 10000-600 systems running Alpha AXP/VMS will support the following devices on the XMI bus.

Table 1–3 Device Types for DEC 7000-600 System XMI Modules

Option	Maximum per XMI bus	Maximum per System	Description
CIXCD-AC[1]	6	10	XMI to CI adapter
KZMSA-AB	8	12	XMI to SCSI adapter
KFMSB-AA[2]	To be determined	To be determined	XMI to DSSI adapter
DEMNA	4	16	XMI to NI adapter
DEMFA	4	7	XMI to FDDI adapter
KDM70[3]	3	12	XMI to SI adapter

[1]CIXCD-AC, T2080-YA, new ucode for interlocks.

[2]This is the KZMSA module with new ucode, expected in the future.

[3]KDM70, new ucode. Number per XMI and number per system to be determined.

NOTE

These configuration limits are subject to change. See the Systems and Options Catalog for the latest configuration information.

Physical Description

The 7000/10000 cabinets are not part of the system's EMI shield. Each unit in the cabinet has its own EMI integrity.

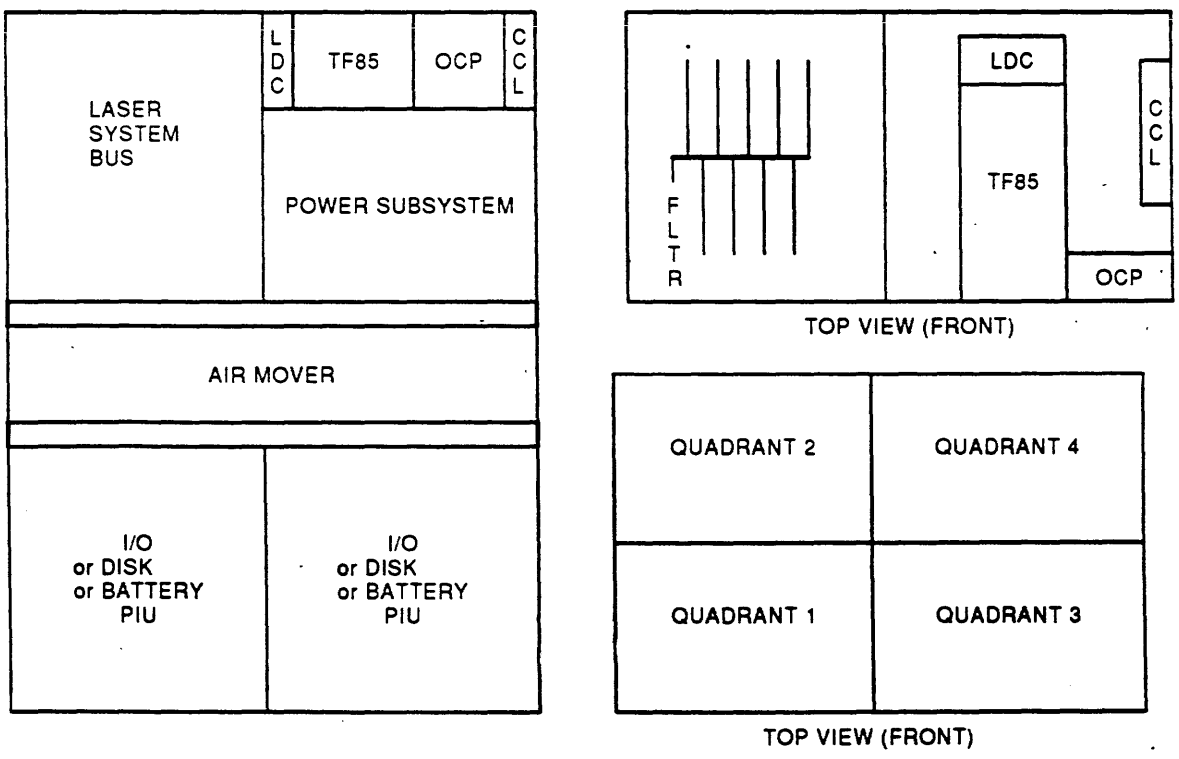
The 7000/10000 main and expander cabinets use metric hardware and are similar in outside size and shape to the VAX 6000 system main cabinet. They are slightly larger than the VAX 6000 cabinet as noted here.

Table 1–4 7000 and 10000 Main and Expander Cabinet Sizes

Dimension	Metric	English
Height	170.0 cm	67.0 in
Width	80.0 cm	31.5 in
Depth	87.5 cm	34.4 in

The main cabinet is illustrated in Figure 1–9, and a list of the major parts follows the illustration.

Figure 1–9 VAX 7000-600 Main Cabinet (Front)



- LASER system bus (LSB) in sheet metal enclosure with front and rear doors
  - — LSB centerplane
  - — Processors
  - — Memory
  - — I/O interface
  - — Unoccupied LSB slots will be sealed off with dummy modules (air flow)
  - — Electrical filters for 48 Vdc and CCL signal lines
- Power area
  - — AC input box
  - — One to three H7263 bulk 48 Vdc power supplies
  - — 48 Vdc distribution box
- Console control area
  - — Operator control panel (OCP)
  - — TF85D-AA (optional load device)
  - — Cabinet control logic module (CCL) (similar to VAX 6000 XTC module)
  - — Local disk converter (LDC) powers optional load device
- Air mover
  - — Air in at top and bottom, dual impeller
  - — Air out at waist level, front and back
  - — Runs from 48 Vdc
- Plug-in units (PIUs) XMI, VAXBI, disk, or UPS

Figure 1–10 VAX 7000-600 Main Cabinet (Rear)

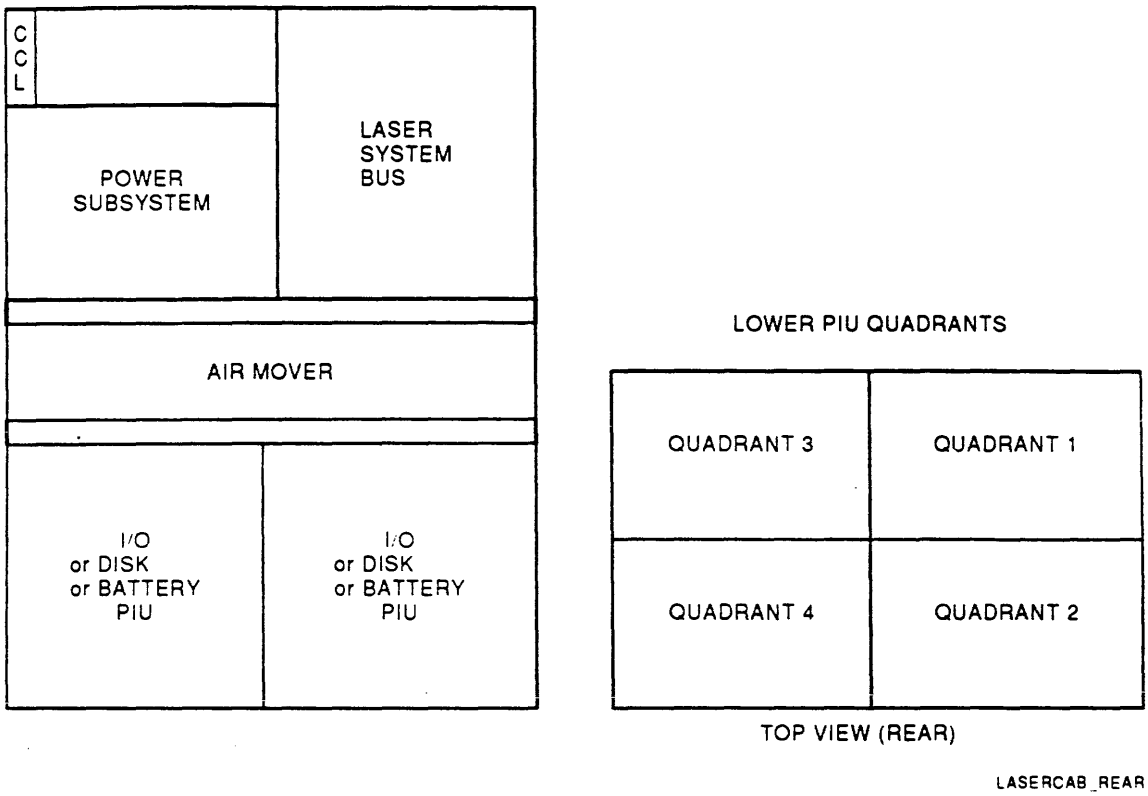
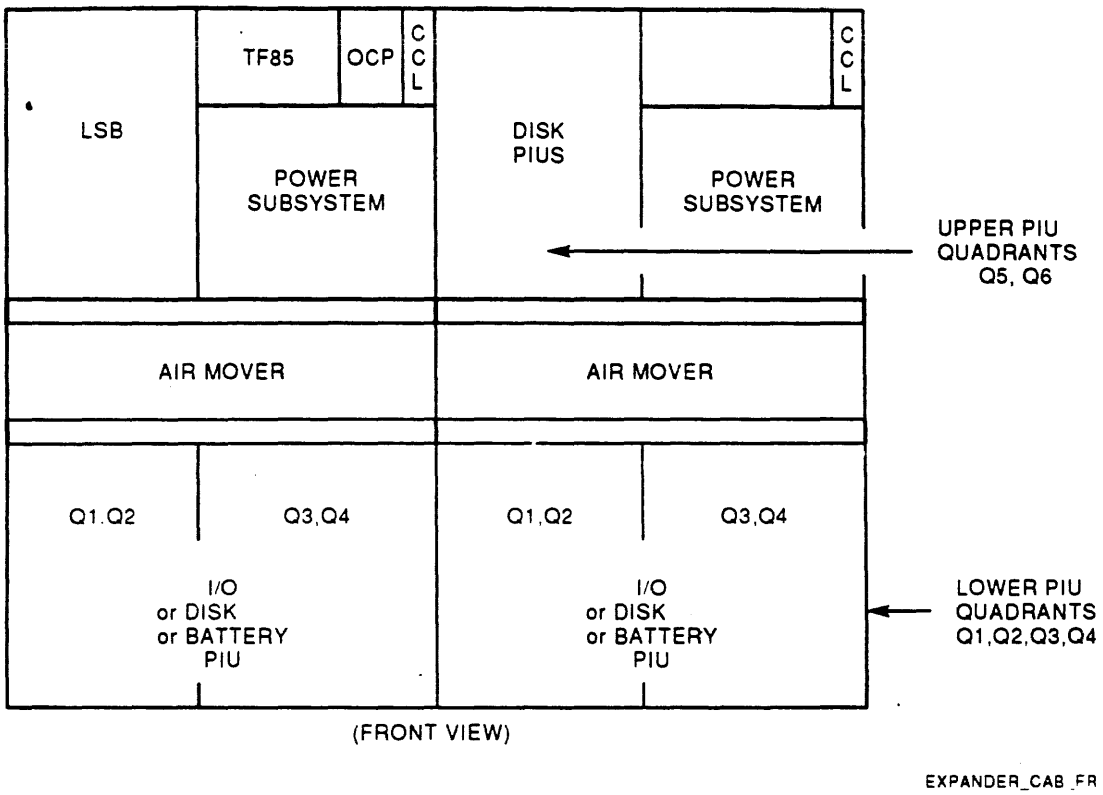


Figure 1–11 7000 and 10000 System Expander Cabinet (Front)



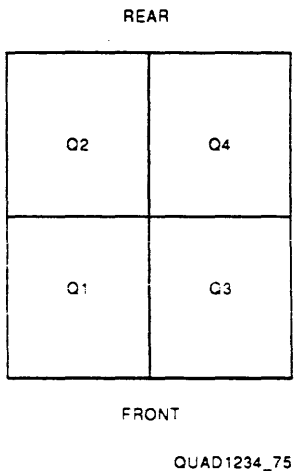
The main VAX 7000-600 cabinet can be connected to two VAX 7000-600 expander cabinets. The first expander cabinet is to the right, and a second to the left. The expander cabinet uses the same hardware as the main cabinet but differs in configuration. An expander cabinet contains:

- Air mover
  - — Air in at top and bottom, dual impeller
  - — Air out at waist level, front and back
  - — Runs from 48 Vdc
- Cabinet control logic (CCL) module
- One to three H7263 bulk 48 Vdc power supplies
- Plug-in units (PIUs)

Physical Description of Lower Cabinet (PIUs)

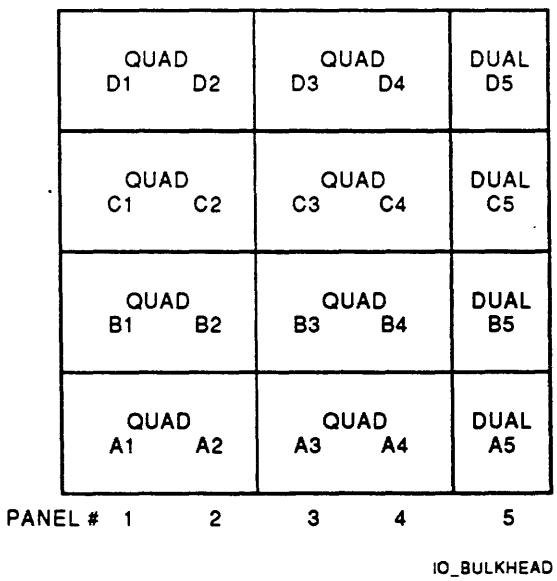
The lower cabinet has four quadrants as shown in Figure 1–12. The lower cabinet can hold XMI, VAXBI, disk, or UPS PIUs.

Figure 1–12 Lower Cabinet Quadrants (Top View)



Each I/O PIU has an I/O bulkhead that is organized as shown in Figure 1–13.

Figure 1–13 PIU I/O Bulkhead



XMI PIU

The XMI PIU takes two quadrants: Q1 and Q2 and/or Q3 and Q4, as indicated in Figure 1–14.

The front quadrant (Q1 or Q3) holds two power supplies and the XMI card cage.

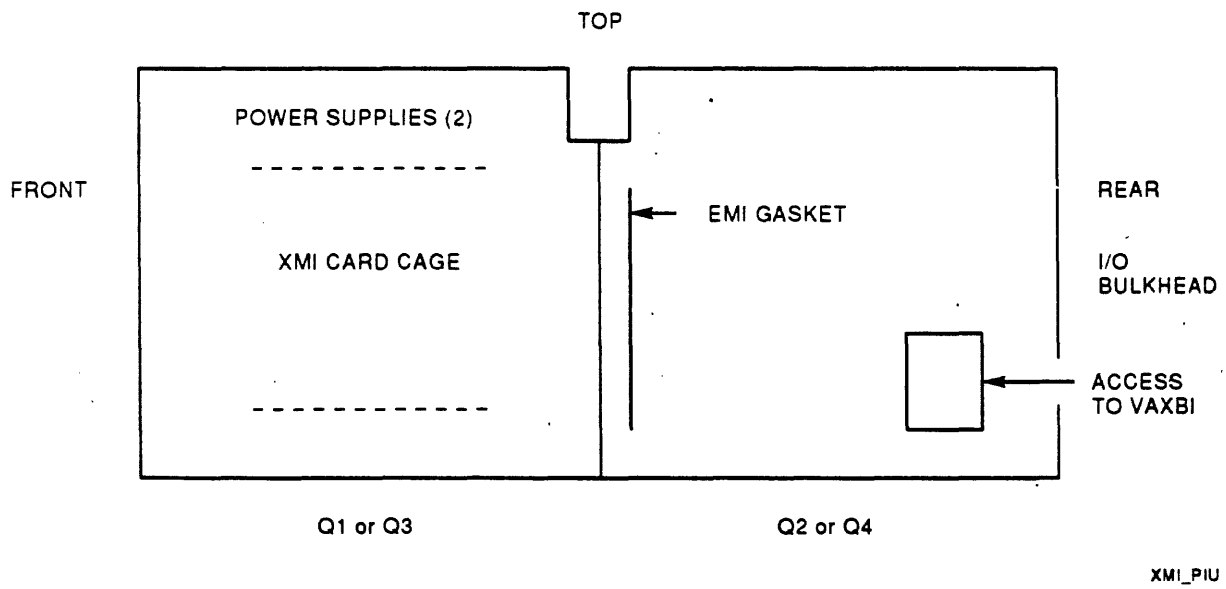
The XMI card cage is similar to the one used on the VAX 9000 series systems. The clock card (T2030-YA) is in slot 7 and the LAMB module (T2028) is in slot 8. If a VAXBI PIU is present, there will be a DWMBB/A module in XMI bus slot 1.

The two supplies convert 48 Vdc to the dc voltages used on the XMI modules.

- Left supply (from front, larger) provides +5.1 Vdc
- Right supply (from front, smaller) provides -5.2 Vdc, -2.1 Vdc, +12 Vdc, -12 Vdc, and four outlets with +13.5 Vdc

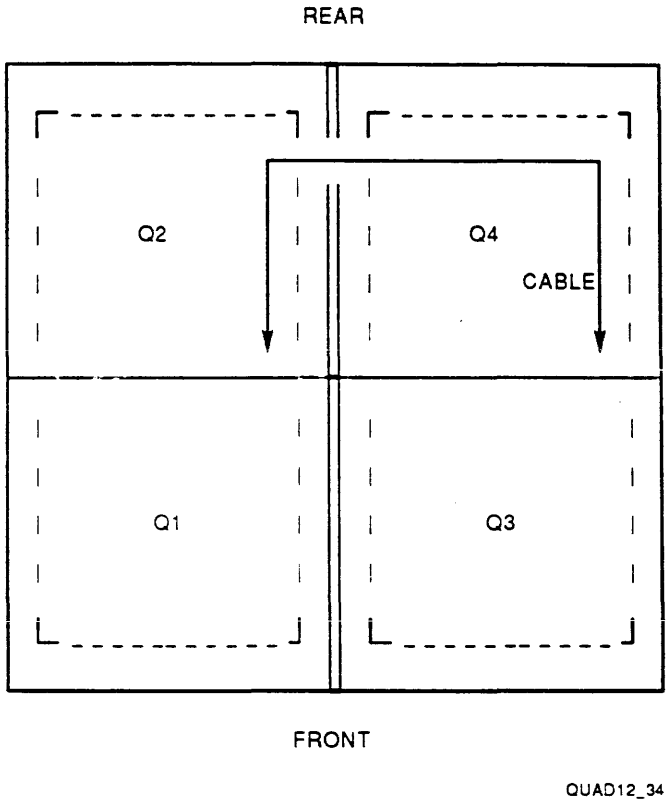
The rear quadrant (Q2 or Q4) provides room for I/O cabling and bulkheads.

Figure 1–14 XMI Plug-In Unit (Side View)



The XMI PIU and VAXBI PIU are enclosed as EMI units as shown in Figure 1–15. There is an EMI gasket where the front and rear sections are joined.

Figure 1–15 EMI Enclosures for XMI and VAXBI PIUs



VAXBI PIU

The VAXBI PIU takes two quadrants: Q3 and Q4, as indicated in Figure 1–16. It will not be installed in quadrants Q1 and Q2.

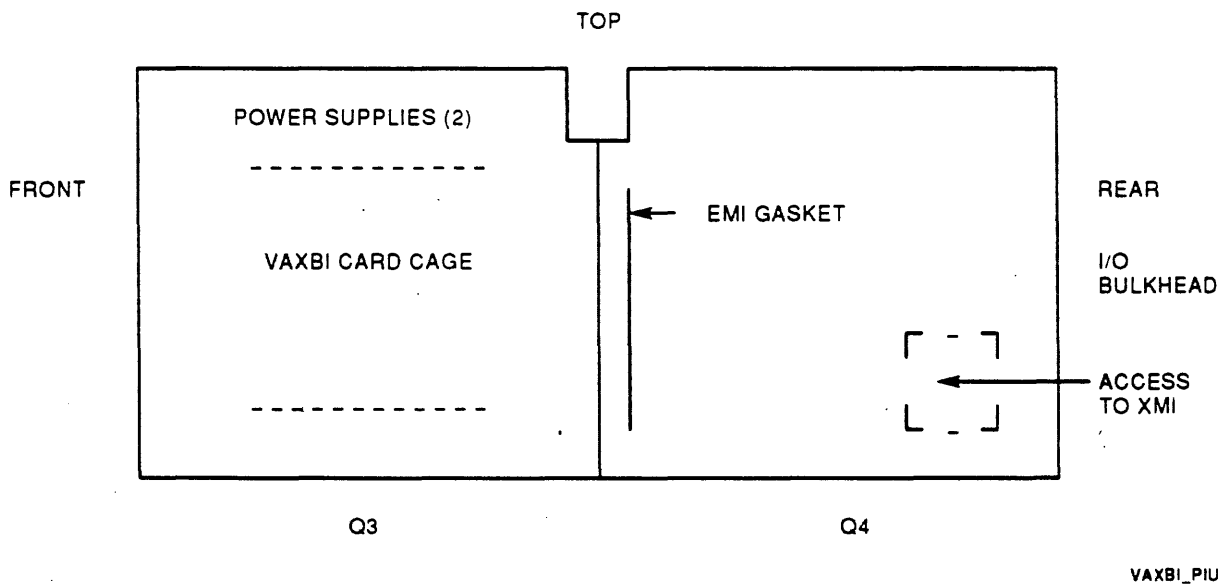
The front quadrant (Q3) holds two power supplies (identical to XMI PIU) and two 6-slot VAXBI card cages combined to make one 12-slot VAXBI. The rear quadrant (Q4) provides room for I/O cabling and bulkheads.

Slot 1 is used for the DWMBB/B module leaving 11 usable I/O slots.

The two supplies convert 48 Vdc to the dc voltages used on the VAXBI modules.

- Left supply (from front, larger) provides +5.1 Vdc
- Right supply (from front, smaller) provides -5.2 Vdc, -2.1 Vdc, +12 Vdc, -12 Vdc, and four outlets with +13.5 Vdc

Figure 1–16 VAXBI Plug-In Unit (Side View)



VAX 7000-600 Uninterruptable Power Supply (UPS) PIU

The VAX 7000-600 system UPS PIU takes two quadrants: Q1/Q2 or Q3/Q4. The B-PIU comes with a 4-battery unit. One or two 4-battery units can be added to the UPS PIU (up to 3 units). Four batteries provide up to 11 minutes of power for each H7263 unit. Each group of 4 batteries is protected by a fuse. The wire harnesses are labelled to ensure that battery groups are connected to the correct H7263.

NOTE

Single battery: Digital part number 12-36168-02

Four battery unit with packaging: H7238-AA

Storme Disk PIU, BA655-AA (Less Drive Units)

The disk PIU for SCSI occupies any one quadrant of the:

- Four lower quadrants in the main system cabinet (Q1, Q2, Q3, Q4)
- Six quadrants in either expander cabinet (Q1, Q2, Q3, Q4, Q5, Q6)

The Storme Disk PIU houses two disk shelves. Each shelf contains a power unit at the bottom and seven spaces for various combinations of bricks and disk units.

- 3 1/2-inch disk (RZ26-VA) uses one space
- 3 1/2-inch tape (TLZ06-VA) uses one space
- 5 1/4-inch disk (RZ73-VA) uses three spaces

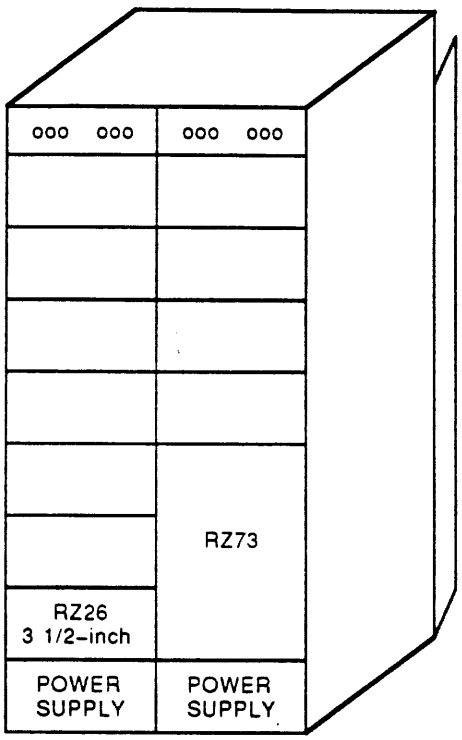
There are two SCSI bus connectors on each shelf. They are accessed from the top front of the cabinet and are located on the back wall of the shelf.

The SCSI bus cable connects to the leftmost connector of the shelf.

If both shelves are to be on one SCSI bus, then a SCSI bus cable goes to the leftmost connector on one shelf. Another cable plugs into the rightmost connectors of the two shelves.

Air flows in the front of the disk drive, into a plenum, and then up or down (depending on lower or upper quadrant location).

Figure 1–17 Storme Disk Plug-In Unit



DSSI Disk PIU, BA654-AA (Less Bricks/Drives)

The disk PIU has a unique LASER front bezel and occupies any one quadrant of the:

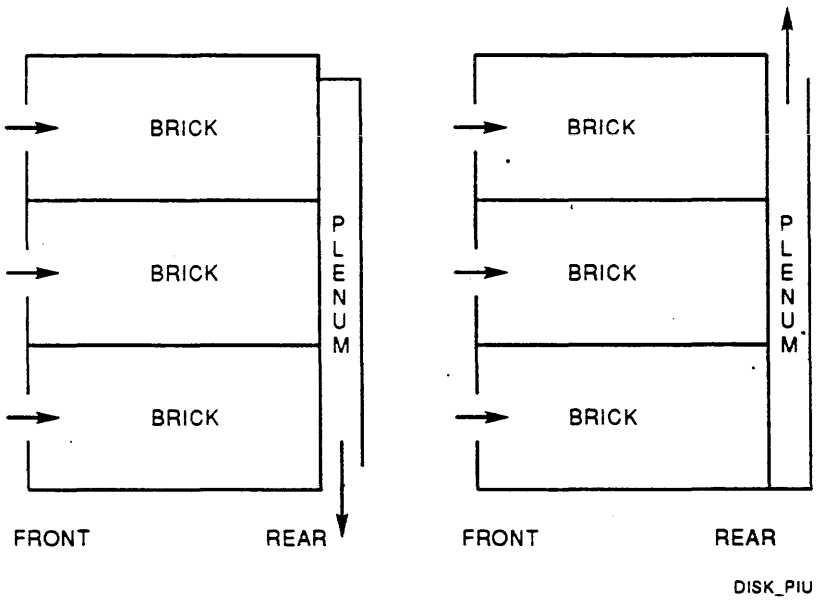
- Four lower quadrants in the main system cabinet (Q1, Q2, Q3, Q4)
- Six quadrants in either expander cabinet (Q1, Q2, Q3, Q4, Q5, Q6)

The disk PIU will house three disk bricks, a power and signal connector, and an EMI/RFI filter assembly. All I/O signal cables will connect to the front of the disk bricks.

A single brick holds two 5 1/4-inch disk drives (RF73), two local disk converters (LDC), and a disk control panel (DCP) interface module.

Air flows in the front of the disk drive, into a plenum, and then up or down (depending on lower or upper quadrant location).

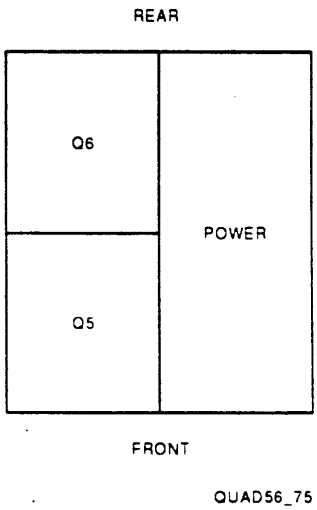
Figure 1–18 Disk Plug-In Unit (Side View)



Physical Description—Expander Cabinet (PIUs)

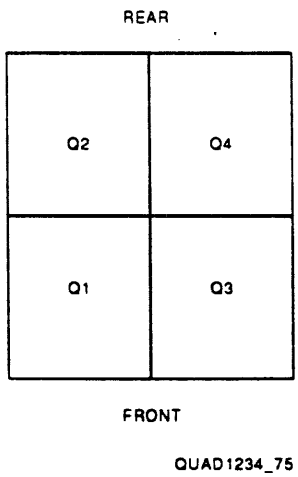
The upper part of the expander cabinet supports a disk PIU in quadrants Q5 and Q6. This is the only PIU allowed in the upper expander cabinet because of cooling and I/O cabling limitations.

Figure 1–19 Upper Expander Cabinet Quadrants (Top View)



The lower part of the expander cabinet can be treated the same as the lower part of the main cabinet. It supports disk, XMI, VAXBI, or UPS PIUs.

Figure 1–20 Lower Expander Cabinet Quadrants (Top View)



VAX 10000-600 Systems Physical Description

The VAX 10000-600 systems have up to 60 minutes UPS capability while VAX 7000-600 systems have up to 11 minutes UPS capability.

The VAX 10000-600 systems come in two versions: three-cabinet and five-cabinet. The three-cabinet version is shown in Figure 1–21, and the five-cabinet version is shown in Figure 1–22.

All four lower quadrants of the main cabinet (Q1, Q2, Q3, Q4) contain battery PIUs with a total of 24 batteries. A dedicated battery cabinet with 24 batteries supplies UPS power for each expander cabinet.

The three-cabinet system has two XMI PIUs, while the five-cabinet system has four XMI PIUs.

Figure 1–21 VAX 10000-600 (3 Cabinets)

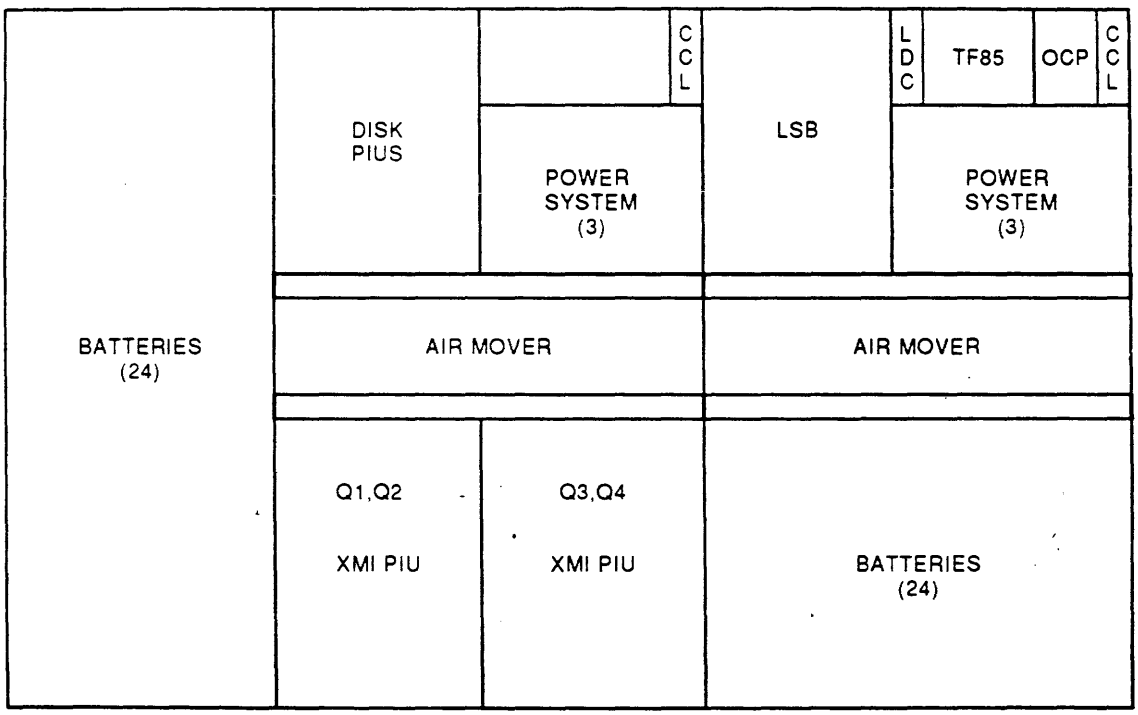
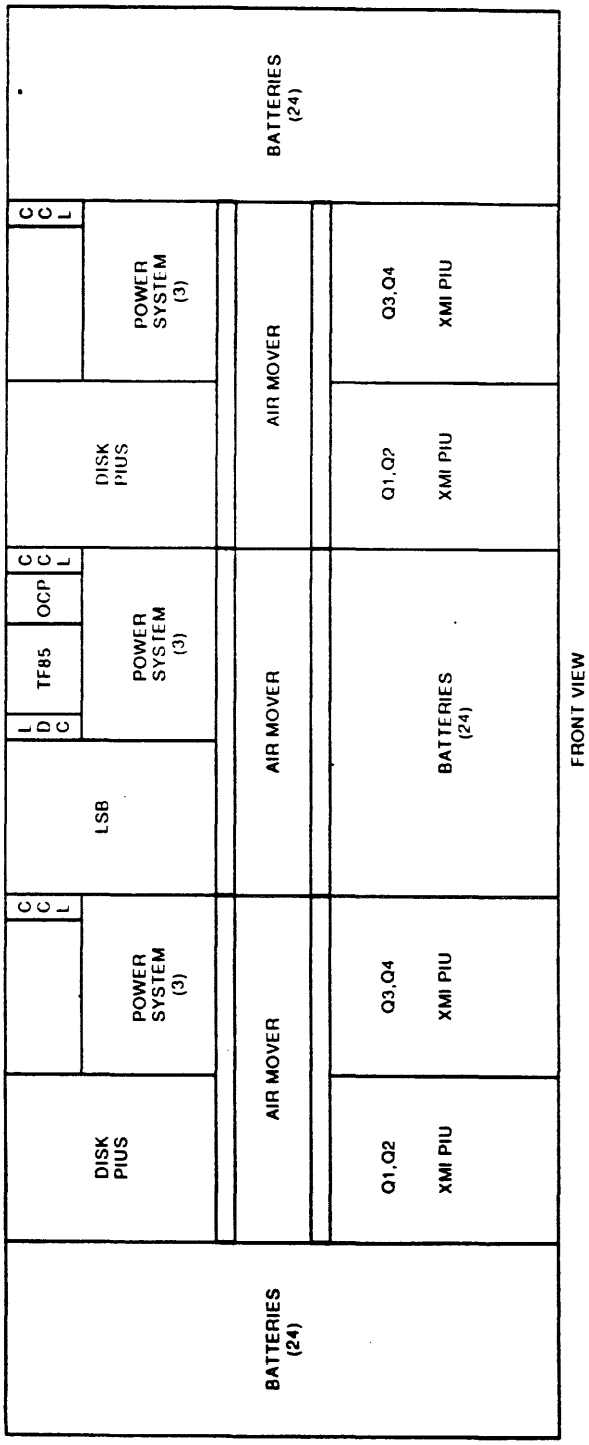


Figure 1–22 VAX 10000-600 (5 Cabinets)



BLASER_5CABS_80R

7000 and 10000 System Power Regulators

The 7000 and 10000 systems have the same power arrangement except for battery backup.

- The 7000 system uses four batteries in series to back up each H7263 regulator.
- The 10000 system uses two sets of four batteries in series to back up each H7263 regulator. The two sets are in parallel and generate 48 Vdc but can store more charge, so they are able to provide current for a longer period.

The systems use the same H7263 regulator, but the regulator must be able to charge the backup batteries at different rates. A nonvolatile status in the regulator determines which charging rate will be used.

This nonvolatile status in the H7263 is set at manufacturing to the correct value for the system.

All replacement H7263 regulators are shipped with the status set for a 7000 system. If it is being placed in a 10000 system, the status must be changed to accommodate the different charging rate.

The method used to set the status (4 or 8 batteries) is shown in Example 1–1. The number *n* is either 4 for 7000 systems or 8 for 10000 systems.

Example 1–1 Setting the H7263 Battery Charge Rate

```
>>>set mode advanced
>>>set power main -b n
>>>set power left -b n
>>>set power right -b n
```

The H7263 has now accepted the temporary battery values. To make the temporary values permanent, turn the OCP keyswitch to Disable. This sets the values.

The battery values are now permanent in the H7263.

7000 Review Exercise

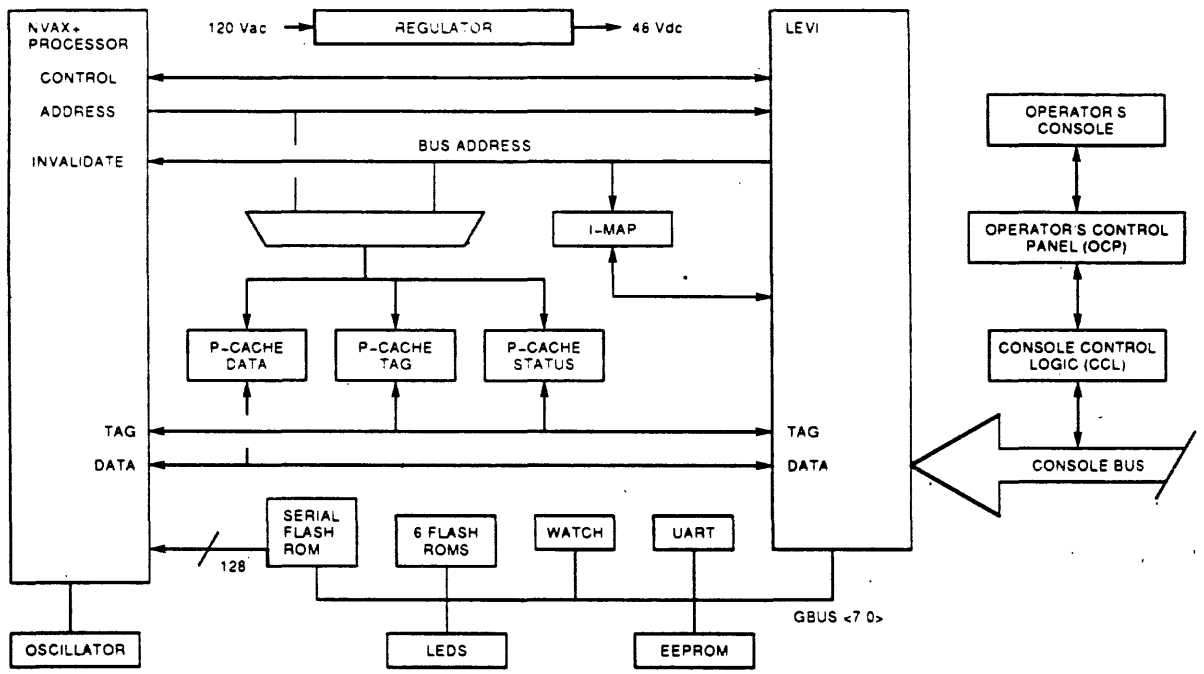
1. A 7000 system can contain up to 7 memory modules in the LSB unit.

   a. True

   b. False

2. There are errors in five areas of the VAX 7000-600 system diagram of Figure 1–23. Mark the errors and note the corrections.

Figure 1–23 Problem 2 Diagram



PROBLEM2_70

3. Which of the following statements describe legal configurations for a 7000 system?

   a. The main cabinet contains three PIUs, and the right expander cabinet contains five PIUs.

   b. The main cabinet contains four PIUs.

   c. The main cabinet contains two PIUs, and the right expander cabinet contains six PIUs.

   d. The main cabinet contains three PIUs, the right expander cabinet contains five PIUs, and the left expander cabinet contains three PIUs.

   e. All of the above.

4. Which of the following statements best describes the differences between a 7000 system and a 10000 system?

   a. The 10000 system has a UPS, while the 7000 system cannot support a UPS.

   b. The 10000 system is always larger than a 7000 system.

   c. The 10000 system UPS can sustain the system for a longer duration than the 7000 system UPS.

   d. There is no difference between the systems.

   e. All of the above.

5. Which I/O buses are or will be supported by the VAX 7000-600?

   a. DSSI, UNIBUS, VME, CI, SCSI

   b. DSSI, VAXBI, VME, SCSI, SDA

   c. DSSI, SCSI, VME, CI, VAXBI

   d. SDA, VME, VAXBI, CI, DSSI

   e. None of the above.

6. An H7263 power regulator for a DEC 10000-600 system must have its battery charge status set with the character *E*.

   a. True

   b. False

7. Which one of the following statements is true?

   a. A STORME PIU can hold fourteen 3 1/2-inch disk drives.

   b. A STORME PIU can hold fourteen 5 1/4-inch disk drives.

   c. A STORME PIU can hold a mixture of 3 1/2-inch SCSI drives and 5 1/4-inch DSSI drives.

   d. A STORME PIU can hold up to fourteen disk shelves.

   e. None of the above.

8. Which one of the following statements is true?

   a. A system can contain six LNP modules having pass 2 LEVI-A/Bs.

   b. A system can contain a mixture of LNP modules having pass 2 LEVI-A/Bs and LNP modules having pass 3 LEVI-A/Bs.

   c. A system containing three LNP modules having pass 2 LEVI-A/Bs and three MS7AA-AC modules (128 MB) is legal.

   d. A system can contain six LNP modules having pass 3 LEVI-A/Bs.

   e. All of the above.

9. Which one of the following statements is true?

   a. A VAX 10000-600 system NVAX+ microprocessor has a cycle time of 11 ns.

   b. A VAX 7000-600 system NVAX+ microprocessor has a cycle time of 11 ns.

   c. A DEC 10000-600 system EV4 microprocessor has a cycle time of 5.0 ns.

   d. A DEC 7000-600 system EV4 microprocessor has a cycle time of 5.5 ns.

   e. All of the above.

Power

Introduction

Details of the 7000/10000 platform power subsystem physical arrangement are described. The subsystem is presented with a discussion of the main power connections and the individual power field replaceable units (FRUs). The sequence of events at power on are shown.

Objectives

A Digital Services Engineer who provides support level maintenance service for the 7000/10000 systems should be able to:

- Recognize the sequence of steps performed by the system during the power on sequence.
- Identify the system units involved in each step of the power on sequence.
- Identify the system LEDs that should be lit during each step of the power on sequence.

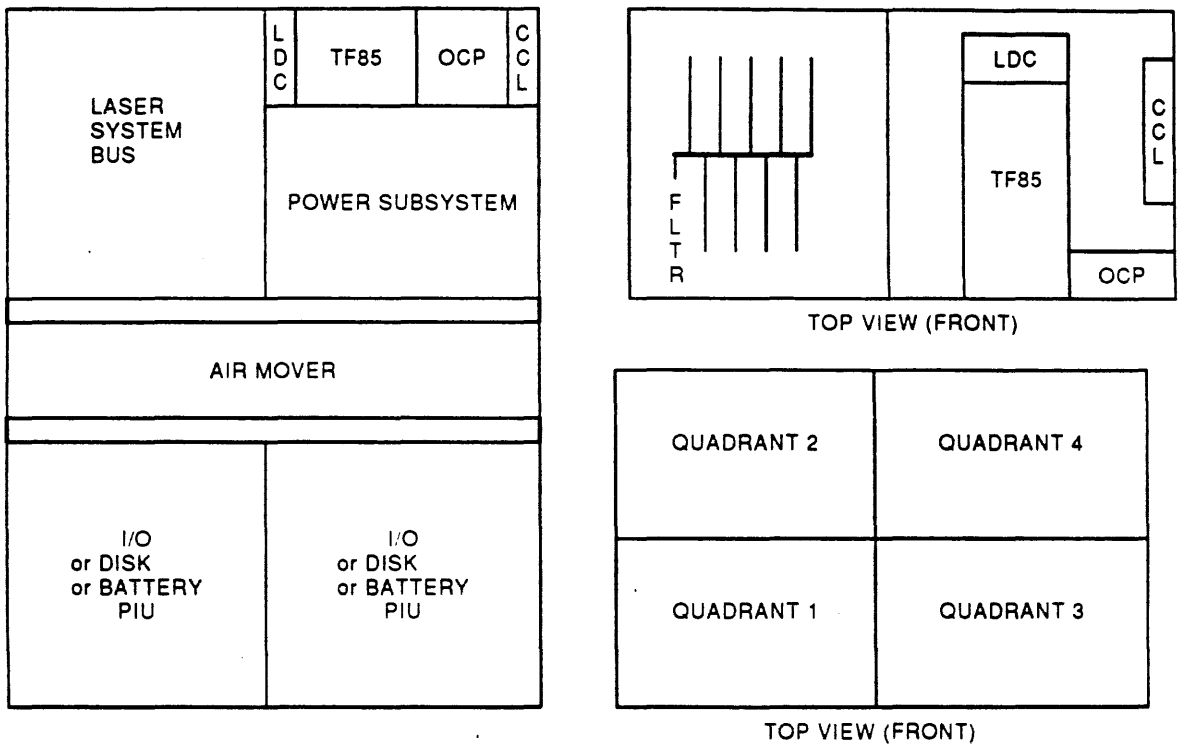
Resources

VAX 7000 Pocket Service Guide	EK-7000A-PG
DEC 7000 AXP and VAX 7000 System Service Manual	EK-7000A-SV
VAX 7000 Advanced Troubleshooting	EK-7001A-TS
DEC 7000 AXP System and VAX 7000 System Installation Guide	EK-700EB-IN

Power System Description

The 7000/10000 platform power subsystem components are located as shown in Figure 2–1 and listed below.

Figure 2–1 Power Subsystem in 7000/10000 Platform Cabinet



- Air mover
- Console control logic (CCL) and operator control panel (OCP)
- AC input box
- DC power distribution box
  - — H7263 bulk 48 Vdc power supplies (1 to 3)
- Local disk converter (LDC)
- DC-to-DC converters in disk and I/O plug-in-units (PIUs)
- DC-to-DC converters on each LSB centerplane module
- Battery plug-in unit (PIU)

System Airflow and Environment Sensors

A centrally located air mover provides air cooling to system components. There are air pressure sensors located within the cabinet to ensure that air movement is adequate. Temperature sensors are used to ensure that cabinet temperatures are within operating limits.

There are also temperature sensors in components, such as power supplies that affect the operation of the component.

Air Mover

The LASER platform air mover gets power from the 48 Vdc bus. The air mover draws air up through the lower cabinet and pulls air down through the upper cabinet area. The air is discharged at the middle of the cabinet, front and rear.

The air mover generates an optically isolated status signal BLOWER OK L. BLOWER OK L is asserted when the air mover is turning greater than 650 r/min. The signal lines BLOWER OK L and BLOWER OK RTN go to the CCL module.

The air mover has a servo unit that senses the temperature of ambient air.

At temperatures

- <25°C the impeller runs at 700 r/min.
- Between 25°C and 29°C the impeller stays at the current speed (tolerance area).
- >29°C the impeller runs at 830 r/min.

Air Pressure Sensors

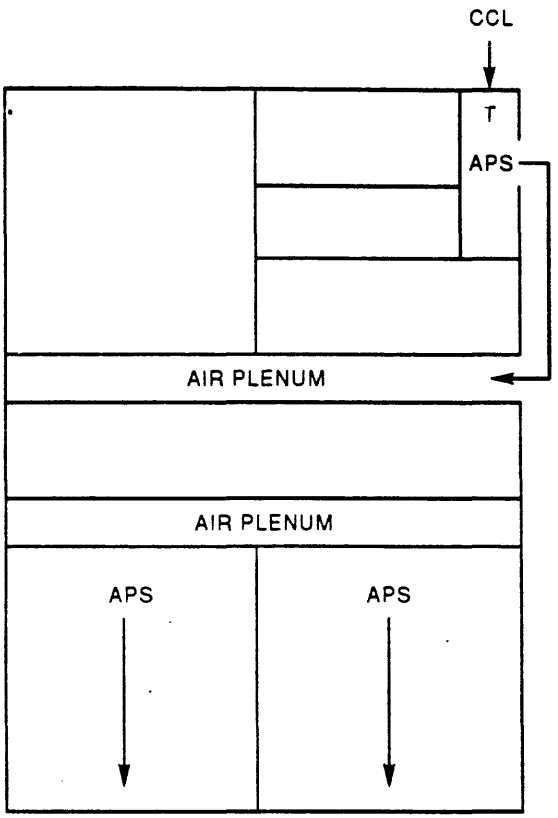
The air pressure sensor for the upper cabinet is located on the CCL module as shown in Figure 2–2. The air pressure sensor checks the difference between the ambient pressure (CCL module area) and the pressure in the upper air plenum. A tube is used to sense the air pressure in the upper plenum.

When the LSB static air pressure is incorrect the sensing switch opens and disables the dc-to-dc converters on the LSB modules. Shorting the sensor switch terminals allows the LSB dc-to-dc converters to come on.

NOTE

Design engineering may remove (or disconnect) the CCL air pressure sensor.

Figure 2–2 Cabinet Air Pressure and Temperature Sensors



There is also an air pressure sensor in each of the main and expander cabinet PIUs. The PIU pressure switch is located near the top of the unit and uses a tube to sense ambient air pressure (bottom of the unit, near the air inlet). The PIU pressure switch is wired in series with the PIU enable signal from the CCL, **PIU n EN**. When the static air pressure is incorrect, the switch opens and the PIU dc-to-dc converters are disabled.

NOTE

Design engineering may remove (or disconnect) PIU air pressure sensors.

Temperature Sensors

There is a temperature sensor (thermal switch, E8) located on the CCL module. When the switch temperature exceeds 68°C, the switch opens disabling the keyswitch ON CMD H signal to the H7263 power supplies. The H7263 power supplies treat this as an OCP key off condition and will disable the 48 Vdc outputs. When in doubt as to cause of a shutdown, short out the thermostat and see if H7263 power supplies turn on.

AC Input

The ac input box distributes and controls ac power to the system. There are three types of ac input units and two types of H7263 power supplies as shown in Table 2–1.

Table 2–1 AC Input Box and H7263 Power Supply Variations

Area	Nominal Input Voltage	AC Input Box	H7263 Power Supply	Configuration
North America	208 Vac	30-33798-01	30-33796-01	3 phase wye
Europe and GIA	380–415 Vac	30-33798-02	30-33796-02	3 phase wye
Japan	202 Vac	30-33798-03	30-33796-02	3 phase delta

NOTE

H7263-AA option is used in North America.
H7263-AB option is used in Europe, GIA, and Japan.

The ac input box includes the following components:

- Input ac line (4.5 m)
- Input circuit breaker (CB)
  - — Four visual indicators (1 per phase and 1 for CB manual position sense) green for open CB, red for closed CB
  - — Auxiliary contact connector J5 with two signal lines: BKR ON L and BKR ON L RTN. BKR ON L is deasserted if the circuit breaker is opened manually. The H7263 power supplies will not enable UPS (batteries) if the signal is deasserted.
  - — Can be padlocked in OFF position
  - — Variants 1 and 2 have single-phase overload trip
  - — Variant 3 (Japan) has phases ganged and all phases will open if a single phase has an overload trip
  - — Manual trip opens all poles in all variants
- Input ac line filter with EMI filter (includes a series inductor in green and yellow ground wire)
- AC monitor port connector (J4), protected with three 1 A line fuses (12-17199-04)
- AC power distribution to the H7263 power supplies (J1, J2, J3)

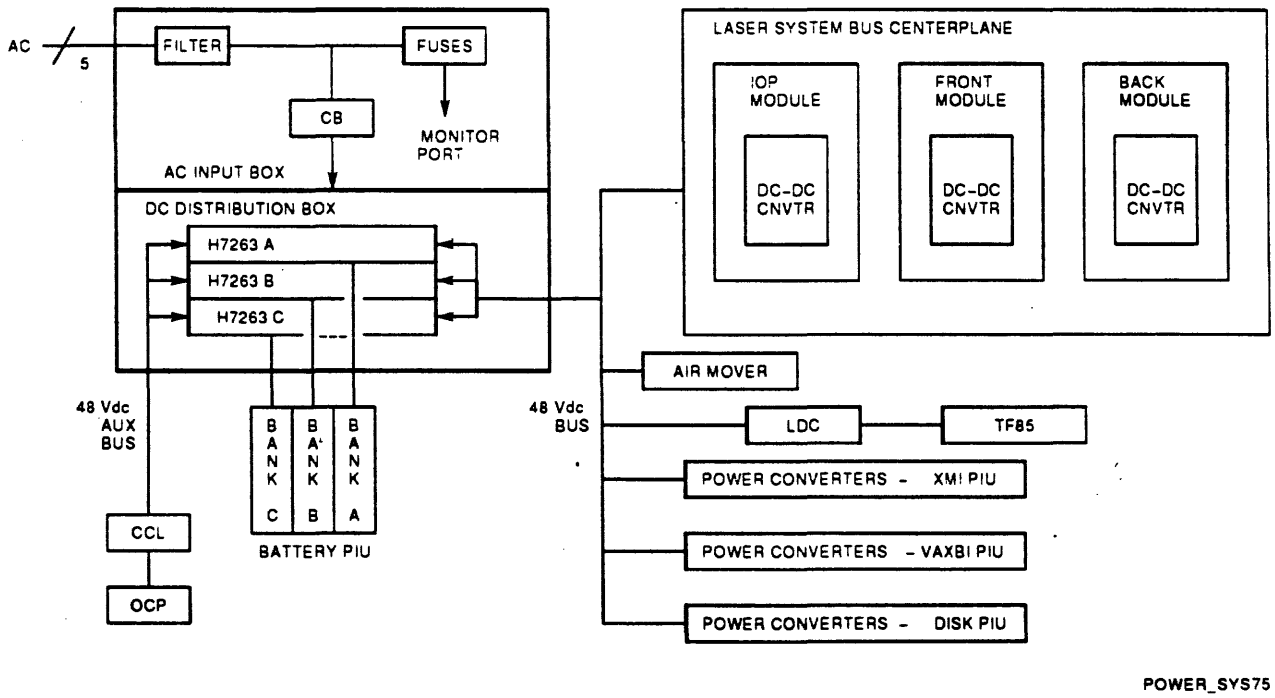
Bulk 48 Vdc Power Distribution Assembly, 30-35143-01

The dc distribution box is mounted to the cabinet and accepts the ac input box and 1 to 3 removable H7263 power supplies. The H7263 power supplies are mounted by sliding them into place and securing them to the box. The dc distribution box provides:

- 48 Vdc bus distribution point
- Battery pack interface to the H7263 power supplies
- Signal interconnect from console command logic to the H7263 supplies

The H7263 power supplies provide power to the 48 Vdc bus, the auxiliary 48 Vdc bus, and 48 Vdc battery packs (while charging). A simplified diagram of power connections is shown in Figure 2–3.

Figure 2–3 Power Subsystem Connections (Simplified)



The 1 to 3 H7263 power supplies provide 48 Vdc power that is distributed over the 48 Vdc bus. The 48 Vdc bus consists of a prewired cable/harness installed in the system.

The 48 Vdc lines attached to the console area LDC and air mover have plug-in connectors. The PIUs are attached to the 48 Vdc bus through blind mated connectors. The 48 Vdc bus is automatically connected to the PIUs when the PIUs are installed.

H7263 Bulk 48 Vdc Power Supplies

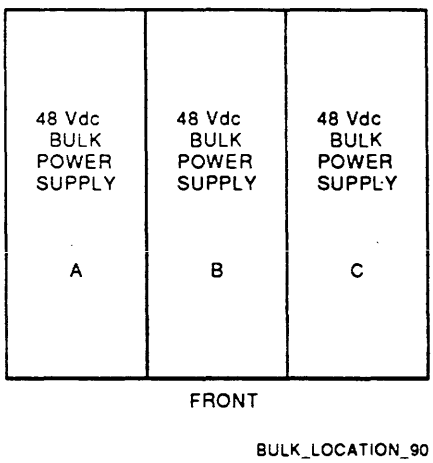
The H7263 power supplies input is single-phase ac and the output is 48 Vdc which is distributed throughout the system. There are two types of H7263 power supplies as listed in Table 2–1.

NOTE

The H7263 power supplies are keyed so that they cannot be mated with an ac input unit with the incorrect voltage.

There are 1 to 3 H7263 power supplies in a LASER platform. One supply is required and another may be needed depending on the system configuration. An additional supply is optional and provides redundant power. The power supply locations are labelled for their European and American input ac phases: A, B, and C as shown in Figure 2–4.

Figure 2–4 H7263 Power Supplies Locations (Front View)



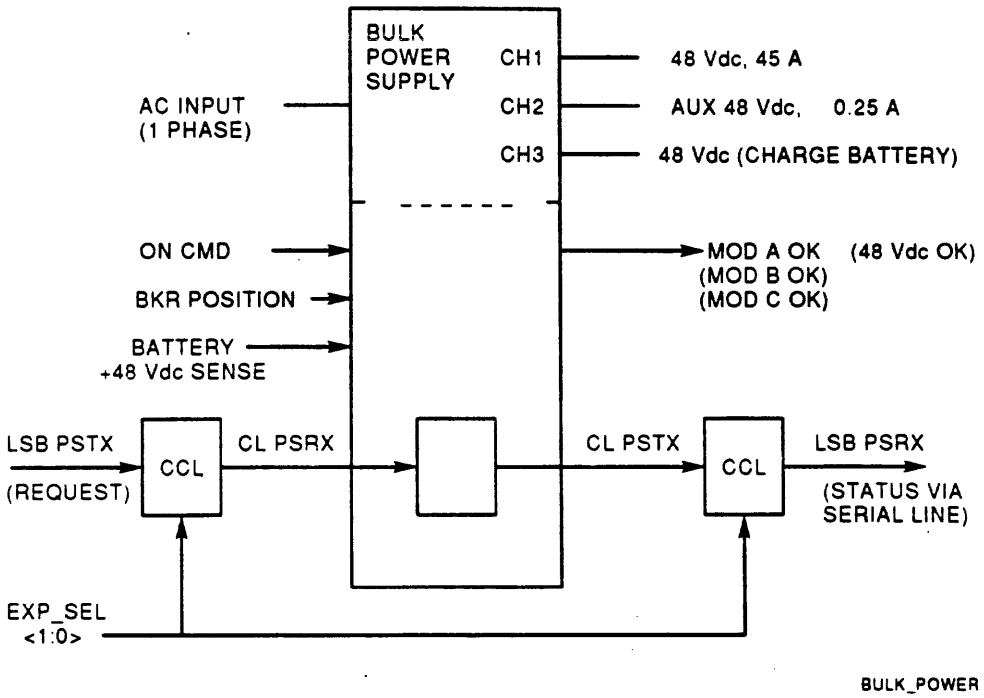
The H7263 power supplies ride through an ac voltage loss of up to 21 ms at full output power. If the outage lasts longer and there is no UPS, the system will fail.

Each H7263 power supply operates from a single ac phase and provides three 48 Vdc outputs:

- CH1, 48 Vdc bus, 45 A maximum —Shares current load in parallel with other units.
- CH2, auxiliary 48 Vdc, 0.25 A maximum—In parallel with other units. This voltage is present whenever ac voltage is present at the input of the H7263 power supplies.
- CH3, 48 Vdc battery charge—Voltage is present on these lines when ac voltage is present at the input of the H7263 power supplies (batteries charging) or when the UPS is providing power to the H7263 (batteries discharging).

When more than one H7263 power supply is present, they share the current load on the 48 Vdc bus. The supplies will source a maximum of 88 A and will source within 10.5 A of each other.

Figure 2–5 H7263 48 Vdc Bulk Power Supply



Each H7263 power supply contains a microprocessor and a serial interface to LSB signal lines. The microprocessor executes a self-test at power on and responds to status inquiries from the operator console and the operating system.

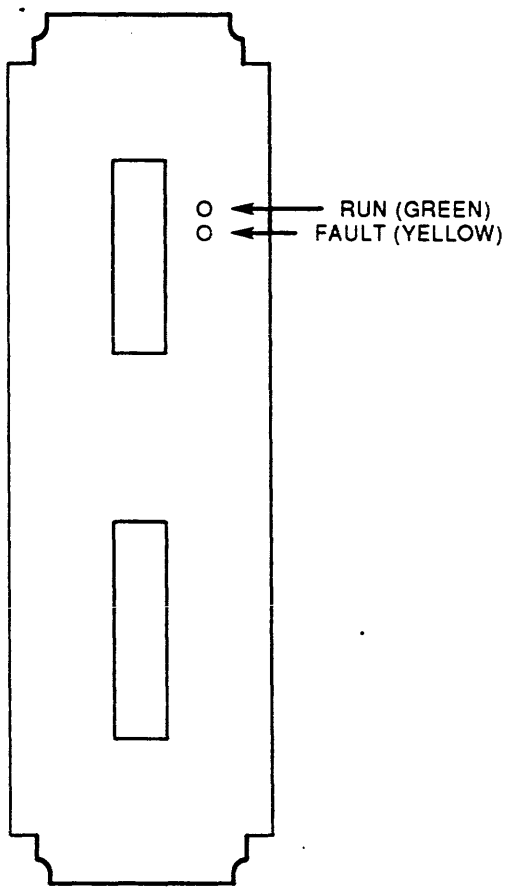
The microprocessor and serial interface are used by the console when implementing the SHOW POWER command described in Example 2–1. The sequence of communications follows:

- The console sends a command packet to a particular H7263 power supply in a particular cabinet using the signal line CL PSRX H. The console uses the signal lines LSB EXP SEL <1:0> to select the cabinet and asserts a line (A, B, or C) to select the H7263 power supply.
- The power supply microprocessor responds to the console by sending its status in a data packet (full, brief, or history) using the signal line CL PSTX H.

The microprocessor also manages the UPS feature of the power supply using CH3 to charge the batteries and switching to battery output when the ac source fails.

The supplies have two status LEDs visible from the front of the unit assembly shown in Figure 2–6.

Figure 2–6 H7263 Power Supply LEDs



RUN (Green)	FAULT (Yellow)	H7263 Power Supply Status
OFF	OFF	Not powered or internal bias failure.
OFF	ON	Fatal fault detected (48 Vdc out, not OK)
FLASHING (7Hz)	OFF	Power is applied, signal ON CMD H not yet received
ON	FLASHING (7Hz)	Nonfatal fault detected • Battery failure, charger inhibited • BBU not available • Boost failure • Heatsink temperature warning
ON	FLASHING (1Hz)	Battery discharge mode (BBU or test mode)
ON	OFF	Normal run mode

Failure Protection

The H7263 power supply has overvoltage and overload protection.

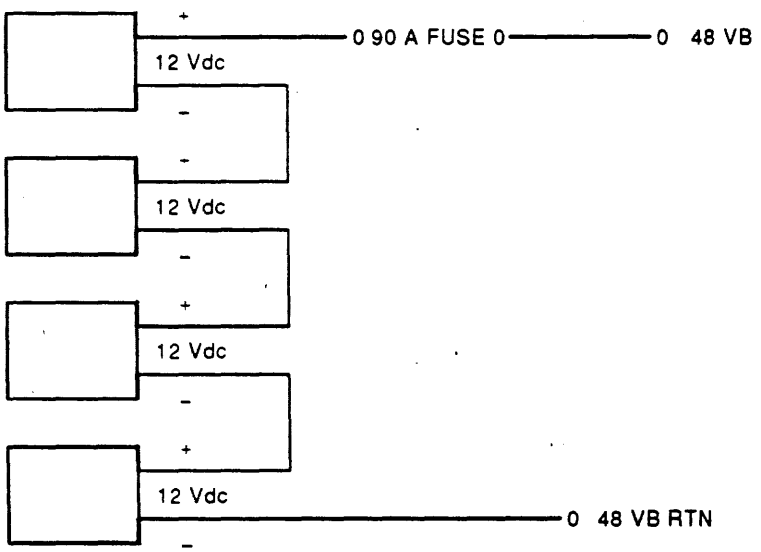
Overload protection reduces the 48 Vdc on the bus as current demand exceeds 50 A per supply. When the voltage is reduced to less than 36 Vdc, the 48 Vdc bus shuts down and a cold start will be initiated after approximately 1 second. At startup, overload shutdown is disabled for 0.9 seconds to allow charging of the load capacitance. MOD <A,B,C> OK L is deasserted in overload condition.

Overvoltage protection shuts down the 48 Vdc bus when the bus voltage exceeds 60 Vdc. The supplies are latched in the off condition. To restart, the main circuit breaker must be placed in the off position for 1 minute and then placed in the on position.

Uninterrupted Power Supply (UPS)

The uninterrupted power supply (UPS) batteries are located in a dedicated PIU and use a pack of four 12 Vdc batteries shown in Figure 2–7.

Figure 2–7 UPS Connections



UPS Modes of Operation

The UPS, controlled by the H7263 power supplies, has three modes of operation. They are charge, discharge, and off. UPS operation is summarized here:

- If the main circuit breaker has been manually opened, the UPS is off.
- If the battery packs are not present, the UPS is off.
- If the ac line voltage is not present but ON CMD is asserted, the UPS is discharging.
- When the UPS has normal conditions but is performing a test, it is discharging.
- With normal conditions, the UPS is charging until fully charged.

UPS Backup Time Capability

A battery pack will supply the 48 Vdc power of one H7263 power supply for approximately 11 minutes when new. The duration depends on battery charge state, age, and load.

The battery end of life occurs when the fully charged battery pack is able to provide only 8 minutes of backup power or requires more than 100 mA constant charge after a certain number of hours (to be determined) of charge time. The H7263 power supply calculates the battery pack capacity and reports the minimum time available in battery backup mode.

The UPS recharges the battery pack within 98% of capacity in less than 18 hours. The H7263 power supply will notice that ac power is going away and if the main breaker switch is closed, it will switch to the battery packs for the 48 Vdc on channels 1 and 2.

The three H7263 power supplies are independent of each other and each decides in which mode it will operate. In the rare case where a single ac phase fails, it is possible for a H7263 supply to be in BBU mode while the other supplies are operating from normal ac input. The console SHOW POWER command can be used to check for this condition.

UPS Tests

The UPS is programmed to perform two tests, deep discharge (11-minute test) and a 5-second test.

11-Minute Test

The UPS performs a deep discharge (11-minute) test a certain time (to be determined) after a battery pack is installed and also when requested by the console. The UPS supports the entire system for up to 11 minutes or until the undervoltage protection circuit is activated. The test is considered successful if the UPS is able to support the system for 8 or more minutes. The results are reported to the console.

5-Second Test

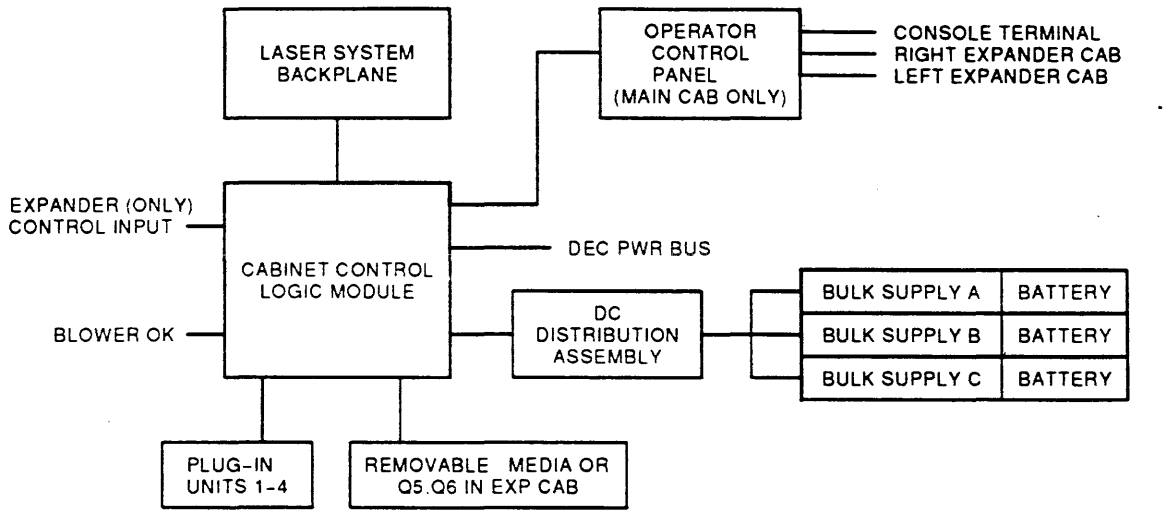
The UPS performs a test by supplying dc power to the entire system for 5 seconds.

- 5 minutes after every power on
- After each 500 hours of elapsed on time
- When requested

Console Control Logic

The console control logic (CCL) module is located behind the operator's control panel (OCP). The CCL module is connected as shown in Figure 2–8.

Figure 2–8 Console Control Logic Connections



There are 8 cables connecting the CCL module and other units as described in Table 2–2.

Table 2–2 Console Control Logic Module Cables

Connecting Unit	Cable Description
Air mover (BLOWER OK)	2-conductor cable
LASER backplane	50-conductor flat cable
Operator control panel	40-conductor flat cable
DC distribution assembly	20-conductor flat cable (via ac input box)
Plug-in units (4)	40-pin connector at CCL via custom cable harness to 4 PIUs and dc distribution assembly
Local disk converter	26-conductor cable
Expander cab control input	BC16E
Digital power bus	3-connector cable

The cable connections for the CCL module used in the expander cabinet differ from those in the main cabinet.

- No OPC cable.
- Main cab LDC cable is used for Q5 and Q6 optional DPIUs.

CCL Indicator LEDs

The CCL has an amber LED and four green LEDs that are visible at the back of the open LASER cabinet.

- +5 Vdc present (amber)
- PIU 1 ENABLED (green)
- PIU 2 ENABLED (green)
- PIU 3 ENABLED (green)
- PIU 4 ENABLED (green)

When power on sequencing is complete, the related green LED is turned on if a PIU is present and its status is satisfactory. If not turned on, the PIU may not be cabled correctly, has an airflow fault, or for an XMI PIU its XMI clock card is not installed correctly.

H7263 Power Supply Interface

The H7263 power-supply interface cables are shown and listed in Appendix A of the *DEC 7000 AXP and VAX 7000 System Service Manual.*

The CCL module has serial-line connections via the ac input box in the main cabinet to the H7263 power supplies. The H7263 power supplies have three ID lines used to identify the separate supplies: A, B, C.

There are also CCL module serial-line connections via the OCP to the H7263 power supplies in the optional right and left expander cabinets. The connections through the respective ac input boxes is identical to the main cabinet.

The receive and transmit multiplexers on the CCL module in the main cabinet are used to obtain information from the H7263 supplies located in the main and expander cabinets. The primary processor can be used to query the H7263 power supplies as shown in Example 2–1.

- AC voltage and current at H7263 power supply input
- DC voltage and current at H7263 power supply output
- Internal dc voltage
- Internal heat sink temperature
- Internal bulk voltage
- Battery status: capacity and time remaining
- Ambient temperature

Example 2–1 SHOW POWER Command (Repeated)

```
>>>show power
Cabinet:      Main:        Regulator:     A             B             C
-------------------------------------------------------------------------
          Primary Micro Firmware Rev:   2.0           2.0           2.0
        Secondary MIcro Firmware Rev:   2.0           2.0           2.0
                  Power Supply State:   NORMAL        NORMAL        NORMAL
             AC Line Voltage (V RMS):   113.71        114.35        115.93
               DC Bulk Voltage (VDC):   227.02        227.02        227.02
            48V DC Bus Voltage (VDC):   47.57         47.57         47.57
            48V DC Bus Current (ADC):   30.17         29.68         29.58
      48V Battery Pack Voltage (VDC):   50.85         50.75         47.91
      24V Battery Pack Voltage (VDC):   25.56         25.56         23.95
   Battery Pack Charge Current (IDC):   2.91          2.90          0
     Ambient Temperature (Degrees C):   26.22         24.80         24.75
                Elapsed Time (Hours):   290.00        290.00        290.00
Remaining Battery Capacity (Minutes):   8.00          8.00          8.00
     Battery Cutoff Counter (Cycles):   0             1.00          1.00
               Battery Configuration:   4 Batteries   4 Batteries   4 Batteries
                     Heatsink Status:   NORMAL        NORMAL        NORMAL
                 Battery Pack Status:   CHARGING      CHARGING      DISCHG'G
                Last UPS Test Status:   PASSED        PASSED        PASSED
LDC POWER Status     : 0
PIU Primary Status   : 0
PIU Secondary Status : 0
>>>
```

Module Power Supplies

There are dc-to-dc converters on the LSB modules and in the PIU units. These converters take 48 Vdc at the input and supply lower voltages such as +5 Vdc and -2 Vdc which are used on the modules.

LSB DC-to-DC Converters

Each LSB module has onboard dc-to-dc converters that supply its voltage needs. The 7000/10000 system CPU, LMEM, and IOP modules use the same general type of dc-to-dc voltage converters. The converters generate these dc voltages: +5.0 Vdc, +3.3 Vdc, +2.0 Vdc, +12 Vdc, and +15 Vdc. All voltages are not used on all modules.

The CCL turns on the LSB module dc-to-dc converters in three groups. They are turned on in sequence by asserting these signal lines: IOP ENABLE L, BACK ENABLE L, FRONT ENABLE L.

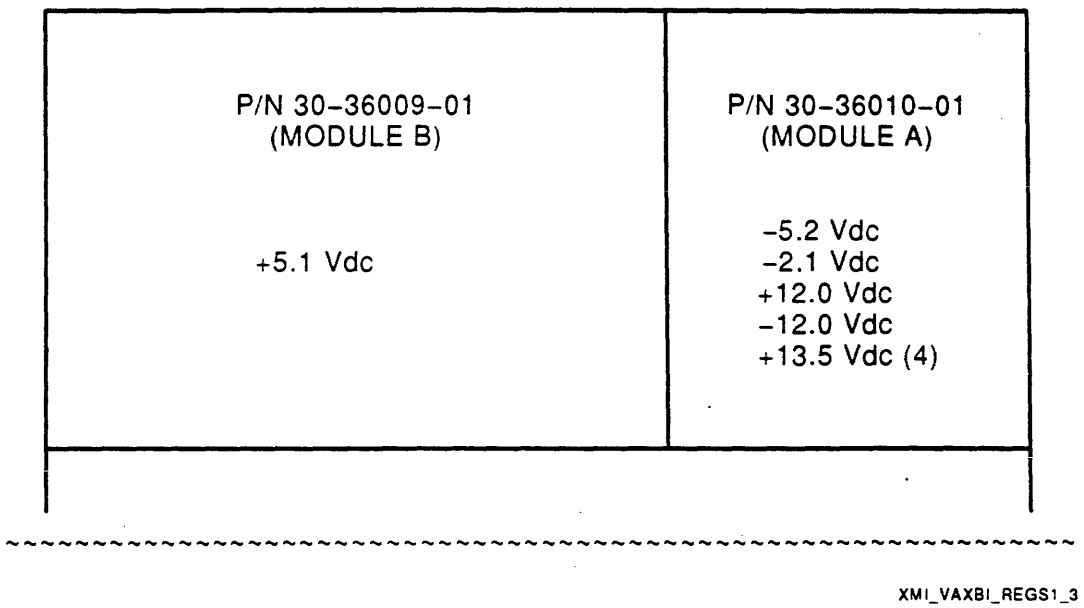
All group converter outputs except +2.0 Vdc are enabled by the one signal. The +2.0 Vdc output on all modules is also controlled by a signal line, 2V ENABLE H. The signal line has a pullup on the IOP module and is accessible to all modules on the LSB. The signal line is asserted low by each module until the module's onboard +5.0 Vdc output is within specification.

Each module has a 5 A fuse, not field replaceable, on its 48 Vdc input line.

XMI and VAXBI Power Supplies

The XMI and VAXBI PIUs have an identical set of two power supplies that generate lower dc voltages. These lower voltages are brought to bus bars on the respective backplanes. The two dc-to-dc power supplies are located at the top of the XMI and VAXBI enclosures shown in Figure 2–9.

Figure 2–9 XMI and VAXBI Enclosure Power Supplies (Front)



The CCL turns on the XMI and VAXBI PIU power supplies by asserting the signal lines: PIU <1:4> EN H.

Module B (left) delivers +5.1 Vdc. There is a 25 A fuse, not field replaceable, on its 48 Vdc input line.

Module A (right) provides the other voltages. There is a 5 A fuse, not field replaceable, on its 48 Vdc input line.

There is a thermal sensor in each supply. When the temperature exceeds the safe maximum the supply shuts down. It will remain shut down until the fault condition is removed and then it will reset itself.

NOTE

Module B develops bias voltage that is used by Modules A and B. If this voltage is not available, then neither voltage source will be able to generate power.

The two power supplies have status LEDs on their front panels. The LEDs on Module B (left) are shown in Figure 2–10 and listed in Table 2–3.

Figure 2–10 Module B LEDs and Switches

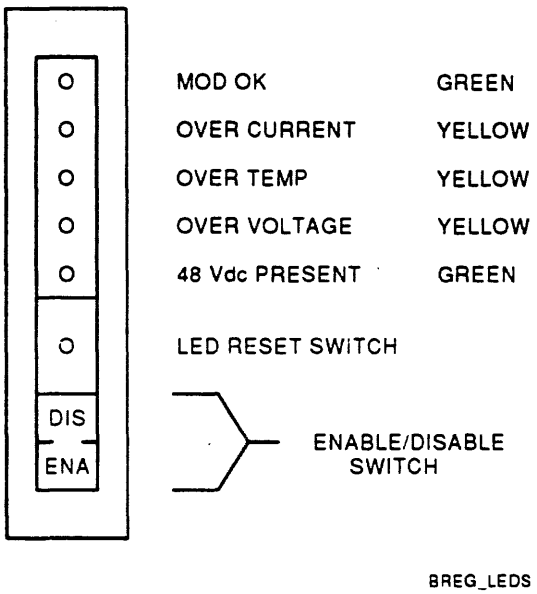


Table 2–3 Module B LEDs and Switches

LED Label	Description
MOD OK	Power supply within limits.
Over Current	LED remains latched after supply recovers.
Over Temperature	LED remains latched after supply recovers.
Over Voltage	LED and supply remain latched.
48 Vdc Present	Before input fuse (fuse not replaceable).
LED Reset Switch	Reset LEDs in both supplies.
Enable/Disable Switch	Turns off power to both power supplies.

The LEDs on Module A (right) are shown in Figure 2–11 and listed in Table 2–4.

Figure 2–11 Module A LEDs

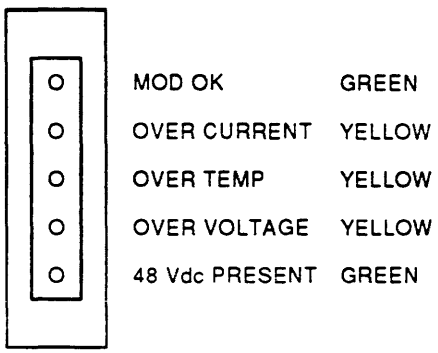


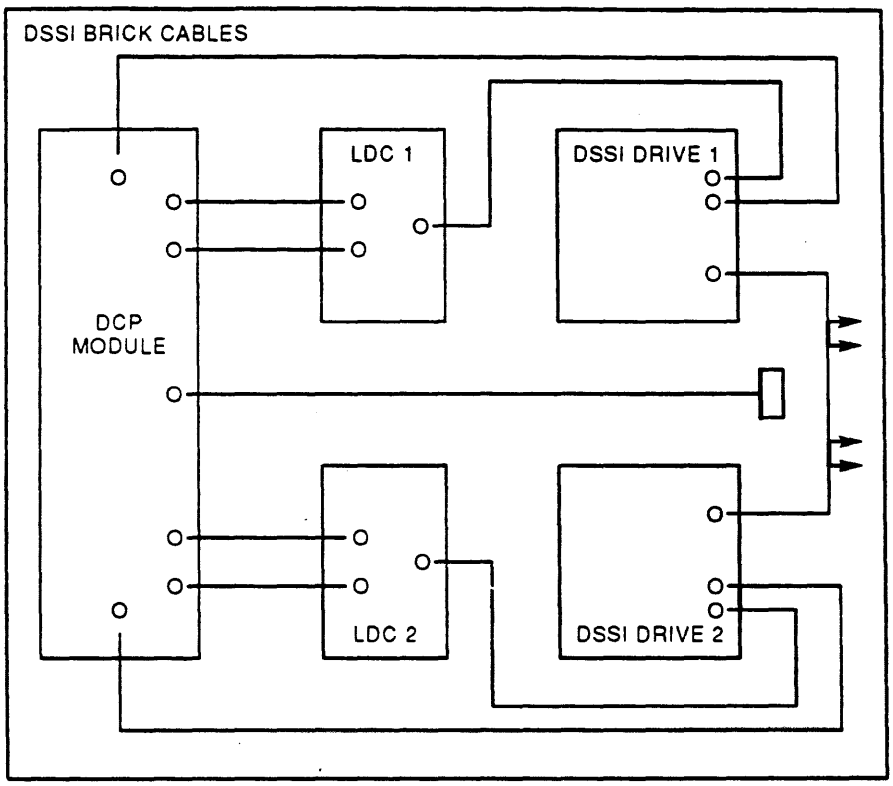
Table 2–4 Module A LEDs

LED Label	Description
MOD OK	Supply within limits.
Over Current	LED remains latched after supply recovers, monitors -2.1 Vdc and -5.2 Vdc current.
Over Temperature	LED remains latched after supply recovers.
Over Voltage	LED and supply remain latched, monitors -2.1 Vdc, -5.2 Vdc, +12 Vdc, and -12 Vdc voltages.
48 Vdc Present	Before input fuse (fuse not replaceable).

DPIU Power Supplies

The 5.25-inch disk PIU (DPIU) brick shown in Figure 2–12 has two local disk converter (LDC) units and a disk control panel (DCP).

Figure 2–12 DSSI Brick Local Disk Converters



The dc-to-dc power converters in the LDC generate +5 Vdc, +12 Vdc, and VTERM (5.0 Vdc). The CCL turns on the DPIU LDC power converters by asserting the signal lines PIU <1:4> EN H as follows:

- PIU <1> - Q1 and Q5
- PIU <2> - Q2 and Q6
- PIU <3> - Q3
- PIU <4> - Q4

The drives spin up in two groups: A and B.

- Signal line PRM ACLO A causes the DPIUs in Q1 and Q2 to spin up.
- Signal line PRM ACLO B causes the DPIUs in Q3 and Q4 to spin up.

There is a 5 A fuse, not field replaceable, on the 48 Vdc input line.

There is a green LED on the LDC that indicates +5 Vdc and +12 Vdc are above their specified minimum requirements.

The LDC has a thermal sensor that detects an overtemperature condition. When the safe temperature has been exceeded, the sensor will latch and disable the output voltages. The LDC must be initialized before it can turn on again.

Internal Local Disk Converter (REMOTE MEDIA LDC)

The local LDC dc-to-dc power converters generate +5 Vdc, +12 Vdc, and VTERM (+5.0 Vdc). The CCL turns on the LDC power converters by asserting the signal REMOTE MEDIA EN L.

There is a 5 A fuse, not field replaceable, on the 48 Vdc input line.

There is a green LED on the LDC that indicates +5.0 Vdc and +12.0 Vdc are above their specified minimum requirements.

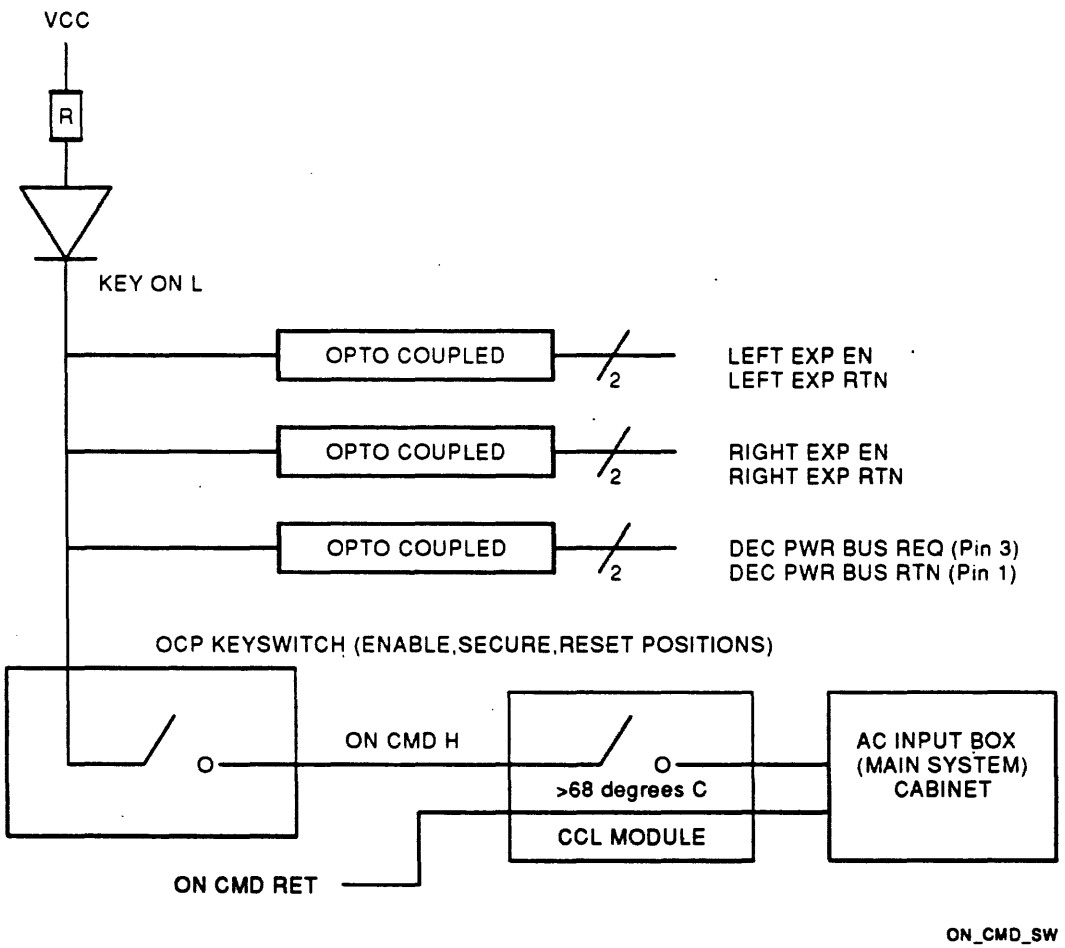
The LDC has a thermal sensor that detects an overtemperature condition. When the safe temperature has been exceeded, the sensor will latch and disable the output voltages. The LDC must be initialized before it can turn on again.

Power On and Reset Operation Sequences

The power on and reset operation sequences are described in this section.

The operator turns on system power by rotating the operator's control panel switch to the ENABLE, SECURE, or RESET positions. Turning the keyswitch to any of those positions asserts the KEY ON signal and lights the KEY ON LED. The ON CMD H signal is sent to the main system cabinet and the RIGHT EXP EN H and LEFT EXP EN H signals are sent to the expander cabinets as shown in Figure 2–13.

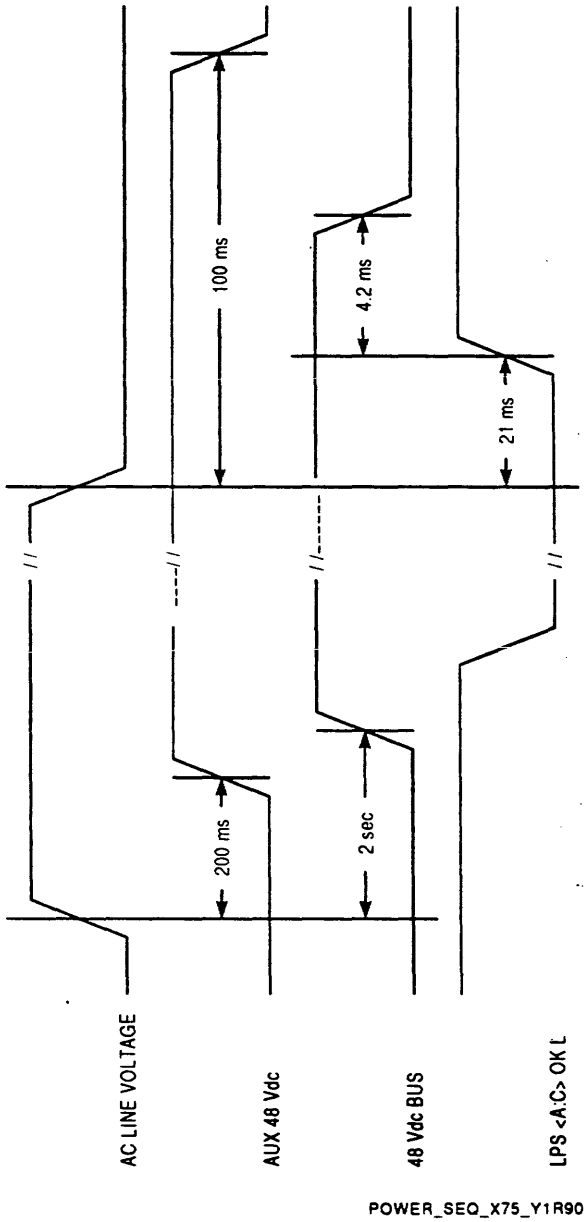
Figure 2–13 OCP ENABLE and SECURE Keyswitch Positions



The DEC PWR BUS REQ signal is also generated on the OCP module. The LASER platform DEC PWR BUS REQ is a master-only signal. This opto-isolated signal is available at the rear of the console control area.

System power sequences at power on and power off are shown in Figure 2–14.

Figure 2–14 System Power On and Power Off Times (Not to Scale)



Power On Sequence of Operation

The power on sequence starts when the operator turns the key of the operator control panel to ENABLE or SECURE. The sequence is complete when all ac and dc power is available to the system components.

Figure 2–15 Power On Flowchart

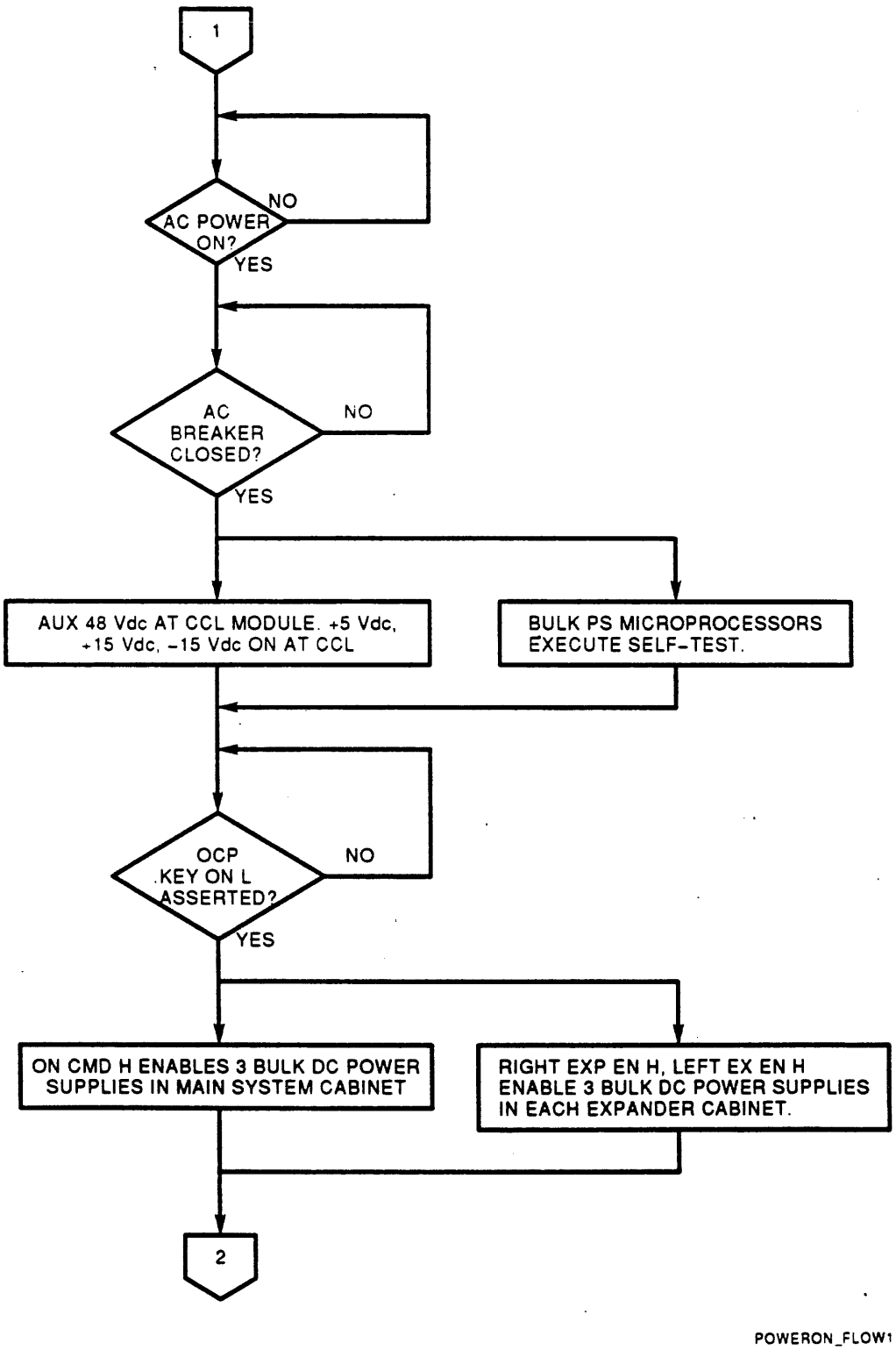


Figure 2–15 Cont'd on next page

Figure 2–15 (Continued) Power On Flowchart

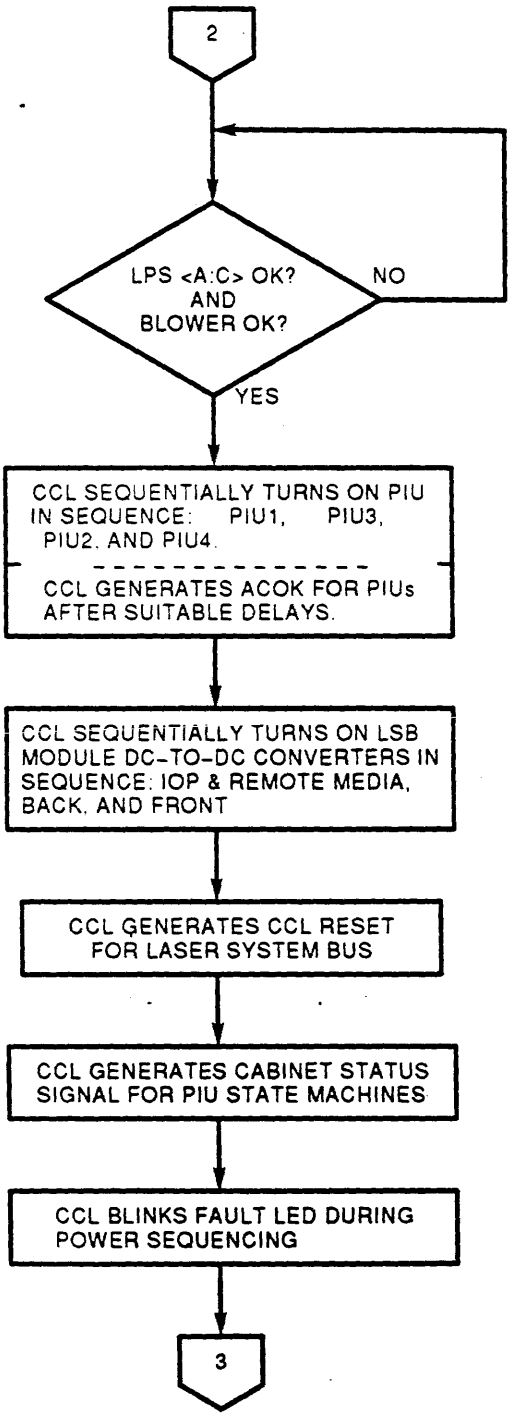
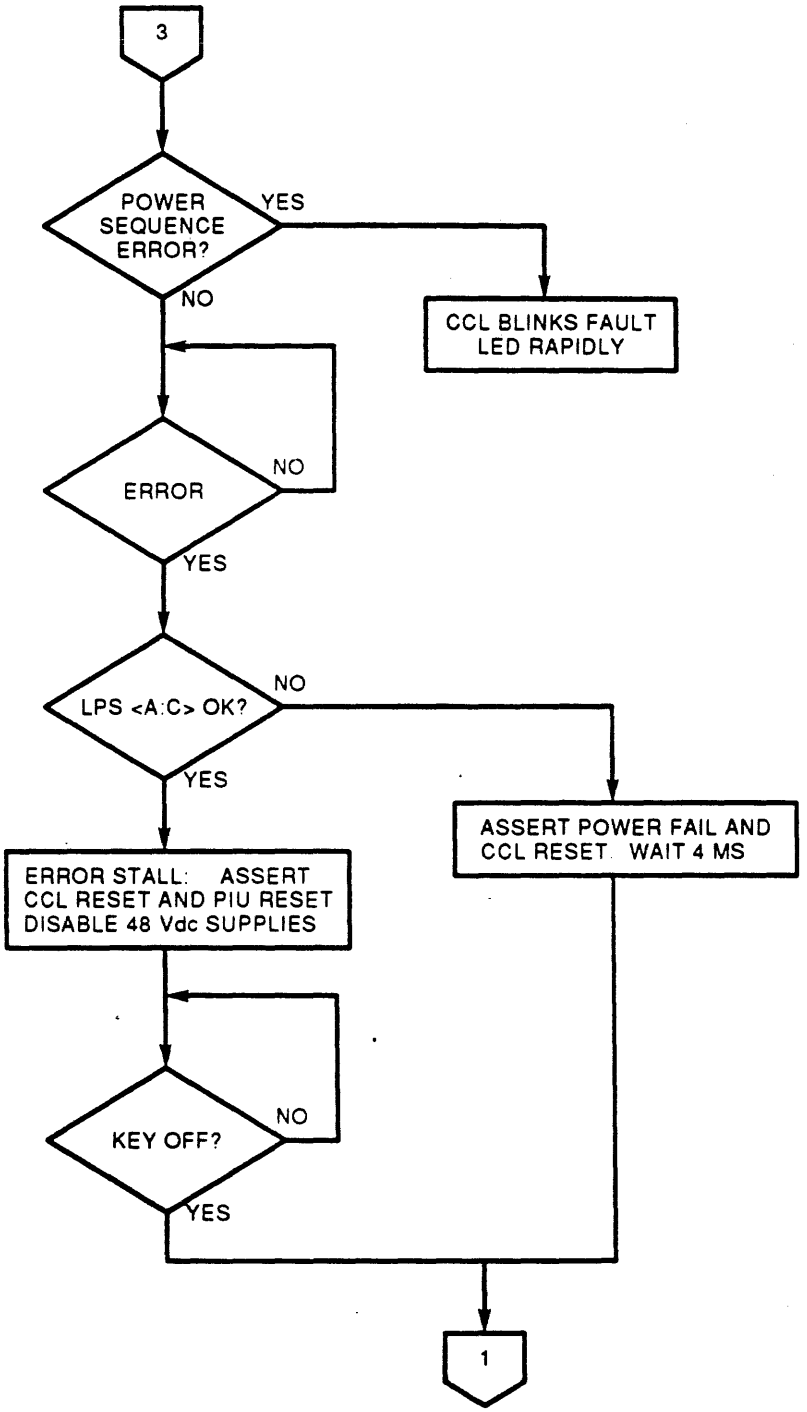


Figure 2–15 Cont'd on next page

Figure 2–15 (Continued) Power On Flowchart



Reset Sequence of Operation

The reset sequence starts when the operator turns the key of the operator control panel to RESTART.

Figure 2–16 Reset Flowchart

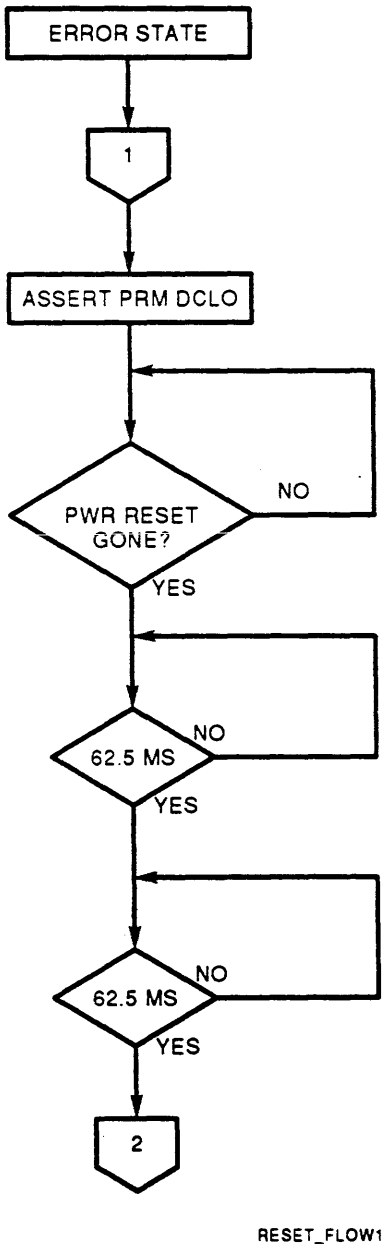
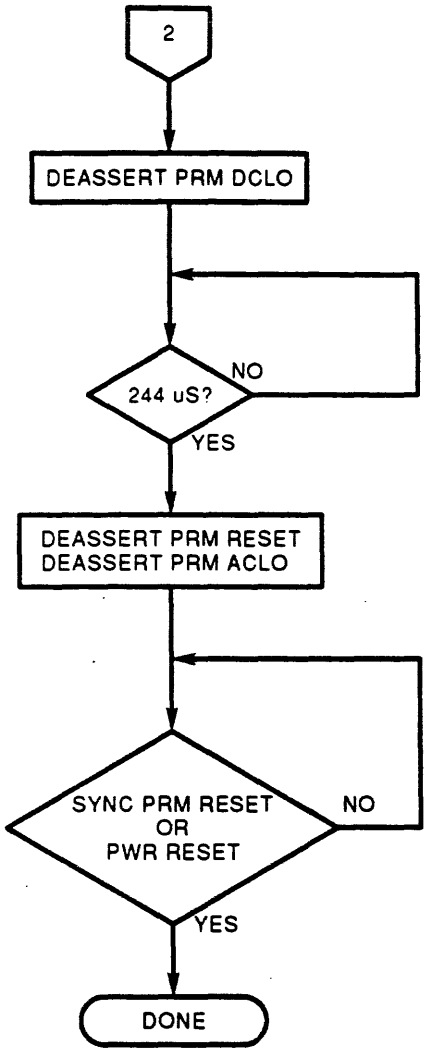


Figure 2–16 Cont'd on next page

Figure 2–16 (Continued) Reset Flowchart



Power Review Exercise

1. At least two H7263 power regulators must be providing 48 Vdc for a system to operate correctly.

   a. True

   b. False

2. Which one of the following statements is true?

   a. The BLOWER OK signal is used by an H7263 regulator to decide whether to turn battery backup on.

   b. The BREAKER ON signal is used by an H7263 regulator to decide whether to turn battery backup on.

   c. The CCL sends the ON CMD signal to the PIUs to turn them all on at once.

   d. The CCL turns off the H7263 regulators when either the PIU <1:4> MOD A OK signal or the PIU <1:4> MOD B OK signal is deasserted.

   e. None of the above.

3. Which one of the following statements is true?

   a. There are three types of ac input boxes and two types of dc distribution boxes.

   b. There are three types of ac input boxes and two types of H7263 regulators.

   c. There are three types of ac input boxes and one type of H7263 regulators.

   d. There is one type of ac input box and two types of H7263 regulators.

   e. None of the above.

4. There is a power control/status cable used to carry signals between the remote media LDC and the CCL in the main cabinet. When located in an expander cabinet, this same cable is used to carry control/status signals between the CCL and the PIUs in Q5 and Q6.

   a. True

   b. False

5. An XMI PIU Module B power supply provides a bias voltage for the Module A power supply. If the Module B power supply fails, it is likely that the Module A power supply will fail also.

   a. True

   b. False

6. In a console terminal SHOW POWER display, the battery cutoff counter (cycles) line indicates the number of times an H7263 supply has completed its deep discharge operation.

   a. True

   b. False

7. Which one of the following statements is true?

   a. It is possible to communicate with the H7263 supplies with the circuit breaker in the OFF position.

   b. The system must be in console mode to communicate with the H7263 supplies.

   c. It is possible to communicate with the H7263 supplies with the OCP keyswitch in the RESET position.

   d. It is possible to communicate with the H7263 supplies with the OCP keyswitch in the DISABLE position.

   e. All of the above.

8. The H7263 power supplies in a 10000 system with battery backup are identical to those in a 7000 system with battery backup.

   a. True

   b. False

Advanced Console

Introduction

This module provides the student with a detailed description of the system initialization and boot sequence. The student learns to use console advanced and diagnostic mode commands.

Objectives

A Digital Services Engineer who provides support level maintenance service for the 7000/10000 systems should be able to:

- Recognize the sequence of steps performed by the system during initialization and booting.
- Identify the system units involved in each step of power on and initialization.
- Determine which memory area is used to support the system console.
- Use the memory interleave scheme to determine which memory module is supporting a particular LSB physical address.
- Use the console to examine system hardware registers.

Resources

DEC 7000 AXP System VAX 7000 Operations Manual	EK-7000B-OP
VAX 7000 Advanced Troubleshooting	EK-7001A-TS
DEC 7000 AXP System Pocket Service Guide	EK-7700A-PG
VAX 7000 Pocket Service Guide	EK-7000A-PG

Console Subsystem Overview

The console subsystem is composed of the NVAX+ processor running console dedicated instructions and some hardware components distributed throughout the system.

Initialization

The console software is initialized as listed here:

- When power is first applied, the processor loads instructions from the SROM into virtual instruction cache (VIC) and starts running those instructions.
- The SROM instructions initialize the processor, run some tests, initialize backup cache, and then load GROM instructions from the other six Flash ROMs (GBus) into backup cache.
- The GROM instructions run the remaining self-tests, including memory tests over the LSB. The primary processor, still running console instructions, sets up console software structures, including the HWRPB, in memory. The console and diagnostic programs from the other six Flash ROMs are then loaded into memory. The console then loads PALcode into memory and puts PALcode pointers into the HWRPB.
- The primary processor prints the console prompt on the console terminal

Console Architecture

The console architecture is built on a shell (like an operating system). It is multitasking and uses interrupts: Priority 0 (lowest) through Priority 7 (highest). The processes are shown in Example 3–1.

- Timer process is Priority 7 (wakes up sleeping processes).
- Power on process is Priority 6 (gone before you see >>>).
- Entry process is Priority 5 (from reset or error).
- Shell process is Priority 3 (parses commands and qualifiers creating new processes).
- Idle process is Priority 0 (lowest priority, runs when no other process is ready).

There are eight ready queues, one at each priority (Priority 0 through Priority 7).

A new process is created by the shell process when you give a console command as shown in Example 3–3.

- Diagnostic tests are processes.
- Console commands are executable files that become processes.

Example 3–1 Show Process Command

```
>>>ps
   ID        PCB     Pri CPU Time Affinity  CPU   Program    State
--------  --------  ---  ------  --------  ---  ----------  -----------------
0000000b  0018bbe0   3      405  00000008   3          ps  running on tt_ctrl
0000000a  001868a0   3    35305  00000008   3          sh  ready
00000008  0018d860   5     1247  ffffffff   3    dup_poll  waiting on tqe
00000007  0018f240   5     1249  ffffffff   3   mscp_poll  waiting on tqe
00000006  00196dc0   5     1333  00000008   3    entry_03  waiting on entry_03
00000004  0019a300   2     1233  ffffffff   3  dead_eater  waiting on dead_pcb
00000003  0019bd60   7    11057  ffffffff   3       timer  waiting on timer
00000002  001a1800   6     1054  ffffffff   3  tt_control  waiting on tt_ctrl
00000001  00151f80   0   453525  00000008   3        idle  ready

>>>
```

Example 3–2 Process Status One

```
>>> set mode diagnostic

>>> mem_ex -p 0 &

>>> show status

15-JUN-145 139:35:44
ID  Program          Device     Pass Hard/Soft Bytes Written    Bytes Read
-------- -------- --------------- -------- --------- ------------ -------------
0000001a   mem_ex              mem     1204     0    0       9854976       9854976

>>> show status

15-JUN-145 139:35:53
ID  Program          Device     Pass Hard/Soft Bytes Written    Bytes Read
-------- -------- --------------- -------- --------- ------------ -------------
0000001a   mem_ex              mem     5091     0    0      41697280      41697280

>>> ps
ID       PCB      Pri CPU Time Affinity CPU  Program    State
-------- -------- --- -------- -------- --- ---------- -----------------------
0000001d 0013ad20 3          0 00000001 0          ps running
0000001a 0013d2a0 2    1466000 00000001 0      mem_ex ready
00000019 00140080 2       1000 00000001 0       sh_bg waiting on child_done
00000015 00143940 2          0 00000001 0   mopdl_exa0 waiting on mop_exa0_dlw
00000014 001480c0 5          0 00000001 0 process_exa0 waiting on exa0_isr
00000011 0014f7a0 1          0 00000001 0    pua_poll ready
00000010 00162b40 6       1000 00000001 0  pua_receive waiting on puu_receive
0000000e 00151f20 3       9000 00000001 0          sh ready
0000000b 001598c0 5          0 00000001 0   mscp_poll waiting on tqe
00000007 00161160 5          0 00000001 0    entry_00 waiting on entry_00
00000004 001662c0 2          0 ffffffff 0  dead_eater waiting on dead_pcb
00000003 00167d20 7      19000 ffffffff 0       timer waiting on timer
00000002 0016f2a0 6          0 ffffffff 0  tt_control waiting on tt_control
00000001 000777b0 0    1493032 00000001 0        idle ready

>>>
```

A process executes and when it is done goes into the DeadProcess queue. Later another process clears aways these dead processes and returns allocated memory space to the heap.

Example 3–3 Process Status Two

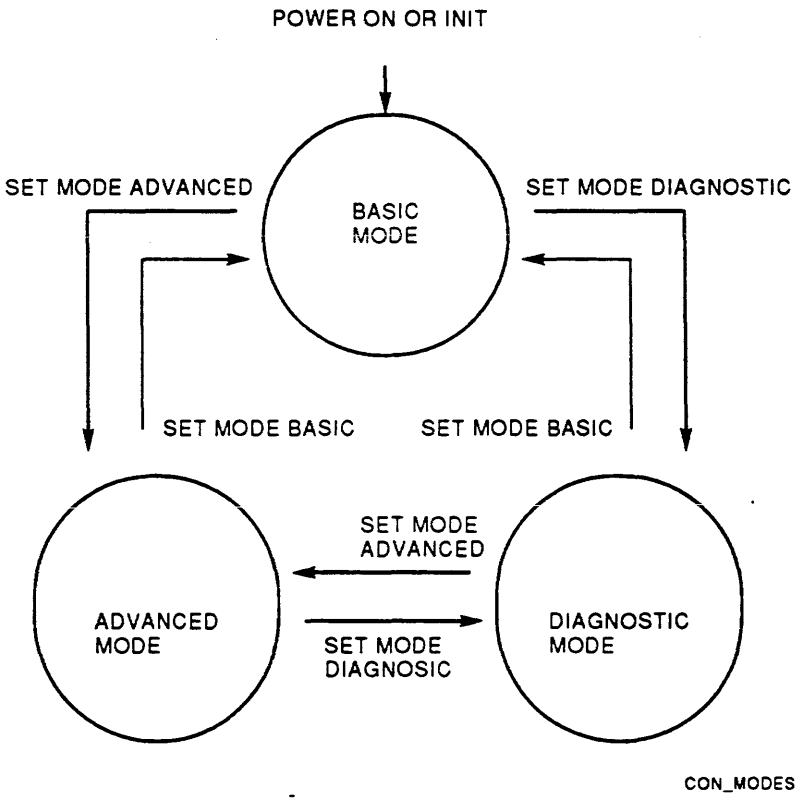
```
>>> kill 1a
>>> ps
ID       PCB      Pri CPU Time Affinity CPU  Program    State
-------- -------- --- -------- -------- --- ---------- ----------------------
0000001d 0013ad20 3          0 00000001 0           ps running
00000019 00140080 2       1000 00000001 0        sh_bg waiting on child_done
00000015 00143940 2          0 00000001 0   mopdl_exa0 waiting on mop_exa0_dlw
00000014 001480c0 5          0 00000001 0 process_exa0 waiting on exa0_isr
00000011 0014f7a0 1          0 00000001 0     pua_poll ready
00000010 00162b40 6       1000 00000001 0  pua_receive waiting on puu_receive
0000000e 00151f20 3       9000 00000001 0           sh ready
0000000b 001598c0 5          0 00000001 0    mscp_poll waiting on tqe
00000007 00161160 5          0 00000001 0     entry_00 waiting on entry_00
00000004 001662c0 2          0 ffffffff 0   dead_eater waiting on dead_pcb
00000003 00167d20 7      19000 ffffffff 0        timer waiting on timer
00000002 0016f2a0 6          0 ffffffff 0   tt_control waiting on tt_control
00000001 000777b0 0    1493032 00000001 0         idle ready
>>>
```

The console can move from CPU to CPU when the operator uses the SET CPU *n* command. The shell process runs on the primary processor. Secondary CPUs run their console but do not implement the shell process as they are not parsing commands.

Using the Console

The console operates in one of three modes: basic, advanced, or diagnostic. When the system is initialized, the console starts up in basic mode. The operator can change modes as shown in Figure 3–1.

Figure 3–1 Console Operating Modes



NOTE

Invoke the HELP command in each mode to obtain a list of commands that are valid in that mode.

Environmental Variables

Environmental variables equate a name with a value and are used in the console operating environment. There are two types of environmental variables:

- User defined variables (user selects name, assigns value)
  - — Memory variable
  - — EEPROM variable (-nv)

- System defined variables
  - — LASER-specific
  - — Alpha AXP System Reference Manual (SRM)
  - — Diagnostic

LASER-Specific Environmental Variables

The LASER-specific group of system-defined environmental variables is shown in Table 3–1 and are manipulated as shown in Example 3–4.

Table 3–1 LASER-Specific Environmental Variables

LASER-Specific Variable	Values	Definition
baud	9600 (default) 300 600 1200 2400 4800	Baud rate of the console terminal.
clock	<value> ns	The clock rate of processor *x*.
cpu	<value>	The LSB node number of the primary processor.
cpu_enabled	<mask>	Bitmask indicating which processors are enabled to run.
cpu_primary	<mask>	Bitmask indicating which processors can become the primary processor.
interleave	Default None <explicit>	Memory interleave specification.
pal	<revision>	The version of PALcode currently in use. Used by DEC 7000-600, not used by VAX 7000-600
pal_flags	<mask>	Mask containing PALcode build flags. Used by DEC 7000-600, not used by VAX 7000-600
version	<value>	The version of LASER console code currently executing.
update	OFF ON	Allows the Flash ROMs and EEPROM to be written.

The SET, CLEAR, CREATE, and SHOW commands are defined here:

- **CREATE[<env_var>.....]** creates the variable.
- **CLEAR [<env_var>.....]** erases the variable.
- **SHOW [<env_var>.....]** displays the current setting of a variable.
- **SET [-d] [<env_var>.....] [<value>]** where:
  - -d sets the variable to its original value.
  - <env_var> is the environmental variable or list of variables.
  - <value> is the value to be assigned to a variable.

Environmental variables are manipulated as shown in Example 3–4.

Example 3–4 Using LASER-Specific Environmental Variables

```
>>>set baud 9600      ❶
>>>show baud
   baud  9600

>>>set -d baud       ❷

>>>create -nv billy      ❸
>>>set billy 5.812
>>>show billy
billy      5.812
>>>clear billy       ❹

>>>create myname 8      ❺
>>>show myname
myname 8

>>>set interleave default  ❻

>>>set update off     ❼

>>>set -nv fred '-fl 0 dua0.0.0.3.0'  ❽
>>>show fred
fred       -f 0 dua0.0.0.3.0
>>>boot fred      ❾
```

❶ Set the operator's console baud rate to 9600.

❷ Set the baud rate to its original default value.

❸ Create a variable called billy in EEPROM and set its value. User-created environmental variables can be up to 128 characters in length.

❹ Erase the variable called billy from EEPROM.

❺ Create a user variable called myname and assign it the value 8.

❻ Set the memory interleave variable to the default condition.

❼ Inhibit updates of EEPROM or FEROM.

❽ Set the nonvolatile boot string called fred.

❾ Boot the system using fred.

Alpha AXP SRM-Defined Environmental Variables

The SRM group of system-defined environmental variables is shown in Table 3–2.

Table 3–2 SRM Environmental Variables

SRM Variable	Values	Definition
auto_action	HALT BOOT RESTART	Specifies action the console should take following an error halt or power failure.
bootdef_dev	<device>	Device specification from which to boot when no path is specified by a BOOT command.
booted_dev	<device>	Device specification from which the last system boot was attempted.
boot_file	<filename>	A file name to be associated with the next system bootstrap.
boot_osflags	<value>	Additonal default parameters to be passed to system software during the next system bootstrap.
booted_osflags	<value>	The parameters passed to the system software during the previous system bootstrap.
boot_reset	ON OFF	Causes the console to perform a system reset prior to booting the system.
dump_dev	<device>	Device specification to which operating system dumps should be written.
enable_audit	OFF ON	Indicates whether audit-trail messages are to be displayed during bootstrap.
license	MU SU	Indicates whether this system is licensed for multiple users or a single user.
char_set	0	Indicates that ISO-LATIN-1 characters are supported.
language	<value>	Contains a hexadecimal code indicating the language of console terminal output.
tty_dev	0	Indicates the console terminal unit number. Not used by VAX 7000-600.

Example 3–5 SHOW Variables

```
>>>show *
envar
adv_diag              off
auto_action           HALT
baud                  9600
boot_reset            ON
char_set              0
clock_0               12.0 ns
cpu                   0
cpu_enabled           ff
cpu_primary           ff
d_bell                off
d_cleanup             on
d_complete            off
d_eop                 off
d_group               field
d_harderr             halt
d_loghard             on
d_logsoft             off
d_oper                on
d_passes              1
d_quick               off
d_report              summary
d_softerr             continue
d_startup             off
d_status              off
d_trace               off
def_term              local
enable_audit          ON
envar
exa0_driver_flags     0
exa0_loop_count       0
exa0_loop_inc         0
exa0_loop_patt        ffffffff
exa0_loop_size        2e
exa0_lp_msg_node      800
exa0_msg_buf_size     5ee
exa0_msg_mod          0
exa0_msg_rem          0
exa0_version          1
exdep_location        0
exdep_size            4
exdep_space           pmem
exdep_type            2
interleave            default
language              English
language_code         3
license               MU
prompt                ella_libkind>
remote                0
repeatcount           1
tty_dev
update                off
version               X2.2-1338 Feb  5 1992 10:08:45
xctintrate            500000
```

Diagnostic Environmental Variables

The diagnostic group of system-defined environmental variables is shown in Table 3–3 and are manipulated as shown in Example 3–6.

Table 3–3 Diagnostic Environmental Variables

Diagnostic Variable	Values	Definition
D_BELL	OFF ON	Bell on a detected error
D_CLEANUP	ON OFF	Should cleanup code be executed after test completion?
D_EOP	ON OFF	Print end-of-pass messages?
D_GROUP	FIELD MFG <section>	Execute which diagnostic section?
D_HARDERR[1]	CONTINUE HALT LOOP	What action is taken following hard error detection?
D_LOGERR	ON OFF	Log error to the console error log?
D_OPER	OFF ON	Operator present?
D_PASSES	1 <pass_count>	Run how many passes of a diagnostic module?
D_QUICK	OFF ON	Normal or quick verify mode?
D_REPORT[1]	SUMMARY FULL OFF	Which level of information provided by diagnostic module status and error reports?
D_SOFTERR[1]	CONTINUE HALT LOOP	What action is taken following soft error detection?
D_STATUS	OFF ON	Display status message?
D_TRACE	OFF ON	Display test progress messages?

[1]Command is effective in basic console mode while using the TEST command.

The DSE runs diagnostic and exerciser programs from the console diagnostic mode. The diagnostic environmental variable can be used to modify operations of these programs.

Example 3–6 Using Diagnostic Environment Variables

```
>>>set mode diag    ❶

>>>show d_*    ❷
d_bell                  off
d_cleanup               on
d_complete              off
d_eop                   off
d_group                 field
d_harderr               halt
d_loghard               on
d_logsoft               off
d_oper                  on
d_passes                1
d_quick                 off
d_report                full
d_softerr               continue
d_startup               off
d_status                on
d_trace                 on
def_term                local

>>>show d_bell    ❸
d_bell     off

>>>set d_bell on    ❹

>>>show d_bell    ❺
d_bell     on

>>>set d_trace off    ❻

>>>set d_passes 5    ❼
```

❶ Put console in diagnostic mode.

❷ Show status of all diagnostic variables.

❸ Show status of d_bell variable.

❶ Bell on if error is detected.

❺ Show status of d_bell variable.

❻ Do not display test progress trace.

❼ Default run will be 5 passes of the test.

Common Console Commands

Some commonly used console commands are described in this section. Remember that all commands are executable files.

SHOW Command

The SHOW command can be used to show the system hardware.

- SHOW CONFIGURATION in Example 3–7
- SHOW MEMORY in Example 3–8
- SHOW DEVICE in Example 3–9
- SHOW POWER [-h, -s] [main, left, right] -h=history and -s=currrent status

Example 3–7 SHOW CONFIGURATION Command

```
>>>show config -v    ❶
Name      Type     Rev       Mnemonic   IntVec
LSB
0+     KA7AA     (8002)  0000  ka7aa0     000        ❷
7+     MS7AA     (4000)  0000  ms7aa0     4000000    ❸
8+     IOP       (2000)  0002  iop0       150        ❹

C0 XMI                           xmi0       ❺
2+     KDM70     (0C22)  1E11  kdm700     170 ❻
8+     DWLMA     (102A)  0003  dwlma0     180 ❼
E+     DEMNA     (0C03)  0608  demna0     190 ❽
```

❶ Show configuration (from console configuration file) with vectors.

❷ LNP processor module is at node 0.

❸ LASER memory module, LMEM, is at node 7.

❹ I/O Port module, IOP, is at node 8.

❺ IOP hose 0 (CO) is attached to XMI0.

❻ KDM700 is in Slot 2 of XMI0.

❼ DWLMA0 (LAMB) module is in Slot 8 of XMI0.

❽ DEMNA0 module is in Slot E of XMI0.

Example 3–8 SHOW MEMORY Command

```
>>> show mem
Set   Node   Size   Base Addr   Intlv   Position
---   ----   ----   ---------   -----   --------
A      6     64Mb   00000000   2-Way      0
A      7     64Mb   00000000   2-Way      1
```

Example 3–9 SHOW DEVICE Command

```
>>> show device
dua0.0.0.3.0        duA0                          RF73
dua1.1.0.3.0        duA100                        RF73
dua2.2.0.3.0        duA200                        RF73
dua3.3.0.3.0        duA300                        RF73
dua4.4.0.3.0        duA400                        RF73
dua5.5.0.3.0        duA500                        RF73
>>>
```

Example 3–10 SHOW POWER Command

```
>>>show power
Cabinet:      Main:        Regulator:     A             B             C
-------------------------------------------------------------------------
          Primary Micro Firmware Rev:   2.0           2.0           2.0
        Secondary MIcro Firmware Rev:   2.0           2.0           2.0
                  Power Supply State:   NORMAL        NORMAL        NORMAL
             AC Line Voltage (V RMS):   113.71        114.35        115.93
               DC Bulk Voltage (VDC):   227.02        227.02        227.02
            48V DC Bus Voltage (VDC):   47.57         47.57         47.57
            48V DC Bus Current (ADC):   30.17         29.68         29.58
      48V Battery Pack Voltage (VDC):   50.85         50.75         47.91
      24V Battery Pack Voltage (VDC):   25.56         25.56         23.95
   Battery Pack Charge Current (IDC):   2.91          2.90          0
     Ambient Temperature (Degrees C):   26.22         24.80         24.75
                Elapsed Time (Hours):   290.00        290.00        290.00
Remaining Battery Capacity (Minutes):   8.00          8.00          8.00
     Battery Cutoff Counter (Cycles):   0             1.00          1.00
               Battery Configuration:   4 Batteries   4 Batteries   4 Batteries
                     Heatsink Status:   NORMAL        NORMAL        NORMAL
                  Battery Pack Status:   CHARGING      CHARGING      DISCHG'G
                 Last UPS Test Status:   PASSED        PASSED        PASSED
LDC POWER Status      : 0
PIU Primary Status    : 0
PIU Secondary Status  : 0
>>>
```

DEPOSIT and EXAMINE Commands

The DEPOSIT and EXAMINE commands use the qualifiers listed and defined in Table 3–4 and the parameters listed and defined in Table 3–5.

Table 3–4 DEPOSIT and EXAMINE Command Qualifiers

Qualifier	Definition
-b, -w, -l, -q, -o, -h	Data type is byte, word, longword, quadword, octaword, hexword
-n <value>	Specifies the number of consecutive locations to write or read
-s <value>	Specifies address increment size (registers spacing is 40[hex])
-d	Tells the console to decode the instruction data

Table 3–5 DEPOSIT and EXAMINE Command Parameters

Parameter	Definition
<address>	The hexadecimal address or one of the following: * previous address + previous address increment - previous address decrement @ previous address deferred
<device>	One of the following console-defined devices: PMEM: VMEM: GPR: FPR: PT: IPR: Device mnemonic as seen in SHOW CONFIG
<mnemonic>	Any of the following SRM defined keywords: RO through R15 (R31 for DEC 7000-600) F0 through F31 PT0 through PT31 PC SP PS IPR n
<data>	Hexadecimal data to be deposited. Greater than longword in longword spaced increments.

Some uses of the DEPOSIT command are shown in Example 3–11

Example 3–11 DEPOSIT Command

```
>>>deposit -l pmem:0006000 1000151588
Deposit data too large for specified data type
>>>deposit -l pmem:00006000 10001515
>>>deposit -w 00006000 1515
>>>deposit -b 00006000 15
>>>d -l 00004000 5A5A5A5A
>>>DEP  -L KDM700:4 0
>>>DEP  -L XMI0:61880004 5800191F
>>>DEP  -Q pmem:4000 61880004 5800191F        ! 61880004 is high longword
```

Some uses of the EXAMINE command are shown in Example 3–12

Example 3–12 EXAMINE Command

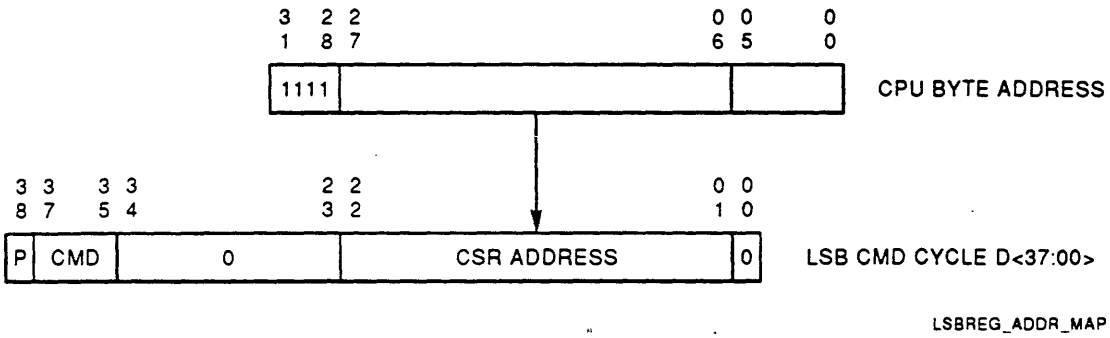
```
>>>examine -l -n 2 pmem:40000
00040000 425B0413
00040004 61E57733
00040008 5A443387
>>>EXAMINE pmem:fa00000
0fa00000 00022000
>>>EXAMINE XMI0:61880000
xmi0: 61880000 0000102A
>>>exam -w pmem:6000
00006000 5511
>>>e r5
gpr: 00000005  (R5)  00115534
>>>e -n 2 r5
gpr: 00000005  (R5)  00115534
gpr: 00000006  (R6)  55238910
gpr: 00000007  (R7)  66718927
>>>E -L -N 4 demna0:8
```

VAX 7000 and 10000-600 System Register Addresses on LSB

VAX 7000/10000-600 system LSB nodes have control and status registers that can be accessed using the system console.

VAX 7000-600 system physical addresses that have bits <31:28> set access LSB register space (example: F880 0004): When a CSR read or write transaction occurs , address bits <31:06> will be mapped to LSB D<34:00> as shown in Figure 3–2.

Figure 3–2 LSB Register Address Mapping



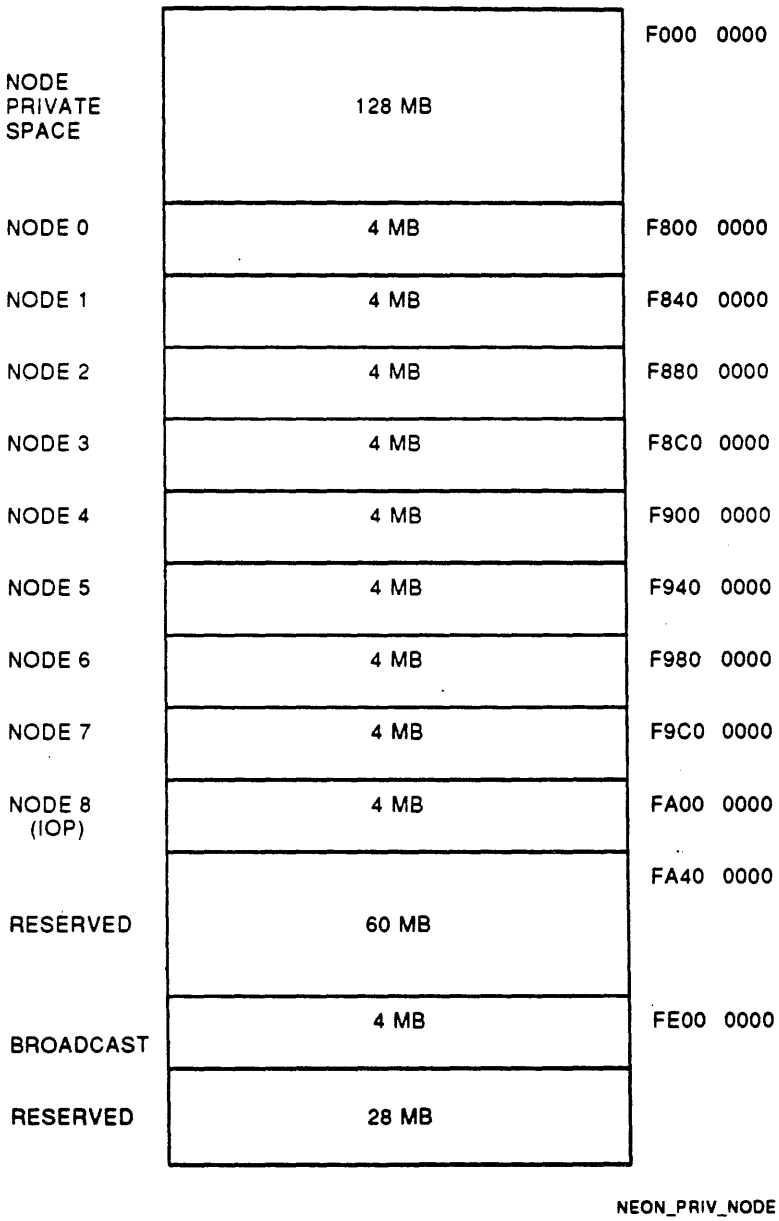
The base address for each node and the broadcast space address is listed in Table 3–6.

Table 3–6 VAX 7000-600 LSB Node Base Addresses (BB and BSB)

Node	Module/Node	VAX 7000-600 Base Physical Address	LSB Base Bits D<22:00>
0	CPU 0	F800 0000	40 0000
1	CPU/MEM 1	F840 0000	42 0000
2	CPU/MEM 2	F880 0000	44 0000
3	CPU/MEM 3	F8C0 0000	46 0000
4	CPU/MEM 4	F900 0000	48 0000
5	CPU/MEM 5	F940 0000	4A 0000
6	CPU/MEM 6	F980 0000	4C 0000
7	CPU/MEM 7	F9C0 0000	4E 0000
8	IOP 8	FA00 0000	50 0000
Broadcast	BSB	FE00 0000	70 0000

The address ranges allocated for node private space and node LSB CSRs is shown in Figure 3–3.

Figure 3–3 VAX 7000 and 10000 Node Private Space and CSR Space



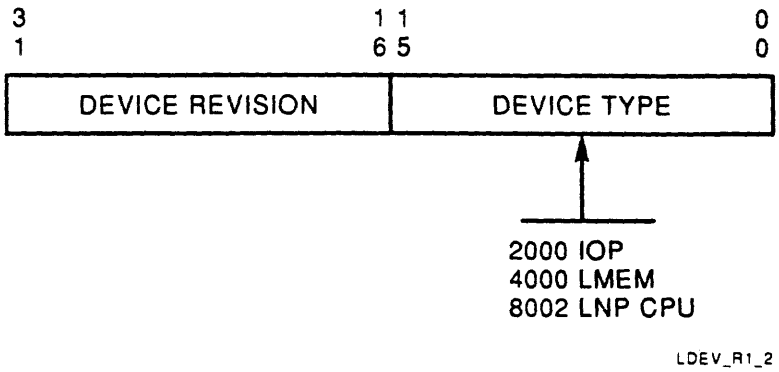
All LSB node CSRs are aligned on natural 64-byte boundaries. The LSB protocol requires the following registers to be maintained by some combination of CPU, memory, or I/O nodes on the LSB as shown in Table 3–7. Nodes can maintain registers beyond this limited set.

Table 3–7 LSB Required Control and and Status Registers

Byte Offset	Mnemonic	Register Name	Register Use
BB+000	LDEV	LASER Device Register	CPU, MEM, I/O
BB+040	LBER	LASER Bus Error Register	CPU, MEM, I/O
BB+080	LCNF	LASER Configuration Register	CPU, MEM, I/O
BB+0C0	LIBR	LASER Information Base Repair Register	MEM, I/O
BB+200	LMMR0	LASER Memory Mapping Register 0	CPU, I/O
BB+240	LMMR1	LASER Memory Mapping Register 1	CPU, I/O
BB+280	LMMR2	LASER Memory Mapping Register 2	CPU, I/O
BB+2C0	LMMR3	LASER Memory Mapping Register 3	CPU, I/O
BB+300	LMMR4	LASER Memory Mapping Register 4	CPU, I/O
BB+340	LMMR5	LASER Memory Mapping Register 5	CPU, I/O
BB+380	LMMR6	LASER Memory Mapping Register 6	CPU, I/O
BB+3C0	LMMR7	LASER Memory Mapping Register 7	CPU, I/O
BB+600	LBESR0	LASER Bus Error Syndrome Register 0	CPU, MEM, I/O
BB+640	LBESR1	LASER Bus Error Syndrome Register 1	CPU, MEM, I/O
BB+680	LBESR2	LASER Bus Error Syndrome Register 2	CPU, MEM, I/O
BB+6C0	LBESR3	LASER Bus Error Syndrome Register 3	CPU, MEM, I/O
BB+700	LBECR0	LASER Bus Error Command Register 0	CPU, MEM, I/O
BB+740	LBECR1	LASER Bus Error Command Register 1	CPU, MEM, I/O
BB+A00	LILID0	Interrupt Level 0 Ident Register.	I/O
BB+A40	LILID1	Interrupt Level 1 Ident Register	I/O
BB+A80	LILID2	Interrupt Level 2 Ident Register	I/O
BB+AC0	LILID3	Interrupt Level 3 Ident Register	I/O
BB+B00	LCPUMASK	CPU Interrupt Mask Register	I/O
BB+C00	LMBPR0	Mailbox Pointer Register 0	I/O
BB+C40	LMBPR1	Mailbox Pointer Register 1	I/O
BB+C80	LMBPR2	Mailbox Pointer Register 2	I/O
BB+CC0	LMBPR3	Mailbox Pointer Register 3	I/O
BSB+000	LIOINTR	I/O Interrupt Register	CPU
BSB+040	LIPINTR	IP Interrupt Register	CPU

The LDEV Register format is show in Figure 3–4.

Figure 3–4 LDEV Register



LSB Address Exercise

LSB Address Problem 1

You want to use the console to examine the LBER register in the VAX 7000-600 processor in Slot 1, shown in Figure 3–5. Calculate the hexadecimal address of the LBER register.

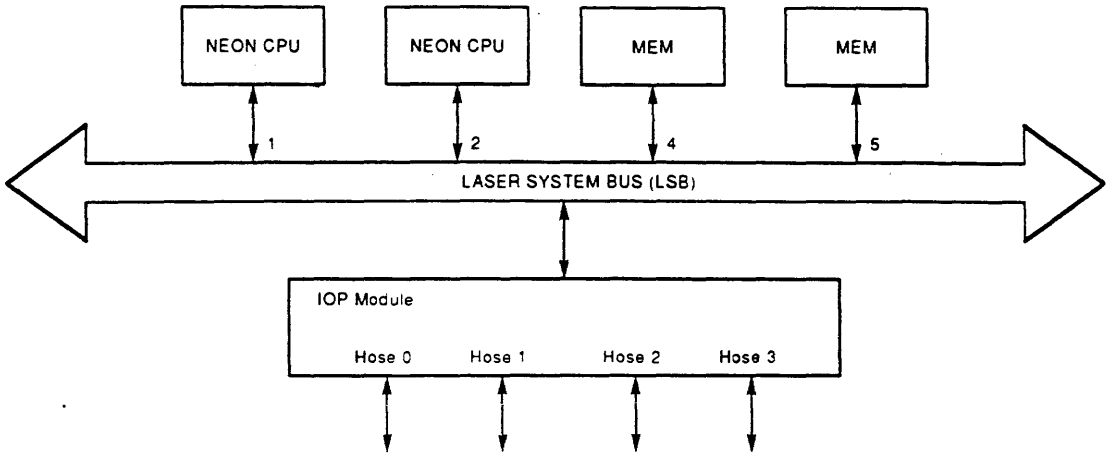
Node base address (BB) for node ____ is _____.

Offset for the LBER register is _____.

Address for node ____ LBER register is _____.

Console command: _____.

Figure 3–5 LSB Address Problem 1, 2, and 3 Diagram



LSB Address Problem 2

Using the diagram in Figure 3–5, calculate the hexadecimal address of Mailbox Pointer Register 0 of the IOP node shown in Figure 3–5.

Node base address (BB) for node ____ is _____.

Offset for LMBPR0 register is _____.

Address for node ____ LMBPR0 register is _____.

Console command: _____.

LSB Address Problem 3

The IOP module in Figure 3–5 sent an interrupt at IPL 15 to CPU1. Calculate the hexadecimal address used by the program to read the vector from the LILIDx register.
Hint: IPL17 is associated with LILID3.

Node base address (BB) for node ____ is _____.

Offset for the LILIDx register is _____.

Address for node ____ LILIDx register is _____.

Console command: _____.

UPDATE Command

The UPDATE command is used to copy the contents of the EEPROM and FEROMs from the primary CPU to a secondary CPU. The environmental variable UPDATE must be set to the on state to allow writing to the EEPROM.

Example 3–13 UPDATE Command

```
>>>show cpu
cpu        0
>>>show update
update       off
>>>set update on
>>>update f 3    ❶
>>>update e 3    ❷
>>>update 3      ❸
>>>set update off  ❹
>>>set cpu 3     ❺
>>>set update off  ❻
>>>set cpu 0     ❼
```

❶ Update the FEROMs on LNP at node 3 from LNP at node 0.

❷ Update the EEPROM on LNP at node 3.

❸ Update the FEROMs and the EEPROM on LNP at node 3.

❹ Set node 0 EEPROM update variable status to off.

❺ Change the primary processor to the LNP at node 3.

❻ Set node 3 EEPROM update variable status to off.

❼ Change the primary processor back to the LNP at node 0.

INITIALIZE Command

The INITIALIZE [<device>] command causes the console to issue an INIT to the device name or LSB node ID. The device name to use can be obtained using the SHOW CONFIG command.

If no device is specified a system reset is performed. This command is used to reset the system to its initialized state at power on.

Miscellaneous Console Commands

The HELP command is used to list the available commands.

The HALT command causes the console to issue a NODE HALT to the device name or LSB node ID. Use the SHOW CONFIG command to get the device-name or LSB node ID.

The START command causes the console to transfer system control to the physical address specified. If no address is specified, the console will attempt an operating system restart.

The CONTINUE command causes the console to restore the saved state of the machine and pass control to the saved PC address.

The CRASH command causes the console to attempt to restart the operating system and passes a CRASH HALT code to the system software. System software then performs a crash dump.

Console EEPROM

The 8 kB EEPROM is organized into 13 sections at present as shown in Example 3–14.

Example 3–14 Processor EEPROM Areas

```
>>>show eeprom areas

EEPROM Address: 001c1fe0
Header - Checksum: 28B87EE9  Length: 0054  Version: 1

Area  Offset  Length     Cksum     Key
  0    0064      76  FFD8BFFF    5.0 product
  1    00B0      28  1BFFFFE1    1.0 console parameters
  2    00CC     444  F217FFFF    2.0 boot specs
  3    0288     660  52A83C26    4.0 environment
  4    051C    1036  1D3230E5   18.0 powerup script
  5    0928    1036  7979F031   18.1 test script
  6    0D34      44  B1FFFE1F   32.0 fru descriptor
  7    0D60      24  8CD7B546   33.0 system serial number
  8    0D78      76  FFD17FFF   34.0 system configuration
  9    0DC4     296  FFFFFFFF   48.0 diagnostic log
 10    0EEC     656  FFFFFFFF   49.0 symptom log
 11    117C      76  FFFFFFFF   50.0 halt code counters
 12    11C8      76  FFCCBFFF   53.0 service call
 13    1214     856  FFC8CFFF   54.0 module history
>>>
```

The EEPROM commands SET, SHOW, BUILD, CLEAR, and DUMP are supported in BASIC mode.

Use the following commands to see which parts of the EEPROM can be examined:

SHOW EEPROM [help-key]

Supported SHOW EEPROM commands include:

```
>>> SHOW EEPROM AREAS [module] *
>>> SHOW EEPROM CONSOLE
>>> SHOW EEPROM DIAG_TDD [module] *
>>> SHOW EEPROM DIAG_SDD [module] *
>>> SHOW EEPROM ENVIRONMENT
>>> SHOW EEPROM FIELD [module] *
>>> SHOW EEPROM HEADER [module] *
>>> SHOW EEPROM MANUFACTURING [module] *
>>> SHOW EEPROM REPAIR [module] *
>>> SHOW EEPROM SERIAL
>>> SHOW EEPROM SYMPTOM [module] *
```

NOTE

Commands with asterisks next to them are supported for both CPU and serial (I/O and memory) EEPROMs. Use the SHOW CONFIG command to get the module name.

Supported SET EEPROM commands include:

```
>>> SET EEPROM CONSOLE
>>> SET EEPROM DIAG_TDD [module] *
>>> SET EEPROM DIAG_SDD [module] *
>>> SET EEPROM ENVIRONMENT
>>> SET EEPROM FIELD [module] *
>>> SET EEPROM MANUFACTURING [module] *
>>> SET EEPROM REPAIR [module] *
>>> SET EEPROM SERIAL
>>> SET EEPROM SYMPTOM [module] *
```

Example 3–15 SET and SHOW EEPROM Commands

```
>>>set eeprom <help-key>
 environment field     manufacturing
 script  serial      symptom
 diag_tdd diag_sdd

>>>show eeprom <help-key>
 areas       diag_sdd diag_tdd environment
        field        manufacturing         repair
        script       serial  symptom

>>>set eeprom serial
System Serial Number> BXB000001
>>>

>>>set eeprom environment ?
    auto_action  baud  baud_local baud_ps       baud_rd
    bootcmd_dev  boot_file boot_osflags boot_reset    char_set
    cpu_enabled   cpu_primary dump_dev enable_audit  interleave
    license  language page  password      prompt
    repeatcount         term     tty_dev  update
>>>set eeprom env baud 9600

>>>set eeprom script power
Script>SET AUTO_ACTION OFF
Script>^Z

>>>show eeprom script power

Powerup Script:

 SET AUTO_ACTION OFF

>>>
```

NOTE

Do not put INIT or BOOT commands in power or test scripts. Present consoles are unable to recover from an unending loop.

The following commands are supported when manipulating or displaying the contents of serial EEPROMs. The syntax for manipulating and displaying serial EEPROM areas is as follows:

>>>SET EEPROM DIAG_TTD
>>>SHOW EEPROM DIAG_TTD
>>>SET EEPROM REPAIR
>>>SHOW EEPROM REPAIR

The current system configuration can be saved in EEPROM, and later the last saved configuration can be displayed.

>>> SET CONFIG—Records the current system configuration in the EEPROM.

>>> SHOW CONFIG -S—Displays the last saved configuration.

When the console notices that the EEPROM has been corrupted, it displays a message on the terminal: NO EEPROM. A new image can be created using the BUILD EEPROM command as shown in Example 3–16.

Example 3–16 BUILD EEPROM Command

```
>>>build eeprom
Creating new EEPROM Image
System Serial Number> BXB00000001
Module Serial Number> LNP01
Module Type> LNP
Module Revision> 1
>>>
```

The CLEAR EEPROM command can be used to clear the value of an EEPROM parameter as shown in Example 3–17.

Example 3–17 CLEAR EEPROM Command

```
>>>clear eeprom ?
 diag   environment log      repair
 symptom
>>>
```

The DUMP EEPROM command can be used in advanced mode to get a printout of the EEPROM contents as shown in Example 3–18.

Example 3–18 DUMP EEPROM Command

```
>>>set mode advance
>>>dump eeprom
0/ AA5500FF 5500FFAA 00FFAA55 FFAA5500 28B87EE9 00010000 00000054 00000064
20/ 000000B0 000000CC 00000288 0000051C 00000928 00000D34 00000D60 00000D78
40/ 00000DC4 00000EEC 0000117C 000011C8 00001214 00000000 00000000 00000000
60/ 00000000 FFD8BFFF 00000005 0000004C 00000000 00000000 00000000 00000000
 Zeros
a0/ 00000000 00000000 00000000 00000000 1BFFFFE1 00000001 0000001C 00000000
c0/ 00000000 00000000 0000003C F217FFFF 00000002 000001BC 00000000 00000000
 Zeros
280/ 00000000 00000000 ABC34730 00000004 00000294 00000218 00000012 00000088
2a0/ 00000097 000000A8 000000BB 000000C5 000000D4 000000E5 000000F3 00000101
2c0/ 00000113 00C00011D 00000127 00000132 00000143 00000166 000001BF 000001F4
2e0/ 0000020A 00000000 00000000 00000000 00000000 00000000 00000000 00000000
300/ 00000000 C0000000 0000C000 00000000 00000000 00000000 00000000 746F6F62
320/ 7365725F 4F3D7465 6C004646 75676E61 3D656761 6C676E45 00687369 65746E69
340/ 61656C72 643D6576 75616665 6200746C 3D647561 30303639 6F6F6200 69665F74
360/ 783D656C 0030696D 5F757063 6D697270 3D797261 30696D78 66656400 7265745F
380/ 6D783D6D 64003069 5F706D75 3D766564 30696D78 616E6500 5F656C62 69647561
3a0/ 6D783D74 6D003069 3D65646F 30696D78 67617000 6D783D65 70003069 706D6F72
3c0/ 73633D74 7561003E 615F6F74 6F697463 41483D6E 6300544C 655F7570 6C62616E
3e0/ 313D6465 6F6F6200 736F5F74 67616C66 2C303D73 30303031 6F620030 6564746F
400/ 65645F66 6D783D76 62003069 5F746F6F 3D766564 30696D78 70786500 65746365
420/ 6E655F64 3D797274 54494E49 62002020 5F746F6F 3D766564 30696D78 70786500
440/ 65746365 6E655F64 3D797274 54494E49 6200315F 5F746F6F 6C66736F 3D736761
460/ 00302C30 65707865 64657463 746E655F 493D7972 2054494E 6F620020 645F746F
480/ 783D7665 0030696D 65707865 64657463 746E655F 493D7972 5F54494E 6F620031
4c0/ 746F6F62 5F666564 3D766564 35616B64 6F6F6200 66656474 7665645F 696D783D
4e0/ 6F620030 645F746F 783D7665 0030696D 746F6F62 7665645F 696D783D 6F620030
500/ 645F746F 783D7665 0030696D 783D7972 0030696D 003D7665 00000000 1D3230E5
520/ 00000012 0000040C 746F6F00 20662D20 6B642030 00003061 00000000 00000000
 Zeros
920/ 00000000 00000000 7979F031 00010012 0000040C 776F6873 70656520 206D6F72
940/ 69726373 69007470 0074696E 00000000 00000000 00000000 00000000 00000000
 Zeros
d20/ 00000000 00000000 00000000 00000000 00000000 B1FFFE1F 00000020 0000002C
d40/ 00003530 00000000 00000000 00003250 00000000 00000000 00000000 00000031
d60/ CCD7A746 00000021 00000018 59425552 00003831 00000000 FFD17FFF 00000022
d80/ 0000004C 00000000 00000000 00000000 00000000 00000000 00000000 00000000
 Zeros
dc0/ 00000000 FFFFFFFF 00000030 00000128 00000000 00000000 00000000 00000000
 Zeros
ee0/ 00000000 00000000 00000000 FFFFFFFF 00000031 00000290 00000000 00000000
 Zeros
1160/ 00000000 00000000 00000000 00000000 00000000 00000000 00000000 FFFFFFFF
1180/ 00000032 0000004C 00000000 00000000 00000000 00000000 00000000 00000000
 Zeros
11c0/ 00000000 00000000 FFCCBFFF 00000035 0000004C 00000000 00000000 00000000
 Zeros
1200/ 00000000 00000000 00000000 00000000 00000000 FFC8CFFF 00000036 00000358
 Zeros
1fe0/ 00000000 00000000 00000000 00000000 0055AAFF 55AAFF00 AAFF0055 FF0055AA
```

Example 3–19 EEPROM Command

```
>>>show eeprom areas

EEPROM Address: 001c1fe0
Header - Checksum: 28B87EE9  Length: 0054  Version: 1

Area  Offset  Length     Cksum     Key
0     0064      76  FFD8BFFF    5.0 product
1     00B0      28  1BFFFFE1    1.0 console parameters
2     00CC     444  F217FFFF    2.0 boot specs
3     0288     660  52A83C26    4.0 environment
4     051C    1036  1D3230E5   18.0 powerup script
5     0928    1036  7979F031   18.1 test script
6     0D34      44  B1FFFE1F   32.0 fru descriptor
7     0D60      24  8CD7B546   33.0 system serial number
8     0D78      76  FFD17FFF   34.0 system configuration
9     0DC4     296  FFFFFFFF   48.0 diagnostic log
10     0EEC     656  FFFFFFFF    49.0 symptom log
11     117C      76  FFFFFFFF    50.0 halt code counters
12     11C8      76  FFCCBFFF    53.0 service call
13     1214     856  FFC8CFFF    54.0 module history


>>>show eeprom console
Console Parameters are:
Version      = 0
Flags        = 00000000
Flags1       = 00000000
CPU_TST TMO = 60
cs>show eeprom diag_tdd
eeprom                console


>>>show eeprom env
boot_reset=OFF
language=English
interleave=default
baud=9600
boot_file=xmi0
cpu_primary=xmi0
def_term=xmi0
dump_dev=xmi0
enable_audit=xmi0
mode=xmi0
page=xmi0
prompt=cs>
auto_action=HALT
cpu_enabled=1
bootdef_dev=xmi0
boot_osflags=0,0
bill=38a77
boot_dev=xmi0
expected_entry=INIT_1


>>>show eeprom field

System Serial Number = SYSTEM2

Module Serial Number = 05
Module Type = P2
Module Revision = 1
```

Example 3–19 Cont'd on next page

Example 3–19 (Continued) EEPROM Command

```
Diagnostic
Logging is enabled
Logging Control Field value = 0000
Actions logged = 0

Symptom
Logging is enabled
Logging Control Field value = 0000
Actions logged = 0

LARS # =
Message =
No Repair History

>>>show eeprom man

Module Serial Number = 05
Module Type = P2
Module Revision = 1
cs>show eeprom repair
No Repair History

>>>show eeprom ser

System Serial Number SYSTEM2

>>>show eeprom symptom

Symptom
Logging is enabled
Logging Control Field value = 0000
Actions logged = 0
```

Advanced Mode Commands

These are unsupported console commands, and they should be used with the understanding that the results may be unpredictable for future versions of the console.

LASER Console Support Commands

Table 3–8 LASER Console Support Commands

Command	Qualifier	Parameter	Function
cat		file	Copy files to standard output.
chmod	-rwxb	file	Change the mode of a file.
cp		file 1 file 2	Copy one file to another.
dynamic			Show memory state (heap).
echo	-n	arg	Echo command line.
eval		exp	Evaluate an arithmetic expression.
grep	-n	file exp	Search for regular expression.
hd		file	Dump a file using hex radix.
head		file num	Print the beginning of a file.
ls	-l, -a, -la		Directory of a file system – -l, long; -a, all.
more		file num	Print a file in page chunks (use space bar).
ps			Print process statistics.
rm		file	Delete a file.
semaphore			Show semaphores.
shell			Start another shell process.
sleep		time	Suspend execution.
tail		file num	Print the end of a file.
wc	-lwc	file	Count lines, words, characters in file.

LASER Console Operators

LASER console operators are used to modify operation of console commands.

Table 3–9 LASER Console Operators

Command	Qualifier	Parameter	Function
>	Write	<destination	Write output to destination.
>>	Append	>>destination	Append output to destination
<	Read	<source	Read input from source.
|	Pipe	cmd1 |	Pipe output of first command to input of second command.
;	Sequence	cmd1 ; cmd2	Run first command to completion before running second command.
&	Background	cmd&	Run command in background, do not wait for command to complete.
&p	Affinity	&p *m*	Sets the processor affinity mask to allow this process to run on the CPUs defined by mask *m*. Multiple processors can also be specified as a list or range.
()	Grouping		Used to override precedence of pipe, sequence, and background operators.
$string	Environment variable	$boot_device	The string is treated as a legal environment variable and translated.
'xxx'	String with no substitution		The string is passed untouched.
'String'	String with substitution		The string is passed after wildcards and environment variables are expanded.
'cmd'	Command substitution		Treat the string as a command string. Execute it and substitute in the resulting output.

Example 3–20 Advanced Commands

```
>>>ps
   ID        PCB    Pri CPU Time Affinity  CPU   Program    State
-------- -------- --- ------ -------- --- ---------- ----------------
0000000b  0018bbe0   3    405  00000008   3         ps  running on tt_ctrl
0000000a  001868a0   3  35305  00000008   3         sh  ready
00000008  0018d860   5   1247  ffffffff   3   dup_poll  waiting on tqe
00000007  0018f240   5   1249  ffffffff   3  mscp_poll  waiting on tqe
00000006  00196dc0   5   1333  00000008   3   entry_03  waiting on entry_03
00000004  0019a300   2   1233  ffffffff   3 dead_eater  waiting on dead_pcb
00000003  0019bd60   7  11057  ffffffff   3      timer  waiting on timer
00000002  001a1800   6   1054  ffffffff   3 tt_control  waiting on tt_ctrl
00000001  00151f80   0 453525  00000008   3       idle  ready

>>>ls -a
b               boot            build           cat             chmod           clear
cmp             continue        crc             d               dave            decode
deposit         dg_pidlist      dump            dynamic         e               echo
el              eval            ex              exa0            examine
examine_render                  exer            exit            fily            gpr
grep            hd              help            helptxt         init            ipr
kill            leds            ls              memtest         metric          mopdl
moplp           more            mp              net             ni_db_exa0      nl
pmem            ps              psr             psr_render      rm              semaphore
set             sh              shell           sho             show            show_iobq
show_status                     sleep           ss              start           stop
strings.k       tokens.k        tt              tta0            uniq            validate
vmb             vmdm            vmem_render                     wc              xctval
xmi1            xmi3

>>>ls -l | wc > foo

>>>cat foo
     78      390     4197

>>>set xyz dynamic

>>>$xyz
zone        zone      used      used      free      free      util-     high
address     size      blocks    bytes     blocks    bytes     ization   water
--------  --------  --------  --------  --------  --------  --------  ---------
0013A090    524288       198    236224         5    288096     45 %      253856
001BE380   5146624         1        32         1   5146624      0 %           0

>>>
```

Console Firmware Script Facility

The console firmware script facility allows the skilled console operator to create and use executable text files containing console commands.

Scripts are files that can be executed by the shell. There are two types of scripts.

- Built-in scripts
  - — Create a text file with console commands you want to execute.
  - — Register this file as a script by entering the name of the file into the console source file: NEON_SCRIPTS.LIST.
  - — Build the console firmware. The resulting console image, when run, shows the new SCRIPT.FILE.
- I/O redirection scripts (temporary)
  - — Using console commands, redirect output to the file you want to be the script.
  - — Execute the file.

NOTE

I/O redirection files are kept in memory and do not carry through system resets or when the power has been turned on or off.

A script is created and used as shown in Example 3–21.

Example 3–21 Creating and Executing a Script

```
>>>echo 'for i in $1 $2 $3' > my_script
>>>echo 'do' >> my_script
>>>echo '$i' >> my_script
>>>echo 'done'>> my_script
>>>cat my_script    !to see my_script contents
for i in $ 1 $2 $3
do
$i
done
>>>my_script ls ps init
  {the script will then loop through the three console commands}
>>>
```

Console Firmware Event Logger

The console firmware message facility produces two types of ASCII text messages.

- Informational messages—sent only to the event logger
- Error messages—sent to the event logger and the console terminal

The event logger keeps a record of events reported by the console message facility. The event logger maintains an event log that exists only in memory. The logged events are not written to EEPROM or recorded on a storage device.

The event log can be up to 129 lines of 80 characters per line. When the event log becomes full, new lines are logged and the oldest lines will be lost.

The event log is visible in the RAM-based file system as the file *el* shown in Example 3–22.

Example 3–22 Event Log in RAM-based File System

```
>>>ls -l el
rw--    el              0/0                0       el
>>>
```

The contents of the event log can be displayed as shown in Example 3–23.

Example 3–23 Displaying the Event Log

```
>>>cat el
found xna on hose 3, xmi node 1    ! newest
assigned vector is 0180
exa0 node reset failed - retrying...  !   to
exa0 node reset failed.
driver_init_done.     ! oldest

>>>cat el
>>>
```

NOTE

The event log is cleared each time it is read.

The contents of the event log can also be saved into another RAM-based file as shown in Example 3–24.

Example 3–24 Saving the Log to a RAM-Based File

```
>>>cat el > saved_el
>>>cat el      ! Log empty
>>>cat saved_el
found xna on hose 3, xmi node 1    ! newest
assigned vector is 0180
exa0 node reset failed - retrying...  !   to
exa0 node reset failed.
driver_init_done.     ! oldest
>>>
```

A TEE command argument can be used to display and save the event log simultaneously as shown in Example 3–25.

Example 3–25 Using a TEE Command Argument to Display and Save a Log

```
>>>cat el > tee:tt/saved_el  ! tt is console terminal
found xna on hose 3, xmi node 1
assigned vector is 0180
exa0 node reset failed - retrying...
exa0 node reset failed.
driver_init_done.
>>>cat el
>>>cat saved_el
found xna on hose 3, xmi node 1
assigned vector is 0180
exa0 node reset failed - retrying...
exa0 node reset failed.
driver_init_done.
>>>
```

Laser Firmware Update Utility (LFU)

The LFU provides firmware update support for the following devices:

- LNP and LEP Processors
- KDM70, KFMSA, KZMSA
- CIXCD, DEMNA

The LFU finds devices by scanning the configuration tables and not by checking for devices in the system.

The loaded LFU image contains all the new firmware for supported devices. The firmware is not in separate files residing on the distribution media.

The LFU supports updating devices with corrupted firmware. If a device cannot be identified by the LDE (in the configuration table), a device name of UNKNOWN will be assigned. If the user wants to update a device of type UNKNOWN, the LFU will ask for device name and hardware revision. The LFU will then try to update the device.

NOTE

The device type will remain UNKNOWN until the LDE is rebooted.

Table 3–10 LFU Functions

Function Name	Definition
DISPLAY	Displays the system's current configuration table.
EXIT	Returns to the loadable offline operating environment.
LIST	Lists all devices and their update revisions supported by this revision of the loadable image.
MODIFY	Change device-specific attributes for a particular device.
SHOW	Displays the major and minor revisions of the firmware and hardware for the selected device.
UPDATE	Updates the firmware on the device.
VERIFY	Compares the loadable image from the distribution media with what currently resides on the desired module.
HELP or ?	Provides online help.

CAUTION

SAVE OLD MEDIA!

The LFU media will only have the latest version of firmware for each device.

Initialization Sequence Description

When power is turned on or the system is reset, the processors initialize themselves and start the self-test as shown in Figure 3–6.

Figure 3–6 System Initialization Sequence

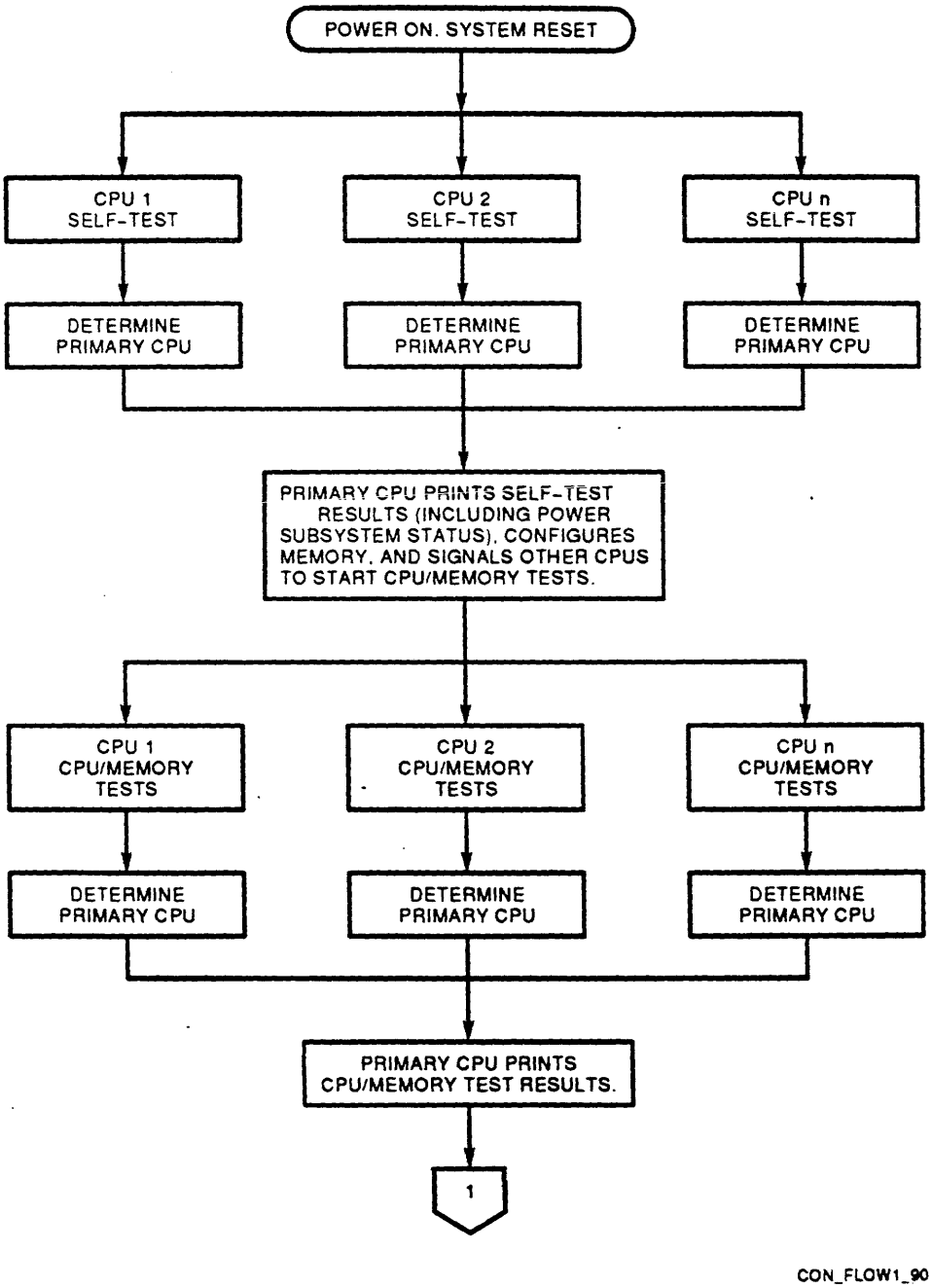


Figure 3–6 Cont'd on next page

Figure 3–6 (Continued) System Initialization Sequence

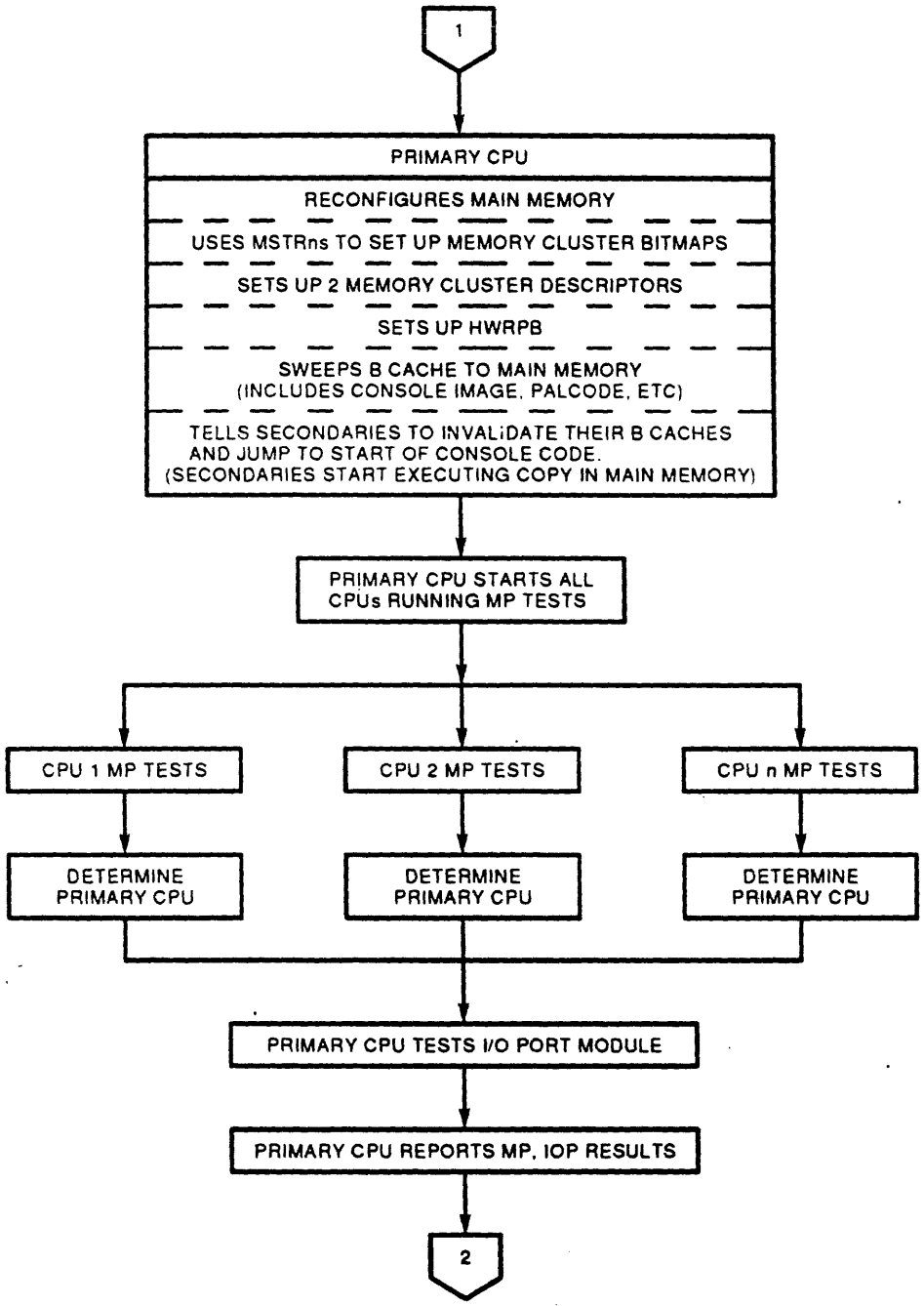
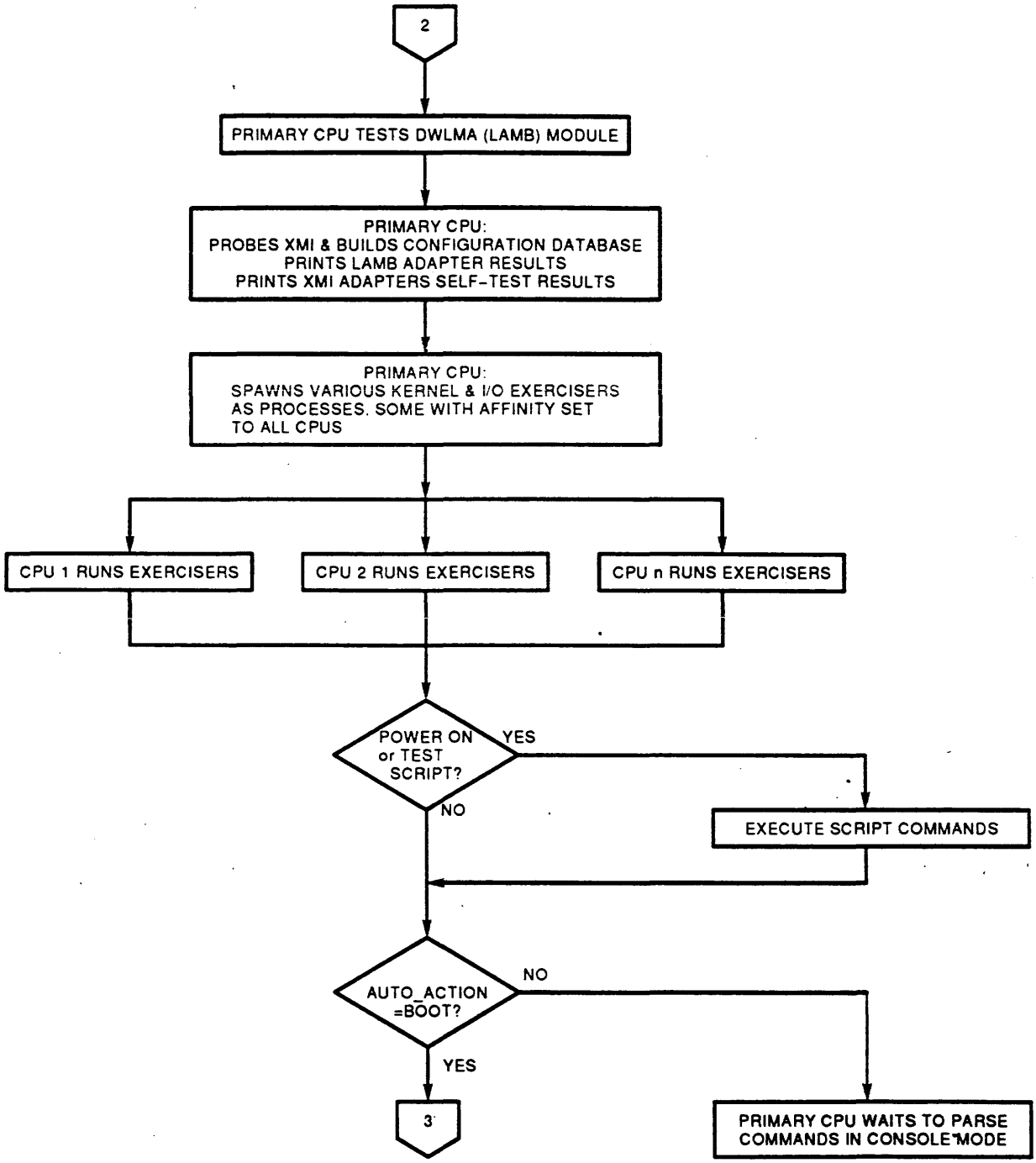


Figure 3–6 Cont'd on next page

Figure 3–6 (Continued) System Initialization Sequence



Serial ROM (SROM is 5th Flash ROM)

The SROM holds the primary startup system console code. It also contains bootstrap primitive information and Flash ROM recovery code (FRRC). When the information in Flash ROMs 1-4, 6, or 7 is corrupted FRRC may be able to recover use of the processor module.

At reset, the SROM data is loaded into virtual instruction cache (VIC). The microprocessor starts executing instructions at address 0.0000.0000, which is a HIT in VIC.

- The VAX 7000/10000 microprocessor has a 2 kB virtual instruction cache.
- The DEC 7000/10000-600 microprocessor has an 8 kB virtual instruction cache.

Running from VIC, the microprocessor initializes some parts of itself then tests and initializes the backup cache.

If successful, the processor executes transition code (final SROM test) which performs a checksum on the the data in the other six Flash ROMs. The six 128 kB GBus Flash ROMs hold the console program and diagnostic firmware. They also hold the PALcode for the DEC 7000/10000-600 systems.

If the checksum is OK, the console loads console code and other data from the other six Flash ROMs into the backup cache. PALcode for Alpha AXP operating systems is loaded into backup cache at this time.

GBus Flash ROMs (6)

After successfully running the SROM code, the processor starts executing GROM self-tests from the backup cache.

The microprocessor runs GROM self-tests, and if they complete successfully, the primary processor configures LSB memory. It creates the hardware restart parameter block (HWRPB), loads a copy of console code, and console data from the other six Flash ROMs into LSB memory. The console cluster memory size is about 1 MB.

The primary processor then starts executing console code from LSB memory. When the system is booted, the operating system starts attached processors running from the copy of console code in LMEM memory.

System Self-Test Description

Processor and memory modules on the LSB perform a self-test when power is turned on or at initialization. IOP modules do not perform a self-test. After successfully completing their self-test, processor modules perform tests on memory, the IOP module, and LAMB module. All power on self-tests will be completed within 30 seconds.

An example of a successful power on self-test sequence is shown in Example 3–26. If a processor fails any phase of self-test, another processor continues and completes the printout. If all processors fail the initial phase, there will be no print out.

Example 3–26 Console Initialization Sequence Printout

```
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #  ❶
                            A   M   .   .   .   .   .   .   P   TYP     ❷
                            o   +   .   .   .   .   .   .   +   ST1     ❸
                            .   .   .   .   .   .   .   .   B   BPD     ❹
                            o   +   .   .   .   .   .   .   +   ST2     ❺
                            .   .   .   .   .   .   .   .   B   BPD     ❻
                            +   +   .   .   .   .   .   .   +   ST3     ❼
                            .   .   .   .   .   .   .   .   B   BPD     ❽
    +   .   .   .   .   .   +   .   .   .   .   .   +   .   C0  XMI +   ❾
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C1          ❿
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C2          ⓫
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C3          ⓬

                            .   A0  .   .   .   .   .   .   .   ILV     ⓭
                            .   64  .   .   .   .   .   .   .   64Mb    ⓮

Firmware Rev = *.*      SROM Rev = V*.*      SYS SN = **********      ⓯
>>>  ⓯
```

❶ Node number header

❷ TYP for device type

- — A for I/O adapter
- — M for memory
- — P for processor
- — T for test device
- — X for unknown device type

❸ ST1—Self-test 1 results

- + for pass
- - for fail
- • for does not apply

❹ BPD—boot processor decision

- B for boot processor
- E for nonboot processor but enabled to be boot processor
- D for disabled from being boot processor

❺ ST2—Self-test 2 results

❻ BPD for boot processor decision (again)

❼ ST3—Self-test 3 results (multiprocessor test)

❽ BPD for boot processor decision (again)

❾ Boot processor tests IOP and LAMB on hose 0, prints results

❿ Boot processor tests IOP and LAMB on hose 1, prints results

⓫ Boot processor tests IOP and LAMB on hose 2, prints results

⓬ Boot processor tests IOP and LAMB on hose 3, prints results

⓭ ILV for memory interleave configuration (memory set assignments)

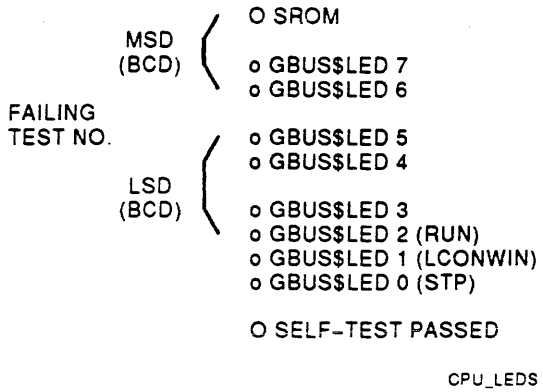
⓮ Memory size

⓯ Console version and system serial number

⓰ Console prompt

LNP processor modules each have a set of status LEDs visible at the outer edge of the module. The ten LEDs are shown in Figure 3–7.

Figure 3–7 Processor Module LEDs



If the CPU fails its self-test during the first two phases (SROM and GROM), the failing test is indicated in the CPU LEDs (MSD and LSD). A lit LED indicates a logical one.

- SROM LED is on a microprocessor dual purpose output line.
  - — Serial output line state used for SROM LED
  - — SROM clock line to fifth FEROM
- RUN LED reflects status of OCP run light.
- LCONWIN LED is turned on by the primary CPU as it becomes the system console.
- STP LED is lit when CPU successfully passes its self-test.

The failing test LEDs count up sequentially as power on tests are performed. The LEDs indicate 1 to 11 for SROM tests and 12 to 56 for GROM tests. The LEDs are lit to indicate the test number at the beginning of a test. If the test fails, the LEDs remain lit.

The LEDs are useful when the processor executes its self-tests and contact with the console terminal has not been established. The status of the processor is indicated in the LEDs.

Eight of the LEDs are driven by bits <7:0> in the GBUS$LEDS register. This read/write register is accessible from the console at physical address F700 0040.

LNP Processor Self-Test

There are two levels of processor self-tests: SROM and GBus Flash ROM. Tests contained in SROM are performed first and if successful they are followed by tests contained in GBus Flash ROM. These power on tests run with LDIAG<FIGN> bit set which causes the module to ignore all LSB activity.

SROM Self-Test

The LNP processor has 11 SROM self-tests that run when power is applied. The SROM self-tests verify:

- Processor chip
- Clocks
- Cache and cache control
- GBus devices
- LSB interface (self-directed transaction on LSB)

GBus Flash ROM Self-Test

The GBus Flash ROM self-tests verify the parts of the processor module not tested by the SROM tests.

There are 45 Flash ROM tests for the LNP processor.

CPU/Memory Self-Test

The LNP processor executes 10 CPU/memory self-tests.

Multiprocessor Self-Test

The LNP processor has 7 multiprocessor tests. Tests 1 and 7 are executed even if only one processor is present.

I/O Port Self-Test

The LNP processor runs 24 IOP module tests and 41 LAMB module tests. They are executed in sequence on all adapters starting with the lowest numbered I/O bus.

At the end of self-test the LSB signal line BAD is deasserted if all:

1. Module's +5 Vdc is OK.
2. Modules pass the self-test.

Main Memory Configuration

The primary processor configures main memory while running the console in initialization mode. 7000-600 and 10000-600 systems memory is in two clusters: a console cluster and a system cluster. There is a memory cluster description for each memory cluster. They are accessed using pointers in the hardware restart parameter block (HWRPB). The size of each cluster is contained in its description.

- Console cluster size can vary with console revision.
- System cluster contains remainder of main memory.

The hardware restart parameter block (HWRPB) is a data structure created by the primary processor in the lower part of the console cluster. The HWRPB can be accessed by processors and the operating system and is used for communications between those units.

The HWRPB contains areas such as:

- General information area
- Per processor slots areas
- Console terminal block
- Console routine block
- Memory cluster 1 description
- Memory cluster 2 description

The HWRPB also contains pointers to other data structures such as:

- Memory cluster 1 bitmap
- Memory cluster 2 bitmap
- PALcode space
- CPU logout area
- CRB pages
- System configuration table
- FRU table

The first quadword of the HWRPB contains the physical address of the block. The second quadword of the HWRPB contains the ASCII representation of HWRPB in hex: 0000 0042 5052 5748.

Booting the System

Booting the VAX 7000/10000-600 System

The VAX 7000/10000-600 systems supports the following boot devices.

Table 3–11 VAX 7000-600 Boot Devices

Adapter	Interface	Internal Label	Console Device Name
DEMNA	NI—networks	XNA	EXx 0
CIXCD-AC	CI—cluster	XCD	DUx
KDM70	SI—storage	Wildcat	DUx
DEMFA	FDDI—fiber network	XFA	FXx 0
KFMSA-BA	DSSI—storage	DASH	DUx, MUx

The VAX 7000/10000 system BOOT command uses the format shown in Example 3–27.

Example 3–27 VAX 7000/10000 Boot Command Format

```
                                CIXCD
                              +---KFMSA    DUauu.A.B.C.D
                             /    KDM70
                            /
>>>BOOT -FLAGS n,mm,pp -----+      DEMNA    EXauu.A.B.C.D
                            \
                             \
                              +-- DEMFA    FXauu.A.B.C.D

NOTE:     n  SHADOW SET VALUE (Not supported by DEC 7000/10000-600 Systems)
         mm  SYSTEM ROOT 00-FF  (SYSmm high level directory)
         pp  VMB OPTION
  FX,EX,DU  DEVICE CODE
                a  CONTROLLER NUMBER a - z
               uu  DEVICE UNIT NUMBER
                A  DEVICE NODE
                B  DEVICE CHANNEL NUMBER
                C  XMI SLOT NUMBER 1-14
                D  IOP CHANNEL (HOSE) NUMBER 0-3
```

Example 3–28 BOOT Command

```
>>>boot -fl 0 dua1.0.0.3.0,dua1.1.0.3.0
```

VMB Options

Table 3–12 VMB Options

Flag Value	Name	Meaning if Set
1	Conversational	Allows the console operator to modify SYSGEN parameters in SYSBOOT.
2	DEBUG	Maps XDELTA to running system.
4	INIBPT	Stops at initial system breakpoint.
8	SECONDARY	Secondary boot from bootblock. Secondary boot is a single 512-byte block whose LBN is specified in GPR4.
10	—	Not used.
20	BOOT BRKPNT	Stops the primary and secondary loaders with a BPT instruction before testing memory.
40	IMAGE HDR	The transfer address of the secondary loader comes from the image header of that file. If this bit is not set, control shifts to the first byte of the secondary loader.
80	SOLICIT	Prompt for the name of the secondary bootstrap file.
1000	—	No effect.
8000	—	Not used.
10000	CRDFAIL	Mark corrected read data error pages bad.
<31:28>	—	Specifies top-level directory number for system disks.

Booting Tasks

The VAX 7000-600 system booting tasks are shown in Figure 3–8.

Figure 3–8 VAX 7000/10000 Console Booting Tasks Flowchart

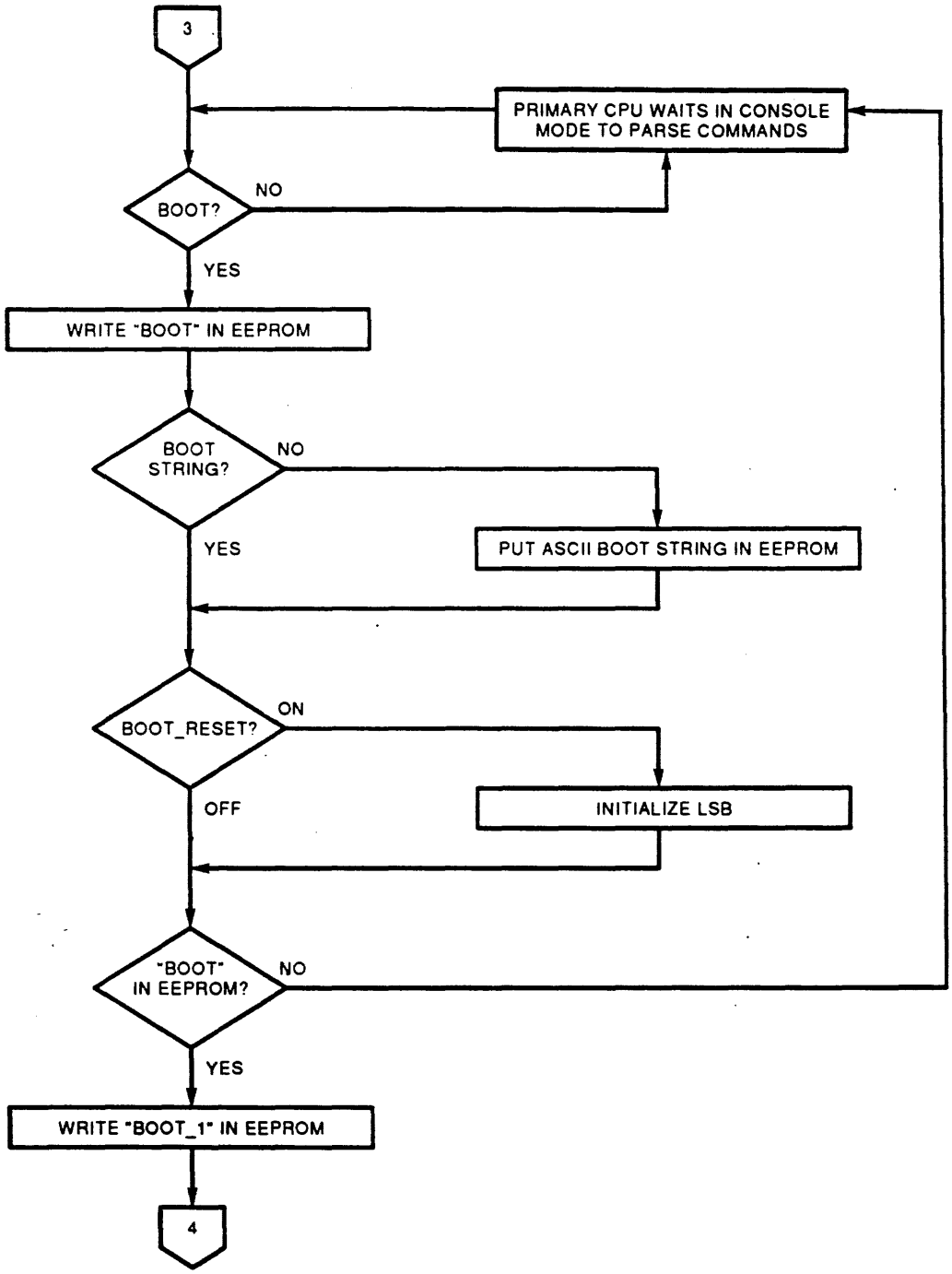
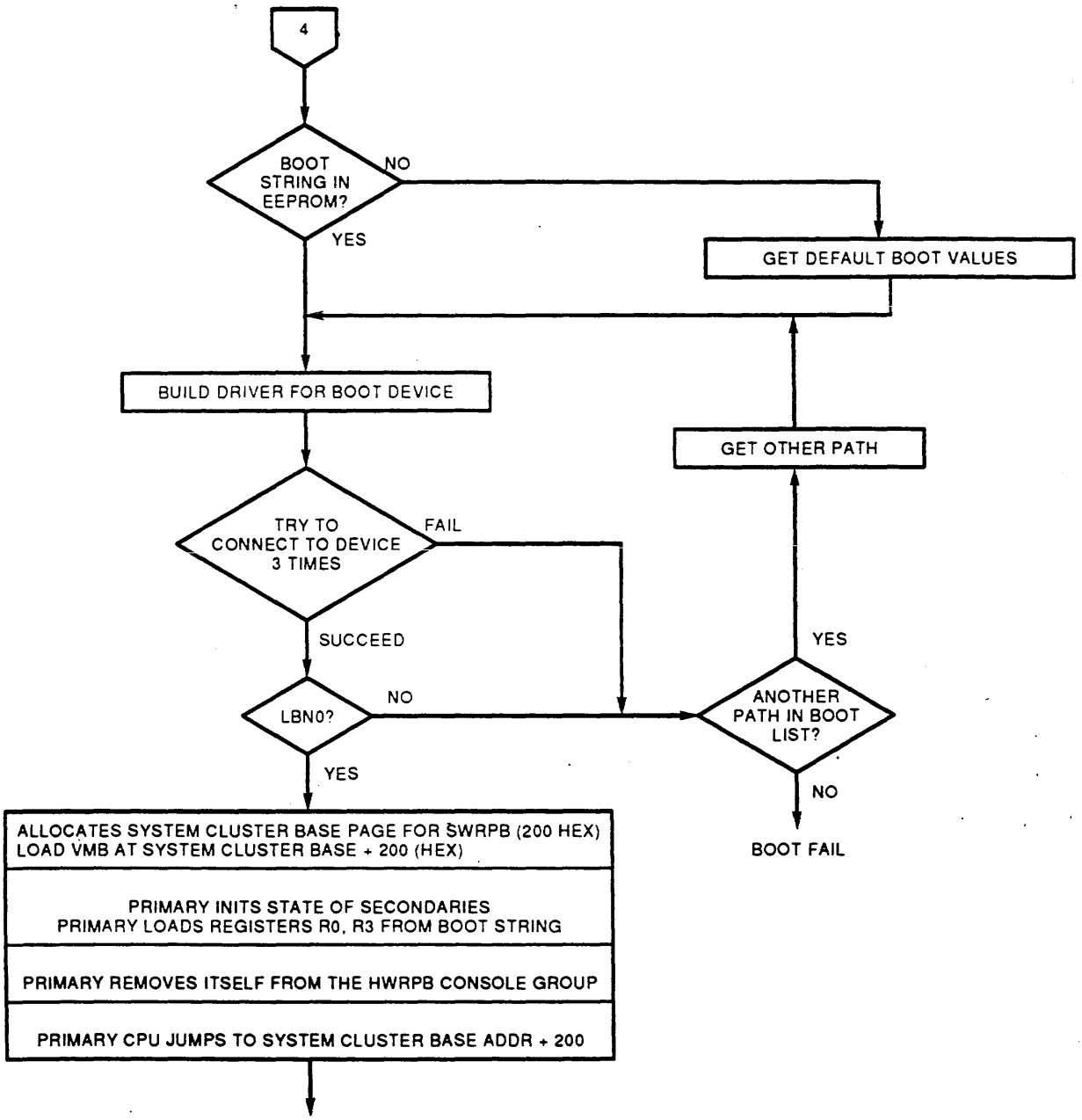


Figure 3–8 Cont'd on next page

Figure 3–8 (Continued) VAX 7000/10000 Console Booting Tasks Flowchart



VAXCON4_BOOT_FLOW90

Boot Display

Example 3–29 VAX 7000-600 System Boot Display

```
>>>show device
polling for units on cixcd0, slot 2, xmi0
dua20.13.0.2.0   $100$DUA20 RA82
dua31.14.0.2.0   $100$DUA31 RA82
dua80.15.0.2.0   $100$DUA80L     RA90
>>>boot -fl 0,4,0 dua20.13.0.2.0     ❶
Initializing...      ❷
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #
                        A   M   .   .   .   .   .   .   P   TYP
                        o   +   .   .   .   .   .   .   +   ST1
                        .   .   .   .   .   .   .   .   B   BPD
                        o   +   .   .   .   .   .   .   +   ST2
                        .   .   .   .   .   .   .   .   B   BPD
                        +   +   .   .   .   .   .   .   +   ST3
                        .   .   .   .   .   .   .   .   B   BPD
    +   .   .   .   .   .   +   .   .   .   .   .   +   .   C0  XMI +
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C1
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C2
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C3

                            .  A0   .   .   .   .   .   .   .  ILV
                            .  64   .   .   .   .   .   .   .  64Mb

Firmware Rev = *.*       SROM Rev = V*.*        SYS SN = ************
>>>     ❸
Booting ...      ❹
Connecting to boot device dua20  ❺
block 0 of dua20 is a valid boot block    ❻
reading 85 blocks from dua20  ❼
bootstrap code read in    ❽
base = 116000, start = 200  ❾
boot device name = dua20.13.0.2.0  ❿
boot flags 0,4,0
boot device type = 2b
controller letter = A
unit number= 20
node ID = 13
channel = 0
slot = 2
hose = 0
jumping to bootstrap at 116200  ⓫
```

❶ Explicit boot string with one path

❷ Initializing (if BOOT_RESET variable equals ON)

❸ Console prompt appears briefly—ignore it

❹ Starting boot indicator

❺ Successful connection to boot device

❻ Verifies that device has a valid boot block

❼ Starting to load VMB into memory

❽ VMB successfully loaded into memory

❾ Base address of VMB (first page allocated to restart parameter block)

❿ Shows load path used (not evident if default had been used)

⓫ Starts executing VMB code

Booting the DEC 7000/10000-600 System

The BOOT command uses the format shown in Example 3–30.

Example 3–30 Boot Command Format

```
                                  CIXCD
                               +-- KFMSB   DUauu.A.B.C.D
                              /    KDM70
                             /
                            +--  KZMSA     DKauu.A.B.C.D
                           /
cs>BOOT -FLAGS mm,pp ------+      DEMNA     EXauu.A.B.C.D
                           \
                            +--  DEMFA     FXauu.A.B.C.D

            NOTE:      mm  SYSTEM ROOT 00-FF (SYSmm HIGH LEVEL DIRECTORY)
                       pp  APB or OSF/1 OPTION

                FX,EX,DU  DEVICE CODE
                        a  CONTROLLER NUMBER a - z
                       uu  DEVICE UNIT NUMBER
                        A  DEVICE NODE
                        B  DEVICE CHANNEL NUMBER
                        C  XMI SLOT NUMBER 1-14
                        D  IOP CHANNEL (HOSE) NUMBER 0-3
```

Example 3–31 BOOT Command

```
cs>boot -fl 0 dka1.0.0.3.0,dka1.1.0.3.0
```

The Open VMS Alpha AXP operating system supports the following boot devices:

Table 3–13 Open VMS Alpha AXP Boot Devices

Adapter	Interface	Internal Label	Console Device Name
DEMNA	NI—networks	XNA	EXx 0
CIXCD-AC	CI—cluster	XCD	DUx
KDM70	SI—storage	Wildcat	DUx
DEMFA	FDDI—fiber network	XFA	FXx 0
KZMSA	SCSI—storage	XZA	DKxx, MKxx

OSF/1 Boot and Alpha AXP Primary Boot (APB) Options

Table 3–14 OSF/1 Boot Options

Option	Function
-a	Boots the system disk to multiuser mode
-d	Do full clumps
-i	Boots to interactive mode plus options
-s	Default boot option

Table 3–15 APB Options

Hex Value	Name	Meaning if Set
1	CONV	Bootstrap conversationally. That is, allow the console operator to modify SYSGEN parameters in SYSBOOT
2	DEBUG	Map XDELTA to running system
4	INIBPT	Stop at initial system breakpoint
8	DIAG	Perform diagnostic bootstrap
10	BOOBPT	Stop at bootstrap breakpoints
20	NOHEADER	Secondary bootstrap image contains no header
40	NOTEST	Inhibit memory test
80	SOLICIT	Prompt for the name of the secondary bootstrap file
100	HALT	Halt before secondary bootstrap
200	SHADOW	Reserved
400	ISL	Reserved
800	BOOTLOG	Reserved
1000	DEBUG_BOOT	Reserved
2000	CRDFAIL	Mark corrected read data error pages bad
4000	NO_SLICE	Reserved
8000	BIT15	Reserved
10000	DBG_INIT	Enable verbose mode in APB,SYSBOOT, and EXEC_INIT
20000	DBG_INIT_ ABBREV	Abbreviated descriptive mode in APB,SYSBOOT, and EXEC_INIT

OpenVMS Alpha AXP Boot Tasks

Figure 3–9 OpenVMS Alpha AXP Console Booting Tasks Flowchart

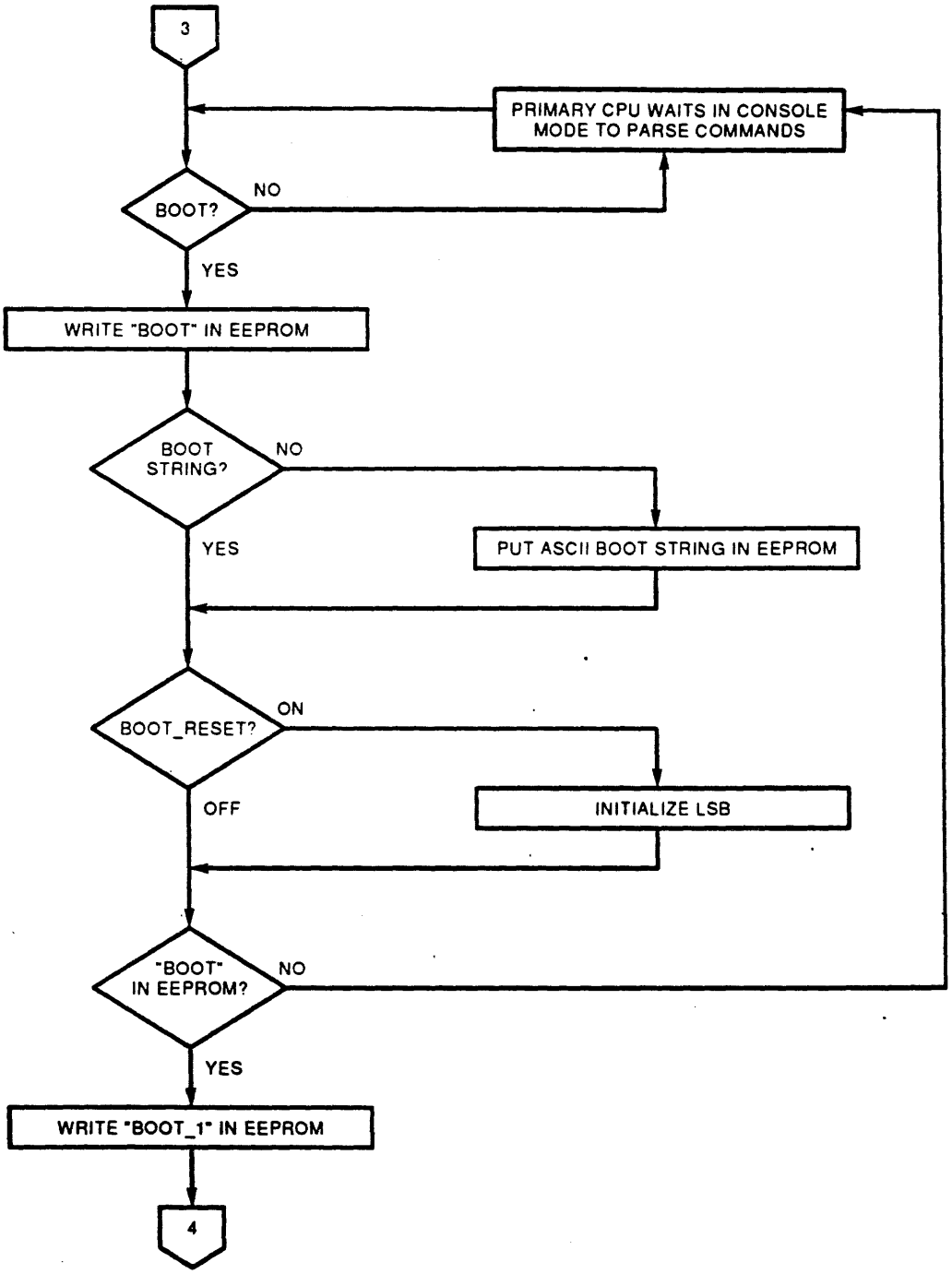
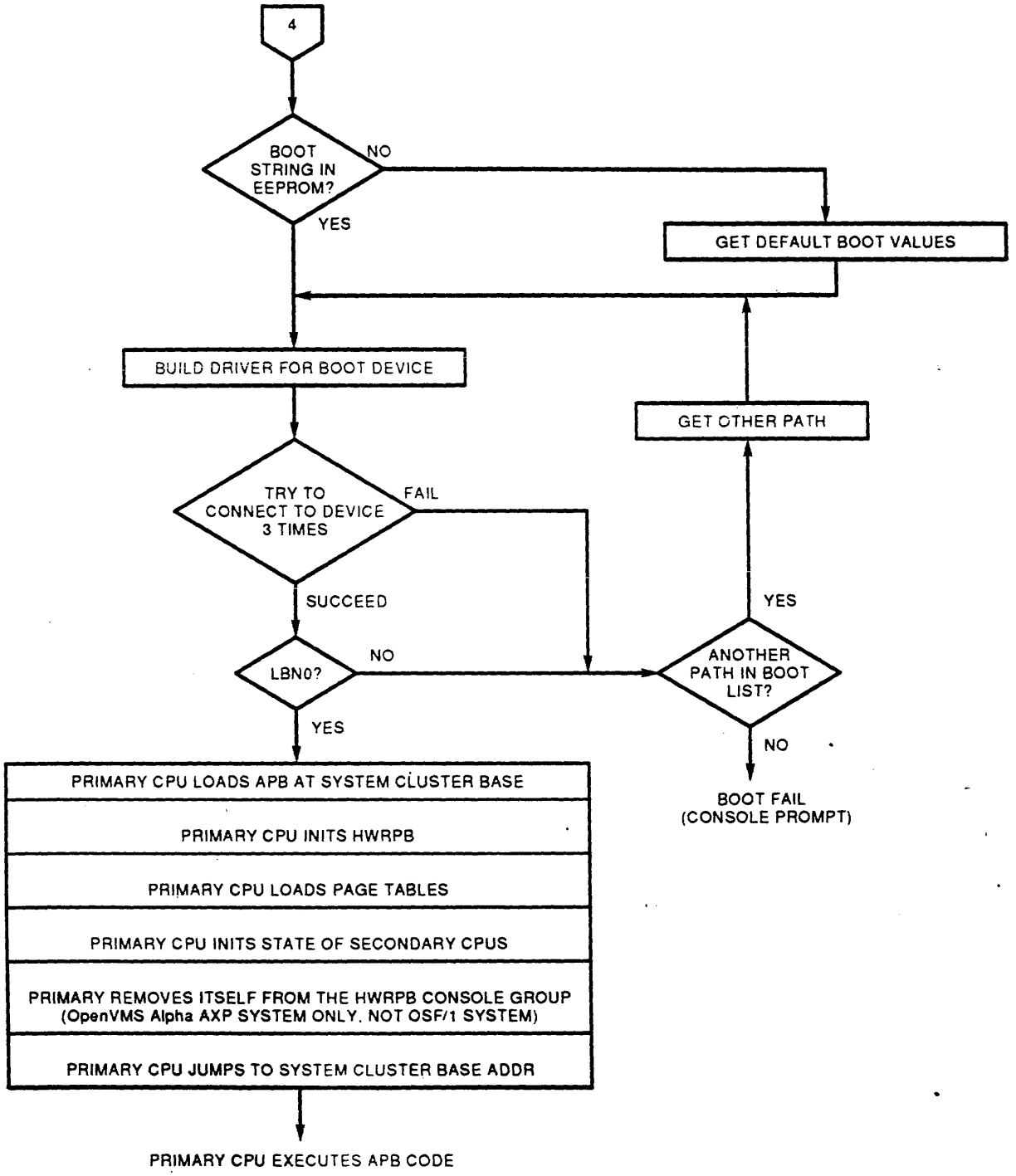


Figure 3–9 Cont'd on next page

Figure 3–9 (Continued) OpenVMS Alpha AXP Console Booting Tasks Flowchart



Example 3–32 Open VMS Alpha AXP System Boot Display

```
>>>show device
polling for units on kzmsa0, slot 3, xmi0..
dka0.0.0.3.0  DKA0  RZ73
dka100.1.0.3.0  DKA100  RZ73
dka200.2.0.3.0  DKA200  RZ73
dka300.3.0.3.0  DKA300  RZ73
dka400.4.0.3.0  DKA400  RZ73
dka500.5.0.3.0  DKA500  RZ73

>>>boot -fl 4,20000    ❶
Initializing...    ❷

F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #
                        A   M   M   .   .   .   .   .   P   TYP
                        o   +   +   .   .   .   .   .   +   ST1
                        .   .   .   .   .   .   .   .   B   BPD
                        o   +   +   .   .   .   .   .   +   ST2
                        .   .   .   .   .   .   .   .   B   BPD
                        +   +   +   .   .   .   .   .   +   ST3
                        .   .   .   .   .   .   .   .   B   BPD
    +   .   .   .   .   .   +   .   .   +   .   +   .   .   C0  XMI +
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C1
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C2
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   C3

                            .  A0   A1   .   .   .   .   .   .  ILV
                            . 128  128   .   .   .   .   .   .   . 256Mb

Firmware Rev = v1.0-226  SROM Rev = V1.0-1    SYS SN = GA21200048
>>>    ❸

Booting ...    ❶
Connecting to boot device dka200 ❺
Created boot device: dka200.2.0.3.0 ❻
block 0 of dka200 is a valid boot block ❼
reading 764 blocks from dka200  ❽
bootstrap code read in  ❾
base = 200000    start=0  ❿
initializing HWRPB at 2000  ⓫
initializing page table at 1f4000 ⓬
initializing machine state  ⓭
jumping to bootstrap at 200000  ⓮

%APB-I-ABPVER, Alpha Primary Bootstrap, Version X5DN
```

❶ Boot string without device - BOOTDEF_DEV will be used, abbreviated debug display will be shown

❷ Initializing (if BOOT_RESET variable equals ON)

❸ Console prompt appears briefly—ignore it

❶ Starting boot indicator

❺ Trying to connect to boot device

❻ Successful connection to boot device

❼ Found valid boot block

❽ Started copying APB to main memory

❾ APB successfully copied to main memory

❿ Base address is 200000, start offset is 0

⓫ Initializing the HWRPB at PA 2000 (second page of memory)

⓬ Loading page tables starting at 1f4000 (DEC 7000 boots virtual)

⓭ Initializing processors

⓮ Starts executing APB code at 200000

Diagnostic Mode Exercisers

The operator may test modules and devices in the system as shown in Example 3–33.

Example 3–33 Using Test Commands

```
>>>test ka7aa0
 { system test sequence runs 600 seconds }
>>>test ka7aa0 -t 60
 { ka7aa0 test sequence runs 60 secs }
>>>test kfmsa0
 { kfmsa0 test sequence runs 60 secs }
>>>test kfmsa*
 { test sequence runs 60 secs testing all kfmsas }
>>>test demna2
 { demna2 test sequence runs 60 secs }
>>>test iop
 { iop test sequence runs 60 secs }
```

When in diagnostic mode the operator can also invoke unsupported exercisers. These are intended for use by manufacturing and design engineering.

The MEM_EX command format is shown and its use is demonstrated in Example 3–34.

Example 3–34 Using the MEM_EX Command

```
>>>set d_* -d    ❶
>>> MEM_EX -T 2 -P 5    ❷

>>> MEM_EX -T 2 -P 0 &    ❸
>>> MEM_EX -T 2 -P 0 &    ❹
>>> MEM_EX -T 2 -P 0 &    ❺
>>> SHOW STATUS    ❻
ID  Program          Device      Pass Hard/. ft Bytes Written    Bytes Read
--------  --------  --------------  --------  --------  -------------  -------------
00000014  MEM_EX               mem     13730     0     0      112467968      112467968
00000017  MEM_EX               mem      2342     0     0       19177472       19177472
0000001a  MEM_EX               mem      1094     0     0        8953856        8953856

>>> SHOW STATUS
ID  Program          Device      Pass Hard/Soft Bytes Written    Bytes Read
--------  --------  --------------  --------  --------  -------------  -------------
00000014  MEM_EX               mem     19487     0     0      159629312      159629312
00000017  MEM_EX               mem      8225     0     0       67371008       67371008
0000001a  MEM_EX               mem      6838     0     0       56008704       56008704

>>> kill 1a    ❼

>>> show status
ID  Program          Device      Pass Hard/Soft Bytes Written    Bytes Read
--------  --------  --------------  --------  --------  -------------  -------------
00000014  MEM_EX               mem     45580     0     0      373383168      373383168
00000017  MEM_EX               mem     34194     0     0      280109056      280109056

>>> kill 14
>>> show status
ID  Program          Device      Pass Hard/Soft Bytes Written    Bytes Read
--------  --------  --------------  --------  --------  -------------  -------------
00000017  MEM_EX               mem    216919     0     0     1776992256     1776992256
```

❶ Set all diagnostic environment variables to default values.

❷ Run Test 2 of MEM_EX for 5 passes on all memory modules.

❸ Run Test 2 of MEM_EX in background forever (creates a process).

❹ Run Test 2 of MEM_EX in background forever (creates a second process).

❺ Run Test 2 of MEM_EX in background forever (creates a third process).

❻ Show three processes running Test 2 of MEM_EX.

❼ Removes Process 1a, leaving two processes running.

Disk Exerciser

The disk exerciser can be run in the background. A nondestructive read and read/compare is performed in Example 3–35. The disk area modifiers are *sb* (starting block), *eb* (ending block), and *bs* (block size). The action string designated by *-a* is defined in Table 3–16.

Table 3–16 Disk Exerciser Action Strings

Action Qualifier	Definition
c	Compare expected buffer with received buffer.
d	Fill expected buffer with default or specified pattern.
D	Fill received buffer with default or specified pattern.
n	Write without lock from expected buffer.
N	Write without lock from received buffer.
r	Read into expected buffer.
R	Read into received buffer.
w	Write from expected buffer.
W	Write from received buffer.
-	Seek to file offset before last read or write.
?	Seek to random block offset within the specified range of blocks.

Example 3–35 Running the Disk Exerciser

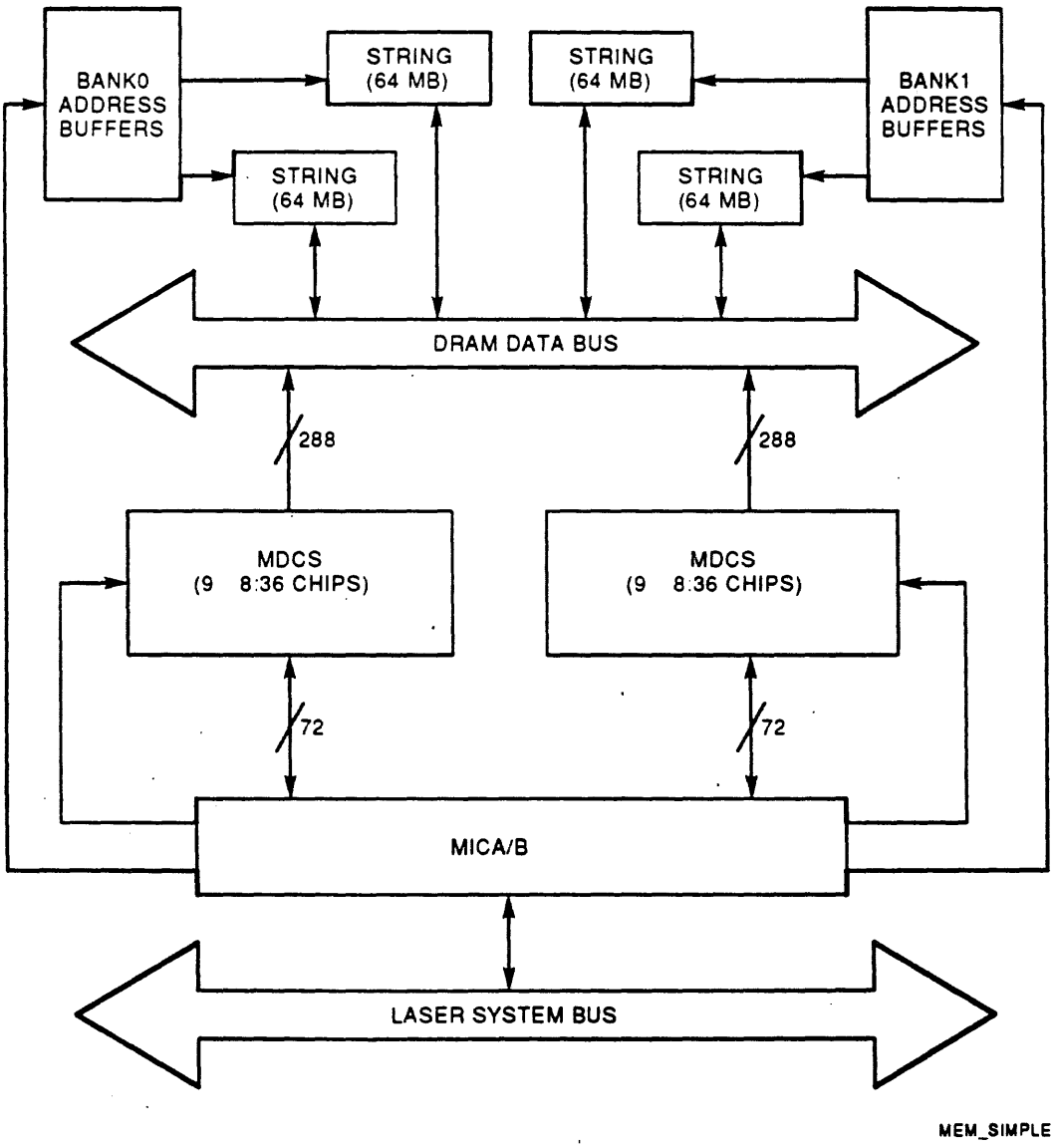
```
>>> sho device
dua112.0.0.2.0     DUA112                          RA70
dua113.0.0.2.0     DUA113                          RA70
dua115.0.0.2.0     DUA115                          RA70
dua192.0.0.2.0     DUA192                          RA90
>>>
>>>dsk_ex -a "?R-rc" -sb 0 -eb 20 -bs 2000 -p 0 dua112 &
>>>dsk_ex -a "?R-rc" -sb 0 -eb 20 -bs 2000 -p 0 dua113 &

>>>show status

ID       Program          Device      Pass Hard/Soft Bytes Written    Bytes Read
-------- -------- ---------------- -------- --------- ------------- -------------
00000014  dsk_ex   dua112.0.0.2.0   1429     0     0     0     112467968
00000017  dsk_ex   dua113.0.0.2.0   1429     0     0     0     112467968

>>>kill 14
>>>kill 17
>>>show status
ID       Program          Device      Pass Hard/Soft Bytes Written    Bytes Read
-------- -------- ---------------- -------- --------- ------------- -------------
>>>
```

Memory Interleaving

The VAX 7000-600 and VAX 10000-600 systems have 64 MB, 128 MB, 256 MB, and 512 MB memory modules available.

A memory string is the smallest part of memory capable of reading or writing 64 bytes of data in one operation. The relationship between a LSB memory string and a block is shown in Figure 3–10.

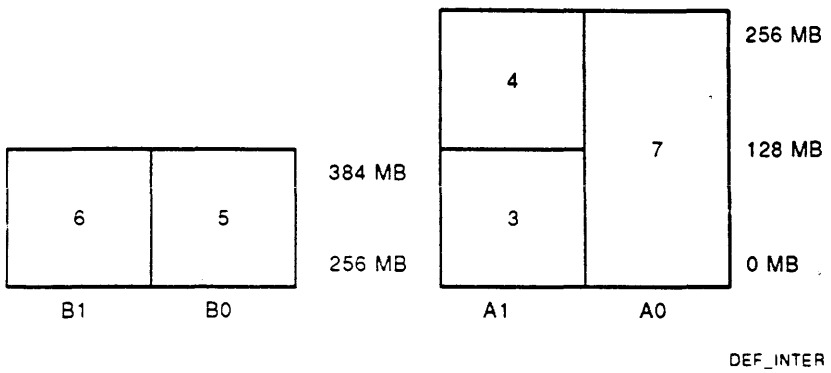
Figure 3–10 Simple LSB Memory Block Diagram



Default Memory Interleaving

The console interleaves memory, if possible, using a default method unless prevented by a previous console command. For example, a system with four 64 MB memory boards at nodes 3, 4, 5, and 6 and a 128 MB board at node 7 would take the default interleave as shown in Figure 3–11.

Figure 3–11 Memory Default Interleave



Explicit Memory Interleaving

The SET INTERLEAVING console command is used by the operator to specify the memory interleave configuration. The command has three variations as shown in Example 3–36.

Example 3–36 SET MEMORY Commands

```
>>SET INTERLEAVE DEFAULT
>>SET INTERLEAVE NONE
>>SET INTERLEAVE "explicit configuration"
```

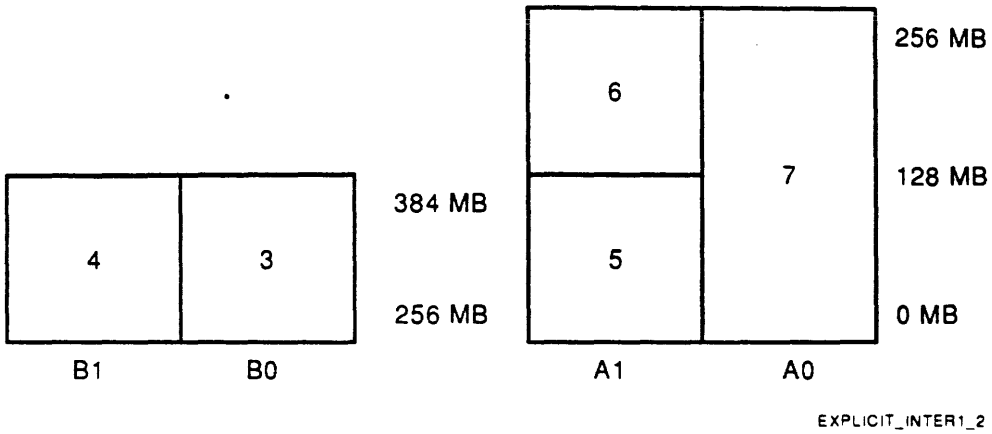
The set interleave command does not change the current memory configuration but changes data in the EEPROM. The next time the console initializes the system, it will use that information when configuring memory.

The explicit configuration parameter list used three operators: ,, +, and : . They are used as shown in Example 3–37 causing the interleave configuration shown in Figure 3–12.

Example 3–37 SET MEMORY Command

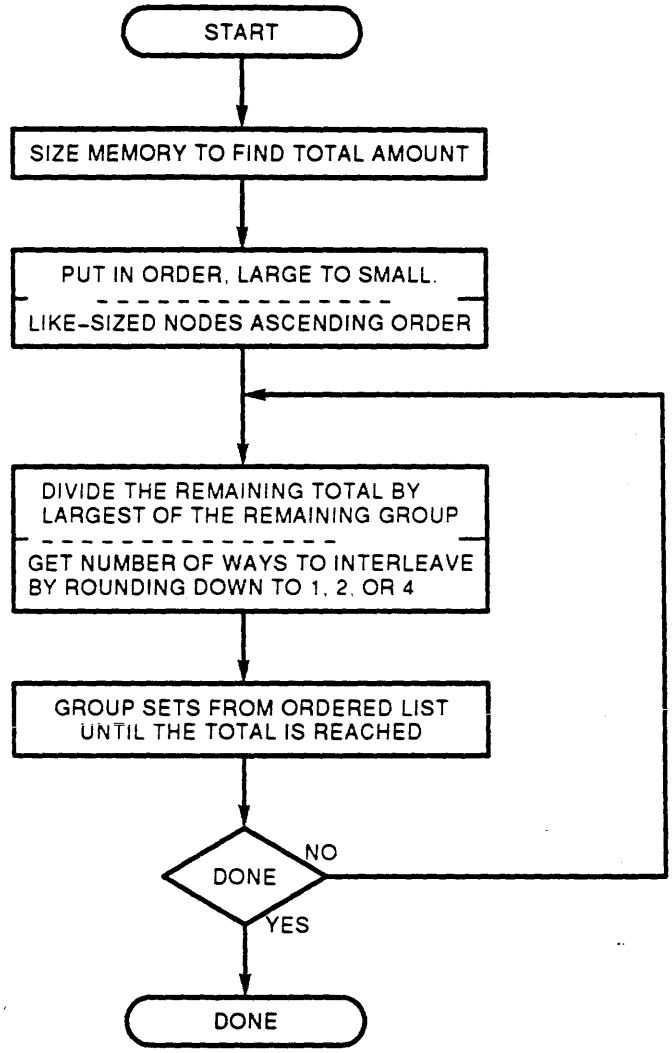
```
>>SET INTERLEAVE 7+5:6,3+4
```

Figure 3–12 Memory Explicit Interleave.



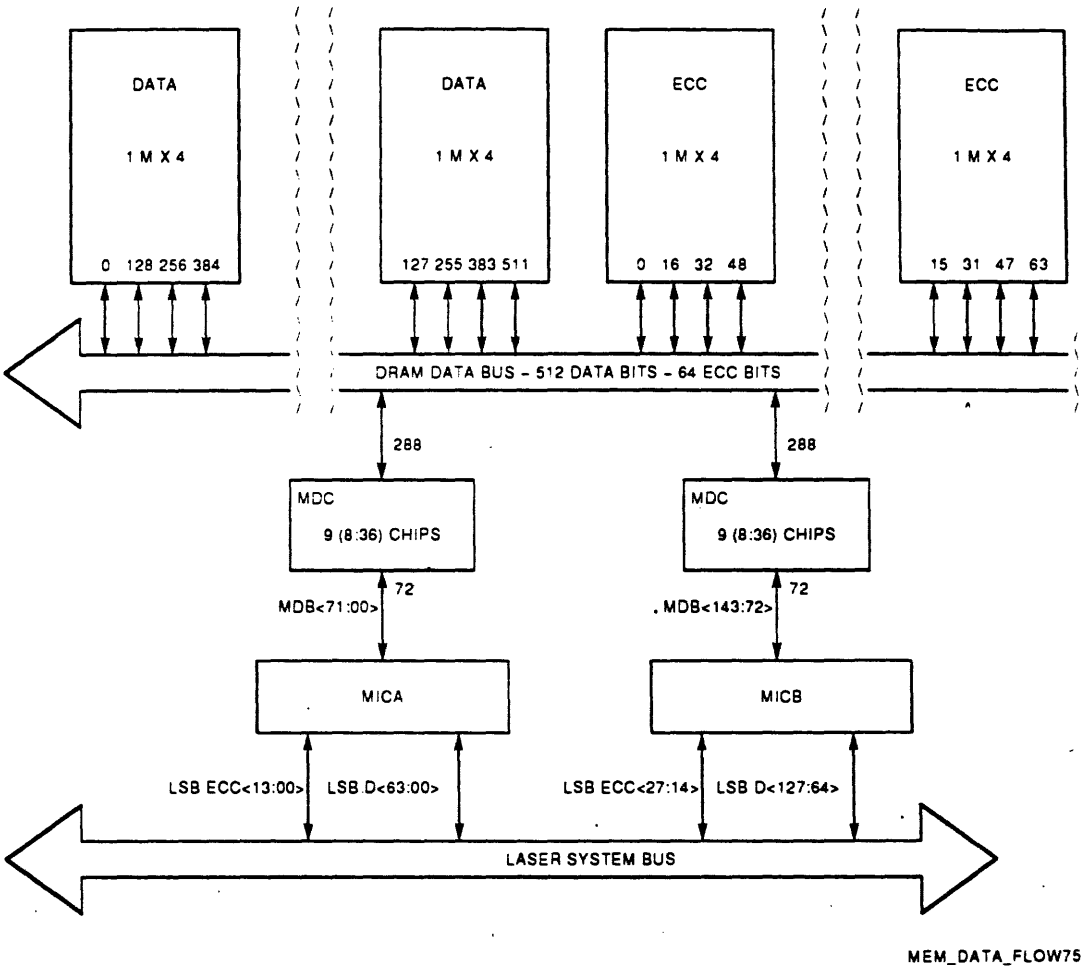
If the memory interleave information in the EEPROM is not consistent with a possible interleave scheme, then the console will print a message on the operator console and use the default algorithm shown in Figure 3–13.

Figure 3–13 Console Default Interleave Algorithm



A memory string is a the smallest part of memory capable of reading or writing 64 bytes of data in one operation. An LSB memory string is shown in Figure 3–14.

Figure 3–14 LSB Memory Module Data Flow



Memory Module Onboard—Two-Way Interleaving

A memory module with more than two strings supports onboard memory interleaving. Each string using 4 Mb chips contains 64 MB of storage space while each string using 16 Mb chips contains 256 MB of storage. The variations of memory modules available is listed in Table 3–17.

Table 3–17 LSB Memory Module Variations

Module Capacity	DRAM Size	Number of Strings
64 MB	4 Mb	1
128 MB	4 Mb	2
256 MB	4 Mb	4
512 MB	4 Mb	8
256 MB	16 Mb	1
512 MB	16 Mb	2
1 GB	16 Mb	4
2 GB	16 Mb	8

Memory banks on the LSB memory module are interleaved as shown in Figure 3–14.

There can be either one or two banks of memory on an LMEM. When there are two banks on an LMEM module, it will automatically support onboard two-way interleaving. If there is only one bank, then there is one-way interleaving (no interleaving).

The LSB memory system also supports two-way and four-way interleaving between LMEM modules. Two LMEM of the same size can support two-way module interleaving and four LMEM of the same size will support four-way module interleaving.

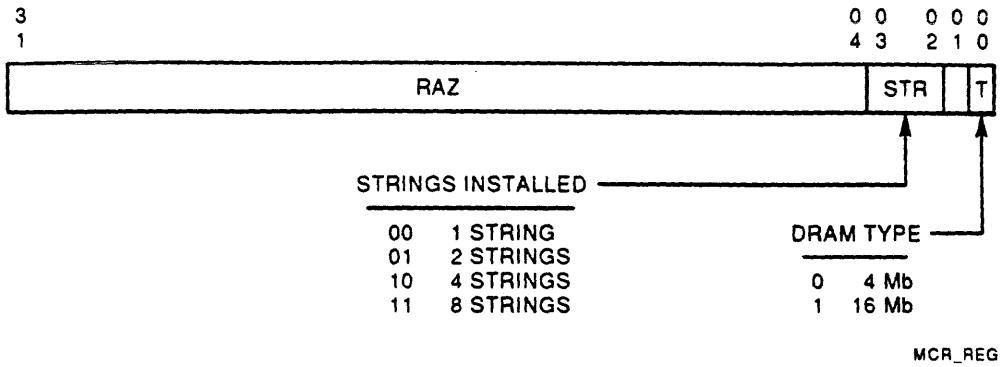
The system console can also combine various size LMEM to appear as like-sized modules and then interleave the combinations. For example with two 64 MB LMEM and one 128 MB LMEM the console can manipulate the two smaller memories to interleave like one 128 MB module.

An LSB memory module uses a storage scheme that returns good data (CRD) even if a single memory chip totally fails.

Memory Module Interleaving Registers

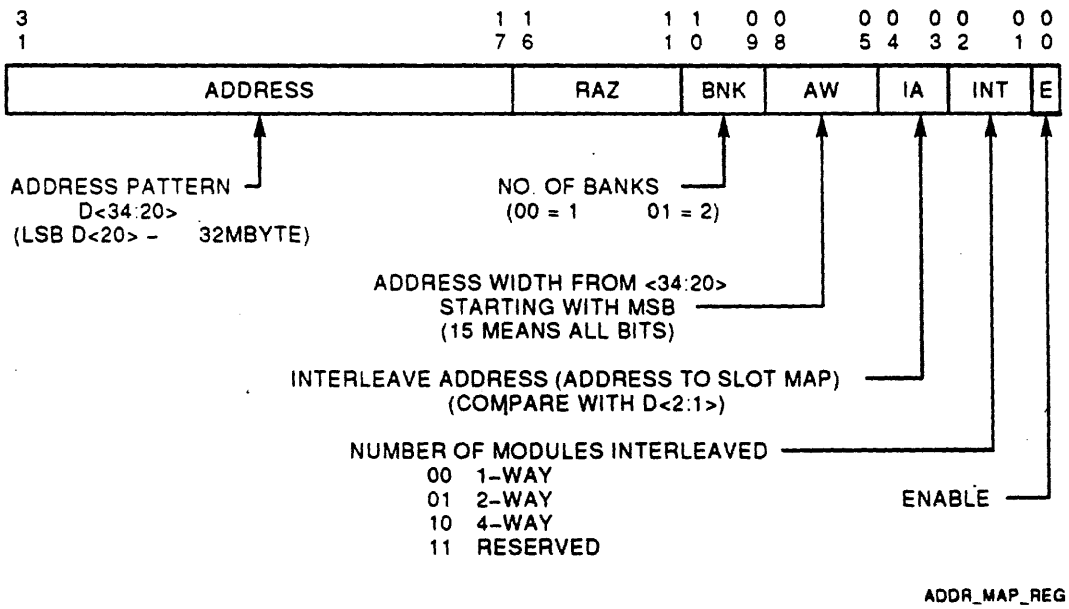
The memory module Memory Configuration Register (MCR, BB+2000) shown in Figure 3–15 is read from each memory module by the initializing program. The information from all modules is then used to determine the system memory interleaving arrangement.

Figure 3–15 Memory Configuration Register (MCR, BB+2000)



When the interleaving arrangement has been determined, the correct data is loaded into the Address Mapping Register (AMR, BB+2040) shown in Figure 3–16 on each memory module.

Figure 3–16 Address Mapping Register (AMR, BB+2040)



A copy of the AMR from each of the memory modules is also loaded into the appropriate mapping register on the IOP module and CPU modules. LMMR0-7 correspond to nodes 0 to 7.

For example: The AMR register from the memory module in Slot 3 would be copied into LMMR3 on the IOP and CPUs modules.

When Slot *n* does not contain a memory module, the enable bit in LMMRn is set to zero.

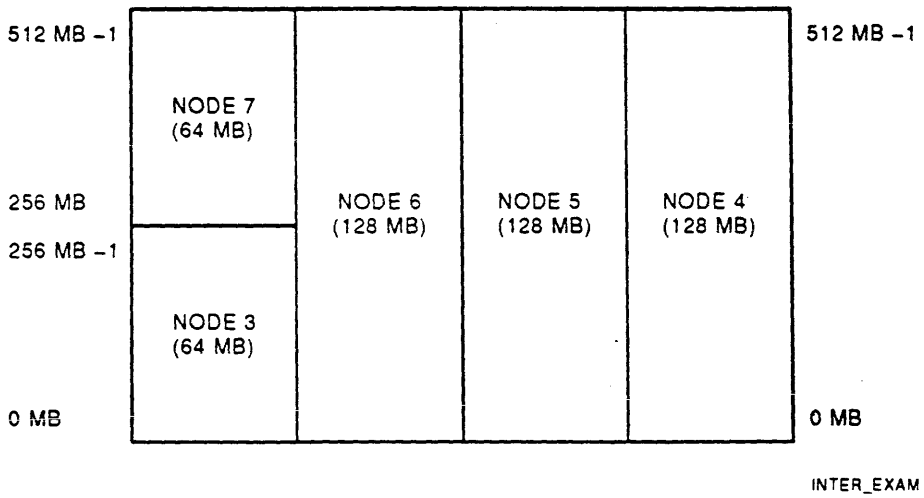
Interleaved Memory Modules Example

Example of a default system memory interleave arrangement set up by the system console when power is applied:

Given: Three 128 MB LMEM at nodes 4, 5, and 6 and two 64 MB LMEM at nodes 3 and 7.

Scheme: The console would interleave the modules as shown in Figure 3–17.

Figure 3–17 LMEM Interleave Configuration Example



As it sets up the interleave scheme, the console:

- Checks the LDEV registers and finds out which are LMEM modules (4000 in LDEV <15:00>).
- Reads the memory configuration register MCR, shown in Figure 3–15, for each of the LMEM nodes. It gets the DRAM type (4 Mb or 16 Mb) and number of strings (1, 2, 4, 8).
- Looks at the interleave variable and finds the interleave variable set to DEFAULT.
- Calculates the default interleave scheme for the given nodes.
- Loads the address mapping registers in LMEM nodes 3, 4, 5, 6, and 7 as shown Figure 3–18.
- Loads the 8 LMMRns in each CPU and I/O module with copies of AMRs that are present and sets ENABLE bit <0> in other LMMRs to zero as shown in Figure 3–19. The other LMMR bits are set to zero.

Figure 3–18 AMR Registers

31 30	29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
0 0	RAZ	0 1 1 0 1 1 0 0 1 0 1	NODE 4 AMR
0 0	RAZ	0 1 1 0 1 1 0 1 1 0 1	NODE 5 AMR
0 0	RAZ	0 1 1 0 1 1 1 0 1 0 1	NODE 6 AMR
0 0	RAZ	0 0 1 1 0 0 1 1 1 0 1	NODE 3 AMR
0 0	RAZ	0 0 1 1 0 0 1 1 1 0 1	NODE 7 AMR

NOTE: AMR <17> CORRESPONDS TO 32 MB

AMRS1_2

Figure 3–19 LMMRs Registers

31 30	29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
0 0	RAZ	0 0 0 0 0 0 0 0 0 0 0	LMMR0
0 0	RAZ	0 0 0 0 0 0 0 0 0 0 0	LMMR1
0 0	RAZ	0 0 0 0 0 0 0 0 0 0 0	LMMR2
0 0	RAZ	0 0 1 1 0 0 1 1 1 0 1	LMMR3
0 0	RAZ	0 1 1 0 1 1 0 0 1 0 1	LMMR4
0 0	RAZ	0 1 1 0 1 1 0 1 1 0 1	LMMR5
0 0	RAZ	0 1 1 0 1 1 1 0 1 0 1	LMMR6
0 0	RAZ	0 0 1 1 0 0 1 1 1 0 1	LMMR7

NOTE: LMMRn <17> CORRESPONDS TO 32 MB

LMMRS1_2

Flash ROM Recovery Code (FRRC)

The SROM (fifth 128 kB Flash ROM) contains flash ROM recovery code (FRRC). FRRC provides a simple user interface at the operator's console. It has the following features.

- It can be used to rewrite the four Flash ROMs.
- It has a simple command interface.
- It uses Kermit protocol.
- FRRC is located in the SROM along with 8 kB SROM image.

The SROM code (TRANSCODE in Test 11) checks the integrity of the first 4 Flash ROMs before loading those instructions into the backup cache. If a checksum error is found the FRRC code is loaded from the SROM into the backup cache and executed. The FRRC in the "sick" CPU checks the CONWIN signal line looking for a "healthy" CPU to become the primary CPU.

Healthy CPU is Present

If a healthy CPU appears, the FRRC sets a bit in the primary CPU's LCON register indicating that this node's Flash ROM code is corrupted.

The sick CPU then polls another bit in the healthy CPU's LCON register. The healthy CPU sets that bit to indicate that there is a good image of the console code in main memory. The sick CPU then starts to execute the good image in main memory.

A secondary processor with a *D* instead of a *P* in the console map may have had a bad checksum in the first four Flash PROMs. The update KA7AAN console command may be used to reload the Flash PROMs.

Healthy CPU is Not Present

If a healthy CPU does not appear within a certain length of time, the lowest numbered CPU running FRRC code asserts CONWIN, prints its FRRC> prompt on the console terminal, and waits for operator action.

FRRC code accepts the following commands using hex numbers.

Table 3–18 FRRC Commands

Command	Parameters	Definition
H	–	Help
R	address	Receive a binary file, using KERMIT receive protocol, and store at address
X	bytes address	Receive *value* data bytes using *X* protocol and store at address
I	–	Execute node reset
C	longwords address	Checksum an image of *number* longwords at *address* in cache
B	–	Blank (erase) four Flash ROMs
W	bytes address	Write *number* bytes of code starting at *address* in four Flash ROMs
EM	address	Examine a memory location at *address*
DM	address data	Deposit *data* to *address* in memory
EI	number	Examine IPR *number*
DI	data number	Deposit *data* to IPR *number*
EB	number address	Examine *number* longwords of memory starting at *address*
F	–	To be determined
P	–	To be determined

The image can be loaded from the RDC via a modem connected to the CPU console port. At 9600 baud, it would take about 40 minutes. At 1200 baud, it would take about 5 hours.

Advanced Console Review Exercise

1. How many memory clusters does a 7000-600 system have?

   a. Four, one for each possible processor

   b. Seven, one for each commander node

   c. One, which supports the primary processor

   d. Five, one for each processor and another for the IOP module

   e. None of the above

2. With pass 3 LEVIs on CPUs which one of the following combinations of memory modules could be interleaved into one set?

   a. 1 at 128 MB and 1 at 64 MB

   b. 2 at 128 MB and 2 at 64 MB

   c. 1 at 128 MB and 3 at 64 MB

   d. 3 at 128 MB and 2 at 64 MB

   e. None of the above

3. Which command would you use to create and set the contents of an environmental variable in the processor EEPROM?

   a. >>>SET -NV FRED '-f 1000 dua0.0.0.3.1'

   b. >>>CREATE FRED '-f 1000 dua0.0.0.3.1'

   c. >>>CREATE -NV FRED '-f 1000 dua0.0.0.3.1'

   d. >>>SET FRED '-f 1000 dua0.0.0.3.1' -NV

   e. None of the above

4. Which command would you use to find which device was last used to boot the system?

   a. >>>EXAMINE BOOT_DEV

   b. >>>SHOW EEPROM FIELD

   c. >>>SHOW BOOT_DEF

   d. >>>SHOW BOOTED_DEV

   e. None of the above

5. Which console mode allows you to use the SHOW_STATUS command?

   a. Basic

   b. Diagnostic

   c. Kernel

   d. Support

   e. None of the above

6. Which address would you use to examine the LBER register on the IOP module?

   a. >>>EX PMEM:500004 -L

   b. >>>EX PMEM:FA000040 -L

   c. >>>EX PMEM:500040 -B

   d. >>>EX PMEM:FA000004 -L

   e. None of the above

7. A DSE using LFU to update the FEROMs on a secondary processor must first update the FEROMs on the primary processor.

   a. True

   b. False

8. Which one of the following statements are true?

   a. The DEC 7000-600 supports the system root argument in the -flags string.

   b. The DEC 7000-600 supports the shadow boot argument in the -flags string.

   c. The VAX 7000-600 supports the console baud argument in the -flags string.

   d. The VAX 7000-600 supports the background argument (&) in the -flags string.

   e. None of the above.

9. Which one of the following statements are true?

   a. Using console diagnostic mode, a DSE can exercise a single target disk in the foreground but not in the background.

   b. Using console basic mode, a DSE can exercise a single target disk in the background.

   c. Using console diagnostic mode, a DSE can exercise a single target disk in the background but not in the foreground.

   d. Using console diagnostic mode, a DSE can exercise a single target disk in the foreground or background.

   e. None of the above.

10. Which one of the following statements is true?

   a. The DEC 7000-600 does not use the HWRPB as there is no concept of warm restart on this system.

   b. The DEC 7000-600 uses the HWRPB, but the VAX 7000-600 does not use it.

   c. The VAX 7000-600 system console creates a HWRPB. It usually starts at address 2000 (hex) in the console memory cluster.

   d. The VAX 7000-600 does not use the HWRPB as there is no concept of warm restart on this system.

   e. None of the above.

LASER System Bus

Introduction

This module introduces the student to the LSB protocol and points out some ways that it is used by the VAX 7000-600 and VAX 10000-600 systems. The DSE becomes familar with LSB arbitration and transactions. Error detection and reporting features of the LSB are discussed for later use in making fault diagnosis.

Objectives

A Digital Services Engineer who maintains a VAX 7000-600 or VAX 10000-600 system should be able to:

- Describe the main features of LSB protocol, including:
  - — Arbitration
  - — Data transactions
  - — Command cycles
  - — Data cycles
  - — Interrupts
  - — Error correction scheme
- Recognize the error detection features of the LSB.
- Describe how detected errors will be reported to the system software.

Resources

MS7AA Memory Technical Manual	EK-MS7AA-TM

LSB Overview

The LASER system bus (LSB) was developed with a high bandwidth for data transfers to support high-speed multiprocessor systems. It has the following characteristics:

- Nonpended
- Synchronous (20 ns cycle)
- LSB clock generated on the IOP module
- 128-bit data bus with longword ECC protection
- Data bus is multiplexed to carry address during command cycles
- Fixed-size memory data transfer (64 bytes, DHW)
- Distributed arbitration
- Supports conditional update writeback cache protocol
- Pipelined
- Limited length

Clocks

The LSB clock signals, PH0 and PH90, are generated on the IOP module. Separate copies are distributed to each slot in the LSB backplane including the IOP itself.

Transaction Overview

The LSB supports six types of transactions on the bus. All transactions use five LSB cycles: a command/address (C/A) cycle and four data cycles. More than one transaction can be in progress at once as the LSB supports pipeline operation.

Typical Transaction

The arbitration cycle begins a transaction. Command and data cycles have a fixed relationship to the arbitration cycle as shown in Example 4–1.

Example 4–1 Typical Cycles

```
                  1111111111222222222233333333334444444444555555555556
CYCLE     12345678901234567890123456789012345678901234567890123456789
ARB       ....X.....................................................
C/A       ......X...................................................
CMD       ......X...................................................
CNF       .........X................................................
SHR/DIR   ..........X................................................
STALL     ...........................................................
DATA      ................XXXX.....................................
```

Best-Case Transaction

Pipeline operation adds to the efficiency of the LSB. The more the pipeline is used, the more efficient the LSB. The most efficient use of the LSB is shown in Example 4–2 where there are no unused cycles between several transactions.

Example 4–2 Best-Case Cycles Transaction

```
                  1111111111222222222233333333334444444444555555555556
CYCLE     12345678901234567890123456789012345678901234567890123456789
ARB       ....1....2.............1....2....3....4.....................
C/A       ......1....2.............1....2....3....4...................
CMD       ......1....2.............1....2....3....4...................
CNF       .........1....2.............1....2....3....4................
SHR/DIR   ..........1....2.............1....2....3....4..............
STALL     ...........................................................
DATA      ..................1111.2222..........1111.2222.3333.4444....
```

Memory Bank Contention

Access to a memory bank is restricted to once per 15 LSB cycles. To implement this rule, the IOP and all CPUs must capture the address of every memory access address from the LSB and save it for at least 15 cycles. When they are going to arbitrate to perform a memory transaction, they check the saved addresses to avoid a 15 cycle per bank conflict.

Each IOP and CPU node has LASER Memory Mapping Registers 0-7 (LMMR0-7). These registers contain copies of the AMR register for each memory module on the LSB. LMMR0-7 are used when deciding whether an address is in the same memory bank as a previous LSB memory access address.

For Example:
The CPU in node 1 is the only node arbitrating for use of the bus.
There is only one memory module: 128 MB with two-way onboard interleaving.
The CPU wants to read data from each physical address in sequence: 00004000, 00004040, 00004080, and 000040C0.

As shown in Example 4–3, the CPU would be able to read the first two blocks without delay as they are in different memory banks. However, the CPU would then have to wait five cycles more before it could access the first bank again.

Example 4–3 Memory Bank Contention

```
                 1111111111222222222233333333334444444444555555555556
CYCLE      12345678901234567890123456789012345678901234567890123456789O
ARB        ....1....1.........1....1.....................................
C/A        ......1....1.........1....1...................................
CMD        ......1....1.........1....1...................................
CNF        .........1....1.........1....1................................
SHR/DIR    ...........1....1.........1....1..............................
STALL      ..............................................................
DATA       ................1111.1111......1111.1111.....................
```

Arbitration

Arbitration can be performed on any cycle when the LSB is idle. Several nodes assert arbitration request lines and the node with the highest priority wins use of the bus. It will assert a C/A on the bus two countable cycles later as shown in Example 4–4.

Example 4–4 Best-Case Cycles Transaction (Repeated)

```
                 1111111111222222222233333333334444444444555555555556
CYCLE      12345678901234567890123456789012345678901234567890123456789O
ARB        ....1....2.............1....2....3....4.....................
C/A        ......1....2.............1....2....3....4...................
CMD        ......1....2.............1....2....3....4...................
CNF        .........1....2.............1....2....3....4................
SHR/DIR    ...........1....2.............1....2....3....4..............
STALL      ..............................................................
DATA       ................1111.2222...........1111.2222.3333.4444....
```

Another arbitration cycle cannot occur until five countable cycles (or more) after a previous arbitration cycle. In Example 4–4, the next cycle does occur on the fifth cycle (the minimum).

Arbitration Priority Scheme

Arbitration on the LSB is distributed. Each node monitors its arbitration status and calculates when it can use the LSB. Arbitration is performed using these LSB signal lines:

- REQ <9:0>
- STALL
- LOCKOUT
- NID<2:0>

The arbitration priority of nodes changes according to who wins and uses the bus. Each node implements a mechanism to monitor its place in the priority scale at any time. There are ten positions on the scale. PRIO<5,9:6,4:0> (highest to lowest). The corresponding level LSB arbitration request lines are REQ5, REQ9, ..., REQ6, REQ4, ..., REQ0.

The 7000/10000-600 system's IOP module has two fixed priority levels: REQ5 (highest) and REQ0 (lowest). When arbitrating, the IOP rotates its priority level using REQ5 (highest) six times and then REQ0 (lowest) two times. This action keeps other nodes from being locked out while giving the IOP the arbitration advantage.

At initialization nodes assume PRIO levels as shown in Table 4–1.

Table 4–1 Initial PRIO Assignment

NODE ID	PRIO Assignment	Corresponding REQ Signal Line
0	PR<1>	REQ1
1	PR<2>	REQ2
2	PR<3>	REQ3
3	PR<4>	REQ4
4	PR<6>	REQ6
5	PR<7>	REQ7
6	PR<8>	REQ8
7	PR<9>	REQ9
IOP	PR<5,0>	REQ5,REQ0

The modules arbitrating for use of the LSB use the following algorithm:

- When the IOP module wins use of the LSB, it still keeps PR<5,0>. No priority changes are made at other nodes.
- When a CPU wins use of the LSB, these changes in arbitration priority take place:
  - — The winning CPU releases its old arbitration level and assumes REQ1, the lowest CPU priority.
  - — A CPU with an arbitration level lower than the winning CPU's old level raises its level by one.
  - — A CPU with an arbitration level higher than the winning CPU's old level keeps its level unchanged.

STALL Signal

The STALL signal is used by a device that is unable to keep up with LSB demands. Usually, the node delays action on the bus while it gets data ready. Asserting STALL has the appearance of making LSB cycles stop.

The STALL signal can be asserted 9 cycles after the C/A cycle. The stall can then be held for any number of contiguous cycles, but long stalls are discouraged as LSB efficiency goes down. The maximum number of cycles for memory refresh is 13.

A STALL cycle causes nodes to be inactive on the LSB two cycles later. Nodes do not count the inactive cycles. STALL cycles have a fixed relationship to other cycles as shown in Example 4–5.

Example 4–5 STALL Cycles

```
CLOCK            1111111111222222222233333333334444444444555555555566
CYCLE     12345678901234567890123456789012345678901234567890123456789012
ARB       ....1....2.......--.......1....2....3....4....................
C/A       ......1....2.....--.........1....2....3....4..................
CMD       ......1....2.....--.........1....2....3....4..................
CNF       .........1....2..--............1....2....3....4...............
SHR/DIR   ..........1....2--..............1....2....3....4.............
STALL     ..............XX.....................................................
DATA      ................--1111.2222..........1111.2222.3333.4444....
```

LOCKOUT Signal

The synchronous LOCKOUT signal can be used by CPU modules to avoid deadlock situations. The signal line is provided by the LSB protocol but its use is implementation dependent. All CPUs using this line on the same LSB must implement the same algorithm

Usually LOCKOUT is asserted by a CPU module that has failed to gain access to a lock variable (for example: VAX 7000/10000-600 system byte-update sequence (BBSS) fails repeatedly). When one or more CPUs assert LOCKOUT, other CPUs delay new outgoing requests until LOCKOUT is released.

I/O and memory modules need not drive or receive this line.

Transactions

All transactions are allocated five cycles on the LSB:

- One C/A cycle (two cycles after arbitration)
- Four contiguous data cycles (starting 11 cycles later)

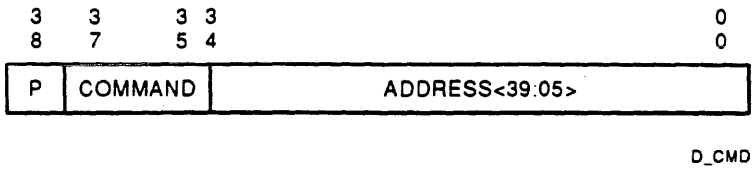
The following LSB signal lines are used during transaction cycles:

- C/A
- ECC<27:0>
- SHARED
- DIRTY
- CNF
- CRD
- D<127:0>
  - — Only D<38:0> are used during command and CSR cycles.

Command Cycle

All nodes monitor the LSB looking for a C/A cycle. The node winning arbitration asserts the C/A on the LSB two cycles later. The commander node asserts odd parity, command type and address on D<38:00> as shown in Figure 4–1. Parity covers D<37:00> while the state of D<127:39> and ECC<27:00> is arbitrary and is ignored.

Figure 4–1 Command Cycle Signals



During the C/A cycle the commander node also asserts its node ID (NID) on REQ <3:0>. This is done for debug and monitoring purposes.

The command field D<37:35> is coded as listed here:

•	000	READ	100	READ CSR
•	001	WRITE	101	WRITE CSR
•	010	RESERVED	110	RESERVED
•	011	WRITE VICTIM	111	PRIVATE

ECC Coding

ECC codes are asserted on ECC<27:00> during data cycles. The code allows nodes to detect and correct single-bit errors and to detect double-bit errors.

Seven ECC bits are used for each longword of data as listed here:

•	ECC<6:0>	D<31:00>
•	ECC<13:7>	D<63:32>
•	ECC<20:14>	D<95:64>
•	ECC<27:21>	D<127:96>

The cycle count is included in the ECC scheme to detect errors that could be caused by an open STALL line. The four groups of data and ECC bits use the same scheme shown in Figure 3-7 of the LASER Memory Specification for D<31:00> and ECC<7:0>.

ECC <3:2> are inverted after the calculation to distinguish null cycles from data cycles.

Write Transaction

During a write transaction the commander asserts its C/A on the LSB two cycles after winning arbitration. Eleven countable cycles later the commander asserts write data on four contiguous cycles. All node check parity on the C/A cycle and ECC on data cycles.

Write Victim Transaction

On the LSB, the write victim transaction appears identical to the generic write transaction. During write victim transactions, nodes which maintain caches do not probe their cache tags to see if the transaction affects cache coherency.

Read Transaction

During a read transaction, the commander asserts its C/A on the LSB two cycles after winning arbitration. Eleven countable cycles later the responder asserts read data on four contiguous cycles. All nodes check parity on the C/A cycle and ECC on data cycles.

The CRD signal line on the LSB is asserted by the responding memory node if the data was corrected (single-bit error). The action taken by the commander is implimentation specific.

CSR Read and Write Transactions

CSR transactions, like data transactions, use a C/A cycle. The CSR C/A cycle has the same format as the read/write C/A cycle. Eleven countable cycles later there are four contiguous data cycles.

The CSR data cycles differ from the read/write data cycles as follows:

- Only the first data cycle has valid data
  - — D<38:0> are used—parity checked by receiving node
  - — D<127:39> are not used—ignored by all nodes
- Three data cycles follow and are counted but not used
  - — Parity and ECC are not checked

The four Mailbox Pointer Registers on the IOP module, (LMBPR0-3), have 38 data bits while all other system registers have 32 bits.

NOTE

CSR and private transactions are all treated as accesses to Bank 0 for bank contention purposes. Therefore, there can only be one access in 15 cycles.

CSR reads and writes actively use only two of the five cycles, as shown in Example 4–6, although all five cycles are countable.

Example 4–6 CSR Cycles

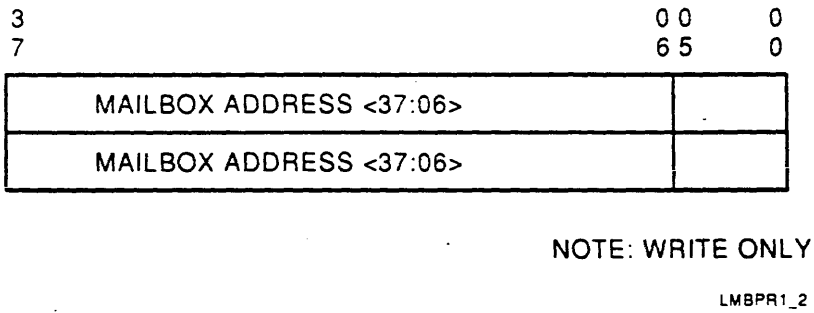
```
                  1111111111222222222233333333334444444444555555555556
CYCLE     12345678901234567890123456789012345678901234567890123456789
ARB       ....1....2.............1....2....3....4....................
C/A       ......1....2.............1....2....3....4..................
CMD       ......1....2.............1....2....3....4..................
CNF       .........1....2.............1....2....3....4...............
SHR/DIR   ...........1....2.............1....2....3....4.............
STALL     ...........................................................
DATA      ................1xxx.2222.........1111.2xxx.3333.4444....
```

Mailbox Transaction

All transfers through the IOP to or from other buses (such as XMI bus) are accomplished using mailboxes. CPUs can handle mailbox transfers. The sequence of steps for a mailbox transfer follows:

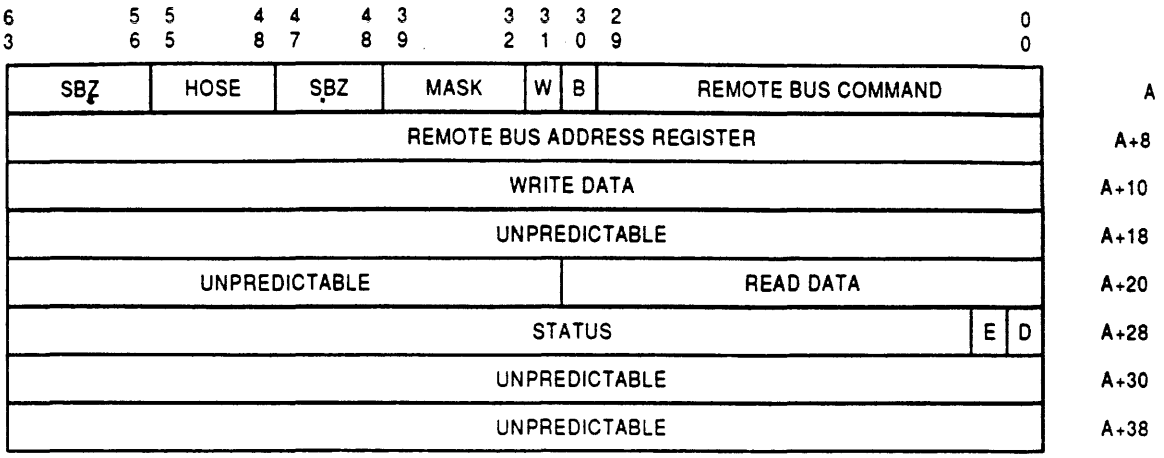
- CPU (0-3) creates a mailbox structure in memory as shown in Figure 4–3 (64 bytes—one transaction)
- CPU (0-3) writes a QW to the corresponding mailbox pointer register in the IOP (LMBPR0-3, one per CPU) as shown is Figure 4–2. Each LMBPRn register accepts and stores the data for two write transactions (hidden queue).
  - — If the selected LMBPRn is holding two writes the IOP does not assert CNF. The CPU must retry.

Figure 4–2 Mailbox Pointer Registers 0-3 (LMBPRn)



- CPU starts polling the mailbox looking for DONE bit set
- IOP and LAMB perform mailbox operation
  - — IOP notices write to LMPRn and reads packet
    - * Makes up mailbox packet
    - * Sends packet down selected hose
  - — LAMB accepts mailbox packet
    - * Performs requested task
    - * Sends mailbox reply packet up hose to IOP with DONE bit set
  - — IOP accepts mailbox reply packet and write information to mailbox in memory
- CPU sees the DONE bit set and checks the ERROR bit and status longword and then it takes action.

Figure 4–3 Mailbox Data Structure



NOTE

MASK=WRITE BYTE MASK
W=WRITE
B=RETURNS ID OF BRIDGE (I/O MODULE) FROM WHAMI REGISTER
D=DONE
E=ERROR

MAILBOX_Y1_X0_9

Private Transaction

The private transaction is used by nodes that want to use LSB cycles for local operations. Private transactions have good odd parity over D<37:00> during the command cycle. Other modules ignore the contents of the data cycles but count them for arbitration and timing purposes.

The commander module:

- Must assert CNF during the proper cycle
- Can drive arbitrary data on the bus during the data cycles
- Can arbitrarily drive SHARED and DIRTY lines during their cycle
- Can assert STALL during the transaction (operates the same when used during read or write transactions)

The VAX 7000/10000-600 system processor initiates private writes to the GBus when it is in diagnostic mode. For example, it writes to the 8 kB EEPROM by way of the LSB.

NOTE

CSR and private transactions are all treated as accesses to bank 0 for bank contention purposes. There can be only one access in 15 LSB cycles.

Interrupts

There are two types of interrupts defined in LSB protocol: vectored and implied.

- **Vectored Interrupts**
  - — Only CPUs at four nodes can service interrupts. NID bits are used to select Nodes 0, 1, 2, 3.
  - — The operating system loads a vector into each I/O port vector register when the system is initialized.
  - — The operating system loads a mask into each I/O port CPU mask register when the system is initialized.
  - — Later the vector is retrieved by the CPU servicing the interrupt if the mask allows. The vector points the CPU to the correct service routine.
- **Implied interrupts**—the CPU uses a dedicated mechanism to enter the correct service routine.

Both types of interrupts are implemented using writes to registers located in broadcast space.

Vectored Interrupts

Vectored interrupts use the LIOINTR (CPU), LCPUMASK (IOP), and LILIDX (IOP) registers as shown in Figure 4–4, Figure 4–5, and Figure 4–6.

Figure 4–4 LIOINTR Register (CPU)

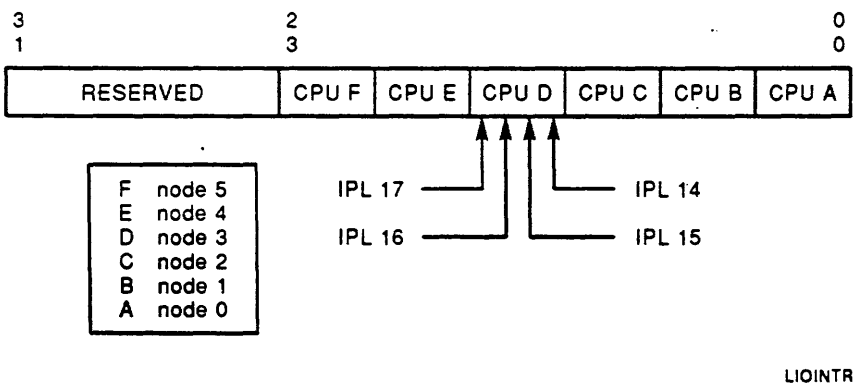


Figure 4–5 LCPUMASK Register (IOP)

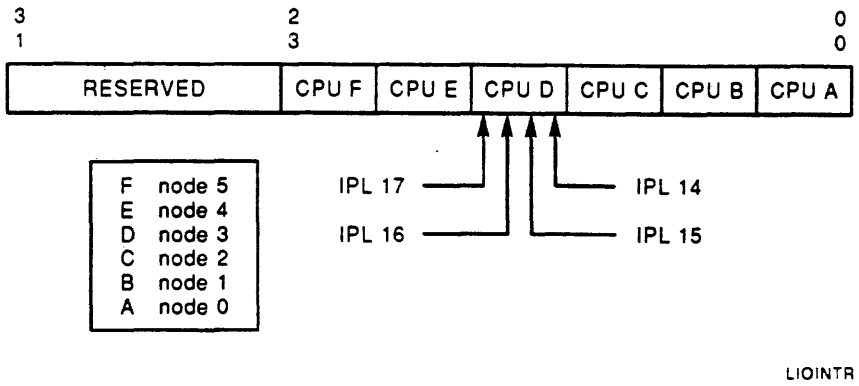
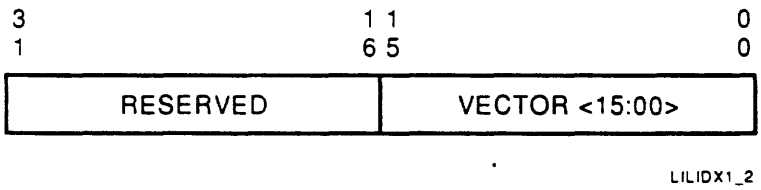


Figure 4–6 LILIDX Register (IOP)



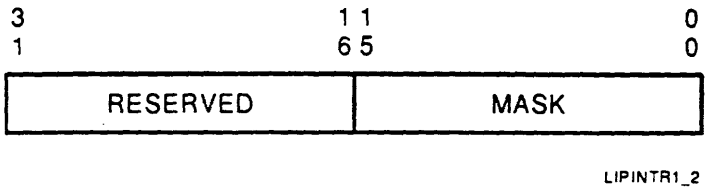
The sequence of events for a vectored interrupt is as follows:

- Interrupting node writes in broadcast space to the LIOINTR Registers on the CPU modules.
  - — Interrupting node assembles IPL level, target CPUs in format like the LCPUMASK register bits
  - — ANDs data with LCPUMASK register and performs a transaction to LIOINTR register (broadcasts to all CPUs)
  - — CPUs assert CNF
- The target CPUs respond at a later date by reading the LILIDx register on the IOP.
  - o The *x* in LILIDx indicates the correct register (0, 1, 2, 3) for interrupt levels IPL14, IPL15, IPL16, or IPL17. The LILIDx register is the top entry in the queue for that IPL.
  - o The I/O node responds with the vector in the (LILIDx register). If the register (queue) is empty, the node returns to zero allowing a passive release.
  - o I/O node considers task completed.

Implied Interrupts

Implied interrupts use the LIPINTR register shown in Figure 4–7.

Figure 4–7 LIPINTR Register



The sequence of events for an implied interrupt follows:

- A node uses broadcast space to write its mask bit in all CPUs LIPINTR registers. Mask bits 3:0 correspond to nodes 3:0.
- The receiving CPUs notice the set bit and enter the correct interrupt service routine.

Interlocks

The LSB protocol does not support any type of interlock command pair such as the INTERLOCK READ and UNLOCK WRITE traditionally found on memory interconnects. The interlock mechanism used on the LSB is implemented by a protocol among commander nodes. The memory node does not actively participate in the protocol.

CPU Interlocks

The mechanism used by CPUs takes place in these steps:

- A CPU wants to perform a read-modify-write operation on a variable Q.
- The CPU fetches the variable Q and stores the address of Q in a LOCK register together with a set LOCK bit.
- The CPU monitors the LSB for writes to the block in memory that contains the variable Q.
  - — If the CPU sees a write to the block that contains Q, it must clear the LOCK bit stored with Q's address.
- When the processor is done working with the variable Q, it attempts to write Q to memory.
  - — If; and only if, the LOCK bit is still set in the LOCK register, the CPU performs the write to memory and reports success.
  - — If the LOCK bit in the LOCK register is clear, the CPU reports failure and will not perform the write to memory.
  - — The CPU will branch and replay the whole operation until it completes successfully.

I/O Interlocks

An emulation of the INSQTI/REMQTI VAX instruction is also required for older VAXport style I/O devices (CIXCD is presently the only such device to be used with the LSB).

VAXport devices use interlock operations only to add and remove entries to or from a VAX self-relative interlock queue structure. These operations can be emulated by the IOP as follows:

- Get the secondary lock by reading the original data from the LSB and writing it back to the LSB setting bit <0>.
  - — If bit <0> is clear, do nothing.
  - — If bit <0> is set, then it is already owned. Replay this operation.
- Release the secondary lock by reading the data from the LSB again and writing it back to LSB setting bit <0>.
  - — If bit <0> is clear, then an error has happened.
  - — If bit <0> is set, then it is already owned. Write data with clear bit <0> to LSB.

Conditional Update Writeback Cache Support

The LSB protocol includes provisions for conditional update writeback cache. The LSB provides the two signal lines SHARED and DIRTY for this purpose. The LSB provides no other physical features to support conditional update writeback caches.

The LSB protocol requires that these two signal lines, when asserted, are asserted five cycles after the command/address cycle as shown in Example 4–7.

Example 4–7 Best-Case Cycles Transaction (Repeated)

```
                 1111111111222222222233333333334444444444555555555556
CYCLE      12345678901234567890123456789012345678901234567890123456789
ARB        ....1....2.............1....2....3....4.....................
C/A        ......1....2.............1....2....3....4...................
CMD        ......1....2.............1....2....3....4...................
CNF        .........1....2.............1....2....3....4................
SHR/DIR    ..........1....2.............1....2....3....4.............
STALL      ...........................................................
DATA       ................1111.2222..........1111.2222.3333.4444....
```

LSB modules implement the conditional update writeback cache protocol as listed here:

- Modules that have cache monitor the SHARED and DIRTY lines, taking actions to support the LSB protocol. The actions of the CPU are described in Chapter 6.
- Memory modules monitor the DIRTY line and take action to support the LSB protocol. The actions of the memory modules are described in Chapter 5.
- The IOP module is unaware of the protocol and does not monitor the SHARED and DIRTY lines.

Console Support

The LSB protocol supports the system console by providing these lines:

- CONWIN—asserted by the primary CPU
- LOCRX and LOCTX—local console serial communications lines, monitored and asserted by primary CPU
- RUN—asserted by primary CPU when at least one CPU is running the operating system (not in console mode)
- SECURE—signal asserted from OCP keyswitch, console ignores ^P from console terminal
- PSRX and PSTX—48 Vdc power supply communications lines to and from the three possible cabinets
- EXP_SEL<1:0>—selects which cabinet 48 Vdc power supplies will be available on PSRX and PSTX lines as listed in Table 4–2

Table 4–2 Cabinet Power Monitor Selection

EXP_SEL<1:0	Path Selected
0 0	Main cabinet
0 1	Right cabinet
1 0	Left cabinet
1 1	Self-test loopback. PSTX goes to PSRX.

Errors

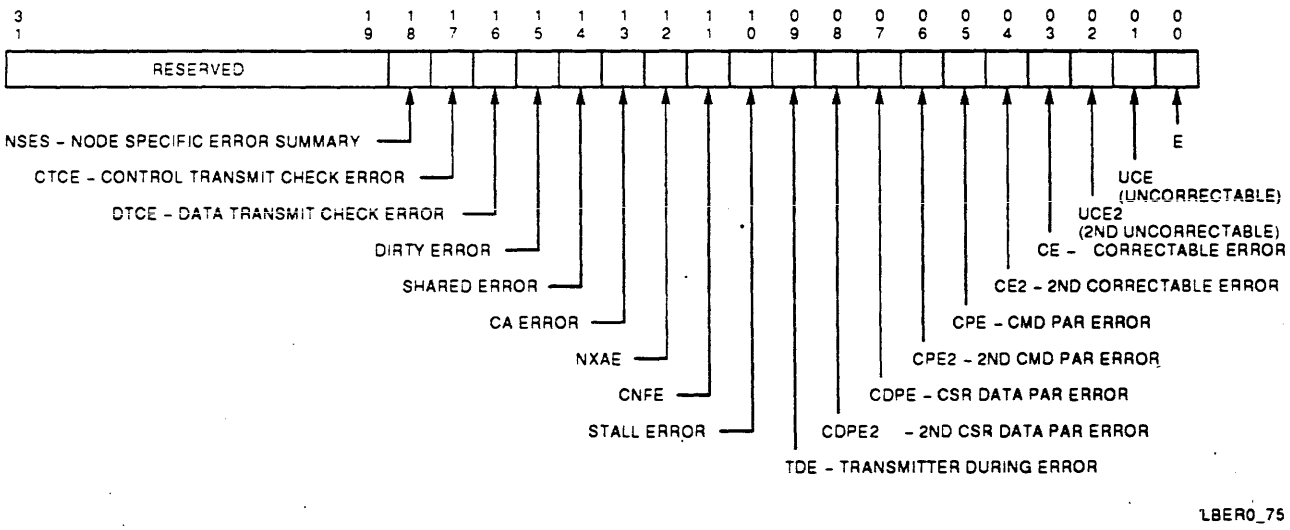
The LSB protocol requires nodes detecting errors to capture the error information, report the error, and continue processing the transaction except for parity errors during C/A cycles. Nodes detecting a C/A parity error capture and report the error but ignore the transaction.

The response of nodes to error bits set in their LBER is module specific. Memory nodes take action that is hardware and firmware dependent. CPU nodes take appropriate action that varies depending on the type of CPU and which operating system is used.

Using the LBER Register

The LBER register, as shown in Figure 4–8, is used for recording LSB errors on each module.

Figure 4–8 LBER Register



LSB ERR Signal Line

All nodes monitor the LSB ERR signal line. A node takes the following actions when LSB ERR is asserted:

- Sets LBER bit E, LBER<0>.
- Inhibits outgoing arbitration for the next 16 cycles.
- Resets its arbitration level as shown in Table 4–1.
- If the node is a CPU, it performs a trap or interrupt if enabled allowing software to handle the error.

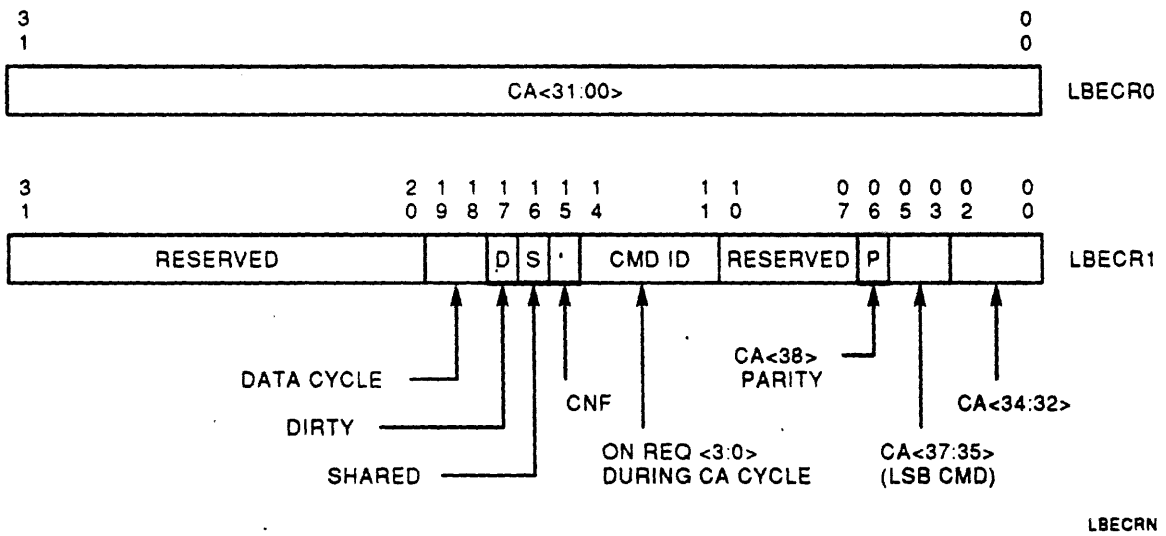
Uncorrectable ECC Error

All nodes monitor memory data cycles for ECC errors. If a node detects an uncorrectable data error, it performs the following actions:

- Asserts LSB ERR for one cycle within four cycles after detection
- Sets UCE bit, LBER<1>
- Captures ECC syndromes in LBESR0 through LBESR3 shown in Figure 4–10
- Captures the associated C/A cycle in LBECRO and LBECR1 shown in Figure 4–9
- Captures the CNF, SHARED, and DIRTY status associated with the command in LBECR1
- Captures the data cycle number (0, 1, 2, 3) of the uncorrectable data in LBECR1

Two LASER bus error command registers (LBECRn) are used to capture the C/A associated with the data ECC error. LBECR0 and LBECR1 are shown in Figure 4–9.

Figure 4–9 LBECRn Register



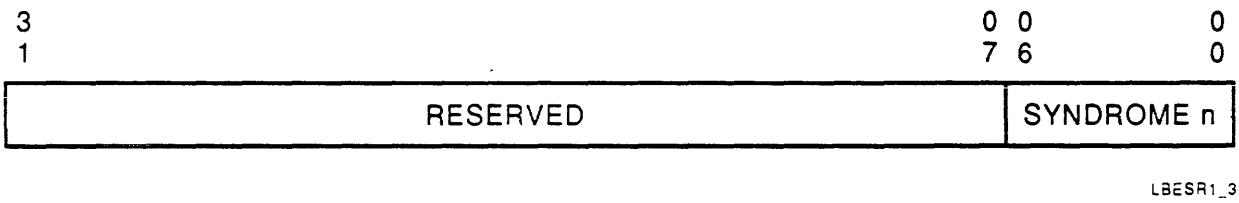
Correctable ECC Error

All nodes monitor memory data cycles for ECC errors. If a node detects a correctable data error and the correction enable bit (CEEN) in its LBCNF is set to allow action, it performs the following actions:

- Sets the CE bit, LBER<03>
- Captures ECC syndromes in LBESR0 through LBESR3
- Captures the associated C/A cycle in LBECRO and LBECR1
- Captures the CNF, SHARED, and DIRTY status associated with the command in LBECR1
- Captures the data cycle number (0, 1, 2, 3) of the uncorrectable data

The four LBESR registers have the same format, as shown in Figure 4–10, and are used to capture the ECC syndrome bits for a data cycle with an ECC error.

Figure 4–10 LBESR Register



The syndrome bits for the four longwords on the bus during the data cycle are in the registers listed here:

- LBESR0 ECC<6:0> for D<31:00>
- LBESR1 ECC<13:7> for D<63:32>
- LBESR2 ECC<20:14> for D<95:64>
- LBESR3 ECC<27:21> for D<127:96>

C/A Parity Errors

All nodes monitor the LSB for C/A parity errors. A node detecting an odd parity error over D<38:00> when C/A is asserted will:

- Assert LSB ERR for one cycle within four cycles after detection
- Set the CPE bit, LBER <05>
- Capture D<38:00> in its LBECR0 and LBECR1
- Ignore the remainder of the transaction (does not check for parity or ECC on data cycles)

CSR Data Parity Errors

CPU and I/O nodes monitor all CSR data cycles for parity errors. If a node detects a parity error, it performs the following actions:

- Assert LSB ERR for one cycle within four cycles after detection
- Sets the CDPE bit, LBER<07>
- Captures the associated C/A cycle in LBECRO and LBECR1
- Captures the CNF, SHARED, and DIRTY status associated with the command in LBECR1
- Sets the data cycle field in LBECR1 to zero

Double Errors

For each CE, UCE, CPE, and CDPE, there is a corresponding second error bit CE2, UCE2, CPE2, and CDPE2. The second error bit will be set if another error of the same type occurs before the first error is cleared.

Transmitter During Error

A node that is driving the bus when a parity or ECC error happens sets the TDE bit, LBER<09>. The TDE bit will be set along with either the CE, UCE, CPE, or CPDE if the node is driving the bus when the error was detected.

STALL Errors

A node detecting STALL asserted during a cycle when STALL is not permitted will assert LSB ERR for one cycle within four cycles after detection and set the STE bit, LBER<10>.

CNF Errors

A node detecting CNF asserted during a cycle when CNF is not permitted asserts LSB ERR for one cycle within four cycles after detection and sets the CNFE bit, LBER<11>.

Nonexistent Address Errors

A commander detecting CNF not asserted during a cycle when CNF should be asserted asserts LSB ERR for one cycle within four cycles after detection and sets the NXAE bit, LBER<12>.

A commander detecting CNF not asserted during an access to LMBPRn in the IOP ignores it and does nothing. When LMBPRn in the IOP is already in use, the IOP ignores transactions to the register.

C/A Errors

A node detecting C/A asserted during a cycle when C/A is not permitted asserts LSB ERR for one cycle within four cycles after detection and sets the CAE bit, LBER<13>.

SHARED Errors

A node detecting SHARED asserted during a cycle when SHARED is not permitted asserts LSB ERR for one cycle within four cycles after detection and sets the SHE bit, LBER<14>.

DIRTY Errors

A node detecting DIRTY asserted during a cycle when DIRTY is not permitted asserts LSB ERR for one cycle within four cycles after detection and sets the DIE bit, LBER<15>.

Data Transmit Check Errors

A node driving a command cycle or a data cycle onto the LSB must also receive the same bits and check them against what was transmitted. If there is a mismatch, the node sets the DTCE bit, LBER<16>.

Control Transmit Check Errors

A node driving any LSB control line must check the received status of that control line at the end of the cycle. If there is a mismatch, the node sets the CTCE bit, LBER<17>.

Node-Specific Error Summary

A node detecting an internal error may set the NSES bit in its LBER. The node is not required to assert LSB ERR but may do so.

LASER System Bus Exercise

1. The LASER system bus (LSB) has four interrupt lines, one each for BR4, BR5, BR6, and BR7.

   a. True

   b. False

2. There are three distinct errors in Example 4–8. Mark the errors and note the corrections.

Example 4–8 Problem 2 Transaction

```
                  1111111111222222222233333333334444444444555555555556
CYCLE     12345678901234567890123456789012345678901234567890123456789
A.B       .....1....2.............1....2....3....4..................
C/A       ......1....2.............1....2....3....4..................
CMD       ......1....2.............1....2....3....4..................
CNF       .........1....2.............1....2....3....4..............
SHR/DIR   ..........1....2.............1....2....3....4............
STALL     ..................................................
DATA      ................11...22............1111222233334444....
```

3. Nodes are arbitrating to use the LSB. The nodes won use of the bus in the sequence: 3, 1, 3, 2, 0. Nodes 0, 1, and 3 now arbitrate for use of the bus. Which one will gain use of the bus next?

   a. Node 0

   b. Node 1

   c. Node 2

   d. Node 3

   e. None of the above

4. Five LSB cycles are allocated for each LSB transaction.

   a. True

   b. False

5. At which priority does the IOP module arbitrate?

   a. 9

   b. 0

   c. 0, 5

   d. 0, 5, 9

   e. None of the above

6. Which node register and bit is the primary indicator of an LSB error?

   a. LDEV register bit 31

   b. XBER register bit 00

   c. LBER register bit 00

   d. BIU_STAT register bit 00

   e. None of the above

7. A node detects a correctable ECC error on the LSB. LBESR0 through LBESR3 of that node contains C, C, C, 19 respectively. Which LSB bit was in error?

   a. LSB D<51>

   b. LSB D<115>

   c. LSB D<147>

   d. LSB D<31>

   e. None of the above

LSB Memory Operation and Error Registers

Introduction

This module introduces the student to the memory array modules used on the LSB. The DSE becomes familar with the following characteristics of the LMEM modules:

- Longword ECC (LSB) with SBE correction and DBE detection.
- Quadword ECC (DRAM) with SBE correction and DBE detection.
- Power on self-test.
- Error detection and error status registers.

Objectives

A Digital Services Engineer who maintains a VAX 7000-600 or VAX 10000-600 system should be able to:

- Be familiar, at block level, with the tasks performed by the LMEM module.
- Recognize the sequence in which the LMEM module performs its tasks.
- Use the contents of LMEM module registers as an aid while diagnosing a system failure.

Resources

MS7AA Memory Technical Manual	EK-MS7AA-TM

Overview of the LASER Memory Module (LMEM)

The LASER memory module (LMEM) is designed specifically for use on the LASER system bus (LSB). Using various populations of 4 Mb memory chips, the LMEM has the memory capacity listed in Table 5–1.

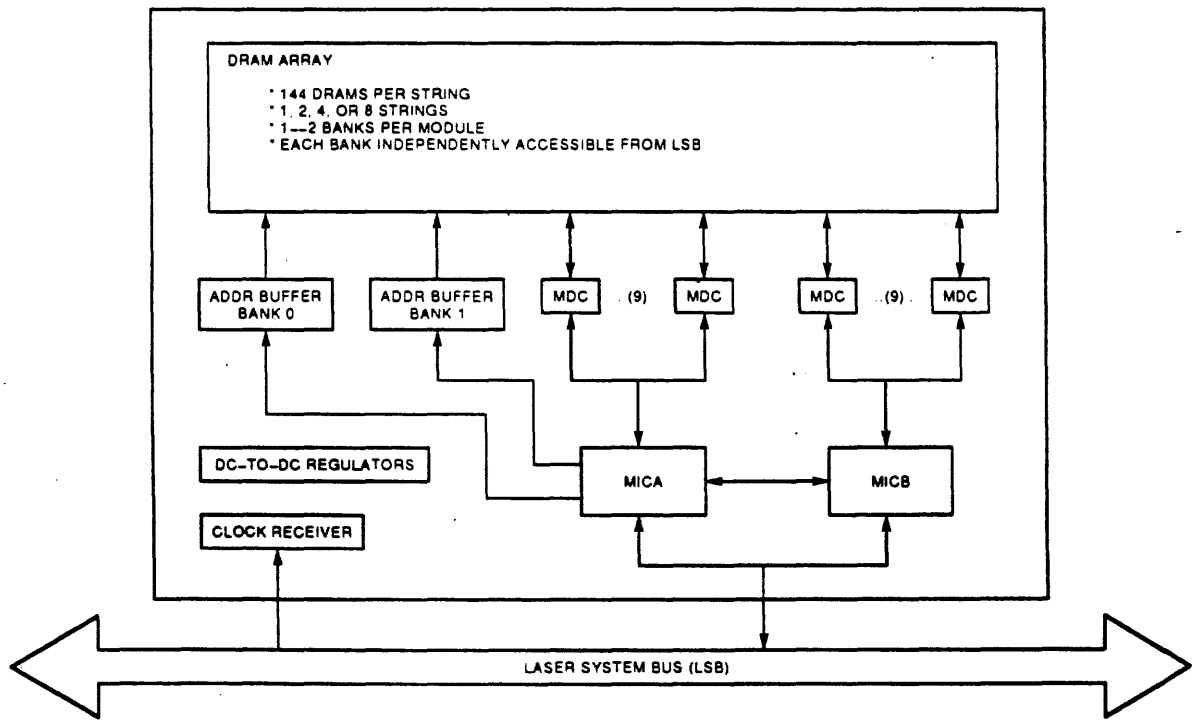
Table 5–1 LASER Memory Module Options

Option	Module	Memory Chip	Memory Size	Strings
MS7AA-AA	E2043-AA	4 Mb	64 MB	1
MS7AA-BA	E2043-BA	4 Mb	128 MB	2
MS7AA-CA	E2043-CA	4 Mb	256 MB	4
MS7AA-DA	E2046-AA	4 Mb (SIMMs)	512 MB	8

Simple Functional Description

A simplified illustration of the LMEM is shown in Figure 5–1.

Figure 5–1 LSB Memory Module Block Diagram



The LMEM responds to reads and writes to its memory address space and to CSR read and writes to its node space. It does not respond to broadcast space addresses.

- **Memory interface controller (MIC—MICA, MICB)**

  MIC is composed of two CMOS gate arrays (MICA controls MICB) that:

  — Provide LSB interface

  — Control DRAM timing and refresh (13 cycles maximum)

  — Provide address and control signals for the two DRAM banks by way of address buffers

  — Provide 4 x 16 bytes read and write buffers

  — Perform read data path checking and conversion

  — Perform write data path checking and conversion

  — Contain all LSB and LMEM specific registers

  — Run memory self-test

  — Control use of SERIAL EEPROM

  — ECC for read and write data path checking

- **DRAM data buffers (MDC) and DRAM arrays**

  — Interface between a 128-bit data path and a 512-bit data path.

  — A DRAMs string has 144 DRAMs that are accessed together.

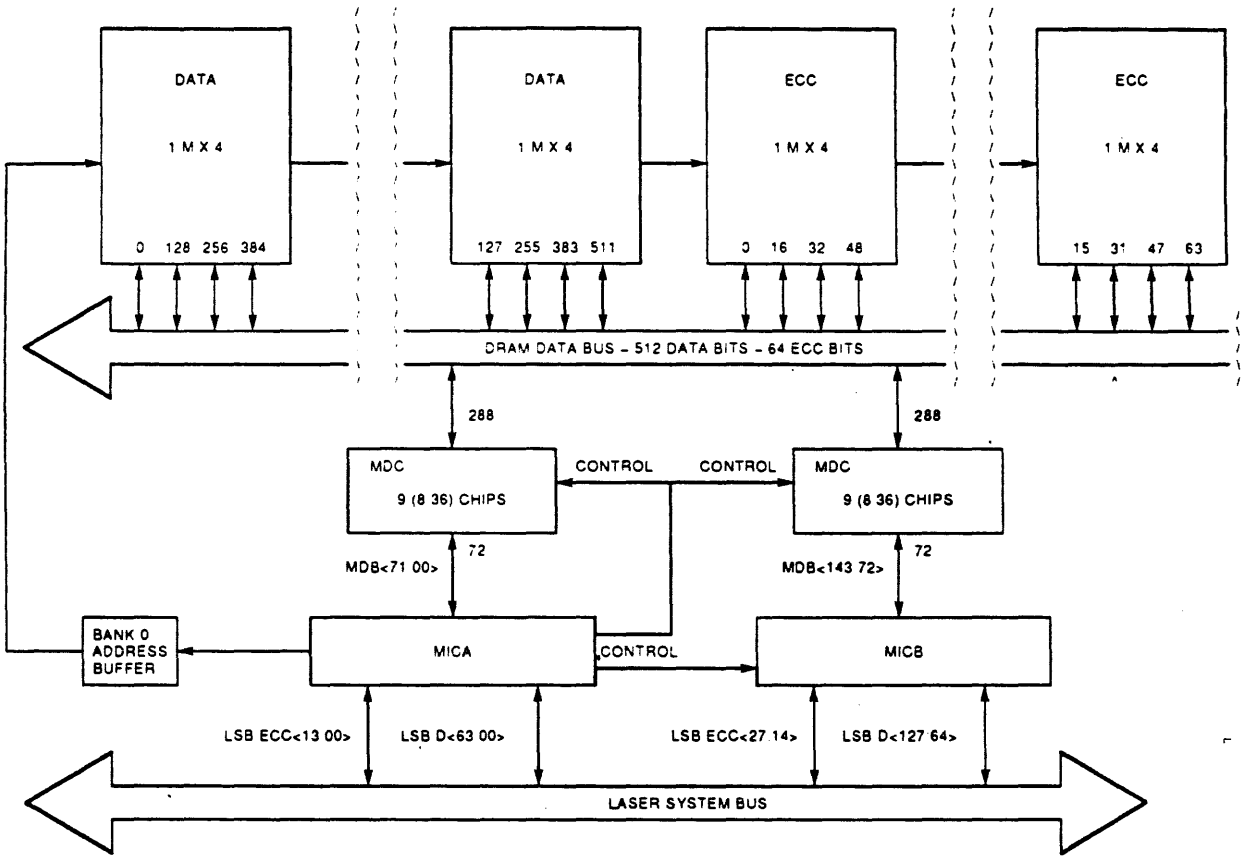
  — DRAMs strings are organized in banks to support interleaving. At least two strings (1 per bank) are required for interleaving (two-way).

- **DC-to-DC power regulator**

Memory Data Storage

Memory data is stored in 64-byte data units at naturally aligned 64-byte addresses. Sixty-four ECC bits are stored with each 64-byte data unit. A 64 MB LMEM module is shown in Figure 5–2. It has one bank in use, but if the other bank was populated with chips, it would be a 128 MB module with two strings and two banks.

Figure 5–2 LSB Memory Module Data Flow



MEM_DATA_FLOW70

LMEM Use of LSB Signal Lines

The LASER memory module (LMEM) implements LSB signal lines as specified by the LSB protocol. LMEM uses some LSB signal lines as follows:

- LSB CRD—asserted with corrected data from the arrays (single-bit-error).
- LSB STALL—when refresh cycles are active LMEM asserts the STALL line for 1 to 14 cycles (refresh takes 13 LSB cycles).
- LSB CA—LMEM is never a commander, so it never asserts LSB CA.
- LSB SHARE—LMEM does not monitor or assert SHARE.
- LSB DIRTY—LMEM monitors but does not assert LSB DIRTY.
- LSB REQ<5:0>—although LMEM does not arbitrate for the LSB, it monitors these lines. When any of these signal lines are asserted the LMEM turns on the address lines to the DRAMs. This is a power-saving measure.

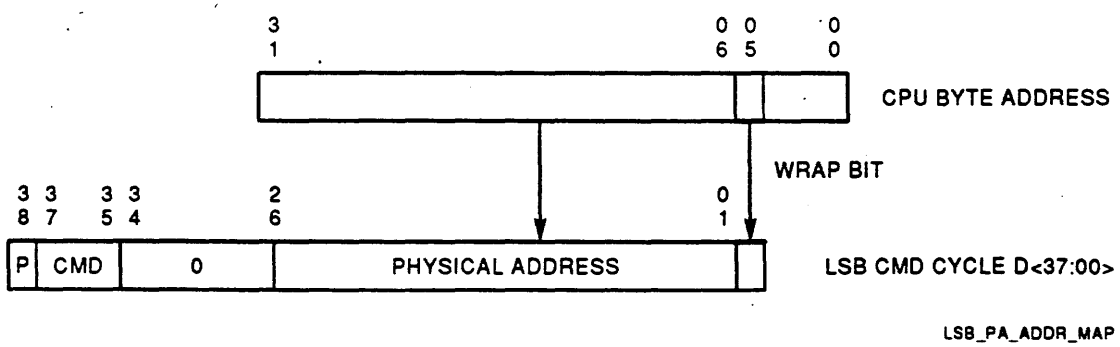
NOTE

The LNP processor does not respond to CRD but should poll for a memory CRD error about once per second.

Mapping Physical Addresses to LSB

VAX 7000/10000-600 system physical addresses are mapped onto the LSB as shown in Figure 5–3.

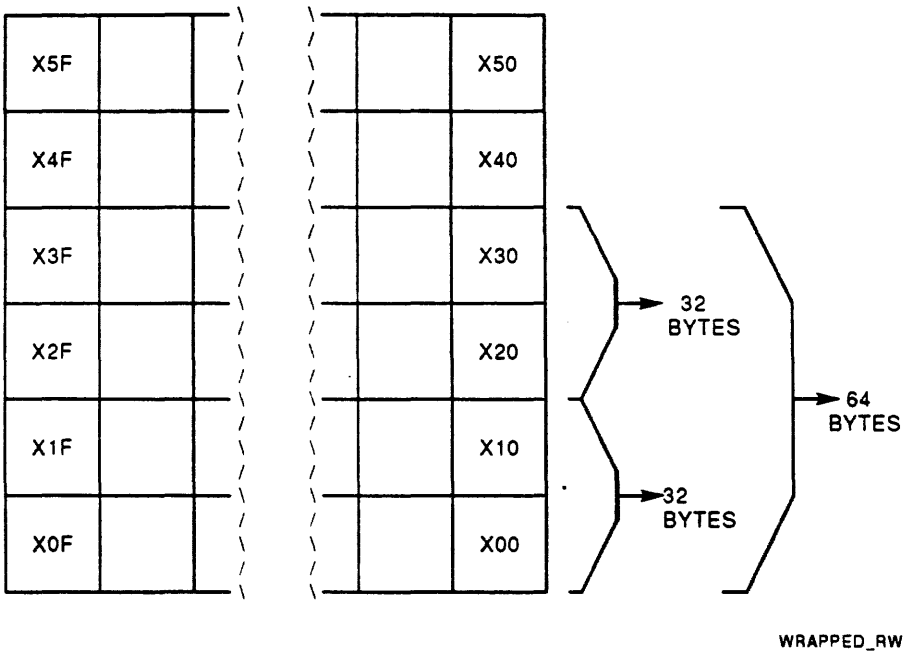
Figure 5–3 VAX 7000/10000-600 System Address Mapping to LSB



Wrapped Memory Reads and Writes

The LASER systems support wrapped read and write transactions on the LSB. The wrap operation is controlled by CPU address <05>. It separates hexwords and its significance is shown in Figure 5–4.

Figure 5–4 LSB Systems Wrapped Reads and Writes



The data for a read to the lower hexword (CPU ADDR <05> = 0) is returned in the sequence: X00, X10, X20, X30.

The data for a wrapped read to the higher hexword (CPU ADDR <05> = 1) is returned in the sequence: X20, X30, X00, X10.

Conditional Update Writeback Cache

LSB protocol does not burden the memory module with much support for conditional update writeback cache. The memory module:

- Ignores the SHARED LSB signal line.
- Monitors the DIRTY LSB signal line. When DIRTY is asserted during READ transactions, the memory lets the node that asserted DIRTY provide the data.

LASER Memory Module (LMEM) Operation

The LASER memory module (LMEM) responds to LSB reads and writes to memory addresses. The LMEM is never a commander on the LSB and never explicitly initiates interrupts. The following two scenarios illustrate how the LMEM in Node 7 performs its tasks during memory reads and writes.

LSB READ

The sequence for a typical LSB memory read operation follows:

1. CPU 0 arbitrates for use of the LSB and wins on LSB cycle 3.
2. On LSB cycle 5, CPU 0 asserts a C/A to read memory address A.
3. LMEM 7 receives the C/A and checks address A with its address mapping register (AMR). The address is on LMEM 7.
4. LMEM 7 asserts CNF on LSB cycle 8.
5. LMEM 7 notices that the DIRTY line is not asserted on LSB cycle 10.
6. MICA on LMEM 7 drives address and timing signals to the DRAM array.
7. DRAM array puts 512 data bits and 64 ECC bits onto the DRAM data bus.
8. The MDC chips for a bank takes the data and ECC bits.
9. The MCA chips take the data and ECC bits for the four cycles into their 4 x 72-bit read buffers.
   a. They take 128 data bits and 16 ECC bits for data cycle 0. Perform check with data and ECC bits. Correct if necessary and possible.
   b. Convert 16 bits of 64-bit ECC to 28 bits of 32-bit ECC.
      - If uncorrectable MICA or MICB inverts its 14 bits of ECC, flagging the data as uncorrectable.
      - Assert onto the LSB during cycle 16.
      - Assert LSB CRD if corrected error.
   c. Repeat for data cycle 1. Assert onto the LSB during cycle 17.
   d. Repeat for data cycle 2. Assert onto the LSB during cycle 18.
   e. Repeat for data cycle 3. Assert onto the LSB during cycle 19.
10. Done!

LSB WRITE and VICTIM WRITE

The sequence for a typical LSB memory write operation follows:

1. CPU 0 arbitrates for use of the LSB and wins on LSB cycle 3.
2. On LSB cycle 5, CPU 0 asserts a C/A writing to memory address A.
3. LMEM 7 receives the C/A and checks address A with its address mapping register (AMR). The address is on bank 0 of LMEM 7.
4. LMEM 7 asserts CNF on LSB cycle 8.
5. The MCA chips take the data and ECC bits for the four cycles into their 4 x 78-bit write buffers.
   a. During LSB cycle 16, the MCA chips take the 128 data and 28 ECC bits for data cycle 0. Perform check with data and ECC bits. Correct if necessary and possible. Convert from 28-bit ECC to 64-bit ECC. If uncorrectable MICA or MICB will invert its 8 bits of ECC, flagging the data as uncorrectable. Pass to MDC chips for bank 0.
   b. During LSB cycle 17 repeat for data cycle 1.
   c. During LSB cycle 18 repeat for data cycle 2.
   d. During LSB cycle 19 repeat for data cycle 3.
6. MDC chips put 512 data bits and 64 ECC bits onto the DRAM data bus.
7. MICA drives address and timing signals to the DRAM array.
8. DRAM array takes 512 data bits and 64 ECC bits onto the DRAM data bus and stores in DRAM chips.
9. Done!

LMEM Registers

LMEM maintains two sets of registers in its node space: LSB required registers and LMEM node-specific registers. Remember all LSB node CSRs are aligned on natural 64-byte boundaries.

LMEM acknowledges (CNF) receipt of a C/A for a CSR transaction to its node space whether the register exists or not. CSR reads to a nonexistent register produces UNDEFINED data. CSR writes to a nonexistent register causes an UNDEFINED operation.

LSB Required Registers

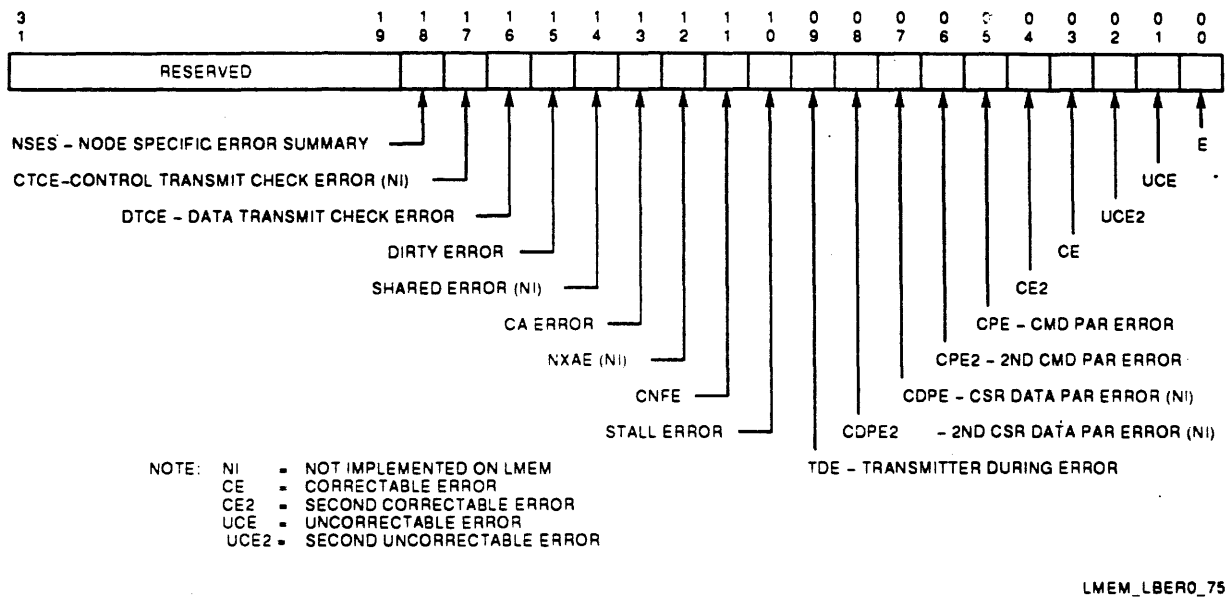
The LSB protocol requires the following registers to be maintained by memory nodes on the LSB as shown in Table 5–2.

Table 5–2 LSB Memory Required CSRs

Byte Offset	Mnemonic	Register Name
BB+000	LDEV	LASER Device Register
BB+040	LBER	LASER Bus Error Register
BB+080	LCNF	LASER Configuration Register
BB+0C0	LIBR	LASER Information Base Repair Register
BB+600	LBESR0	LASER Bus Error Syndrome Register 0
BB+640	LBESR1	LASER Bus Error Syndrome Register 1
BB+680	LBESR2	LASER Bus Error Syndrome Register 2
BB+6C0	LBESR3	LASER Bus Error Syndrome Register 3
BB+700	LBECR0	LASER Bus Error Command Register 0
BB+740	LBECR1	LASER Bus Error Command Register 1

The LMEM module supports the LBER register bit in conformance to LSB protocol except for five bits that are not implemented as shown in Figure 5–5.

Figure 5–5 LMEM LBER Register



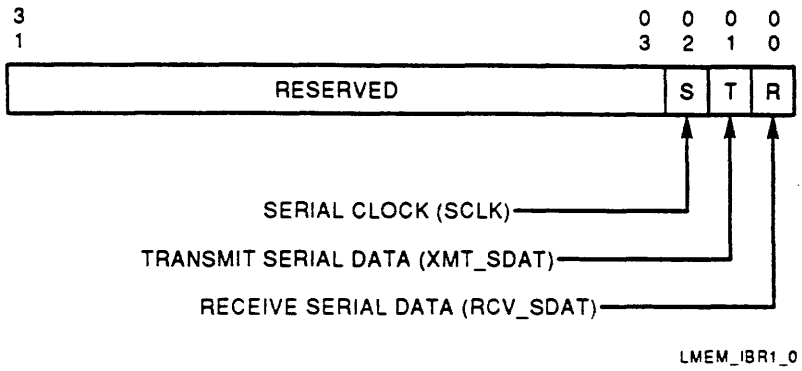
NSES BIT

When an error occurs that causes an error bit to be set in one of the LMEM specific registers the Node Specific Error Summary bit (LBER <18>) is also set.

The LASER Information Base Repair Register (LIBR), shown in Figure 5–6, is used by software to access the LMEM Serial EEPROM (SROM). The LMEM SROM is used to store such information as:

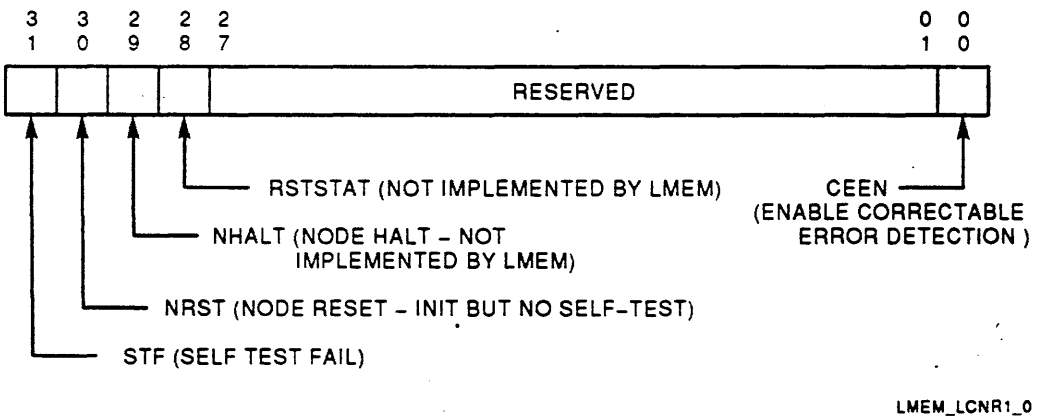
- Serial number of the module
- Module revision
- LMEM error log information to be used at field repair depot

Figure 5–6 LMEM IBR Register



The LMEM module also implements the LASER Configuration Register (LCNR) as shown in Figure 5–7.

Figure 5–7 LMEM LSB Configuration Register



NOTE

The LMEM performs a self-test at power on and LSB RESET. It does not perform a self-test in response to a node reset.

LMEM Specific Registers

In addition to registers required by LSB protocol, the LMEM also maintains the node specific registers listed in Table 5–3.

Table 5–3 LSB Memory Specific CSRs

Byte Offset	Mnemonic	Register Name
BB+2000	MCR	Memory Configuration Register
BB+2040	AMR	Address Mapping Register
BB+2080	MSTR0	Memory Self-test Register 0
BB+20C0	MSTR1	Memory Self-test Register 1
BB+2100	FADR	Failing Address Register
BB+2140	MERA	Memory Error Register A[1]
BB+2180	MYSNDA	Memory Syndrome Register A[1]
BB+21C0	MDRA	Memory Diagnostic Register A[1]
BB+2200	MCBSA	Memory Check Bit Substitute Register A[1]
BB+4140	MERB	Memory Error Register B[2]
BB+4180	MSYNDB	Memory Syndrome Register B[2]
BB+41C0	MDRB	Memory Diagnostic Register B[2]
BB+4200	MCBSB	Memory Check Bit Substitute Register B[2]

[1]Contained in MICA
[2]Contained in MICB

MCR and AMR Registers

The Memory Configuration Register and Address Mapping Registers, which were described in Memory Module Interleaving Registers, are shown in Figure 5–8 and Figure 5–9.

Figure 5–8 Memory Configuration Register (MCR)

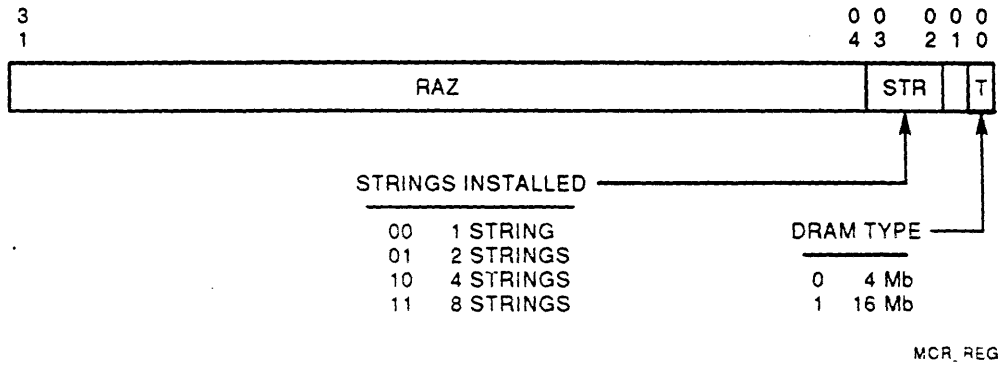
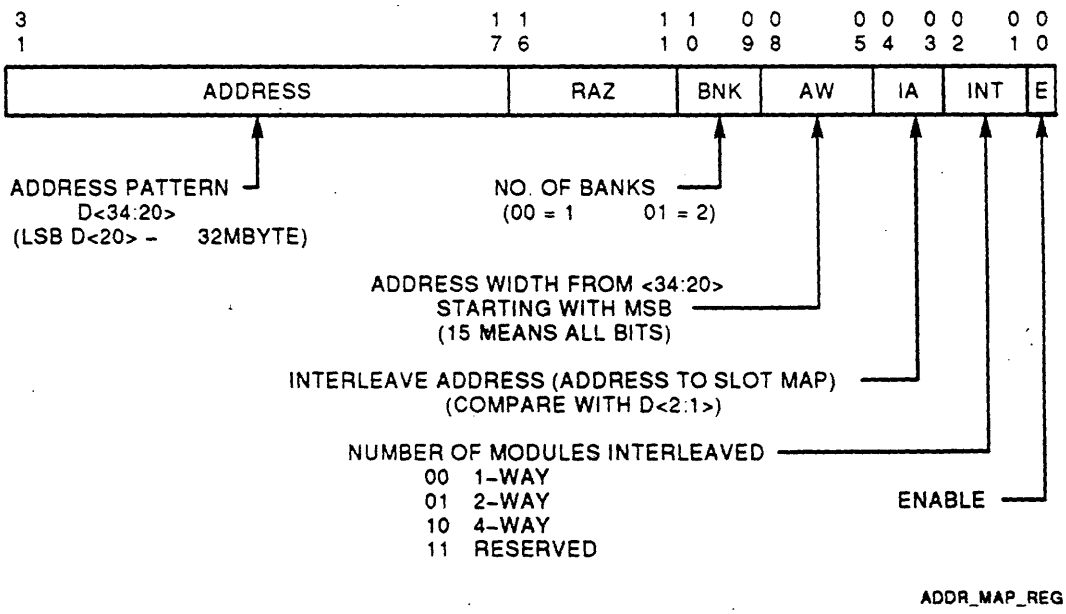


Figure 5–9 Address Mapping Register (AMR)



Memory Self-Test Register 0 and 1 (MSTR0, MSTR1)

The two Memory Self-test Registers are used by the module self-test to report its test results for 64 areas of memory. A bit is set in each register for each area where an uncorrectable error was found. Modules using 4 Mb and 16 Mb DRAMs differ in the size of the 64 areas. Samples of the area address ranges are shown in Table 5–4. .

Table 5–4 Address Ranges for Each Bit in MSTR0/1

DRAM	Failing Bit	MSTR0	MSTR1
4 Mb	0	0000 0000 - 007F FFFF	1000 0000 - 107F FFFF
4 Mb	1	0080 0000 - 00FF FFFF	1080 0000 - 10FF FFFF
.....	.....	.....	.....
.....	.....	.....	.....
4 Mb	30	0F00 0000 - 0F7F FFFF	1F00 0000 - 1F7F FFFF
4 Mb	31	0F80 0000 - 0FFF FFFF	1F80 0000 - 1FFF FFFF
16 Mb	0	0000 0000 - 01FF FFFF	4000 0000 - 41FF FFFF
16 Mb	1	0200 0000 - 03FF FFFF	4200 0000 - 43FF FFFF
.....	.....	.....	.....
.....	.....	.....	.....
16 Mb	30	3C00 0000 - 3DFF FFFF	7C00 0000 - 7DFF FFFF
16 Mb	31	3E00 0000 - 3FFF FFFF	7E00 0000 - 7FFF FFFF

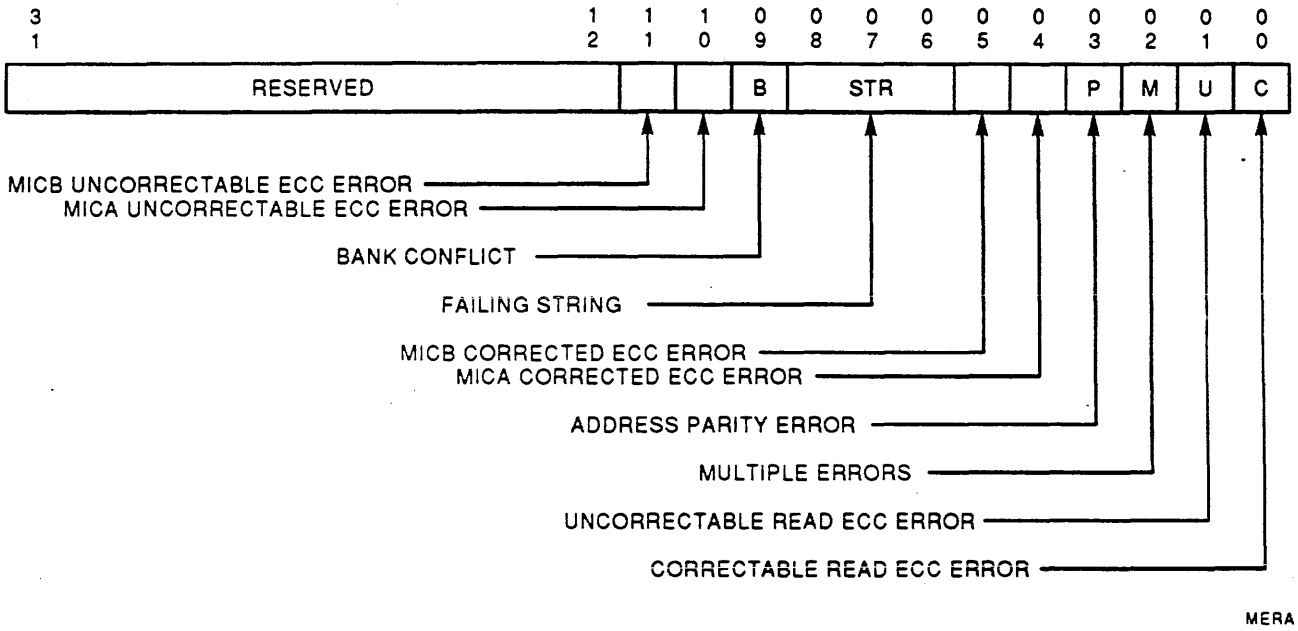
Failing Address Register (FADR)

The Failing Address Register captures the C/A address bits 31:00 from LSB signal lines D<31:00> whenever the LMEM detects a correctable or uncorrectable DRAM ECC error on an LSB read. LSB address bits 34:32 are not captured. If this register is locked with the address of a correctable error when an uncorrectable error occurs, the FADR is updated with the new address.

Memory Error Register A (MERA)

MERA contains MICA error bits and two MICB error bits (<5,11>).

Figure 5–10 Memory Error Register A (MERA)



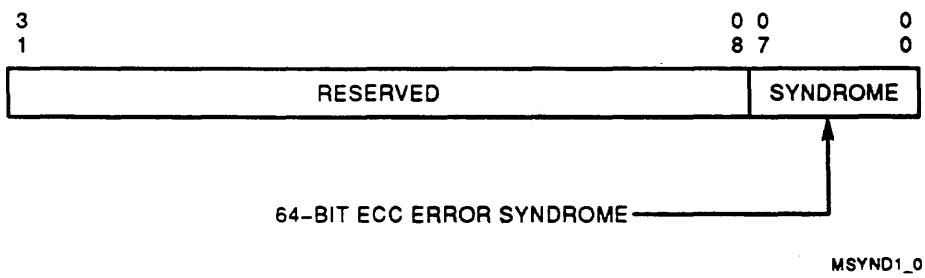
LBER_NSES

MERA bits <11,10,9,5,4> cause the LBER NSES bit to be set.

Memory Error Syndrome Register A (MSYNDA)

MSYNDA contains the syndrome of an ECC error for 64 bits in MICA data path D<63:00>.

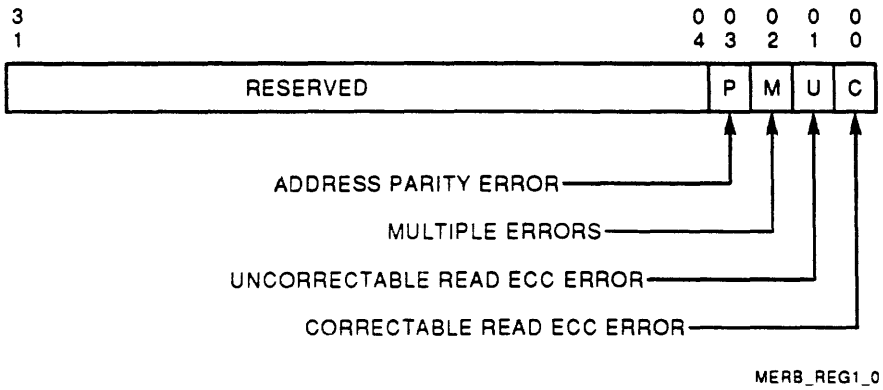
Figure 5–11 Memory Error Syndrome Register A (MSYNDA)



Memory Error Register B

MERB contains MICB error bits.

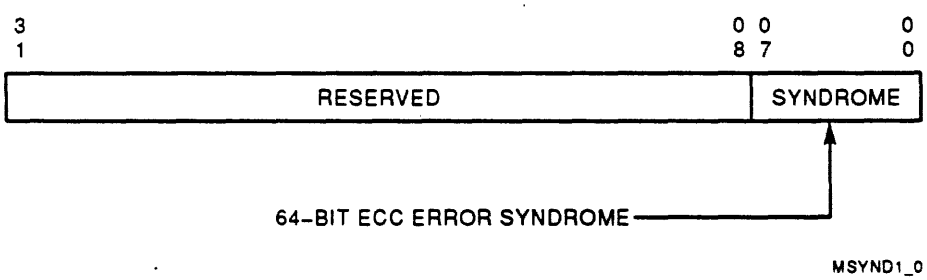
Figure 5–12 Memory Error Register B (MERB)



Memory Error Syndrome Register B (MSYNDB)

Contains the syndrome of an ECC error for 64 bits in MICB data path D<127:64>.

Figure 5–13 Memory Error Syndrome Register B (MSYNDB)



LSB Memory Exercise

1. Which of the following statements is true?

   a. The LMEM module supports a byte-dependent ECC scheme.

   b. The LMEM module supports only one ECC scheme: longword.

   c. The LMEM module supports only one ECC scheme: hexword.

   d. The LMEM module supports two ECC schemes: longword for the LSB and quadword for the DRAMs.

   e. None of the above.

2. The information from MERA, MSYNDA, MERB, and MSYNDB can be used to diagnose correctable errors. Using this information, which of the following is true?

   a. The error can be diagnosed to the DRAM bus bit level.

   b. The error can only be diagnosed to a particular byte.

   c. The error can be diagnosed to the DRAM chip level.

   d. The error can only be diagnosed to the longword level.

   e. None of the above.

3. Which of the following operations is performed by the MDC chips?

   a. The MDC chips multiplex data from one 512-bit data path to 128 bits on two 64-bit data paths, one path MICA and one path MICB.

   b. The MDC chips check 64 bits of ECC code from the DRAM bus and translate it into four 28-bit ECC codes for the LSB.

   c. The MDC chips can be used to perform wrapped read and write operations.

   d. The MDC chips are controlled by MICA.

   e. All of the above.

4. The LMEM module on the LSB supports interlock operations by maintaining a lock bit for each naturally aligned 64-byte block.

   a. True

   b. False

5. Which of the following tasks does the LMEM module perform to support conditional write back cache protocol?

   a. Provides data whenever the SHARED line is asserted during an LSB transaction.

   b. Provides data whenever the DIRTY line is asserted during an LSB transaction.

   c. Aborts an LSB read transaction to its DRAMs if the DIRTY line is asserted during the transaction. The data is supplied by another node.

   d. Aborts an LSB read transaction to its DRAMs if the SHARED line is asserted during the transaction. The data is supplied by another node.

   e. None of the above.

6. The LMEM module receives data with a single-bit error (SBE) from the LSB. The LMEM corrects the SBE and writes the corrected data into its DRAMs.

   a. True

   b. False

7. The LMEM module receives data with a double-bit error (DBE) from the LSB. The LMEM is unable to correct the DBE but writes the bad data into its DRAMs.

   a. True

   b. False

VAX 7000-600 and VAX 10000-600 Processor Operation and Error Registers

Introduction

The VAX 7000/10000-600 processor module (KA7AA-AA, E2045, LNP) contains an NVAX+ microprocessor, backup cache, LSB interface (LEVI), and some auxiliary circuits.

Objectives

A Digital Services Engineer who maintains a VAX 7000/10000-600 system should:

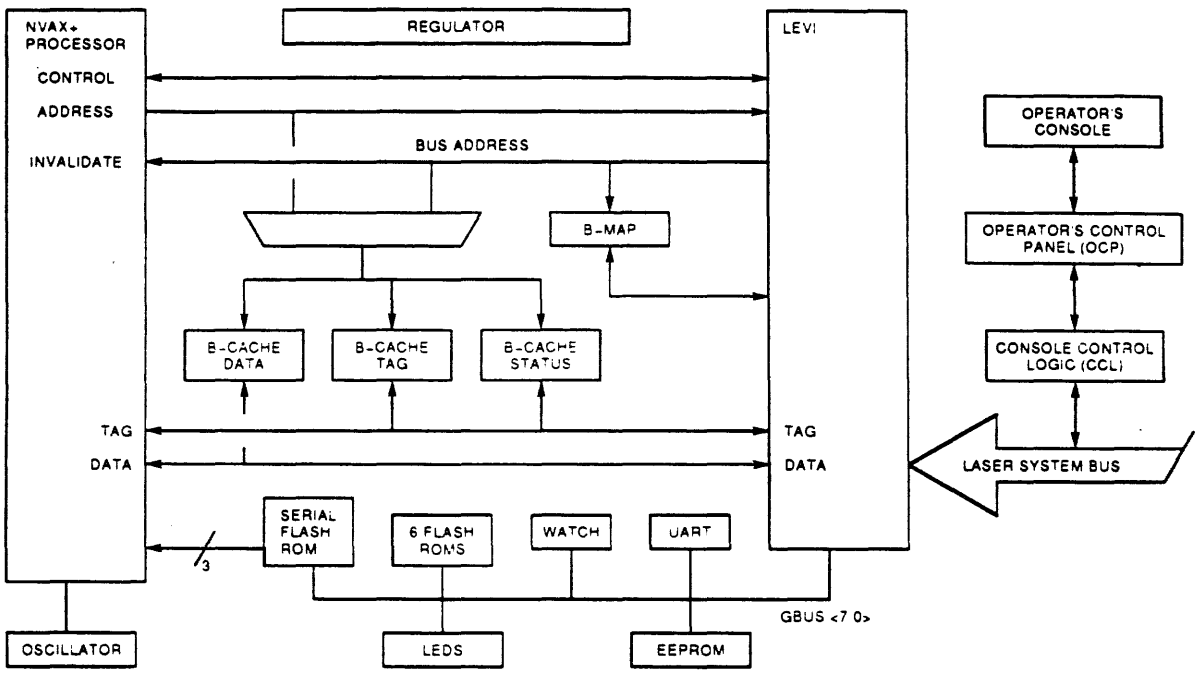
- Be familar, at block level, with the tasks performed by units on the LNP module.
- Recognize the sequence in which the LNP units perform their tasks.
- Use the contents of LNP module registers as an aid while diagnosing a system failure.
- Be familar with conditional update writeback cache protocol.

Resources

VAX 7000/10000 KA7AA CPU Technical Manual	EK-KA7AA-TM
DEC 7000 AXP System and VAX 7000 Console Reference Manual	EK-70C0B-TM

VAX 7000-600 CPU Module (LNP, KA7AA, E2045) Overview

Figure 6–1 LNP Module—E2045 (Repeated)



NVAX_PLUS_70

- NVAX+ is a CMOS-4 macropipelined chip
  - — Similar to NVAX chip but has a different CBox
  - — Implements the basic 242-instruction set including floating point
  - — 8 kB primary cache for instruction and data
  - — 2 kB virtual instruction cache (VIC)
  - — 96 entry translation buffer
  - — 11 ns cycle time

- 4 MB direct mapped backup cache
  - — 64 byte entries
  - — VALID, SHARED, DIRTY bits along with tags
- LSB-to-NVAX+ interface (LEVI)
  - — Two LEVI chips
    - * LEVI-A (DC7300A) provides control and data path <39:00>
    - * LEVI-B (DC7301A) provides most of data path <127:40>
  - — Supports conditional update writeback protocol
  - — PMAP which has list of P cache entries—used to support writeback protocol
- Console support hardware
  - — Flash ROMs
  - — WATCH chip—TOY clock with BBU, TODR, interval clock
  - — Register and status LEDs
- DC-to-DC power regulator (48 Vdc to lower dc voltages)
  - — +5 Vdc after +3.3 Vdc is stable
  - — +2 Vdc after +5 Vdc on all LSB modules is stable

VAX 7000-600 Processor Physical Address Space

The VAX 7000-600 processor can support either 30-bit or 32-bit physical addresses. The 30-bit and 32-bit physical address space is shown in Figure 6–2 and Figure 6–3.

Figure 6–2 VAX 7000-600 30-Bit Address Space

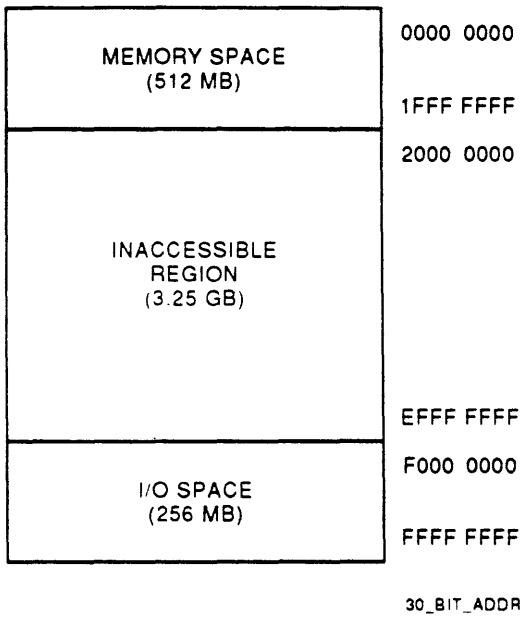
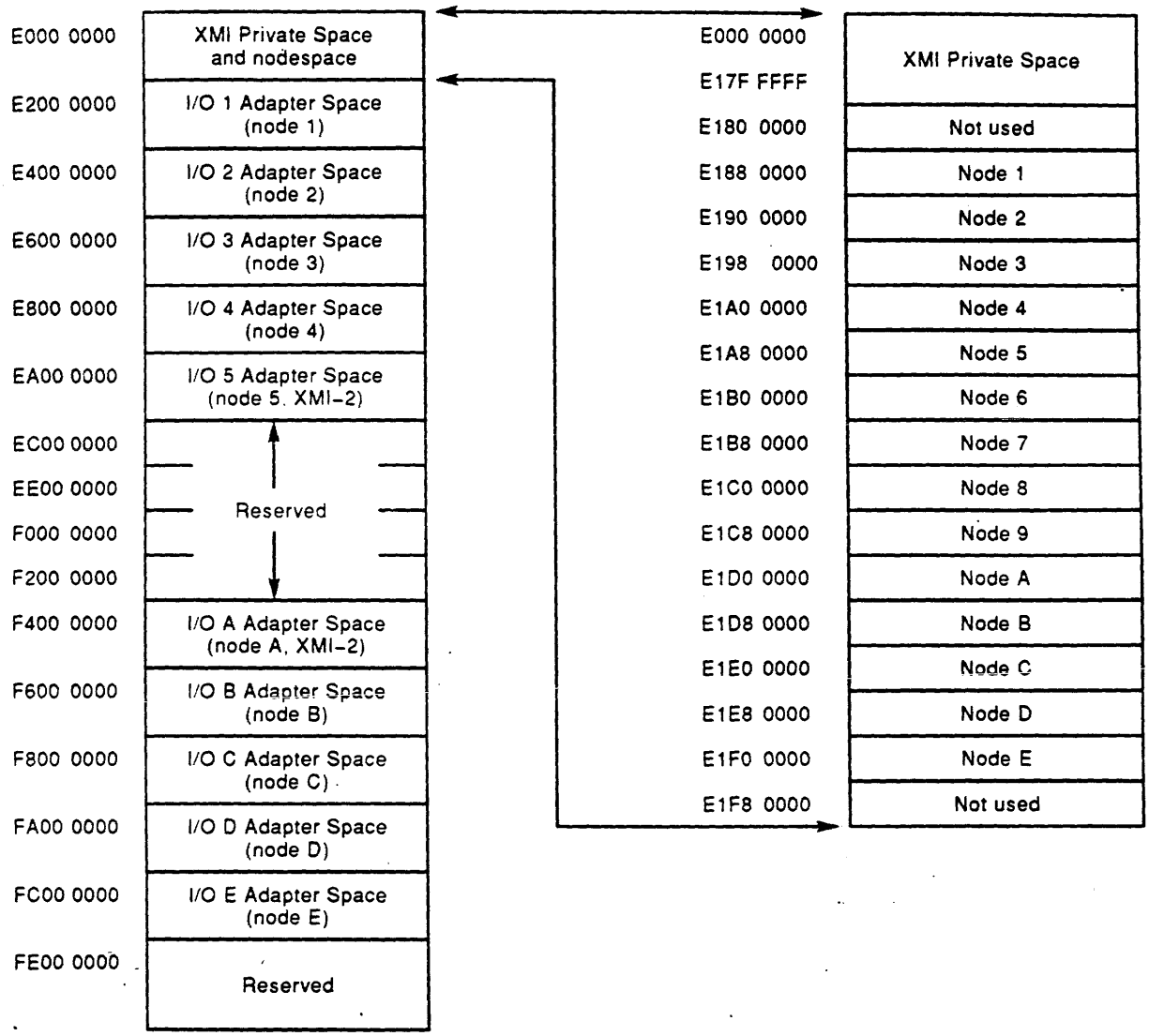
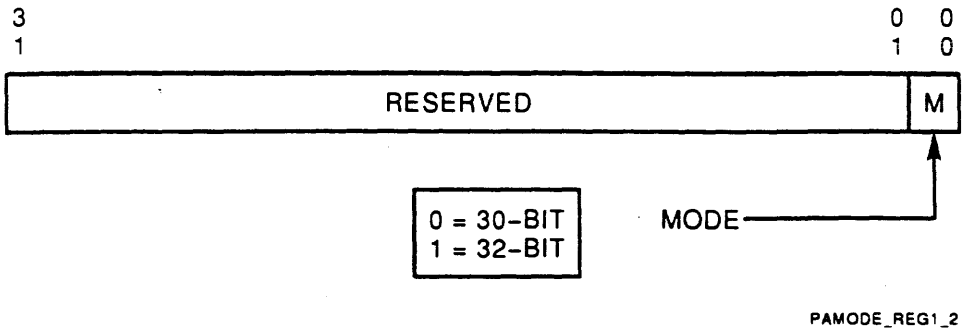


Figure 6–3 VAX 7000-600 32-Bit Address Space



During the power on sequence, console code configures the VAX 7000-600 processor to generate 30-bit physical addresses by writing to the Physical Address Mode (PAMODE, IPR E7) register. The format of the register is shown in Figure 6–4.

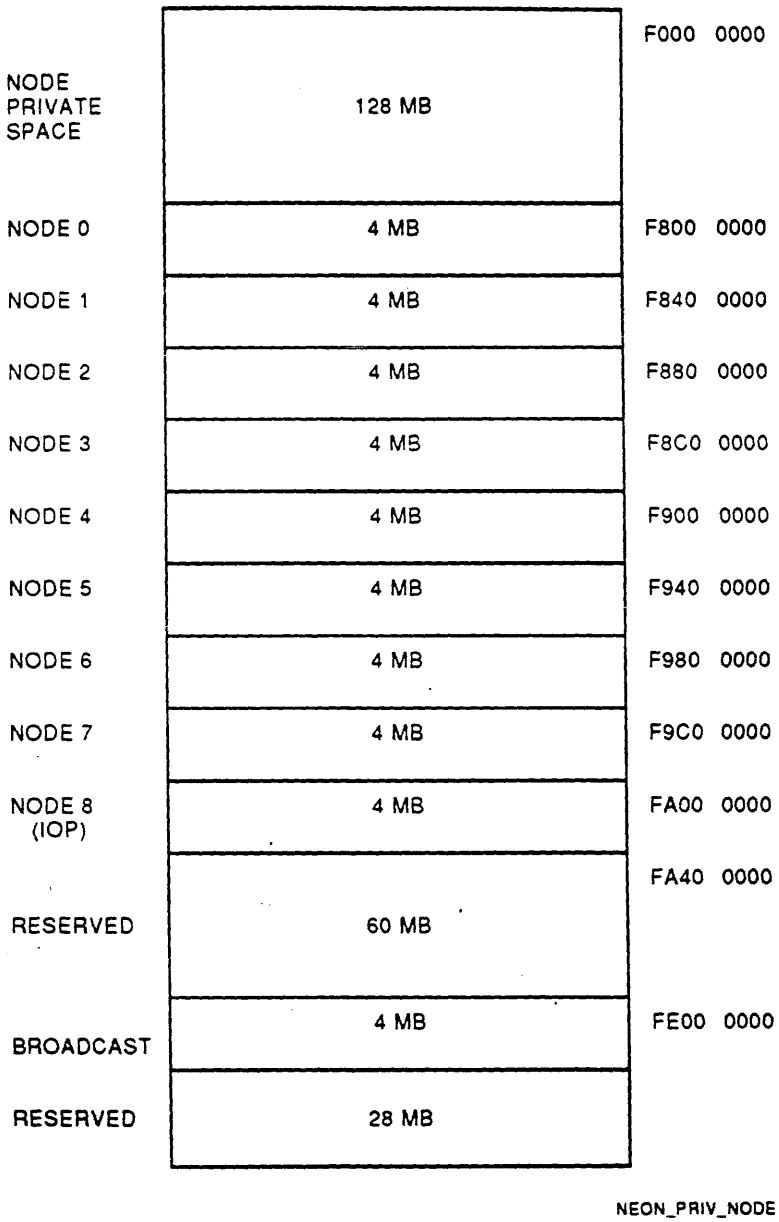
Figure 6–4 PAMODE Register



The address mode affects interpretation of PTEs. In 30-bit mode, PTEs are interpreted in 21-bit PFN format. In 32-bit mode, PTEs are interpreted in 25-bit PFN format although the upper two bits of the PFN field are ignored.

The address ranges allocated for node private space and node LSB CSRs are shown in Figure 6–5.

Figure 6–5 Node Private Space and CSR Space



LNP node private space is shown in Table 6–1.

Table 6–1 LNP Private Space

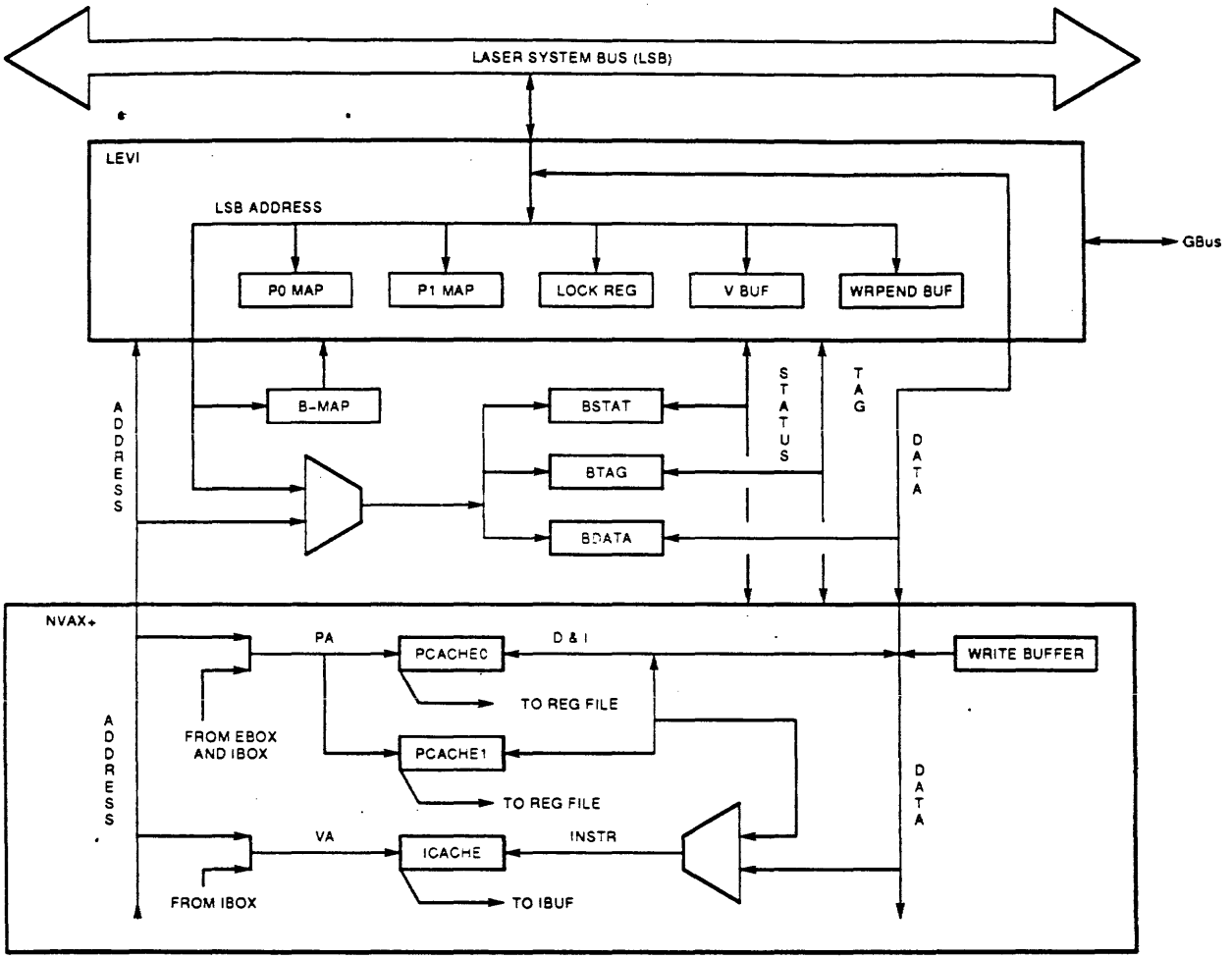
Address Range	Register Name
F000 0000 - F37F FFFF	Seven Console Flash ROMs
F380 0000 - F3FF FFFF	Console EEPROM
F400 0000 - F57F FFFF	UART Registers
F600 0000 - F600 0FC0	WATCH Registers
F700 0000	Gbus$FWHAMI
F700 0040	Gbus$LEDs
F700 0080	Gbus$PMask
F700 00C0	GBus$Intr
F700 0100	Gbus$Halt
F700 0140	Gbus$LSBRST
F700 0180	Gbus$Misc
F780 0000	Gbus$RMode_ENA

Cache Scheme

The LNP uses a three-tiered cache scheme to help keep the NVAX+ processor working at high efficiency.

- There is a very fast virtual instruction cache (VIC) in the NVAX+ IBox. It is used with a 16-byte prefetch queue to help keep IBox instructions available.
  - — First level cache for instructions and a subset of primary cache
  - — Flushed with every REI instruction
- There is a very fast primary cache (P cache) in the NVAX+ EBox with both instructions and data.
  - — Subset of the backup cache.
- There is a fast backup cache (B cache) external to the NVAX+ chip.
  - — Very large (4 MB).
  - — NVAX+ chip and LEVI gate arrays share use and control of the B cache.

Figure 6–6 LNP Cache Arrangement



CACHE_BD_70

Virtual Instruction Cache

The virtual instruction cache (VIC) is a 2 kB, direct-mapped cache with a block and fill size of 32 bytes. It has the following features:

- Subblock size: 8 bytes
- Access bus size: 8 bytes
- One even parity bit per tag
- Four even parity bits per block (1 per subblock)
- Four VALID bits per block (1 per subblock)

Primary Cache

The primary cache (P cache) is an 8 kB, two-way, set associative cache with a block and fill size of 32 bytes. It has the following features:

- Subset of the B cache
- Cache data bus is 16 bytes wide, two cycles to fill a block
- 12 tag bits with one even parity bit
- 12 status bits and 4 even parity bits per block
  (3 status bits with one even parity bit per subblock)
- 32 even parity bits per block (1 per byte)
- Read allocate (no write allocate)
- Write through
- I-stream and D-stream data
- 32 byte block size, 8 byte subblock size
- Access size 8 bytes (access bus is 8 bytes)
- Second level cache for instructions and first level cache for data.

Backup Cache

The backup cache (B cache) is a 4 MB, single-set, direct-mapped cache. The NVAX+ (CBox) and LEVI chips cooperate to access and manage the B cache.

- Block and subblock size of 64 bytes.
- Third level cache for instructions and second level cache for data.
- Cache data bus is 16 bytes wide (4 cycle block fill).
- Each block has a tag (with even parity bit) that is compared to PA <33:22>.
- Each block has three status bits: VALID, SHARED, DIRTY (with even parity bit).
- ECC protection is provided for each data longword (ECC <27:00>.

NOTE

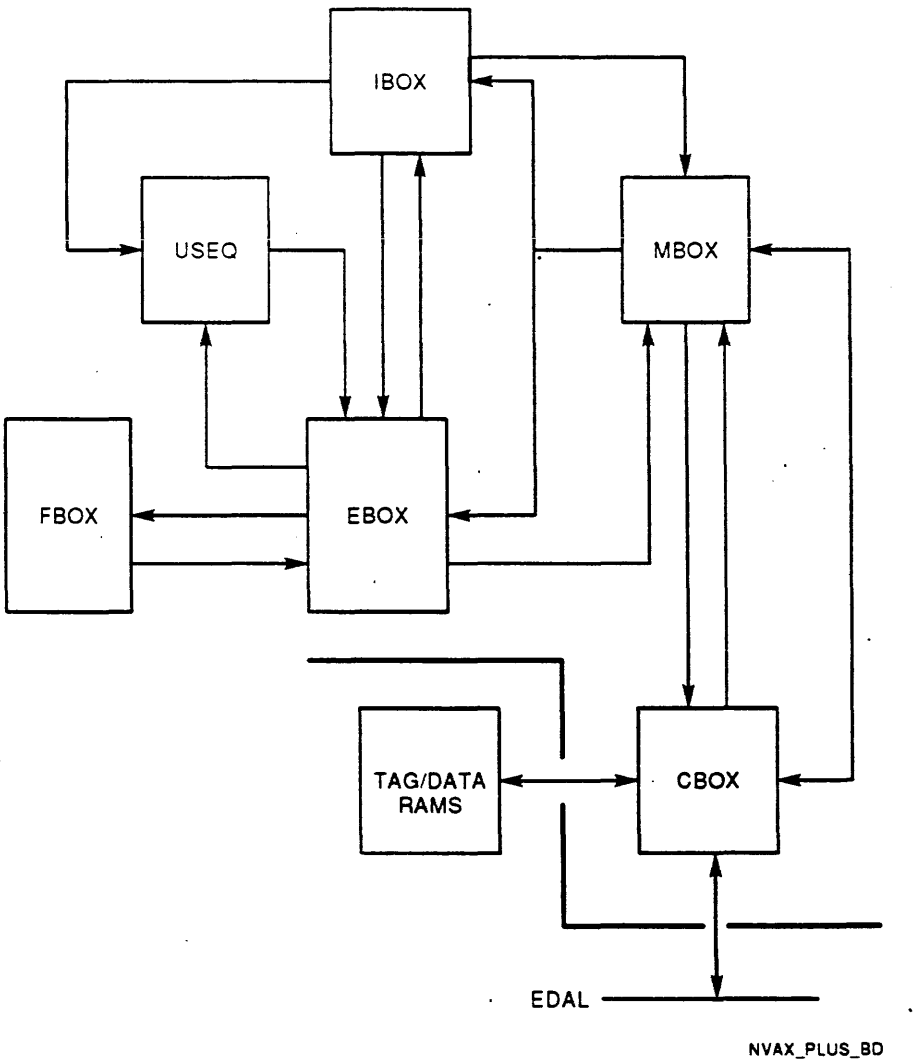
The LNP B cache, unlike previous VAX 6000 system caches, cannot be turned off.

NVAX+ Processor Chip (DC262)

The NVAX+ processor chip (DC262) is similar to the processor chip used on the VAX 6000 Model 600 system. The chip has a few improvements and a new cache box to interface with the EDAL bus. The NVAX+ uses the EDAL as its data bus and uses the LEVI as an interface to the LSB.

- NVAX+ shares B cache with the LEVI chips.
- NVAX+ requests services from LEVI chips.
- NVAX+ accepts interrupts from LEVI chips.

Figure 6–7 NVAX+ Block Diagram



The NVAX+, like the Model 600 processor, has a microcode pipeline and a macrocode pipeline. For example, the IBox is usually one or more macroinstructions ahead of the EBox.

The NVAX+ IBox performs:

- Istream prefetching (16-byte PFQ)
- Istream parsing
- Operand specifier processing
- Branch prediction

The NVAX+ EBox performs:

- Instruction execution
- Instruction coordination between IBox, MBox, FBox
- Trap, fault, and exception handling
- Processor control

The NVAX+ MBox performs:

- VAX memory management
- Memory references from IBox, EBox, CBox
- P cache control

The NVAX+ FBox performs:

- Floating point arithmetic calculations
- Integer arithmetic calculations
- Accepts operands from the EBox
- Returns results to EBox

The NVAX+ CBox performs:

- As an interface between the MBox and the EDAL
- Reads and writes to B cache
- Reads and writes to LEVI
- ECC checks on incoming EDAL data

LEVI Gate Arrays

The LEVI is made up of two gate arrays: LEVI-A (DC7300A) and LEVI-B (DC7301A). LEVI-A contains most of the control and address logic, while LEVI-B provides most of the data path. The LEVI provides the services described here:

- Provides an interface between the EDAL bus and the LSB, translating NVAX+ memory and I/O references to LSB transactions.
- Manages B cache
  - — Controls writebacks to memory when needed
  - — Cache fills from memory when needed
  - — Supports cache invalidates using a B cache map
  - — Removes victim blocks and writes them to memory
  - — Cache status updates
- Supports P cache coherency using a P cache map
- Supports reads and writes to private space (GBus)
- Provides a write pending buffer
- Supports the LOCK mechanism
- Supports interrupts
- Implements all LSB required registers

EDAL Bus

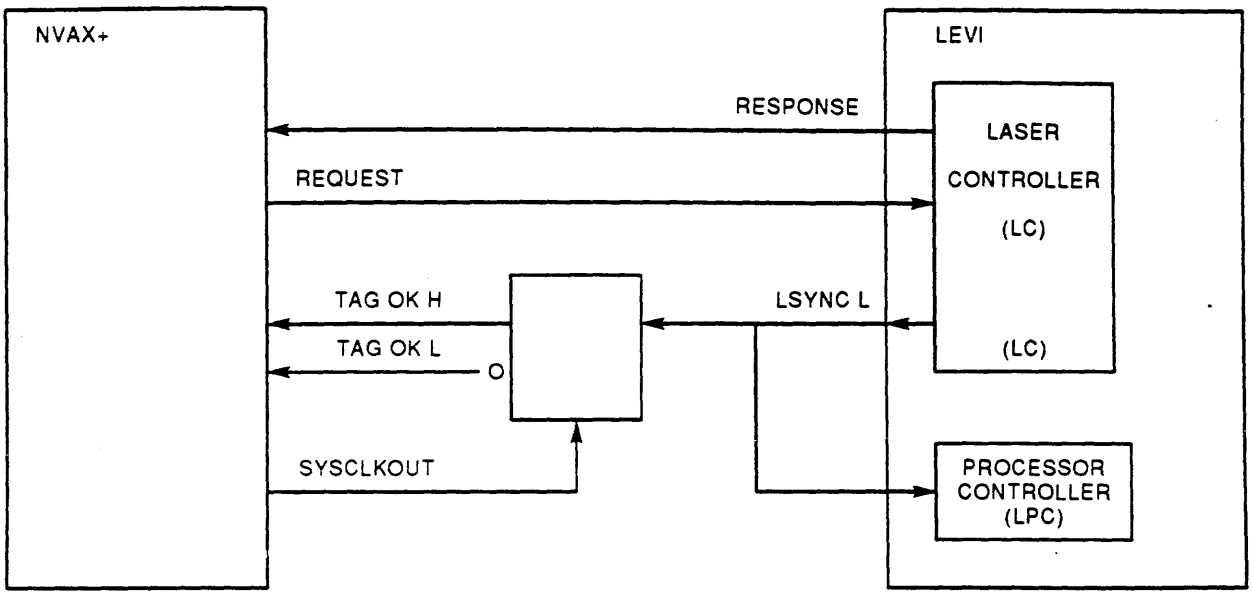
The EDAL bus is for data communication between the NVAX+ and LEVI. The bus interface unit (BIU) in the NVAX+ CBox performs the EDAL tasks for the processor.

Both NVAX+ and LEVI use the EDAL to access B cache.

- EDAL cycle equals two NVAX+ cycles (slower)
  EDAL cycle time is programmable and lasts *n* processor cycles as set in the BIU control register BC_SPD bits <05:04>.
  - — <05:04> = 0 0 for 2 processor cycles
  - — <05:04> = 0 1 for 3 processor cycles
  - — <05:04> = 1 0 for 4 processor cycles
  - — <05:04> = 1 1 for 8 processor cycles
- Has data bits <127:0>, ECC bits <27:00>, and mask bits <7:0>
- NVAX+ and LEVI share dual-ported B cache
- LEVI manages B cache
- NVAX+ requests services of LEVI
  - — Read and write to memory (on B cache miss)
  - — Read and write to CSRs and GBus space

While sharing use of the EDAL, the NVAX+ and LEVI use the signal line LSYNC (TAG OK on NVAX+) shown in Figure 6–8. The signal line indicates who is using or can use the EDAL.

Figure 6–8 LEVI LSYNC—NVAX+ TAG OK



- LEVI LC has priority to assert LSYNC and use the B cache to service LSB transactions.
- LC asserts LSYNC stalling an output handler sequencer in the NVAX+ CBox EDAL bus interface unit (BIU).
- When LSYNC is not asserted, either the NVAX+ or the LPC can immediately access the B cache.

NVAX+ Transactions

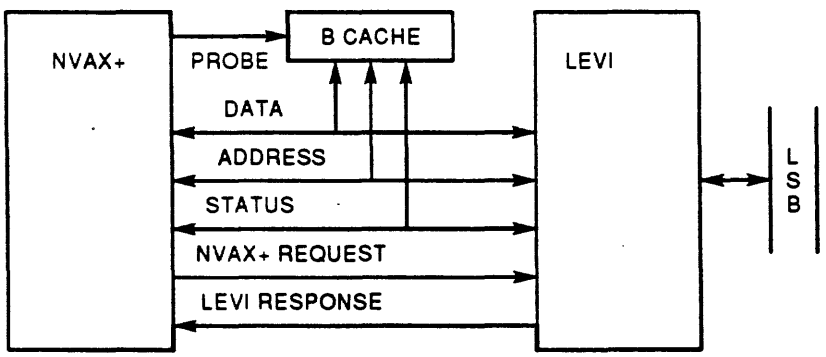
The NVAX+ can only have one transaction request outstanding at a time. The NVAX+ makes a request and then must wait for the LEVI to release it. The NVAX+ requests on cReq_h<2:0> and LEVI responds on cAck_h<2:0>.

- 001 Barrier (NVAX+ will never issue)
- 010,011 Fetch, FetchM (NVAX+ never prefetches)
- 100 READ_BLOCK (B cache miss)
- 101 WRITE_BLOCK (B cache miss)
- 110 LDxL (LOAD LOCK, start LOCK)
- 111 STxC (STORE CONDITION, clear LOCK)

The NVAX+ makes a request, then it is stalled waiting for the LEVI to release it. The LEVI releases it with the following responses on cAck_h<2:0>:

- 001 Hard error
- 010 Soft error
- 011 Store conditional failure
- 100 OK

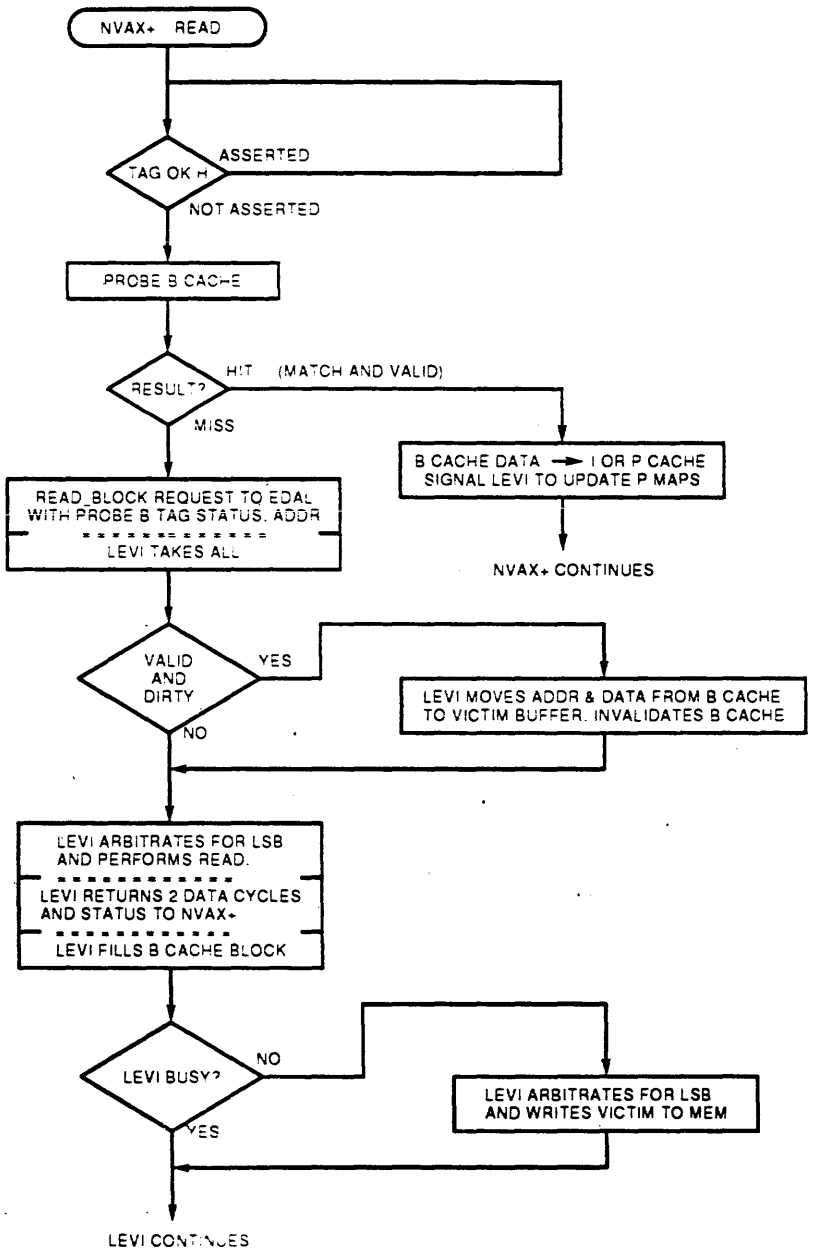
Figure 6–9 NVAX+ and LEVI Cache Arrangement



NVAX+ READ_BLOCK

The NVAX+ and LEVI performs B cache READ_BLOCK operations in the sequence shown in Figure 6–10.

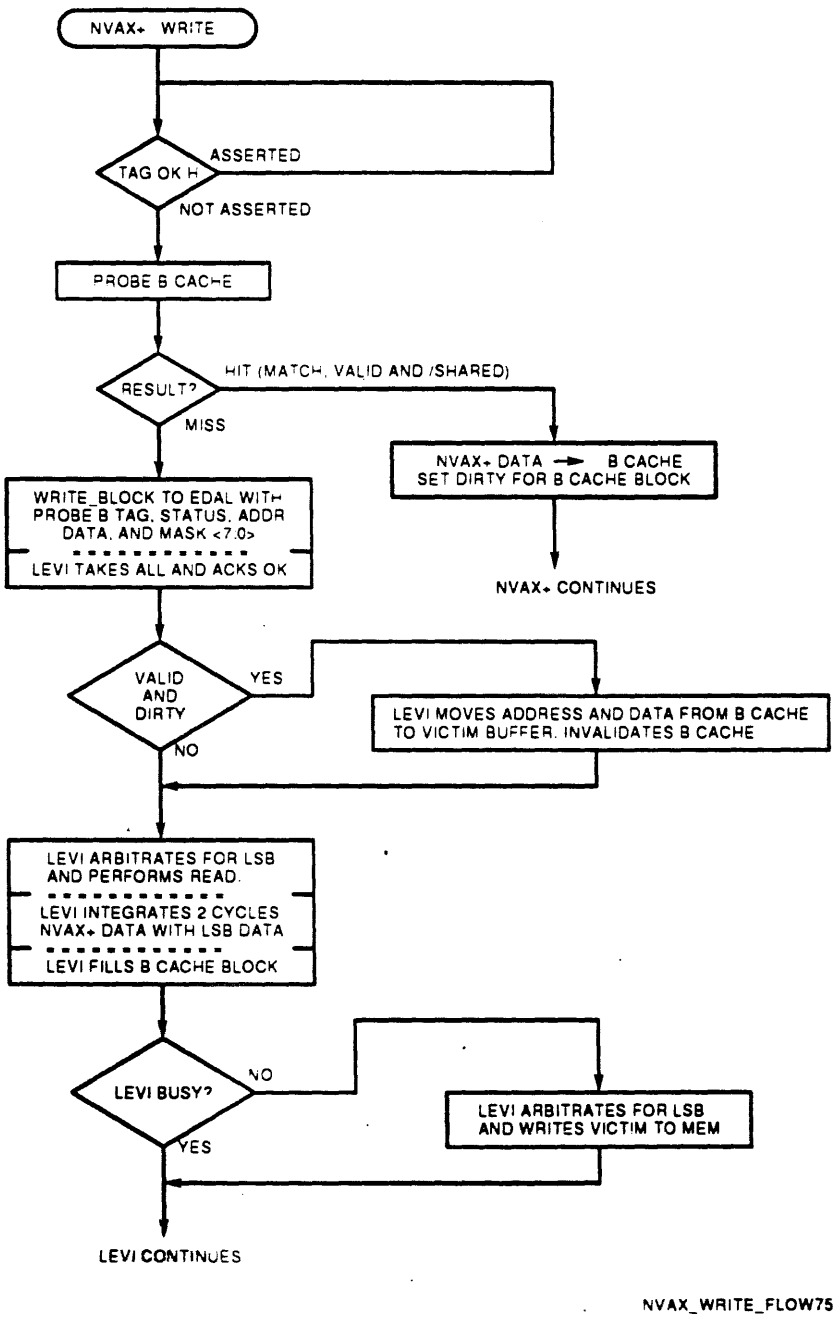
Figure 6–10 NVAX+ READ



NVAX+ WRITE_BLOCK

The NVAX+ performs B cache WRITE_BLOCK operations in the sequence shown in Figure 6–11.

Figure 6–11 NVAX+ WRITE



NVAX+ Lock Transactions

The LSB protocol does not support any type of interlock command pair such as the INTERLOCK READ and UNLOCK WRITE traditionally found on memory interconnects. The interlock mechanism used on the LSB is implemented by a protocol among commander nodes. The memory node does not actively participate in the protocol.

The LEVI maintains a LOCK address register and a LOCK bit to support LNP interlocks. The LNP implements interlocks as shown in Figure 6–12 and Figure 6–13.

Figure 6–12 NVAX+ LDXL (Load Lock)

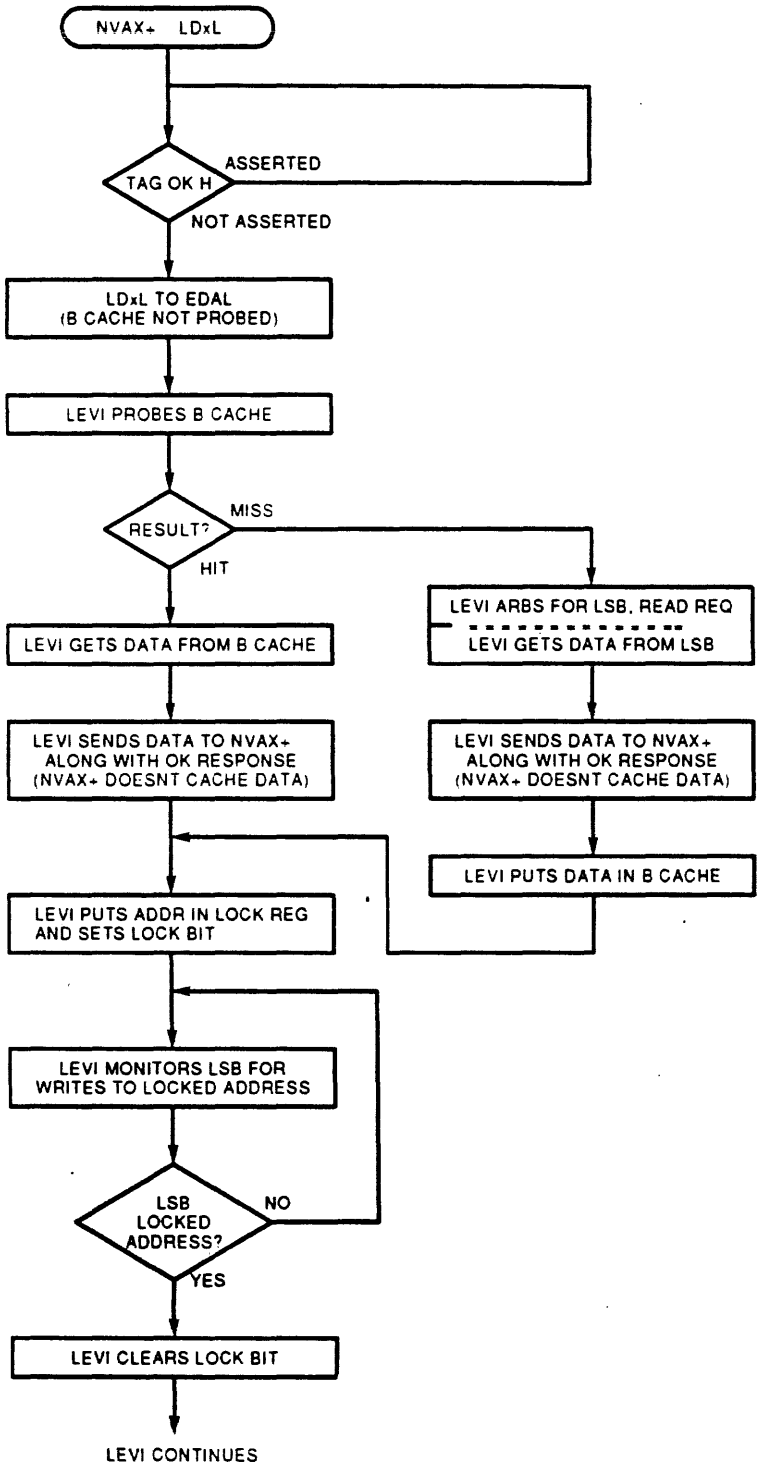
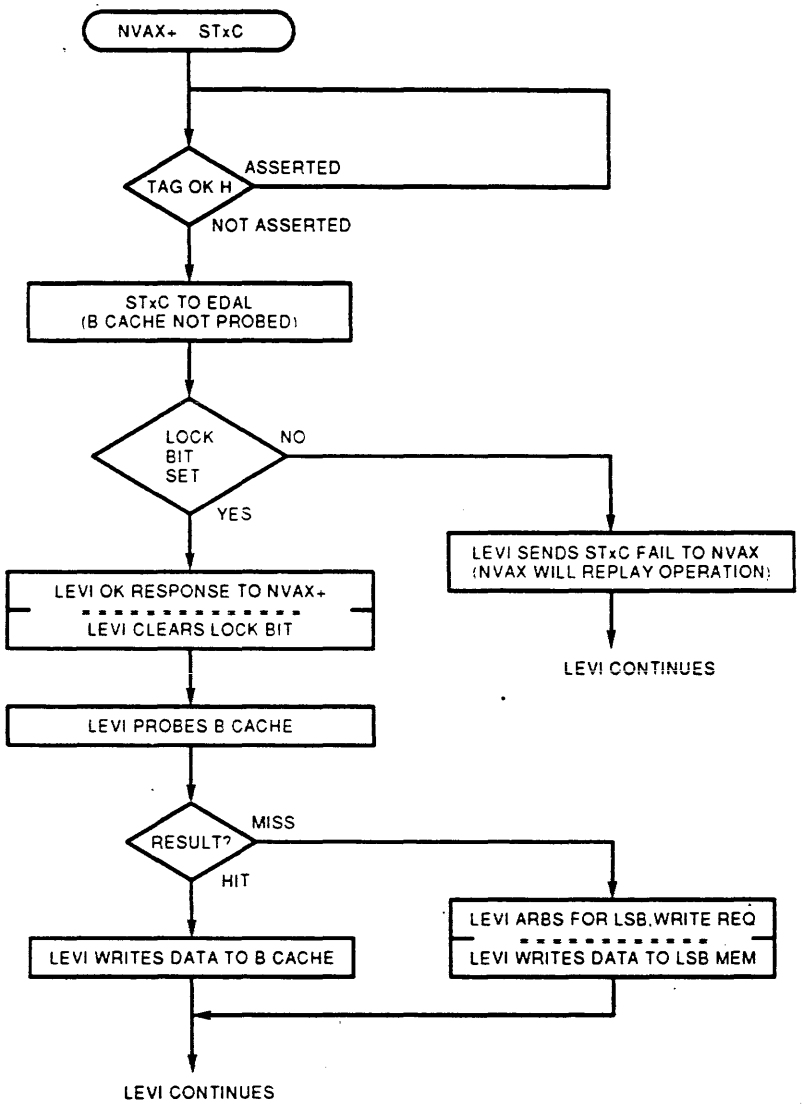


Figure 6–13 NVAX+ STXC (Store Conditional)



Conditional Update Writeback Cache Protocol

The LSB supports a conditional write update cache protocol. This protocol is similar to the write update cache protocol used on the XMI Bus for the VAX 6000 Models 500/600. It differs in that the processor cache controller accepts LSB write updates only if the data is **interesting**.

The LSB implements SHARED and DIRTY signal lines on the bus to support the protocol.

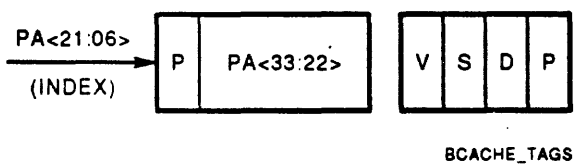
The VAX 7000-600 CPU maintains three status bits: VALID, SHARED, and DIRTY. The significance of these bits is described in Table 6–2.

Table 6–2 Backup Cache Status Bits

VALID	SHARED	DIRTY	State of Cache Entry (with tag match)
0	X	X	Not valid miss.
1	0	0	Valid for read or write.[1] Only cached copy of block and copy in memory is identical.
1	0	1	Valid for read or write.[1] This entry contains the only cached copy of the block. This entry has been modified more recently than the block in memory.
1	1	0	Valid for read or write but writes must be broadcast on the bus. Can be in some other node's cache. The copy in memory is identical.,
1	1	1	Valid for read or write but writes must be broadcast on the bus. Can be in some other node's cache. This entry has been modified more recently than the block in memory.

[1]NVAX+ B cache transactions without LEVI assistance.

Figure 6–14 B Cache Tag Store



NVAX+ Initiated Transactions

Operation to support conditional update writeback cache protocol for NVAX+ initiated transactions is shown in Table 6–3. *NVAX+ write to SHARED B cache line always causes LEVI to write to memory on LSB.*

Table 6–3 Conditional Write Update—NVAX+ Initiated Transactions

NVAX+ Request	B Cache Probe Results	LEVI -> LSB	LSB Status Lines	Next B Cache State	Comment
READ	INVALID	READ	/SHARED	/SHARED, /DIRTY	
READ	INVALID	READ	SHARED	SHARED, /DIRTY	
WRITE	INVALID	READ	/SHARED	/SHARED. DIRTY	
WRITE	INVALID	READ	SHARED	SHARED, /DIRTY	LSB WRITE follows
READ	/MATCH AND /DIRTY	READ	/SHARED	/SHARED, /DIRTY	
READ	/MATCH AND /DIRTY	READ	SHARED	SHARED, /DIRTY	
WRITE	/MATCH AND /DIRTY	READ	/SHARED	/SHARED, DIRTY	
WRITE	/MATCH AND /DIRTY	READ	SHARED	SHARED. /DIRTY	LSB WRITE Follows
READ	/MATCH AND DIRTY	VWRITE. READ	/SHARED	/SHARED, /DIRTY	Victim written back
READ	/MATCH AND DIRTY	VWRITE, READ	SHARED	SHARED, /DIRTY	Victim written back
WRITE	/MATCH AND DIRTY	VWRITE, READ	/SHARED	/SHARED, DIRTY	Victim written back
WRITE	/MATCH AND DIRTY	VWRITE, READ, WRITE	SHARED	SHARED,/DIRTY	Victim written back. LSB WRITE follows.
READ	MATCH	None	None	No Change	
WRITE	MATCH AND /SHARED	None	None	/SHARED, DIRTY	
WRITE	MATCH AND SHARED	WRITE	/SHARED	/SHARED, /DIRTY	
WRITE	MATCH AND SHARED	WRITE	SHARED	SHARED, /DIRTY	

LEVI Responses to LSB Transactions

The LEVI gate arrays support conditional update writeback cache protocol. The actions they perform in response to LSB transactions are shown in Table 6–4.

Table 6–4 Conditional Write Update—LEVI Response to LSB

LSB Operation	B Cache Probe Results	LEVI -> LSB Status	Next Cache State	Comment
READ	/MATCH OR INVALID	/SHARED, /DIRTY	No change	
WRITE	/MATCH OR INVALID	/SHARED, /DIRTY	No change	
READ	MATCH AND /DIRTY	SHARED, /DIRTY	SHARED, /DIRTY	
READ	MATCH AND DIRTY	SHARED ,DIRTY	SHARED ,DIRTY	LEVI supplies data
WRITE	MATCH AND INTERESTING	SHARED, /DIRTY	SHARED, /DIRTY	LEVI supplies data
WRITE	MATCH AND /INTERESTING	/SHARED ,/DIRTY	INVALID	LEVI invalidates B cache

The LEVI monitors LSB read and write transactions (not CSR or VICTIM-WRITE). It checks each transaction address against the B map to see if it is valid in B cache. If it is valid, then the LEVI probes B cache and checks the STATUS bits (V, S, D).

The LEVI performs the tasks listed in LEVI Responses to LSB Transactions as required.

Interesting Data

- If the data is valid in B cache and marked SHARED:
  - — The LEVI checks to see if it is also valid in P cache.
  - — If valid in both B and P caches, then the LEVI assumes it is interesting and updates the B cache and tells the processor to invalidate the location in P cache.

Write Pending Buffer

When the write pending buffer contains an entry, the LEVI checks addresses of LSB reads and writes. If the LEVI gets a "hit" it responds as described in Table 6–5.

Table 6–5 LEVI Action to Support Write Pending Buffer

LSB Operation	Address Register Compare	LEVI Response	LEVI Action
Read	Pending write address match	SHARED, DIRTY	Nothing, continues write to memory
Write	Pending write address match	SHARED, /DIRTY	Accept Update to B Cache

Victim Buffer

When the NVAX+ probes B cache and has a victim miss, it sends the victim to the LEVI and initiates a read to fill the B cache location. The LEVI accepts the victim write and puts it into its single-entry victim buffer. The LEVI starts arbitration for an LSB cycle to write the victim to memory.

When the victim buffer contains an entry, the LEVI checks addresses of LSB reads and writes. If the LEVI gets a victim hit, it responds as described in Table 6–6.

Table 6–6 LEVI Action to Support Victim Buffer

LSB Operation	Address Register Compare	LEVI Response	Action
Read	Victim address match	SHARED, DIRTY	Supply data from victim buffer
Write	Victim address match	/SHARED, /DIRTY	Invalidate victim buffer, remove bus request

Lock Register

When the lock register contains an address, the LEVI checks addresses of LSB reads and writes. If the LEVI gets a hit, it responds as described in Table 6–7.

Table 6–7 LEVI Action to Support Lock Mechanism

LSB Operation	Address Register Compare	LEVI Response	Action
Read	Lock address match	SHARED, DIRTY (from B cache)	Nothing
Write	Lock address match	SHARED, DIRTY (from B cache)	Clear lock bit

NVAX+ GBus References

The NVAX+ uses READ_BLOCK and WRITE_BLOCK requests to reference private space addresses that are in GBus address space. The LEVI recognizes the address range and treatment differs from memory space requests.

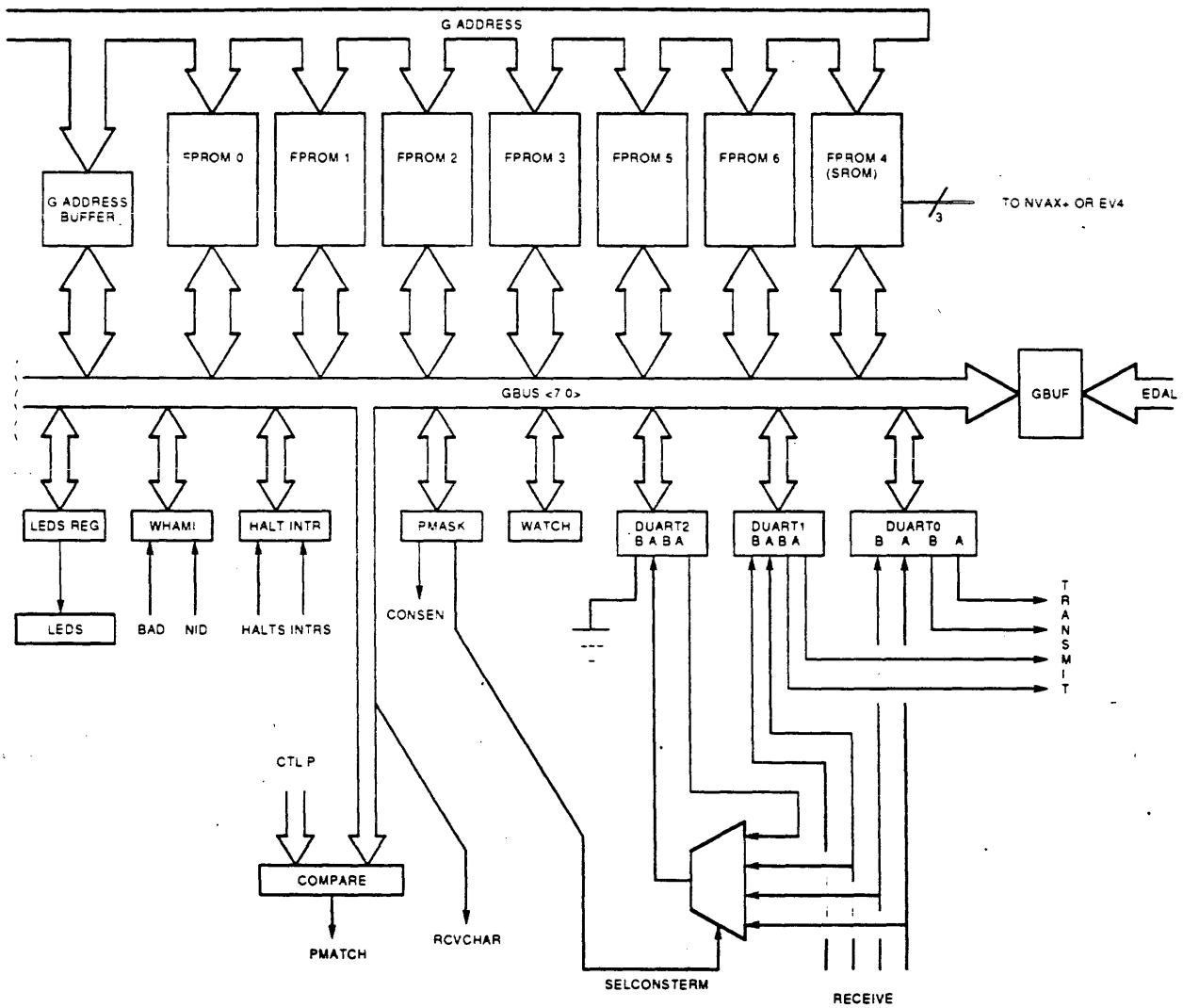
- Read GBus CSR
  - — LEVI waits for LSYNC to deassert, if necessary.
  - — LEVI GBus interface accesses the GBus directly and responds with read data.
  - — LEVI sends data to NVAX+ via EDAL.
- Write GBus CSR
  - — LEVI waits for LSYNC to deassert, if necessary.
  - — LEVI GBus interface accesses the GBus directly and writes the data to the GBus register.

When forced into RUGGED mode by diagnostic routines, LEVI writes to the GBus via the LSB using a private transaction.

GBus Devices

The GBus devices support the LEVI and NVAX+ to provide LNP services that were provided in the VAX 6000 systems by system support chips (SSC). GBus devices are described in this section and shown in Figure 6–15.

Figure 6–15 GBus Devices



GBUS_65

Flash ROMs

There are seven Flash ROMs:

- Six contain the system console and diagnostic programs.
- The fifth (SROM) has startup code, SROM diagnostics, and Flash ROM recovery code (FRRC).

UARTs

There are three dual UARTs (DUARTs) implemented using industry standard 85C30 dual UARTs. They are described in the following list:

- DUART0
  - — UART0A—local console terminal (LOC_RX,LOC_TX)
  - — UART0B—unused
- DUART1
  - — UART1A—unused.
  - — UART1B—power supply status (PS_RX,PS_TX)
- DUART2
  - — UART2A—dedicated to ^P detection. It taps the local console receive line. Transmit line unused.
  - — UART2B—unused.

The default configuration of the serial lines at power on follows:

- Baud rate: 9600
- No parity
- One stop bit
- 8-bit characters
- Local console terminal UART2A selected for ^P character detection

EEPROM

The 8 kB EEPROM holds console parameters, bootstrap information, and error logging information. The present EEPROM contains the 13 areas shown in Example 6–1. More areas could be added.

Example 6–1 Processor EEPROM Areas (REPEATED)

```
>>>show eeprom areas

EEPROM Address: 001c1fe0
Header - Checksum: 28B87EE9  Length: 0054  Version: 1

Area  Offset  Length     Cksum     Key
  0     0064      76  FFD8BFFF    5.0 product
  1     00B0      28  1BFFFFE1    1.0 console parameters
  2     00CC     444  F217FFFF    2.0 boot specs
  3     0288     660  52A83C26    4.0 environment
  4     051C    1036  1D3230E5   18.0 powerup script
  5     0928    1036  7979F031   18.1 test script
  6     0D34      44  B1FFFE1F   32.0 fru descriptor
  7     0D60      24  8CD73546   33.0 system serial number
  8     0D78      76  FFD17FFF   34.0 system configuration
  9     0DC4     296  FFFFFFFF   48.0 diagnostic log
 10     0EEC     656  FFFFFFFF   49.0 symptom log
 11     117C      76  FFFFFFFF   50.0 halt code counters
 12     11C8      76  FFCCBFFF   53.0 service call
 13     1214     856  FFC8CFFF   54.0 module history
>>>
```

Watch Chip

The watch chip provides a time-of-year clock and 50 bytes of battery backed up RAM, both of which have battery backup. The chip contains a built-in crystal oscillator and a 10-year lithium battery. The chip is controlled via a set of registers in the GBus address space.

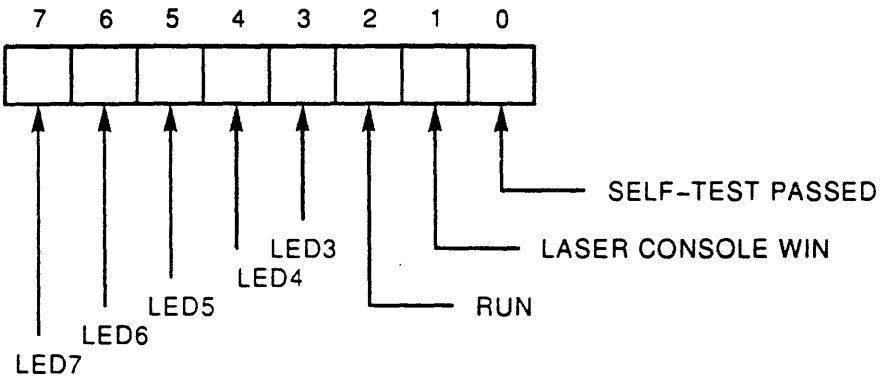
GBus Registers

There are several 8-bit registers located on the GBus. They are addressed on natural 64-byte boundaries.

LED Register

The LED register is used to turn the LNP status LEDs on or off as shown in Figure 6–16.

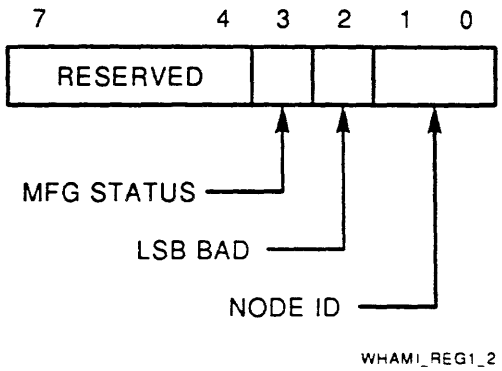
Figure 6–16 LEDs Register



WHAMI Register

The WHAMI register contains the status of the LSB BAD signal line and node ID of the LNP as shown in Figure 6–17.

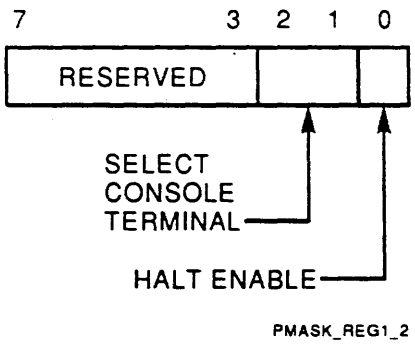
Figure 6–17 GBus WHAMI Register



P Mask Register

The P Mask register is used to select the console terminal and to enable or disable processor halts as shown in Figure 6–18.

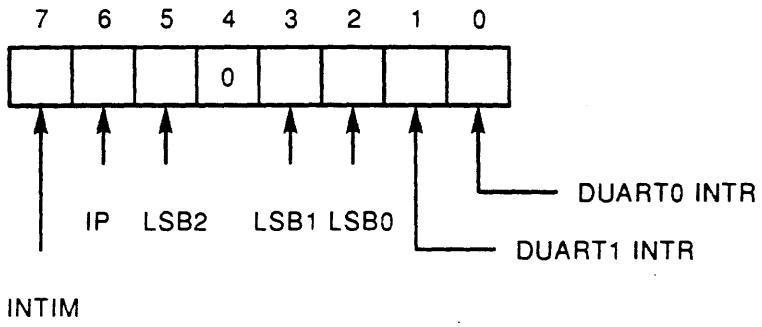
Figure 6–18 GBus P Mask Register



Interrupt Register

The GBus Interrupt register captures interrupts from the DUARTs, the LSB, and the LEVI. These interrupts are later filtered and passed onto the NVAX+.

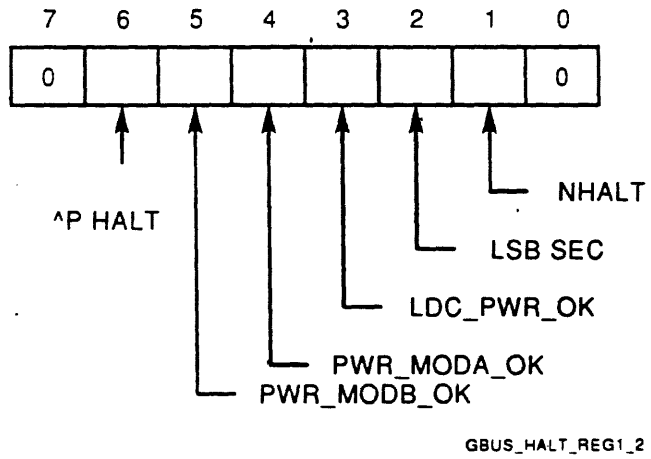
Figure 6–19 GBus Interrupt Register



Halt Register

The GBus Halt register reflects the state of the LEVI halt enable bit and captures ^P status from UART selected in the P Mask register.

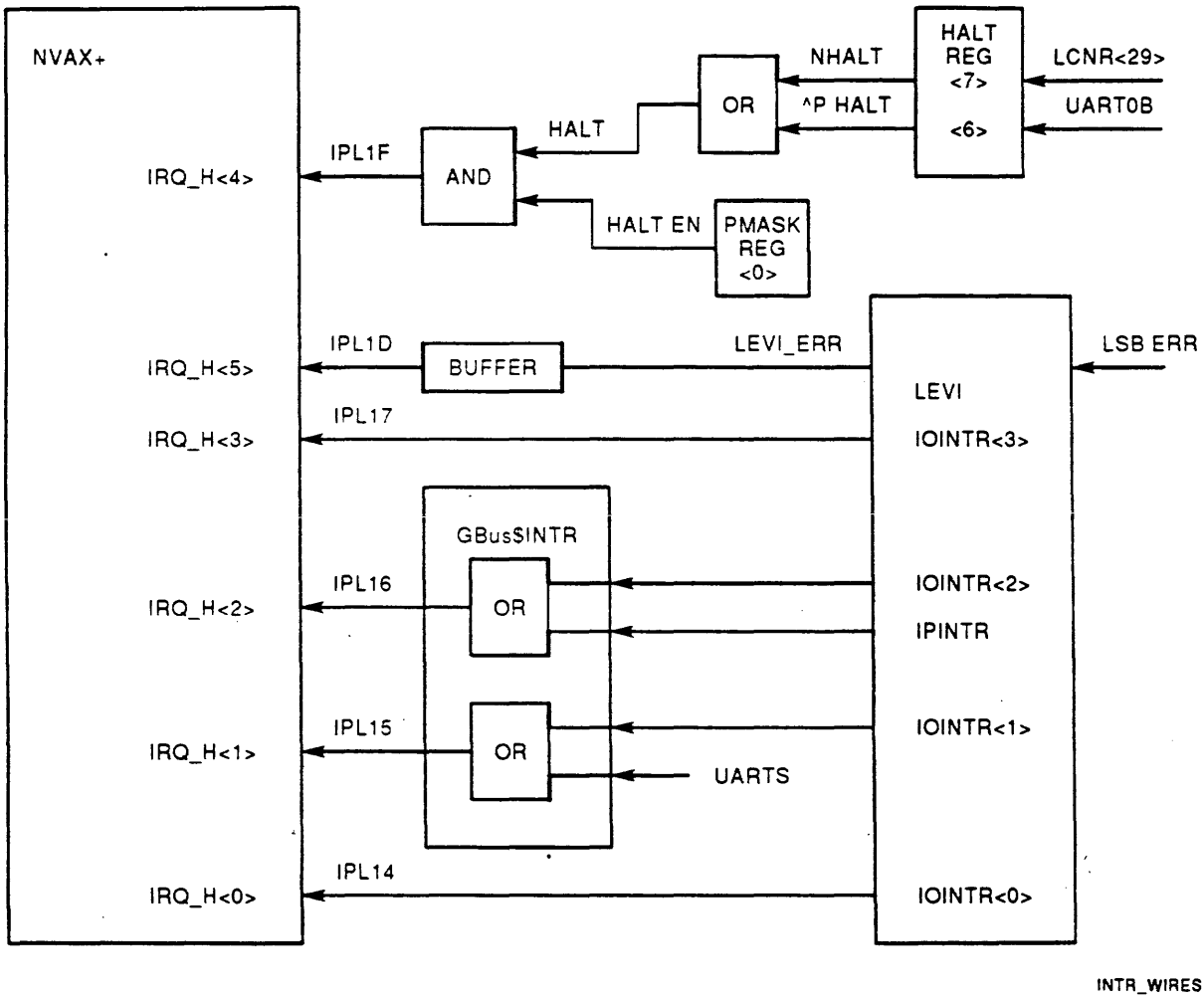
Figure 6–20 GBus Halt Register



Interrupts

The LNP, like all VAX processors, responds to interrupts through the system control block (SCB).

Figure 6–21 NVAX+ Processor Interrupt Connections



NOTE

- **IRQ_H<1>—UARTS have priority over IOINTR <1>.**
- **IRQ_H<2>—IPINTR has priority over IOINTR <2>.**
- **Interval timer interrupt is internal to NVAX+.**

The LNP responds to the interrupts listed in.Table 6–8.

Table 6–8 LNP Interrupts

IPL (Hex)	Label	Devices
1F	HALT_H ^P	Instruction Operator's console
1D[1]	ERR_H	Internal hard error
1A		Internal soft error
17	LSB level 3 interrupt	IOP module
16	LSB level 2 interrupt	IOP module Interval timer Interprocessor interrupt
15	LSB level 1 interrupt	ICP module
14	LSB level 0 interrupt	IOP module
01-0F		Software interrupts

[1]IPL 1D uses the vector 60 as a pointer into the SCB.

NOTE

Notice that the powerfail interrupt, used on previous systems, is not used on the VAX 7000-600 system.

ERR to NVAX+ IRQ_H<5> is asserted by LEVI in response to the assertion of LSB ERR and errors on the LNP module. The assertion of ERR causes the NVAX+ to perform one of the following:

- A machine check through index 04 in the SCB (see machine check parse tree)
- An interrupt through index 60 in the SCB (see interrupt 60 hard error parse tree)

Halts can be caused by a kernel mode halt, a system reset, or when the NVAX+ detects an inconsistency in the internal state. The NVAX+ saves:

- PC goes to SAVPC Register—IPR 42
- PSL goes to SAVPSL Register—IPR 43
- SP goes to IPR0 0, 1, 2, or 3 (dependent on mode)

The NVAX+ then puts contents of Console Base register (CBASE) into the PC and continues.

VAX 7000-600 Mailbox

All processor transaction through the IOP to or from remote buses (such as XMI bus) are accomplished using mailboxes.

A write to LMBPRn may fail if the IOP module has already accepted two writes to the register and has not completed either mailbox operation.

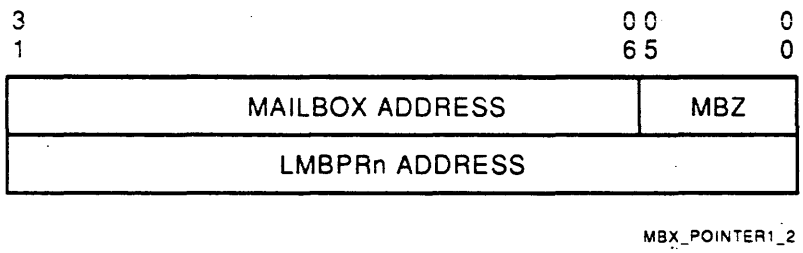
The OpenVMS for VAX operating system cannot directly handle an incomplete write to a physical address, so the write operation is simulated by the LNP module.

The sequence of steps for a VAX 7000-600 mailbox transfer follows:

1. The operating system creates a mailbox structure in memory.
2. The operating system gets the address of the mailbox structure to LMBPRn in the IOP module.

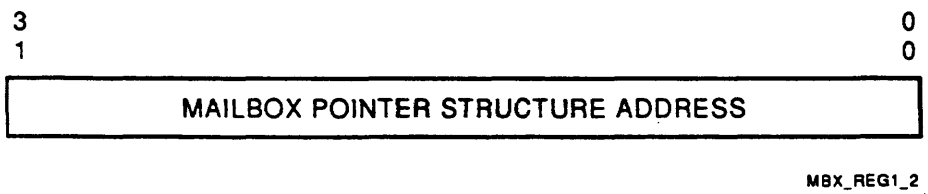
   a. The operating system creates a mailbox pointer structure in memory as shown in Figure 6–22.

Figure 6–22 NVAX+ Mailbox Pointer Structure



   b. The operating system performs a MTPR LMBOX mailbox_pointer_structure_addr. The LMBOX register is IPR 79 (hex) and is shown in Figure 6–23.

Figure 6–23 NVAX+ LMBOX IPR (79 Hex)



c. NVAX+ microcode recognizes the MTPR LMBOX and jumps to a microcode routine to perform a write to LMBPRn.

   1. Normally the write succeeds, and Cack results in the CBox indicating success.

   2. If the selected LMBPRn already holds two mailbox addresses, the IOP does not assert CNF, and the Cack results in the CBox indicating failure.

d. NVAX+ microcode checks the write result code in the CBox and either sets or clears PSL<Z>—sets for passed and clears for failed.

3. If PSL<Z> is clear, the operating system retries the operation until the operation succeeds with PSL<Z> set.

4. The operating system starts polling the DONE bit in the mailbox structure.

5. Meanwhile the IOP services the mailbox request by reading the mailbox and sending a mailbox packet to the LAMB.

6. Later the LAMB returns a mailbox status packet to the IOP. The IOP writes the status packet into the mailbox structure.

7. The operating system (polling) notices the set DONE bit in the mailbox structure and takes action depending on the status part of the mailbox.

LNP Registers and Error Handling

The LNP module has three groups of registers: Internal Processor, LSB required, and module specific.

LNP Internal Processor Registers

The LNP maintains Internal Processor registers (IPR) as required by the VAX System Reference Manual. There are the usual registers such as SCCB and so on and a few registers that are unique to the VAX 7000-600 system. The IPR registers are listed in and described in the *VAX 7000/10000 KA7AA CPU Technical Manual.*

Some of the LNP IPRs (NVAX+ CBox) are listed in Table 6–9. They are often used during troubleshooting as they contain information about EDAL transactions and status.

Table 6–9 NVAX+ CBox Internal Processor Registers

IPR (Hex)	Mnemonic	Register Name
A0	BIU_CTL	BIU Control Register
A1	DIAG_CTL	Diagnostic Control Register
A2	BC_TAG	B Cache Error Tag Register
A4	BIU_STAT	BIU Status Register
A6	BIU_ADDR	BIU Address Register
A8	FILL_SYN	Fill Syndrome Register
AA	FILL_ADDR	Fill Address Register
AC	IPR_STR_COND	STxc Pass/Fail Register

LSB Required Registers

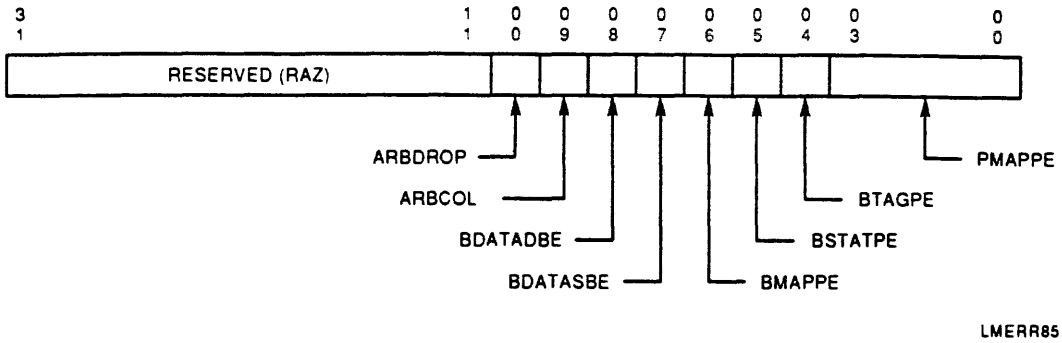
The LNP registers conform to the requirements of the LSB protocol model and maintains all register bits defined in the LSB protocol. The LSB required registers maintained by the LNP module are listed and described in the *VAX 7000/10000 KA7AA CPU Technical Manual.*

LNP Specific Registers

There are several LNP specific registers accessible from the LSB that are used for diagnostic purposes and for capturing error information. They are listed and described in the *VAX 7000/10000 KA7AA CPU Technical Manual.*

The LMERR, shown in Figure 6–24, is used to report B cache and EDAL errors detected by the LEVI.

Figure 6–24 LMERR Register



LNP Error Handling

The LNP error handling methods are described here.

NOTE

The NVAX+ does not implement EDAL timeout errors.

NVAX+ Detection of EDAL Errors

EDAL errors detected or reported to the CBox are reported using the two CBox registers shown in Figure 6–25 and Figure 6–26.

Figure 6–25 BIU Address Register (BIU_ADDR, IPR A6)

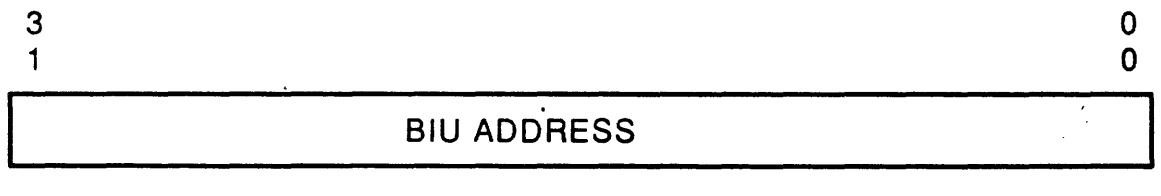
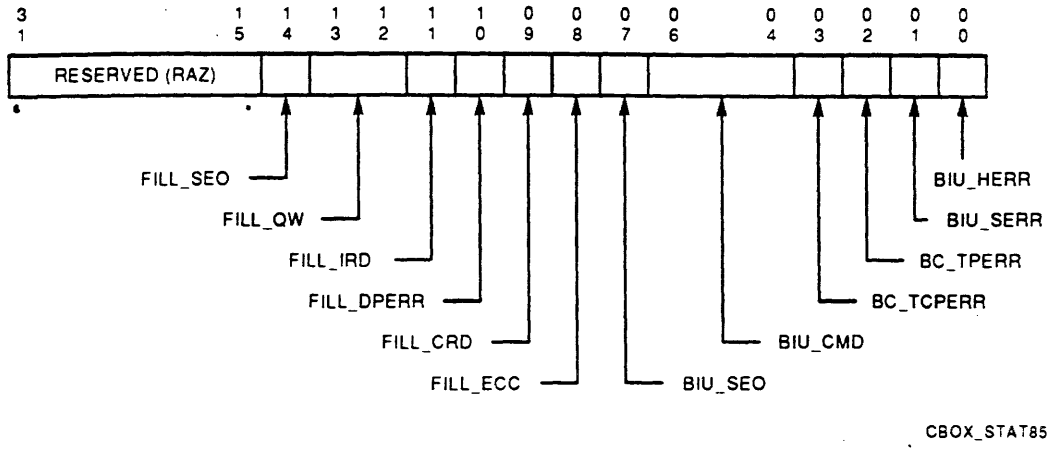


Figure 6–26 NVAX+ BIU Status Register (BIU_STAT, IPR A4)



The bits of particular interest are described here:

- BIU_HERR—EDAL cycle ended with command acknowledge signal pins indicating a hard error (from LEVI).
- BC_TPERR—B cache probe encountered bad parity on tag bits.
- BC_TCPERR—B cache probe encountered bad parity on status bits (VALID, SHARED, DIRTY, even parity).
- BIU_CMD—captures cycle type when any of BIU_STAT <3:0> are set.
- BIU_SEO—second error caused one of BIU_STAT <3:0> to be set when one of BIU_STAT <3:0> was already set.

ECC Corrections on LSB Read Data

The LEVI chips detect ECC errors on read data from the LSB but does not correct the data. LEVI notes the error in error registers to set a level of error isolation.

The NVAX+ performs ECC corrections on SBE read data from the LSB.

LSB CRD Signal Line

The LEVI receives the signal LSB CRD along with LMEM corrected array data. The LEVI passes the status to the NVAX+ chip. The NVAX+ chip ignores status caused by LSB CRD being asserted.

VAX 7000-600 Processor Exercise

1. Which of the following statements is most true?

   a. The LEVI treats the VICTIM buffer as an extension of P cache.

   b. The LEVI treats the VICTIM buffer as an extension of B cache.

   c. When a double-bit error occurs, the VICTIM buffer is used to hold the DBE data.

   d. The VICTIM buffer is used to hold data received from the LSB in a transaction when the DIRTY line was asserted.

   e. None of the above.

2. The VAX 7000/10000-600 systems can support either a 30-bit or a 32-bit address space.

   a. True

   b. False

3. Which one of the following statements is true?

   a. NVAX+ VIC cache holds 8 kB of instructions.

   b. LNP B cache is a subset of NVAX+ P cache.

   c. NVAX+ P cache is a subset of LNP B cache.

   d. When the NVAX+ invalidates an entry in P cache, the LEVI immediately invalidates that entry in B cache.

   e. None of the above.

4. The LNP module B map is probed by the NVAX+ processor to check for valid entries in the B cache.

   a. True

   b. False

5. Which one of the following statements is most true?

   a. An LMEM module always has the best (latest) copy of data.

   b. The LEVI always knows where the best (latest) copy of data is located.

   c. The IOP module asserts the DIRTY line during a transaction if it notices that the transaction address is the same as one of its active mailbox addresses. It then supplies the requested mailbox data.

   d. The NVAX+ always knows where the best (latest) copy of data is located.

   e. None of the above.

6. The LEVI treats its write buffer like an extension of B cache.

   a. True

   b. False

7. The NVAX+ starts counting time when it requests data from the LEVI. It detects timeouts on the EDAL when the LEVI does not respond to the request for more than 128 ms.

   a. True

   b. False

8. The LEVI sees a READ transaction on the LSB to a line that is valid in its B cache and in its P Map. It will accept the update to its B cache and invalidate the NVAX+ P cache entry.

   a. True

   b. False

LSB I/O Operation

Introduction

LSB I/O operations are carried out by the IOP module on the LSB and a remote bus adapter module: LAMB module for the XMI bus. At one interface the IOP module conforms to LSB protocol and at its other interface conforms to hose protocol.

The LAMB module conforms to hose protocol and appears on the XMI bus to be a memory module with special ability to handle interrupts.

Any remote bus adapter module which conforms to hose protocol may interface to the IOP module. This flexible arrangement allows other remote buses to be attached to the 7000/10000 systems in the future.

Objectives

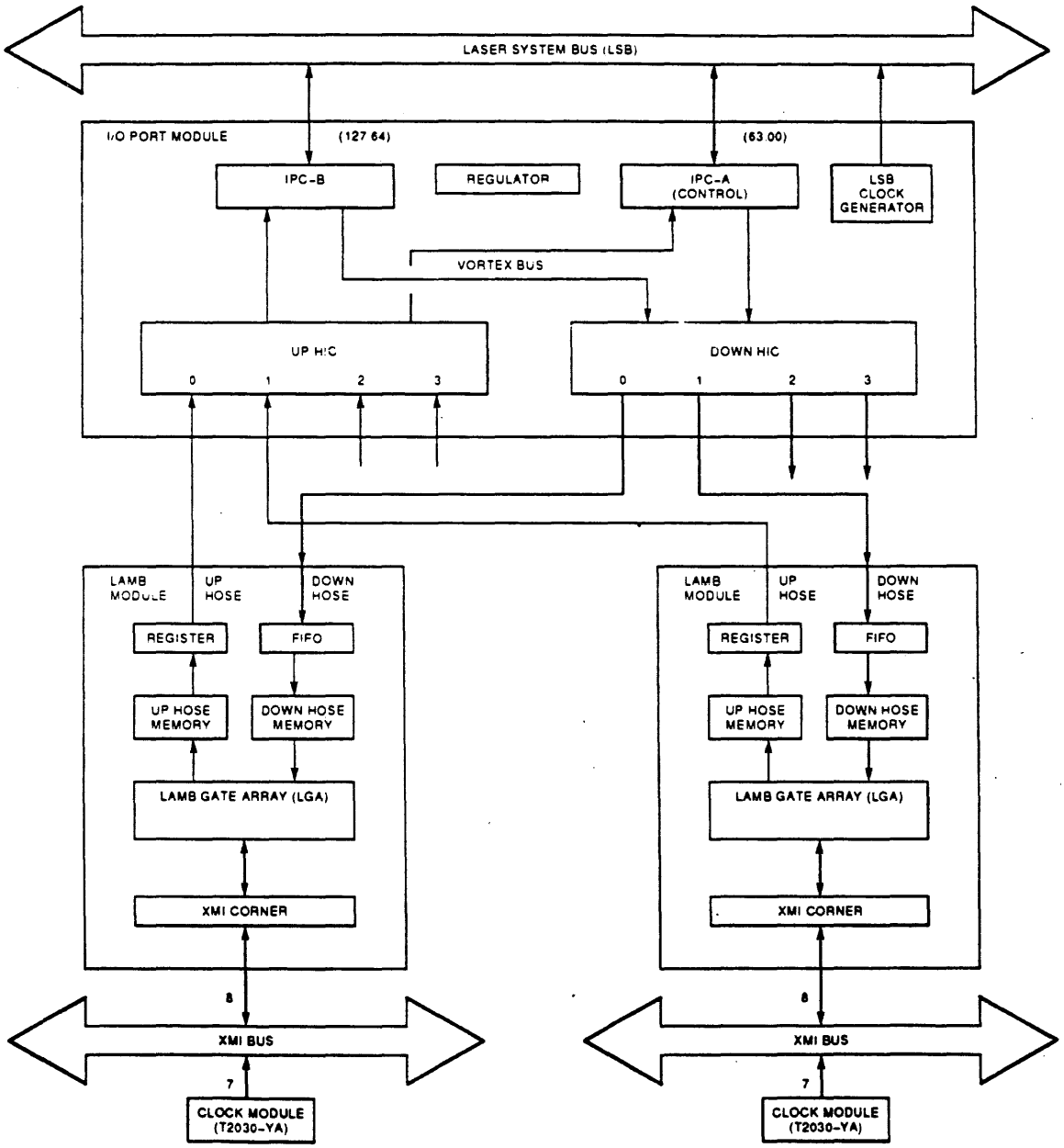
A Digital Services Engineer who maintains a VAX 7000-600 or VAX 10000-600 system should be able to:

- Be familar, at block level, with the tasks performed by the IOP and LAMB modules.
- Recognize the sequence in which the IOP and LAMB modules perform their tasks.
- Use the contents of IOP and LAMB module registers as an aid while diagnosing a system failure.

Overview

A functional block diagram of the 7000/10000 system's I/O subsystem is shown in Figure 7–1.

Figure 7–1 LSB IOP and LAMB Modules (Repeated)



The IOP module on the LSB can be connected via hoses to up to four I/O adapters on remote buses. At present 7000/10000 systems only support connections to the LAMB I/O adapter on the XMI bus. The hose cable assembly may be up to 9 feet long.

NOTE

The XMI bus is contained in a VAX 9000 type backplane and card cage. Processor and memory modules are not allowed on the XMI bus in the 7000/10000 systems.

IOP Module (IOP, E2044)

- IOP module sits in Slot 8 of the LSB centerplane, at the rear of the system.
- Provides two clock signals (PH0, PH90) for each LSB node (including itself).
- LSB reset circuit—any hose can send RESET, and IOP asserts RESET on LSB.
- IOP synchronizes transfers between hoses and the LSB bus.
- Two I/O port chip (IPC, DC7303A) gate arrays between the LSB and the vortex bus.
  - — Buffers to accept up vortex packets
    - * Two DHW data buffers
    - * Two command buffers
  - — Buffers for down vortex packets
    - — Two DHW read data/merge buffers (loaded from LSB)
    - — One interrupt status buffer
    - — One HW mailbox command buffer
- The vortex bus that connects IPCA and IPCB with the UP and DOWN HICs
- Two hose interface chips (HICs, DC7304A) that interface with the hoses.
  - — Each HIC has a 3-hexword buffer for each hose
- Translates between LSB bus protocol and hose interface protocol.
- DC-to-DC power regulators. Convert 48 Vdc to +5 Vdc, +15 Vdc, +2 Vdc.
- No onboard self-test.

Hose Cable Assembly

A single hose (up/down) is used to connect the IOP module to a LAMB module on an XMI bus.

LAMB Module (LAMB, T2028)

- The LAMB module sits in Slot 8 of the XMI bus.
- Conforms to XMI bus protocol.
- Main elements of LAMB are:
  - — One LGA that provides the "brains" for the module.
  - — Up and down hose memories (store commands and data).
- LAMB synchronizes transfers between hoses and XMI bus.
- No onboard self-test.
- No processors or memory on XMI bus (for 7000/10000 systems).

I/O Port Module (IOP, E2044)

The IOP module acts as an interface between the LSB and the four possible hoses.

IOP Transactions

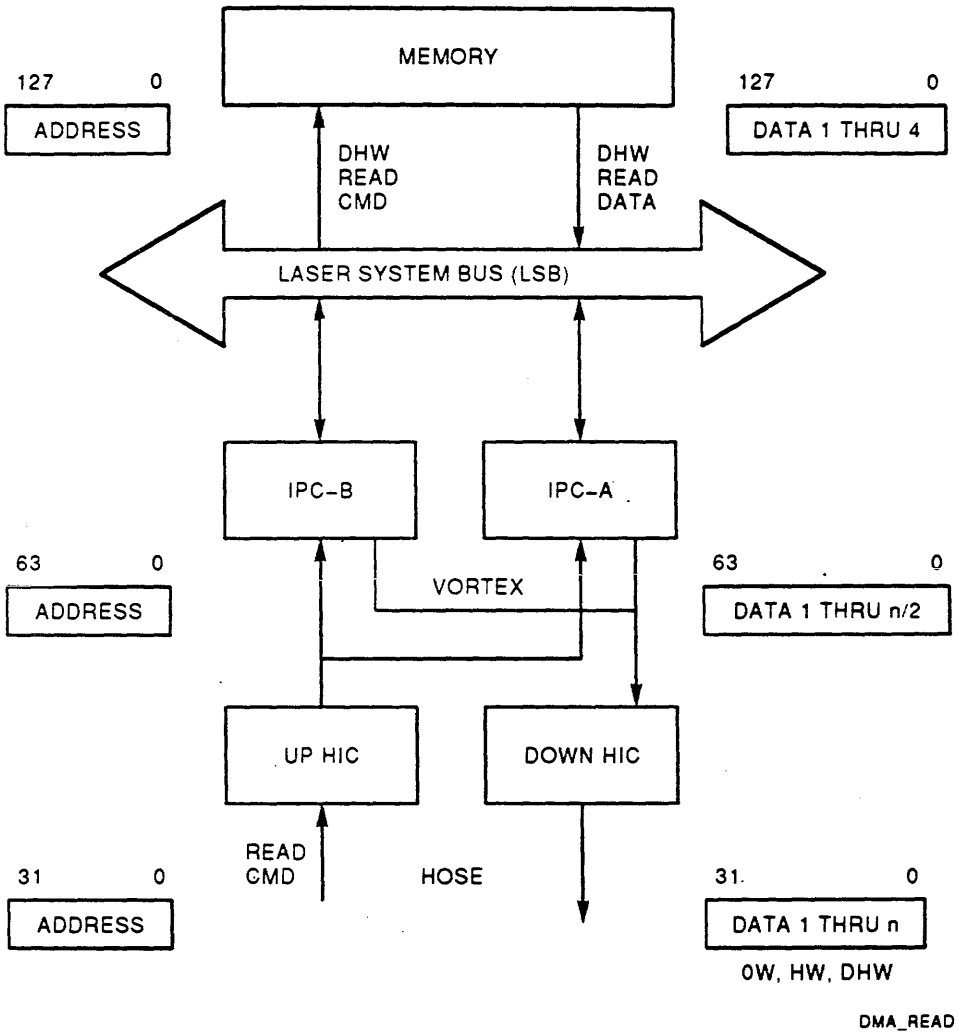
The IOP module can perform the following transactions:

- DMA READ
- DMA WRITE
- DMA READ/MODIFY/WRITE
- INTERLOCK READ (emulated)
- MAILBOX READ or WRITE
- INTERRUPT

DMA Read Transactions

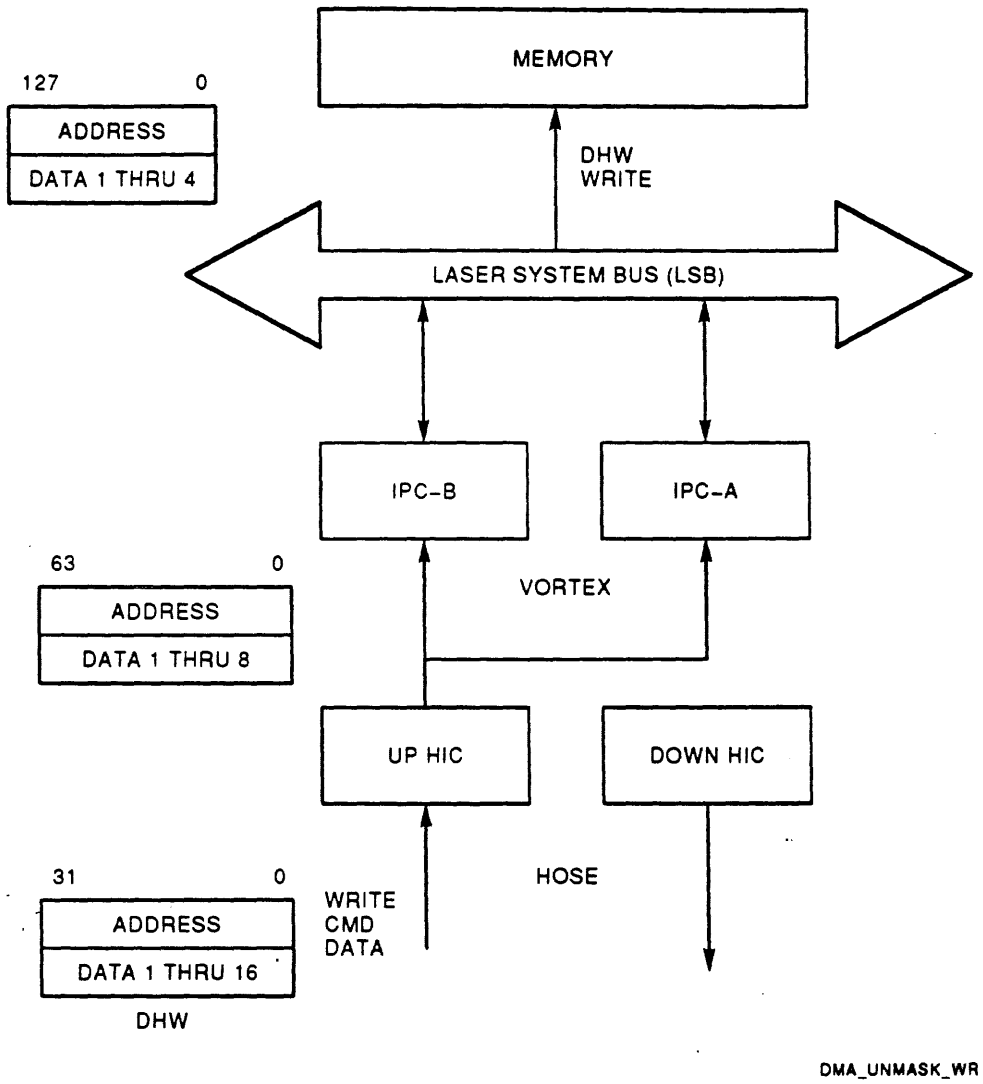
The IOP performs OW, HW, and DHW DMA reads from LMEM. Longword and quadword reads are supported within the OW transaction.

Figure 7–2 DMA READ



DMA Write Transactions (Unmasked)

Figure 7–3 DMA WRITE (Unmasked)



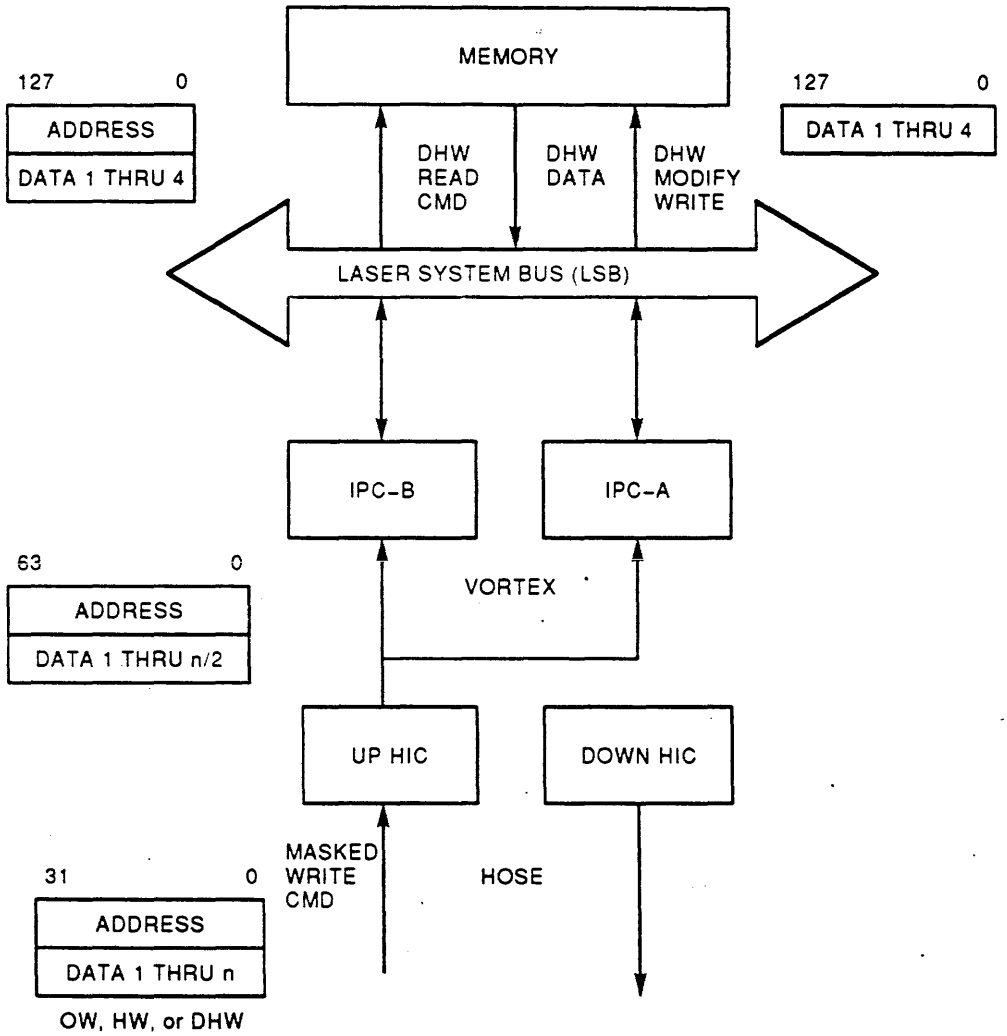
NOTE

Used with LAMB I/O module only if an XMI bus device implements the MORE signal line.

DMA Write Transactions (Masked)

The IOP performs OW, HW, and DHW DMA masked writes to LMEM. Longword and quadword masked writes are supported within the OW transaction.

Figure 7–4 DMA MASKED WRITE



Interlock Transactions

The LSB does not support an INTERLOCK primitive, so the IOP must simulate an interlock to satisfy the CIXCD on the XMI bus.

- IOP reads DHW
- Writes data back setting QW bit <00>
- Another node cannot access data between read and write
  - — Uses PR5 to write back
  - — One bank access per 15 cycles
- Sends OW containing original data down hose

Figure 7–5 IPC Mode Selection Register (IPCMSR)

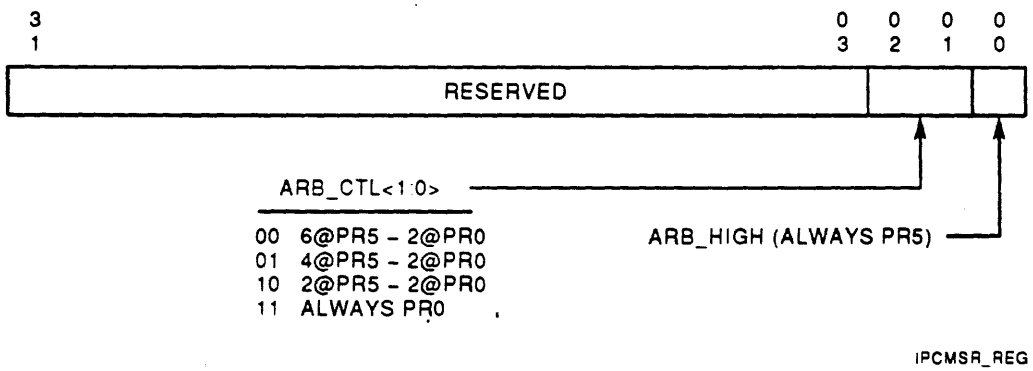
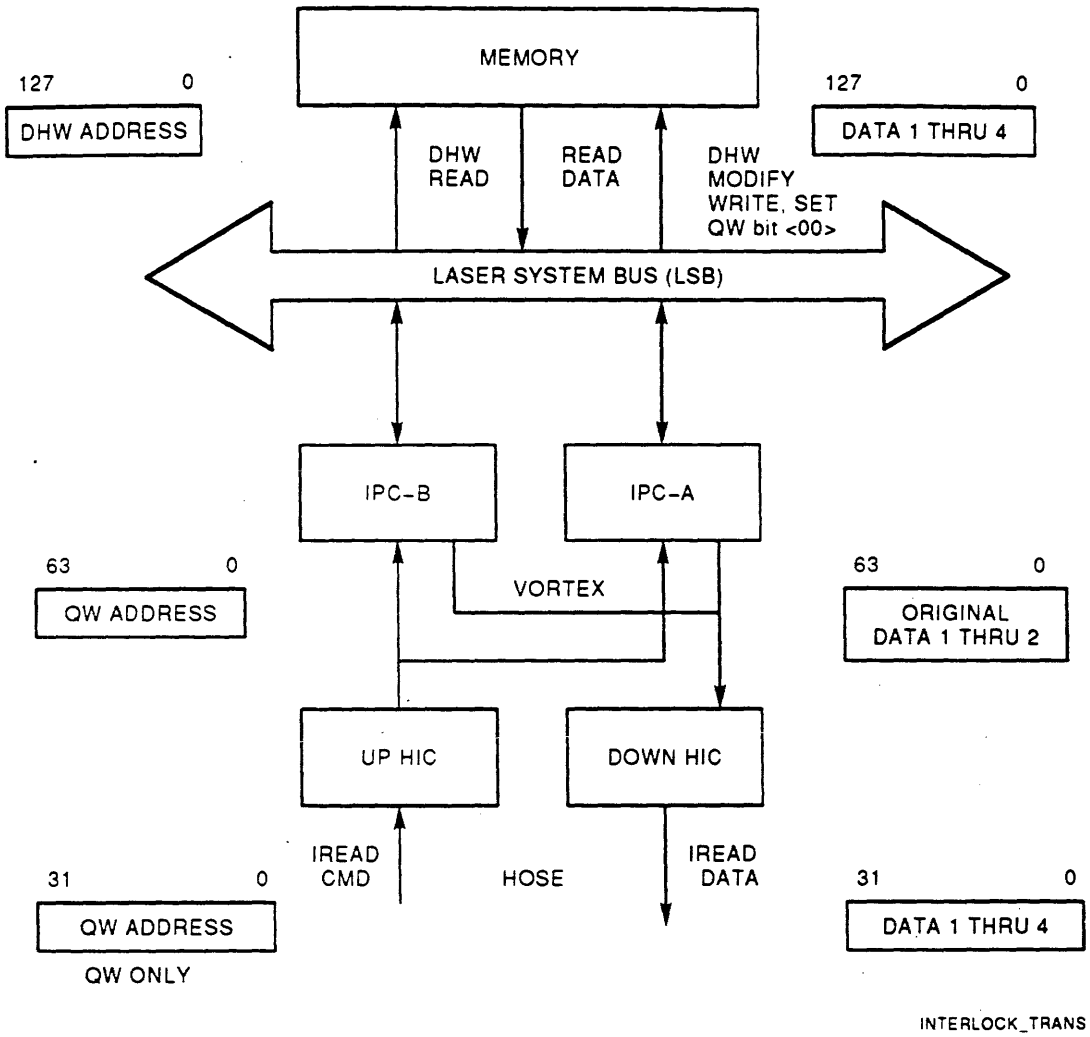


Figure 7–6 INTERLOCK READ (Simulated)

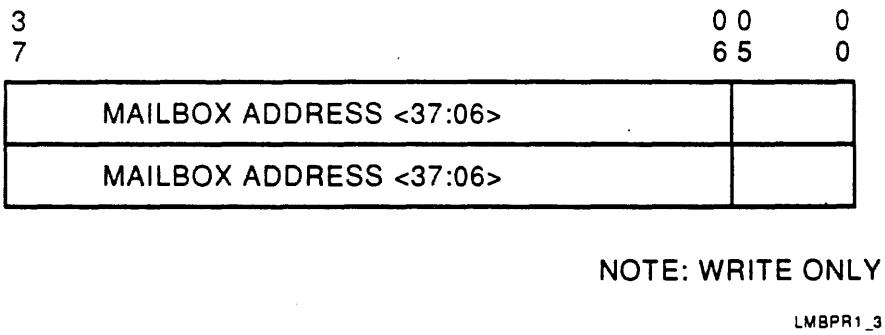


Mailbox Transaction

All transfers through the IOP to or from other buses (such as the XMI bus) are accomplished using mailboxes. Only CPUs at nodes 0-3 handle mailbox transfers.The sequence of steps for a mailbox transfer follows:

1. CPU (0-3) creates a mailbox structure in memory as shown in Figure 7–8 (64 bytes, one transaction).
2. CPU (0-3) writes to the corresponding mailbox pointer register in the IOP (LMBPR0-3, one per CPU) as shown in Figure 7–7.
   - If the selected LMBPRn already contains two entries (hidden queue), the IOP does not assert CNF and the CPU must retry.

Figure 7–7 Mailbox Pointer Register, LMBPRn (Repeated)

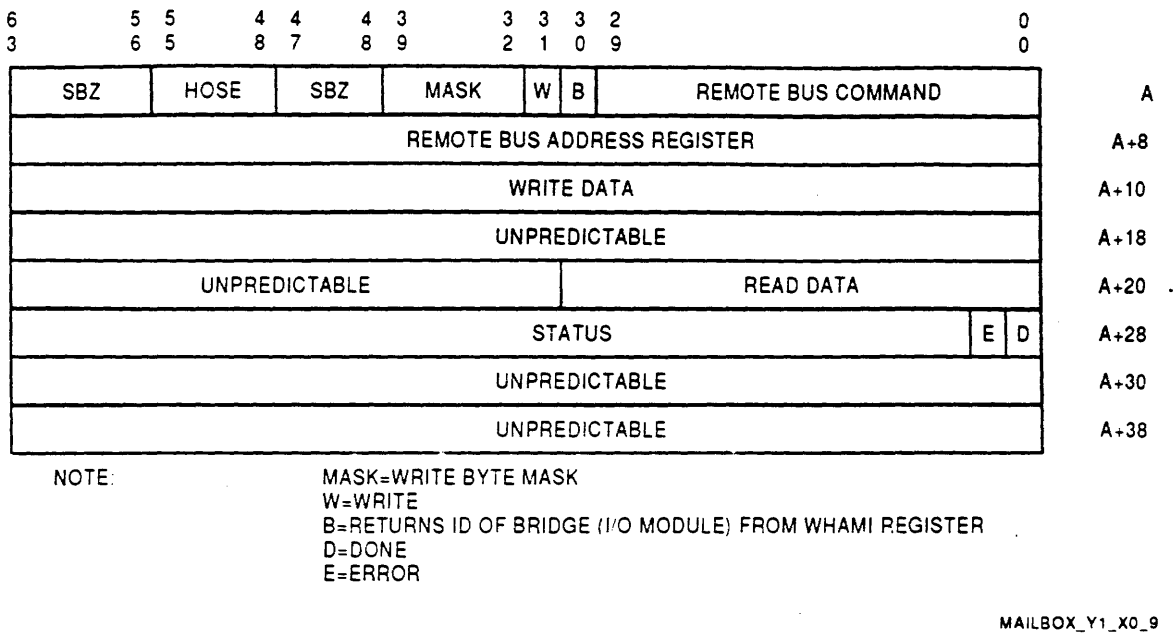


3. CPU starts polling the mailbox looking for the DONE bit set.
4. IOP transmits mailbox to LAMB and waits for mailbox status return packet.
5. IOP writes mailbox status return packet to memory and sets DONE bit.
6. CPU sees DONE bit set and checks the ERROR bit and status longword and takes appropriate action.

NOTE

A mailbox transaction to a nonexistent hose by any CPU causes all mailboxes to hang up (IOP can only have one mailbox packet outstanding at a time).

Figure 7–8 Mailbox Data Structure



The LAMB uses the mailbox fields as shown in Table 7–1.

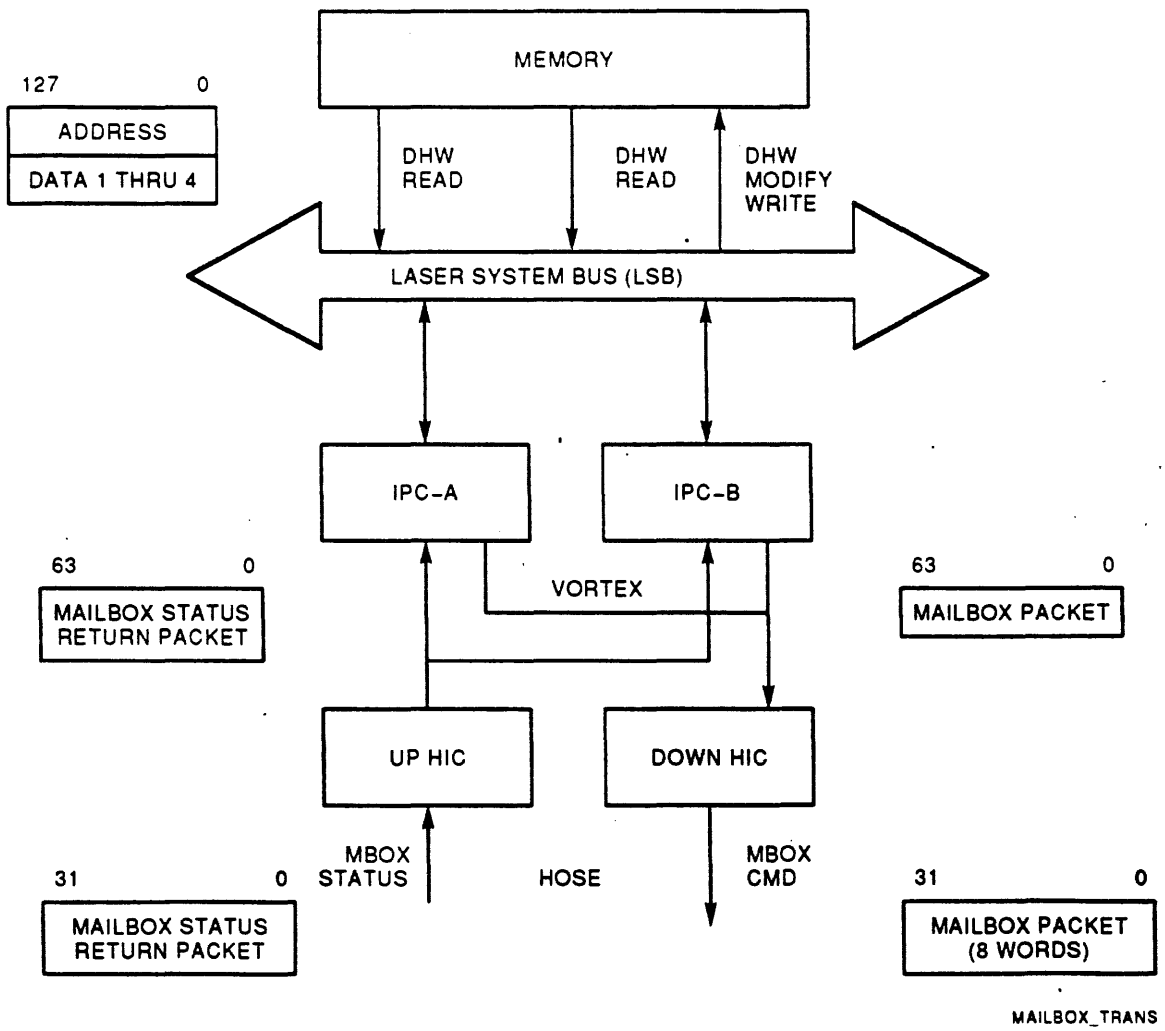
Table 7–1 Mailbox Data Structure Fields

Field Name	Contents Definition
HOSE	Number of the target hose (0 through 3)
MASK	Byte write mask used by LAMB
W	Write bit. Not used by LAMB.
B	Bridge bit. Force the remote bus adapter to respond with WHAMI register identification. Not used by LAMB.
REMOTE BUS COMMAND	LAMB has two commands: 1 for read and 7 for write.

The mailbox operation is performed in the following sequence:

- CPU*n* writes to the LMBPRn register.
- The IOP reads the mailbox.
- The IOP sends a mailbox packet down the hose.
- The LAMB module sends mailbox status packet up the hose.
- The IOP reads the mailbox again and merges the status packet.
- The IOP writes the data back within 15 LSB cycles using Priority 5.

Figure 7–9 Mailbox Transaction

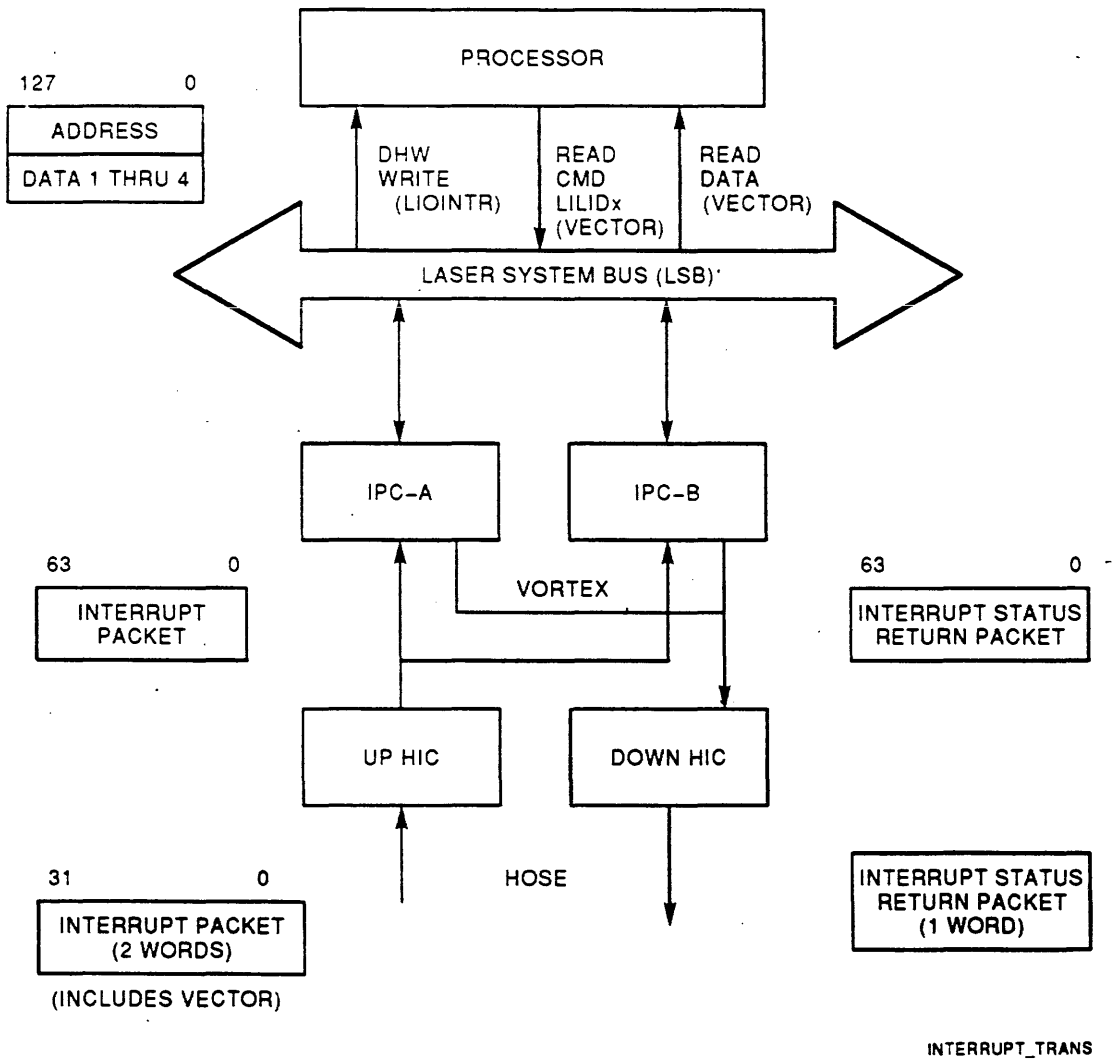


IOP Interrupts

The IOP initiates interrupts on the LSB from itself and also from the LAMB. Interrupts from the LAMB can originate within the LAMB or from a module on the XMI bus.

- IOP broadcast to LIOINTR using its LCPUMASK register
- CPU responds by reading vector from LILIDx register
- IOP sends interrupt status return packet down hose

Figure 7–10 Interrupt Transaction



Vectored interrupts use the LIOINTR (CPU), LCPUMASK (IOP), and LILIDX (IOP) registers as shown in Figure 7–11, Figure 7–12, and Figure 7–13.

Figure 7–11 LIOINTR Register (CPU)

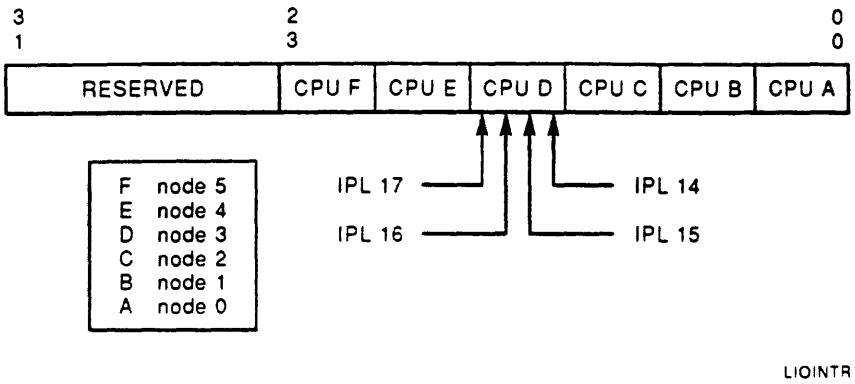


Figure 7–12 LCPUMASK Register (IOP)

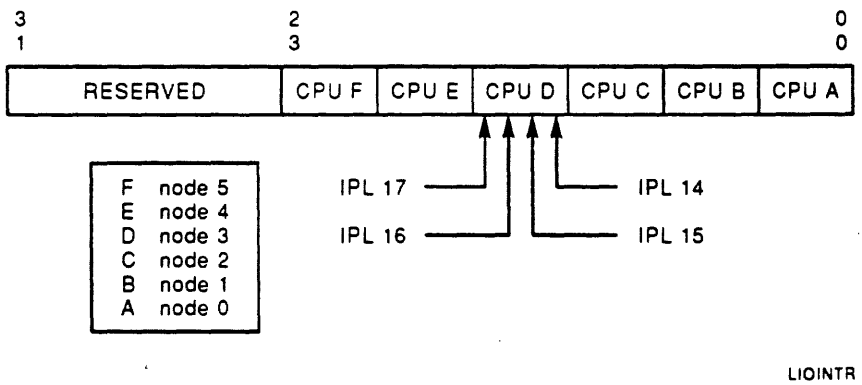
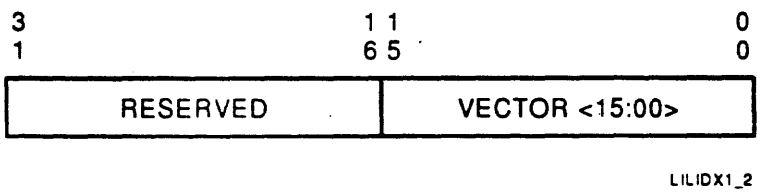


Figure 7–13 LILIDX Register (IOP)



As each interrupt packet is received from a hose, a vector is placed in the correct queue (IPL 14, 15, 16, 17). Each FIFO queue has four locations.

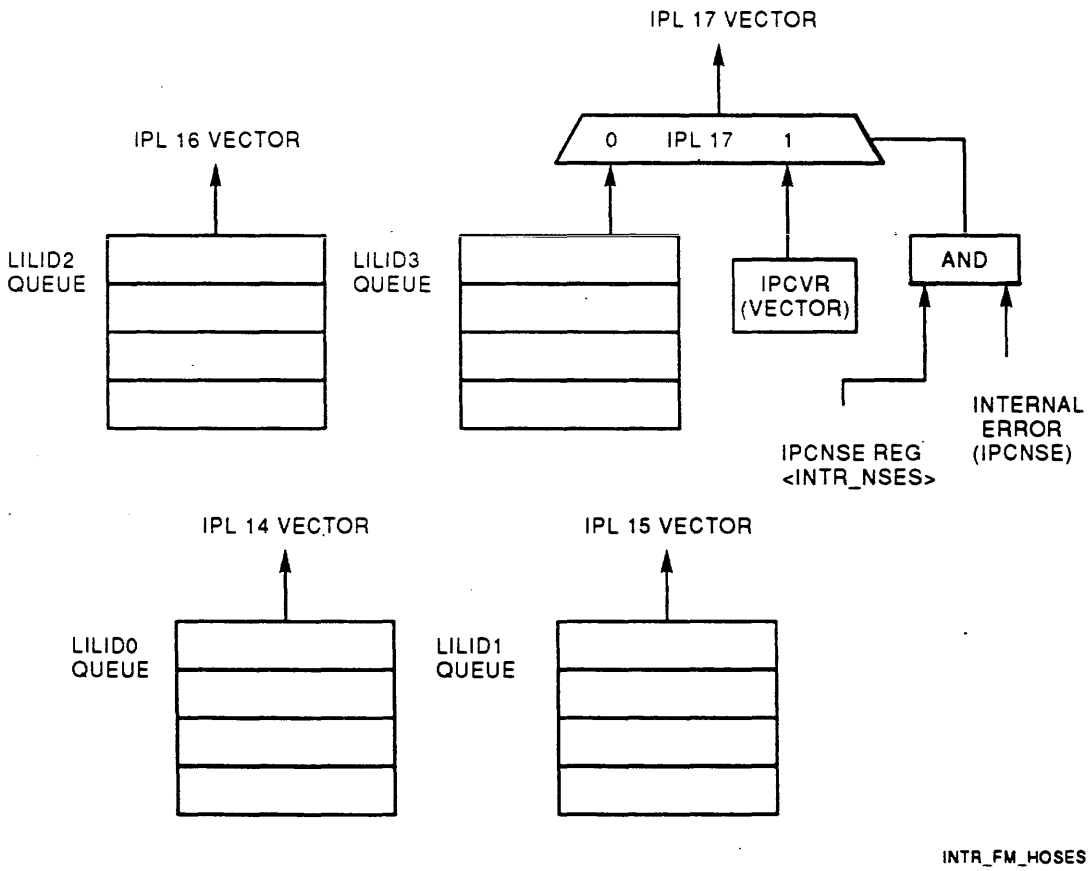
Each I/O module, such as the LAMB, may initiate only one interrupt at each level at the same time. A four-location queue on the IOP holds a vector for each hose at that IPL level.

When a processor reads from the LILIDx register the top entry from the queue is returned. When the queue is empty, the IOP returns zero to the processor.

The IOP uses IPL17 for error interrupts that occur within the module. If enabled, IPCNSE<INTR_NSES> bit set, the IOP returns the contents of IPCVR, its vector register.

Each processor maintains an interrupt pending queue at each IPL. The queue exceeds the maximum of five interrupts that could be read from the IOP at any one IPL.

Figure 7–14 Interrupts from Hoses



Vortex Bus

The vortex bus connects the IPCs to the UP and DOWN HICs. It has a cycle time of 20 ns (same as LSB). The signal lines and packets are described here.

Down Vortex Bus Packets

The IPC is always down vortex bus master. It keeps count of the number of buffer locations available in the HIC for each (3 maximum).

- IPC increments the counter for an HIC when it sends a packet down the vortex to that HIC.
- IPC decrements the counter when the down HIC signals that it has removed a packet from the down HIC buffer. The down HIC uses HIC_DN_DEC_PKT_CNT<3:0> to signal the IPC. The bits <3:0> indicate the hose number 0 through 3.

There are three types of down vortex packets:

- DMA read data return (or read error status)
- INTR/IDENT status
- Mailbox command

Up Vortex Bus Packets

The up HIC is always up vortex bus master. It keeps count of the number of buffer locations available in the IPC (2 maximum).

- Up HIC increments its IPC buffer counter when it sends a packet up the vortex to the IPC.
- Up HIC decrements the counter when the IPC removes a packet from its buffer (IPC signals IPC_DEC_PKT_BUF L).

There are six types of up vortex packets:

- DMA read command
- DMA interlock read command
- DMA write command (unmasked)
- DMA masked write command
- INTR/IDENT request
- Mailbox status

Hose Packet Protocol

The IOP supports a defined hose packet protocol which will accomodate the XMI I/O module (LAMB) module. Other modules which conform to the hose packet protocol could also be used with the IOP module.

The protocol is made up of two parts: down hose protocol and up hose protocol. Both are described here.

Down Hose Protocol

The I/O module indirectly controls the flow of packets on the down hose by limiting its requests.

- A mailbox packet may be sent down hose by the IOP at any time.
  - — The IOP is only allowed to send down one mailbox packet at a time regardless of which hose is its destination. The I/O module keeps one of its three packet buffers available for a mailbox packet.
- The I/O module can send up a DMA READ or an INTR/IDENT if it has a packet buffer location available for the response.

Down Hose Signal Lines

There are no ACK (CNF) signals as an acknowledge process would slow packet flow.

The downclock period is 40 ns (2 x LSB)).

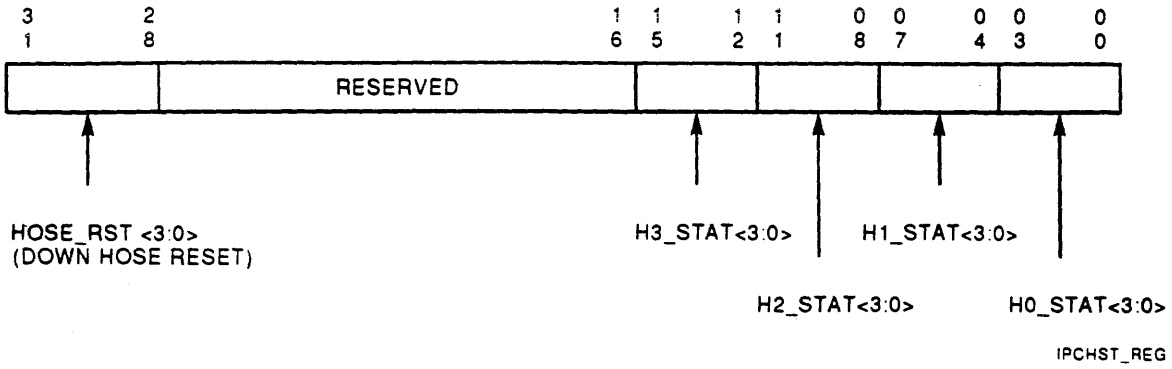
DECPKTCNT L tells the LAMB module that an entry has been removed from the UP HIC 3-location buffer.

When ERROR L is asserted, the LAMB module enters quiescent state—ignores up and down hoses.

Down Hose Packets

The down hose bus has the same three packets as the down vortex bus but with a different format.

Figure 7–15 IPCHST Register



The hose STATUS field <3:0> for each hose is decoded as follows:

- <3> = PWROK transitioned
- <2> = CBLOK present level
- <1> = PWROK present level
- <0> = ERROR transitioned from 0 to 1

UP Hose Packet Protocol

The I/O module controls the flow of packets on the up hose. The I/O module is aware that the UP HIC has a 3-location buffer. The I/O module maintains a counter 0, 1, 2. If the count is less than two the I/O module can send up a packet and increment the counter. When the UP HIC has removed a packet from the buffer (sent it to IPCs), it causes the signal DECPKTCNT L to be asserted. Seeing DECPKTCNT L asserted, the I/O module decrements its counter.

Up Hose Packets

The up hose bus has the same six packets as the up vortex bus but with a different format.

IOP Registers and Errors

IOP registers are listed and their user in reporting I/O subsystem errors is described in the following section.

IOP Registers

The IOP module maintains IOP-specific and LSB-required registers.

IOP Node-Specific Registers

The node-specific registers are listed in Table 7–2.

Table 7–2 IOP Node Specific CSRs

Address	Mnemonic	Register Name
FA00 2000	IPCNSE	IPC Node Specific Error Register
FA00 2040	IPCVR	IPC Vector Register
FA00 2080	IPCMSR	IPC Mode Selection Register
FA00 20C0	IPCHST	IPC Hose Status Register
FA00 2100	IPCDR	IPC Diagnostic Register

LSB-Required Registers on the IOP

LSB protocol requires that the registers listed in Table 7–3 be maintained by LSB I/O nodes.

The IOP conforms to the LSB protocol in maintaining the LSB required registers. The IOP does not implement the DIRTY and SHARED bits in the LBER0 register shown in Figure 7–17. The IOP has no concept of writeback cache.

Table 7–3 IOP LSB Required CSR Registers

Address	Mnemonic	Register Name
FA00 0000	LDEV	LASER Device Register
FA00 0040	LBER	LASER Bus Error Register
FA00 0080	LCNF	LASER Configuration Register
FA00 00C0	LIBR	LASER Information Base Repair Register
FA00 0200	LMMR0	LASER Memory Mapping Register 0
FA00 0240	LMMR1	LASER Memory Mapping Register 1
FA00 0280	LMMR2	LASER Memory Mapping Register 2
FA00 02C0	LMMR3	LASER Memory Mapping Register 3
FA00 0300	LMMR4	LASER Memory Mapping Register 4
FA00 0340	LMMR5	LASER Memory Mapping Register 5
FA00 0380	LMMR6	LASER Memory Mapping Register 6
FA00 03C0	LMMR7	LASER Memory Mapping Register 7
FA00 0600	LBESR0	LASER Bus Error Syndrome Register 0
FA00 0640	LBESR1	LASER Bus Error Syndrome Register 1
FA00 0680	LBESR2	LASER Bus Error Syndrome Register 2
FA00 06C0	LBESR3	LASER Bus Error Syndrome Register 3
FA00 0700	LBECR0	LASER Bus Error Command Register 0
FA00 0740	LBECR1	LASER Bus Error Command Register 1
FA00 0A00	LILID0	Interrupt Level 0 IDENT Register
FA00 0A40	LILID1	Interrupt Level 1 IDENT Register
FA00 0A80	LILID2	Interrupt Level 2 IDENT Register
FA00 0AC0	LILID3	Interrupt Level 3 IDENT Register
FA00 0B00	LCPUMASK	CPU Interrupt Mask Register
FA00 0C00	LMBPR0	Mailbox Pointer Register 0
FA00 0C40	LMBPR1	Mailbox Pointer Register 1
FA00 0C80	LMBPR2	Mailbox Pointer Register 2
FA00 0CC0	LMBPR3	Mailbox Pointer Register 3

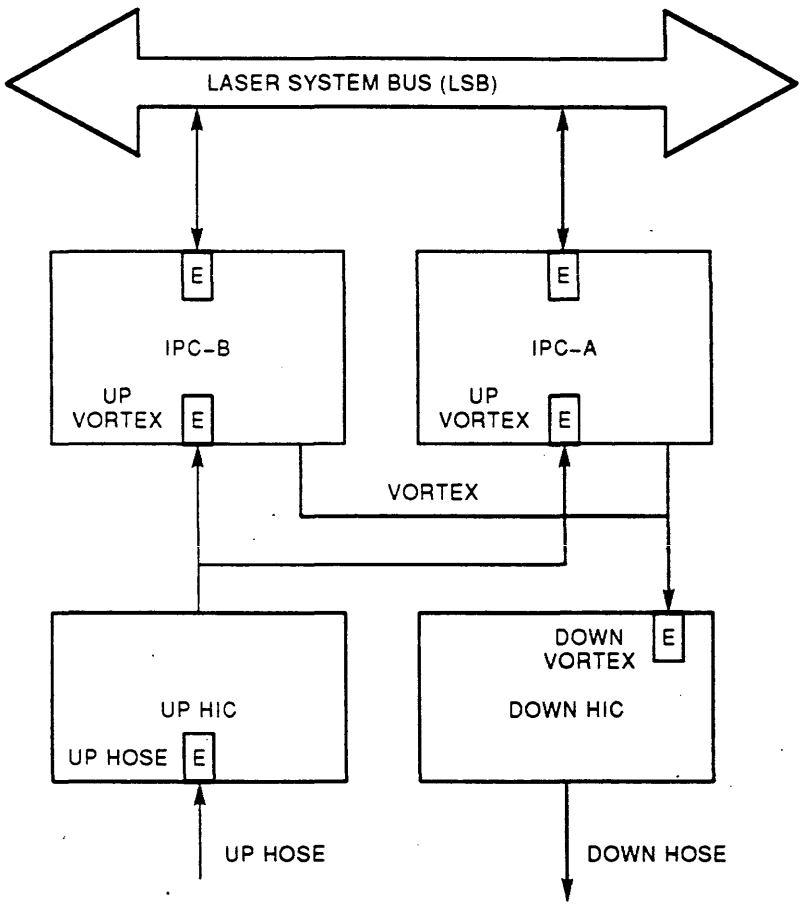
NOTE

All LSB node CSR Registers are aligned on natural 64-byte boundaries.

IOP Errors

The IOP has error detecting circuits located where signals enter a gate array as indicated by each *E* in Figure 7–16. There are also error detection circuits internal to the gate arrays.

Figure 7–16 IOP Error Detection



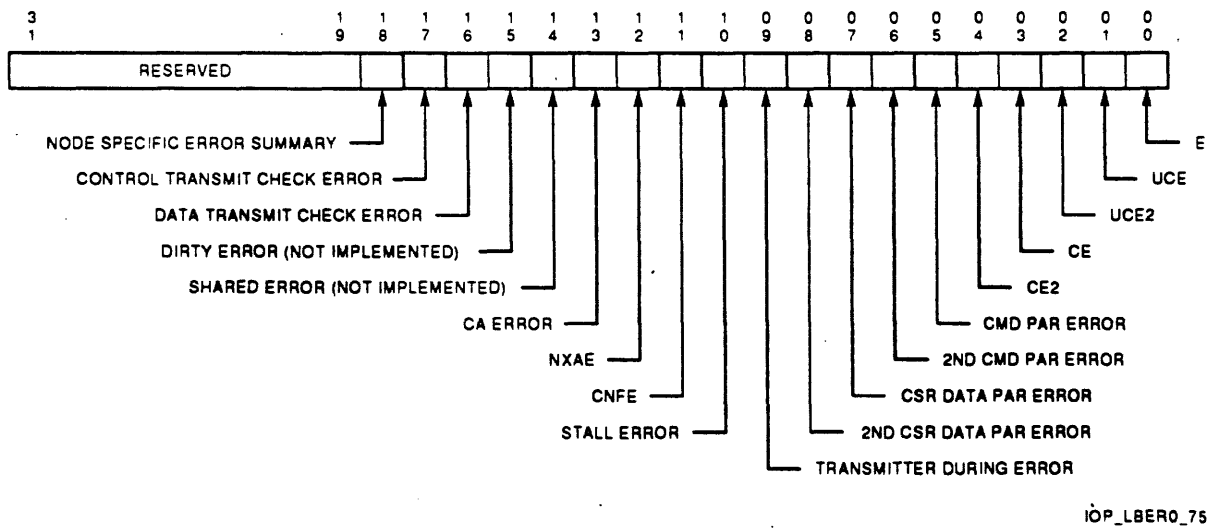
LSB Errors Detectable by IPCs

The IOP asserts the signal LSB ERR when it detects the error listed in Table 7–4.

Table 7–4 Summary of IOP Asserting LSB ERR

LBER Bit	LBER Bit Label	Error Description
<03>	CE	Correctable ECC error during a memory data cycle
<01>	UCE	Uncorrectable ECC error during a memory data cycle
<05>	CPE	Parity error during a command cycle
<07>	CDPE	Parity error during first data cycle of a CSR transaction
<10>	STE	STALL asserted illegally
<11>	CNFE	CNF asserted illegally
<12>	NXAE	IOP was commander when CNF not received for a C/A cycle
<13>	CAE	C/A asserted illegally
<16>	DTCE	Transmit check error on data lines while IOP was driving the bus
<17>	CTCE	Transmit check error on control lines while IOP was driving the bus

Figure 7–17 IOP LBER Register



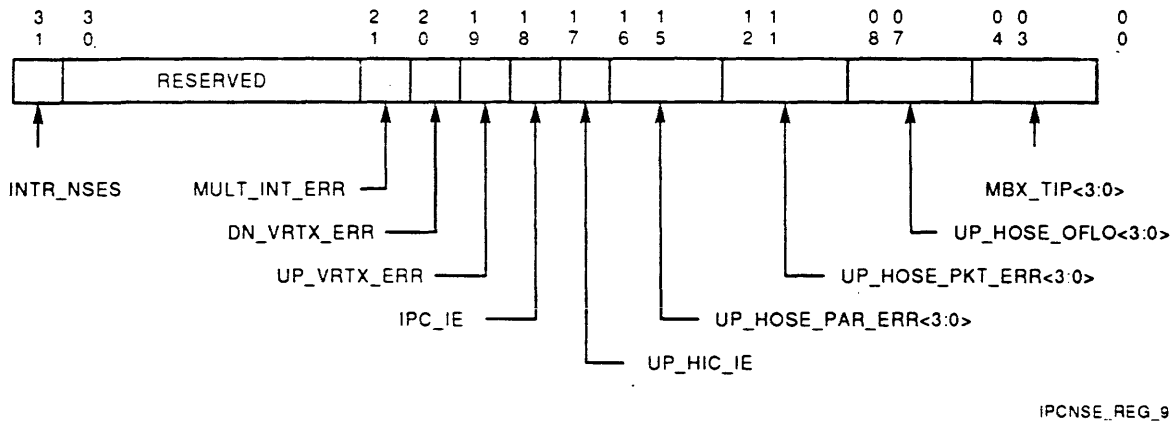
The IOP also monitors the LSB ERR line and takes these actions when ERR is asserted:

- Asserts LBER0 bit E (LBER0<00>)
- Inhibits outgoing arbitration for the next 16 cycles

IOP Internal Errors

Errors that are specific to the IOP module are captured and recorded in the IPCNSE register. These errors cause an interrupt to be posted at IPL 17 if IPCNSE<INTR_NSES> is set. They do not cause the LSB ERR line to be asserted nor LBER<ERR> to be set.

Figure 7–18 IPCNSE Register



Up Vortex Errors Detectable by IPCs

The IPCs can detect the following four types of errors coming from the vortex into the IPCs. IPCNSE<18>, UP_VRTX_ERR, is set if any of these errors occur.

- Parity error—odd parity bit for command and data lines for each IPC
- Illegal command—no LSB transaction
- Sequence error—wrong number of cycles for packet
- Buffer overflow—bad packet will not go onto LSB

IPC INTERNAL ERROR

When the IPC detects an internal error, it sets IPCNSE<17>, IPC_IE.

Down Vortex Errors Detectable by Down HIC

The down HIC can detect the following four types of errors coming from the vortex into the down HIC. IPCNSE<19>, DN_VRTX_ERR, is set if any of these errors occur:

- Parity error
- Illegal command
- Sequence error
- Buffer overflow

DOWN HIC INTERNAL ERRORS

When the down HIC detects an internal error, it also asserts IPCNSE<19>, DN_VRTX_ERR.

Up Hose Errors Detectable by UP HIC

The up HIC can detect the following four types of errors coming from the vortex into the up HIC:

- Up hose parity error—HOSEn_UP_PAR
  - — Odd parity is maintained over HOSEn_UP_DATA<31:00> and HOSEn_UP_CTRL<3:0>.
- Up hose packet error—UP_HOSE_PKT_ERR<3:0>
  - — Illegal command—bad command in HOSEn_UP_CTRL<3:0>
  - — sequence error—packet contains wrong number of cycles or has an illegal length code
- Up hose buffer overflow—UP_HOSE_OFLO<3:0>
  - — This error will occur when all IPC buffers are full and a packet is sent up the vortex bus.

UP HIC INTERNAL ERRORS

When the up HIC detects an internal error, it asserts IPCNSE<16>, UP_HIC_IE.

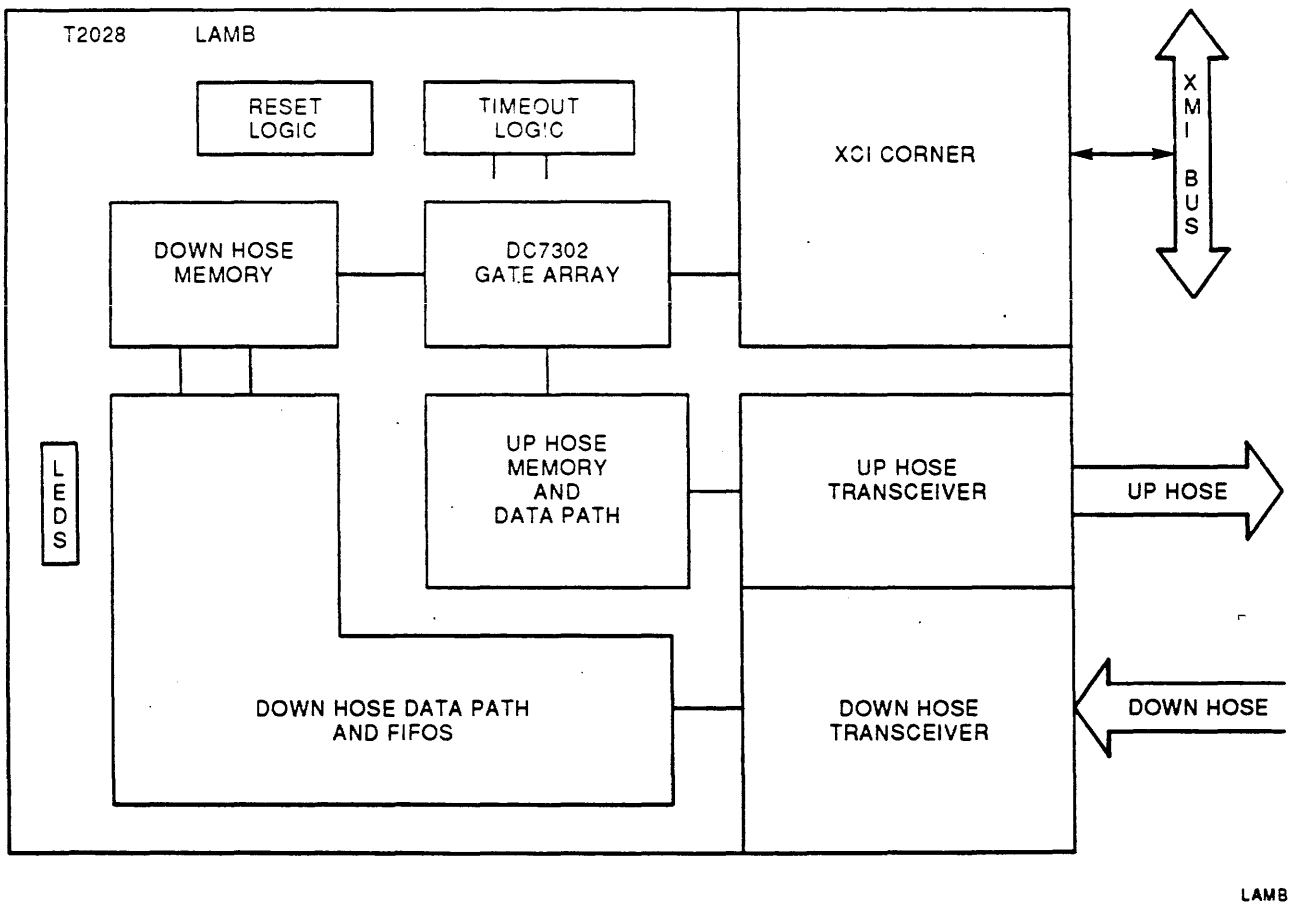
LASER-to-XMI Board (LAMB, T2028)

The LAMB module sits in the XMI card cage backplane and is connected via cables to the IOP module. There is a 4-to-1 cable assembly to the I/O bulkhead and then a hose (up and down cables) to the IOP module.

LAMB Overview

The LAMB module uses only +5 Vdc power and receives that from the power supply in the XMI card cage.

Figure 7–19 LAMB Block Diagram



DMA Read Example (XMI Originated)

- Up Hose (Read command)
  - — XMI corner
  - — LAMB gate array (LGA, DC7302) controls LAMB
  - — Up hose memory and data path
  - — Up hose transceivers
- Down Hose (Read return data)
  - — Down hose transceivers
  - — Down hose data path and FIFOs
  - — Down hose memory
  - — LGA
  - — XMI corner

MAILBOX Read Example (LSB Originated)

- Down Hose (MAILBOX packet)
  - — Down hose transceivers
  - — Down hose data path and FIFOs
  - — Down hose memory
  - — LAMB gate array (LGA)
  - — XMI corner
- Up Hose (MAILBOX status packet, with data)
  - — XMI corner
  - — LGA
  - — Up hose memory and data path
  - — Up hose transceivers

LAMB Operation

The LAMB operations to communicate between the XMI bus and the up and down hoses are described here.

LAMB on the XMI

The LAMB, as a responder, accepts all size generic READ and WRITE memory transactions on the XMI bus. It only accepts QW IREAD and QW UNLOCK WRITE memory transactions. It does not respond (NOACK) to LW, OW, or HW IREADs and nonQW UNLOCK WRITEs.

Data Length

- XMI bus supports QW, OW, and HW memory transactions.
- Up hose only supports OW, HW, and DHW data packets.
- XMI QW transactions are placed in up hose OW packets with QW length code.

XMI MORE Protocol

- Some XMI nodes use the MORE protocol to read and write large amounts of data from and to sequential memory locations.
- Largest XMI bus data length memory transactions are HW.
- LAMB supports MORE stream data by consolidating data and using DHW packets.

Mailbox

- The LAMB receives mailbox packets from the IOP and performs register read and write operations. The LAMB then assembles a mailbox status packet and sends it over the up hose to the IOP. The mailbox status packet contains one of the following:
  - Read data, status (sets E bit if access fails)
  - Write status (sets E bit if access fails)

Timeouts

- The LAMB module uses a timer while waiting for the following responses on the XMI bus:
  - — I/O read data
  - — IDENT vector
  - — XMI GRANT
  - — CNF for I/O command (retry timeout)
- The transaction is tried for 16 ms then will timeout, setting LAMB XBER<13>.

Arbitration

The LAMB module always uses its responder request to arbitrate for use of the XMI bus. It does this even on transactions for which it is a commander: READ I/O, WRITE I/O, and IDENT.

The LAMB module uses ARB SUPPRESS to stop incoming XMI transactions for several purposes, including these:

- While responding over the up hose to a mailbox packet which reads a LAMB module register
- When transmitting INTR/IDENT packets over the up hose
- When there are 6 or less free locations in up hose queue

Command and Data Paths

The simplified diagram of the LAMB module command and data paths is shown in Figure 7–20.

Down Hose Paths

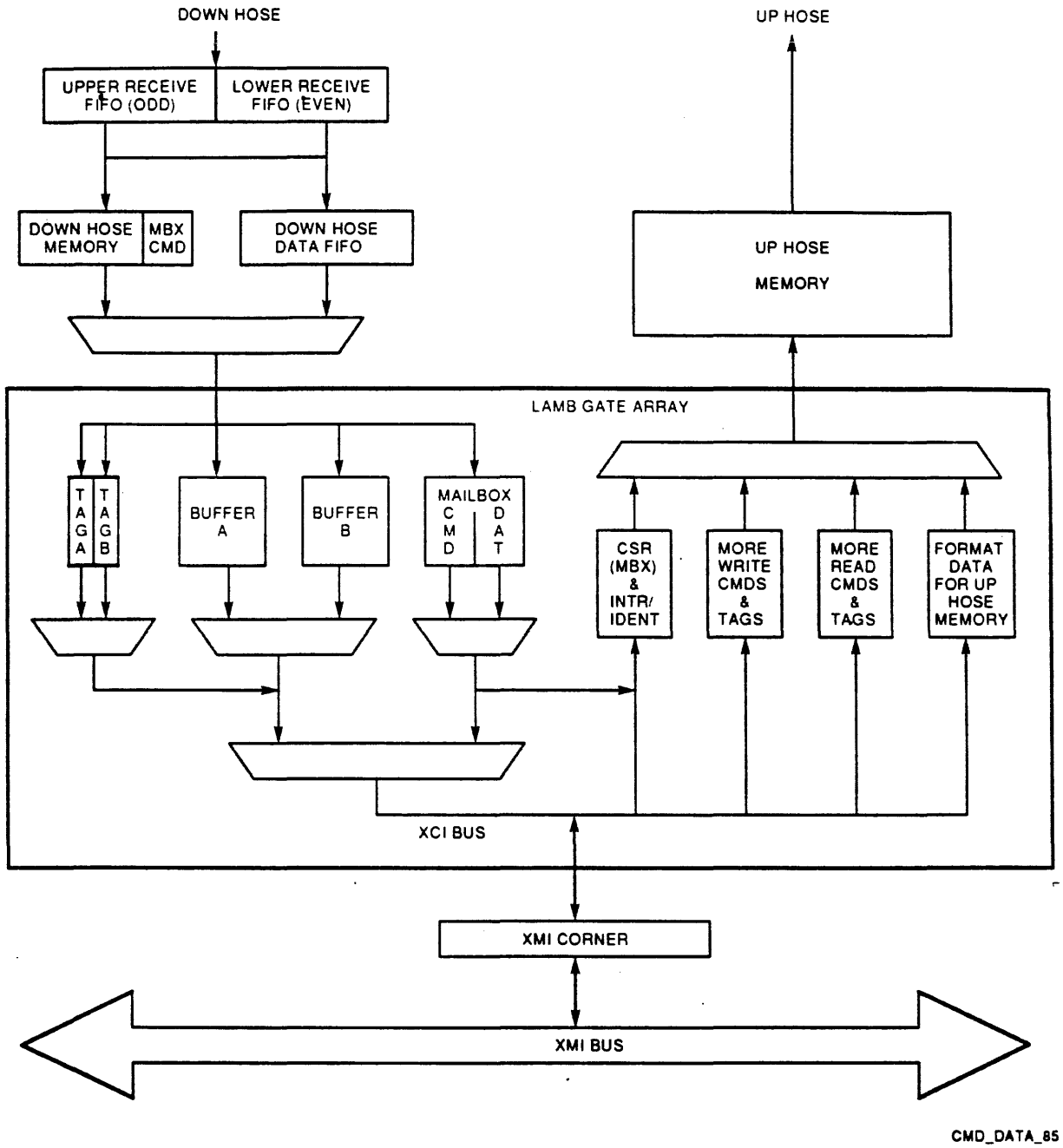
- Mailbox CSR Read
  - — If bit B (bridge) is set in mailbox, the remote bus adapter module returns the contents of its device register. This operation in not implemented in the LAMB.
  - — To a LAMB register, sends request to CSR and INTR logic. Contents of requested LAMB register sent up hose in mailbox response packet.
  - — XMI CSR
    - * WRITE—arbitrates for XMI, receives GRANT, sends mailbox response to up hose via CSR and INTR logic
    - * READ—arbitrates for XMI, sends READ command on XMI then puts returned I/O read data into mailbox response packet to up hose via CSR & INTR logic.
- DMA READ DATA RETURN—Sends data on XMI to requesting node.
- INTR/IDENT STATUS PACKET—Sends to CSR and INTR logic. INTR in progress is cleared.

Up Hose Paths

There is a 16-deep up hose transaction queue in the LGA that monitors packets stored in up hose memory. The LGA sends packets up hose when the IOP has an available buffer.

- DMA READ (VANILLA, NO MORE)—Store command in up hose memory.
- DMA READ (MORE)—Store command in up hose memory.
- DMA WRITE (VANILLA, NO MORE)—Store command and data in up hose memory.
- DMA WRITE (MORE)—Store command and data in up hose memory. Build (or update) WRITE MORE tag.
- MAILBOX CSR READ— Format CSR read return data into a mailbox status packet. Store mailbox return status packet in up hose memory.

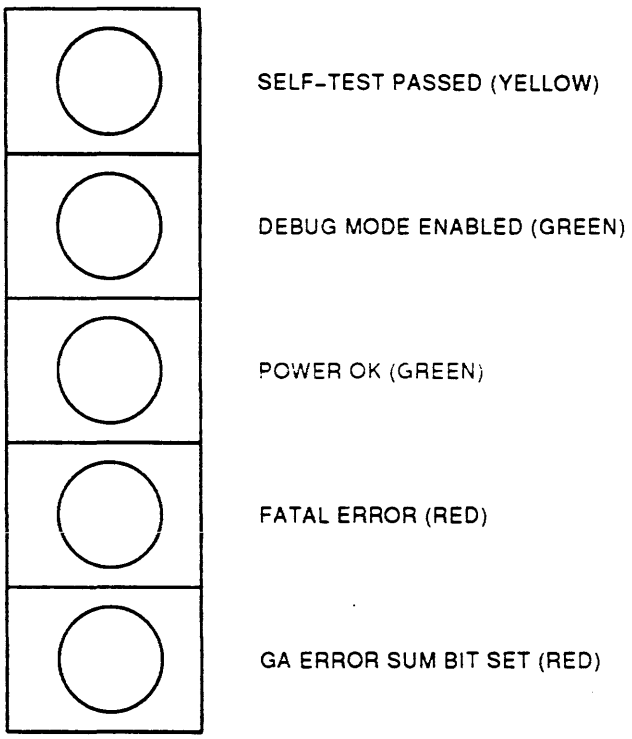
Figure 7–20 Command and Data Paths Block Diagram



LAMB Status LEDs

The LAMB module has five status LEDs located at the front edge of the module as shown in Figure 7–19.

Figure 7–21 LAMB Status LEDs



A LAMB LED is turned on as indicated here:

- Self-test passed—XMI BER <10> must be clear for this LED to be turned on.
- Debug mode enabled—LAMB must be in debug mode if LSB is running slower than 20 ns per cycle.
- Power OK—Power is on.
- Fatal error (hose ERROR Line asserted) caused by any one of the following:
  - — Down hose module fatal errors
  - — LGA fatal errors

- GA error sum (XBER<31>)—caused by any one of the following:
  - — XMI errors
  - — LAMB errors

LAMB Registers

The LAMB module maintains XMI required registers and LAMB specific registers. These registers are listed here and then registers used for interrupts and reporting I/O subsystem errors are described.

MAILBOX ILLEGAL ADDRESS, LERR<12>, is asserted if the register offset is greater than 7F (beyond range of LAMB registers).

XMI Required Registers on the LAMB

XMI protocol requires that the registers listed in Table 7–5 be maintained by XMI I/O nodes.

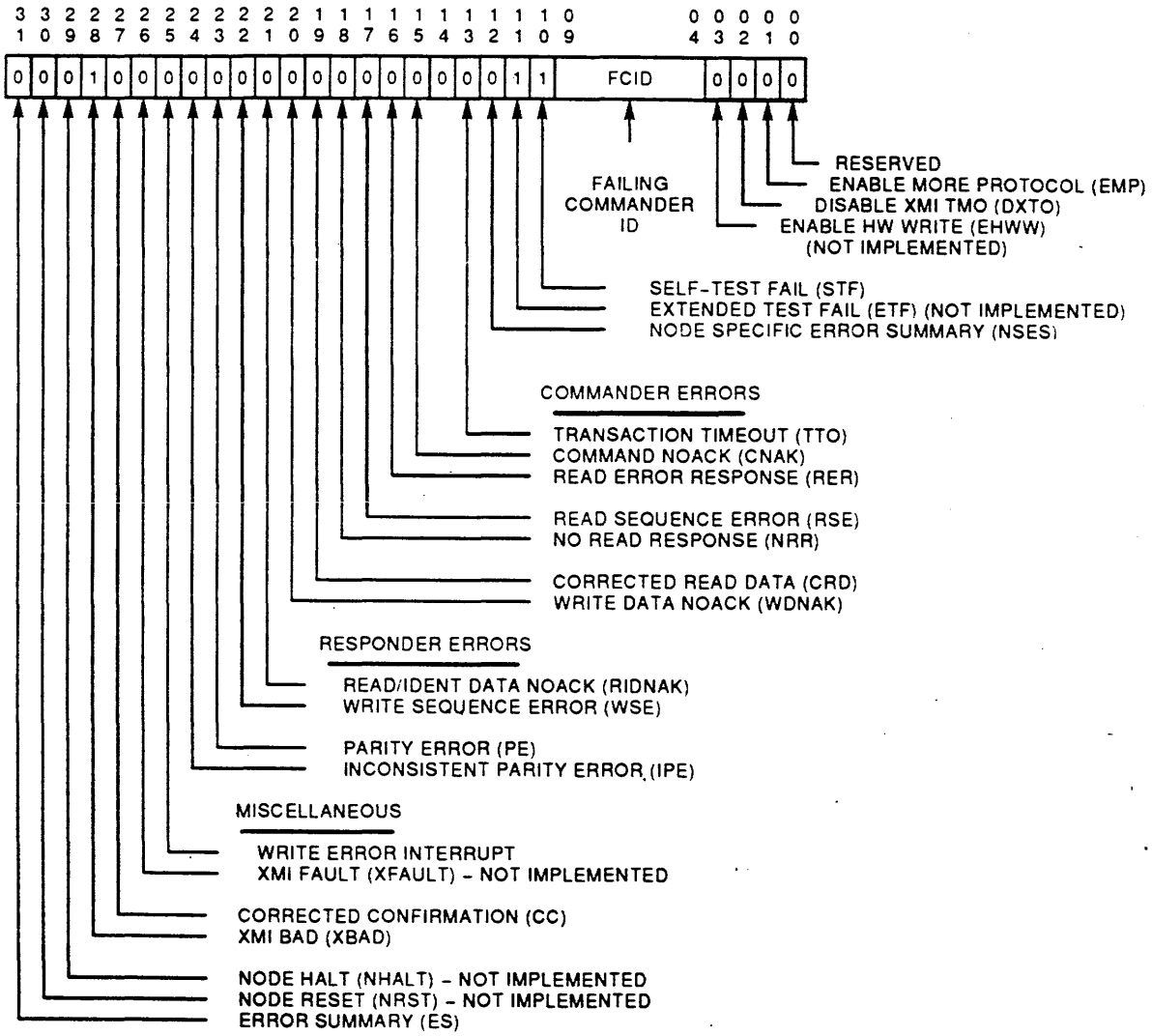
Table 7–5 XMI Bus CSRs on LAMB

Byte Offset	Mnemonic	Register Name
BB+00	XDEV	Device Type Register.
BB+04	XBER	Bus Error Register
BB+08	XFADR	Failing Address Register
BB+10	IBR	Information Base Repair Register
BB+2C	XFAER	Failing Extension Register

NOTE

Mailbox read and write transactions to and from LAMB CSRs do not go out onto the XMI bus.

Figure 7–22 LAMB XBER

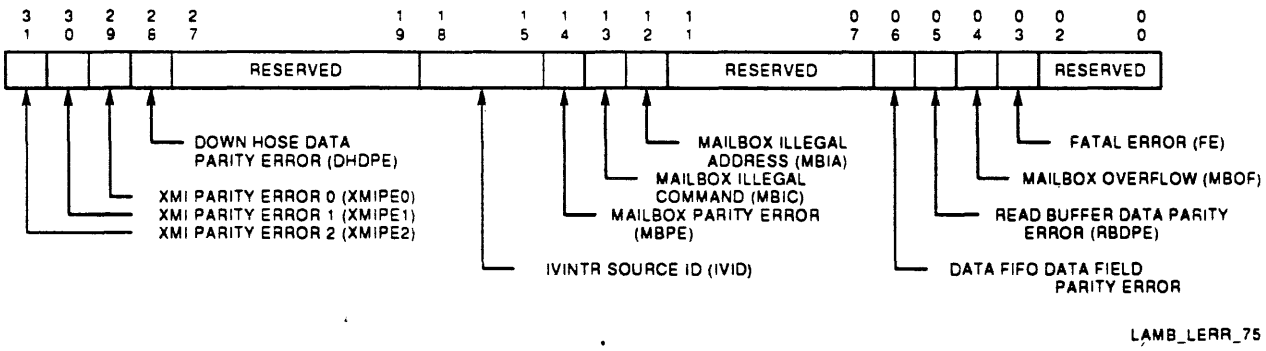


LAMB Specific Registers

Table 7–6 LAMB Specific CSRs

Byte Offset	Mnemonic	Register Name
BB+40	LDIAG	LAMB Diagnostic Register
BB+44	IMSK	LAMB Interrupt Mask Register
BB+48	LEVR	LAMB Error Vector Register
BB+4C	LERR	LAMB Error Register
BB+50	LGPR	LAMB General Purpose Register
BB+54	IPR1	LAMB Interrupt Pending Register 1
BB+58	IPR2	LAMB Interrupt Pending Register 2
BB+5C	IIPR	LAMB Interrupt In Progress Register

Figure 7–23 LAMB LERR



Interrupts from LAMB

The LAMB handles interrupts from two sources: the XMI bus and the module itself. The LAMB can accept an interrupt at each IPL (4) from each device on the XMI (14). It maintains 56 (4 x 14) interrupt pending flops for this purpose.

The status of pending interrupts can be determined by reading the two interrupt pending registers shown in Figure 7–24 and Figure 7–25.

Figure 7–24 Interrupt Pending Register 1 (IPR1)

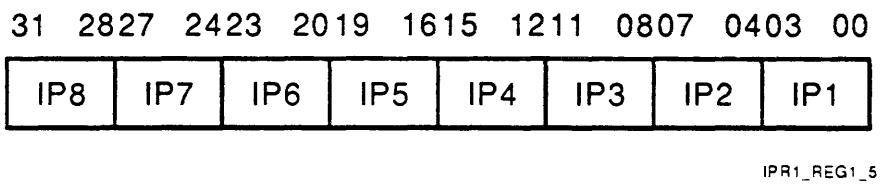
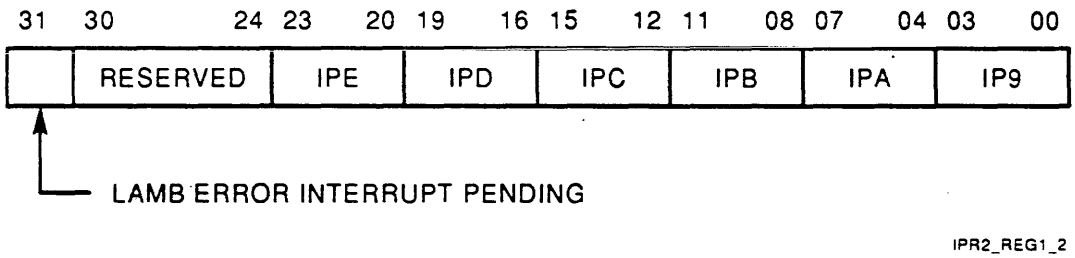


Figure 7–25 Interrupt Pending Register 2 (IPR2)

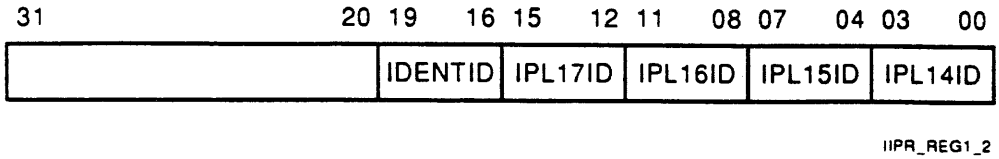


NOTE

The LAMB module itself can have an interrupt pending as noted in bit <31> of IPR2.

The LAMB module can send one interrupt at each IPL to the IOP module at a time. The status of the interrupts in progress is maintained in the Interrupt In Progress register shown in Figure 7–26.

Figure 7–26 Interrupt In Progress Register (IIPR)



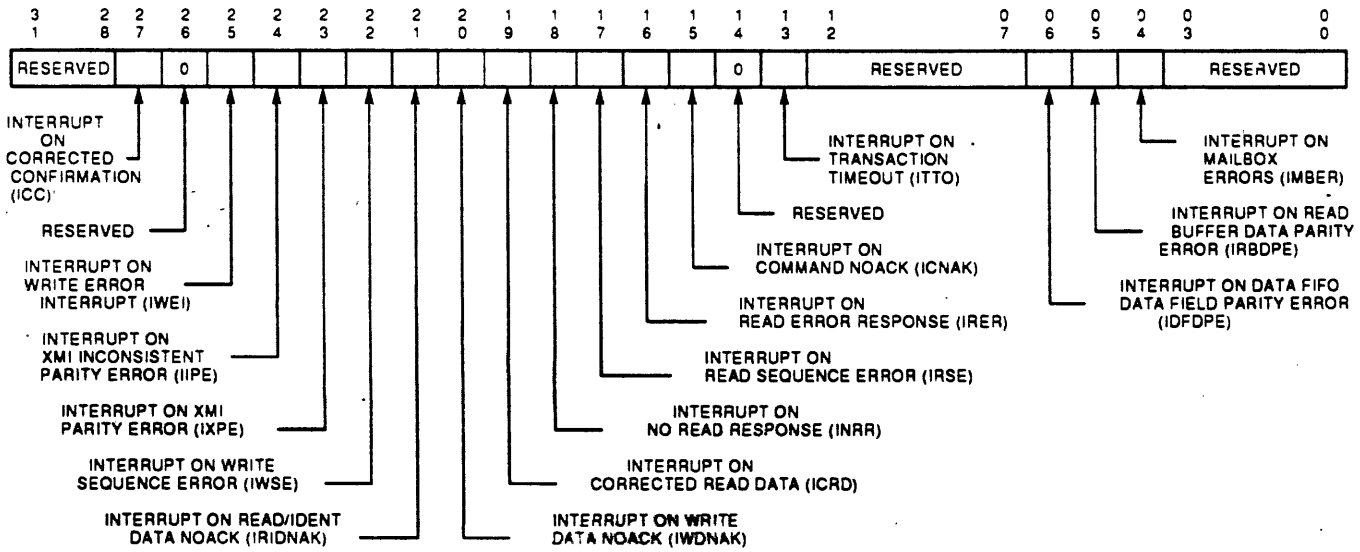
When the LAMB receives an interrupt from a device on the XMI it acknowledges the transaction and sets a bit in one of its IIPR registers. The LAMB later performs an IDENT transaction on the XMI to get the interrupt vector. It can only keep one vector for each IPL at a time (4 total).

The LAMB can have only one IDENT at a time in progress on the XMI. The node for which an IDENT has been sent but a response has not been received is shown in IIPR bits <19:16>.

When the LAMB detects errors at the XMI interface, it posts them in its XBER. Errors which it detects in itself are posted in the LERR.

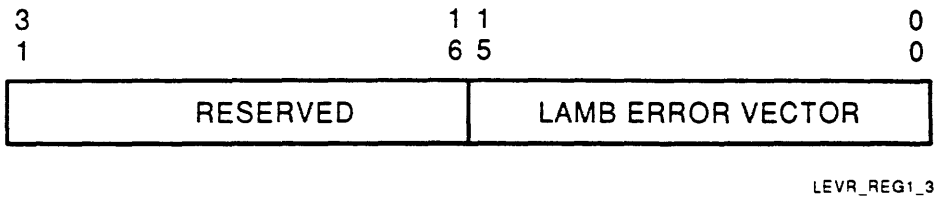
The LAMB uses the Interrupt Mask register, shown in Figure 7–27, to enable or disable error interrupts that occur because of error bits being set in the XBER and the LERR.

Figure 7–27 LAMB Interrupt Mask Register (IMSK)



The LAMB initiates an interrupt at IPL17 when an error occurs in itself (XBER or LERR). The LEVR contains the interrupt vector used by the LAMB for module INTR/IDENT packets to the IOP.

Figure 7–28 LAMB Error Vector Register (LEVR)



NOTE

There can only be one LAMB module interrupt in progress at one time.

The interrupt service routine must write the LAMB LERR (W1C) and then the LAMB XBER, in that order, to clear the module interrupt-in-progress state. The LAMB module is then able to initiate another module interrupt.

IVINTR Transactions

Nodes on the XMI can send IVINTR interrupts to the LAMB by identifying the LAMB node in the destination field of the transaction. The LAMB accepts IVINTRs with the WEI bit set but ignores IVINTRs with the IP bit set.

When the LAMB receives an IVINTR with the WEI bit set, it will:

- Set bit <25>, WEI, in its XBER register.
- Capture the ID of the interrupting node in LERR <18:15>.
- WEI starts a LAMB module interrupt at IPL17 if enabled by IMSK<25>.
- The INTR/IDENT packet contains the LEVR vector.

NOTE

The interrupt handler must check LERR for an IVINTR node ID.

LAMB Errors

The LAMB detects errors in three places:

- On the XMI Bus
- Down hose and down hose path inside LGA
- Up hose path inside LGA

The LAMB only retries transactions on the XMI bus. Transactions on hoses are not retried. If XMI bus transactions are not acknowledged the LAMB will retry them.

- I/O READ
- I/O WRITE
- IDENT

LAMB Error Groups

- XMI bus—XBER<31> is the logical OR function of XBER<27:10>
  - — Parity
  - — Responder
  - — Commander
- Up hose—data path errors
  - — Write sequence
  - — Parity error on MORE data
  - — Transaction queue overflow
- Down hose
  - — LAMB, data parity
  - — LAMB, fatal errors
  - — LAMB, mailbox
  - — LGA, parity
  - — LGA, fatal errors
  - — LGA, mailbox errors

- LAMB detected transaction errors
  - — DMA WRITE
  - — DMA READ
  - — MAILBOX
  - — INTR/IDENT

LAMB Reset and Power On

The LAMB power on sequence follows:

- CCL sequences XMI PRM ACLO and XMI PRM DCLO on the XMI bus
- XMI clock module (Slot 7) synchronizes XMI PRM ACLO and XMI PRM DCLO with the XMI clocks to generate XMI ACLO and XMI DCLO (no clocks, no XMI ACLO L or XMI DCLO L)
- LAMB, and all other XMI nodes, reset on XMI DCLO
- LAMB module receives deasserted XMI ACLO and uses it to drive the up hose signal PWROK H to the IOP module
- LAMB is ready to receive DOWN hose packets

LAMB Reset

The LAMB module only resets in response to a system reset.

- IOP asserts DNRST L to the LAMB.
- LAMB asserts XMI RESET L.
- CCL module senses XMI RESET L and sequences XMI PRM ACLO and XMI PRM DCLO.
- XMI clock module sychronizes XMI PRM ACLO and XMI PRM DCLO to the XMI clocks (no clocks, no XMI ACLO L or XMI DCLO L).
- LAMB sees assertion of XMI DCLO L and stops driving XMI RESET L.

NOTE

The LAMB module does not implement node reset, XBER<30>.

LSB I/O Operation Exercise

1. Which console command would a DSE use to check for the presence of a cable between the IOP module and a LAMB module on hose 1?

   a. >>>ex gpr:10

   b. >>>ex pmem:FA00 20C0

   c. >>>ex pmem:FA0020C0

   d. >>>ex pmem:05020C0

   e. None of the above

2. Which of the following statements describes the maximum number of interrupts that can be processed by the IOP module at one time?

   a. 16 interrupts: one at each level IPL14 to IPL17 from each remote bus interface module (LAMB)

   b. 4 interrupts: one at each level IPL14 to IPL17

   c. 1 interrupt at any of IPL14 to IPL17

   d. 17 interrupts: one at each level IPL14 to IPL17 from each remote bus interface module (LAMB) and an additional one at IPL17 from the IOP module itself

   e. None of the above

3. When the LAMB module suffers a fatal error, it stops communicating with the IOP module. How can a DSE tell if this happened?

   a. If the error summary LED on the LAMB (fifth down of five) is lit (red)

   b. If there is no response to the console command >>>EX PMEM:FA000000 -L

   c. If there is no response to the console command >>>EX PMEM:FA000004 -L

   d. If the fatal error LED on the LAMB (fourth down of five) is lit (red)

   e. None of the above

4. The IOP module is not aware of the conditional update writeback cache protocol.

   a. True

   b. False

5. If a QW WRITE UNLOCK transaction is sent on the XMI bus to the LAMB, how will the LAMB respond?

   a. The LAMB will acknowledge receipt of the data, and send it to the IOP in a WRITE UNLOCK packet.

   b. The LAMB will acknowledge receipt of the data and then abort the transaction.

   c. The LAMB will not acknowledge receipt of WRITE UNLOCK data.

   d. XMI modules will never perform a QW WRITE UNLOCK on the 7000/10000 systems LSB.

   e. None of the above.

6. Which one of the following operations can the IOP perform on the LSB?

   a. The IOP can read 64 bytes from memory, modify some or all of the data, and write it back without allowing any other node to access that memory location.

   b. Broadcast interrupt requests to all CPUs on the LSB during the same transaction.

   c. Work with the LAMB module to simulate an INTERLOCK operation to LSB memory.

   d. Override LAMB interrupts to initiate an IPL17 interrupt from itself.

   e. All of the above.

7. The hose is a synchronous connection that runs at 50 MHz.

   a. True

   b. False

8. Which one of the following statements about IOP participation in mailbox operation is true?

   a. The IOP reads mailbox contents in memory and decodes the priority of the operation.

   b. The IOP reads mailbox contents but does not decode any of it. The IOP just passes the data to the LAMB.

   c. The IOP reads mailbox contents and decodes it to find out which hose to send a packet down.

   d. The IOP does not read the mailbox contents without instructions from the LAMB. It sends the address of the mailbox to the LAMB and waits for instructions.

   e. None of the above.

9. Which one of the following statements is true?

   a. The down hose has a 64-bit data path, while the up hose has a 32-bit data path.

   b. The down hose has a 32-bit data path, while the up hose has a 64-bit data path.

   c. The IOP and hose support full duplex communication.

   d. The IOP and hose support half duplex communication.

10. Which one of the following statements is true?

   a. The LAMB can handle DHW transactions, but the XMI only supports HW transactions.

   b. The LAMB can handle HW transactions, but the XMI only supports QW transactions.

   c. The LAMB can handle DHW transactions, but the IOP only supports HW transactions.

   d. The LAMB can handle QW transactions, but the XMI only supports LW transactions.

11. When the LAMB experiences a fatal error, it ceases communication with the IOP.

   a. True

   b. False

VAX 7000-600 and VAX 10000-600 Systems Fault Management

Introduction

The sequence of actions performed by the VAX 7000-600 system during a machine check operation are described in this training module. The DSE is shown how to obtain and interpret an errorlog printout.

Objectives

A Digital Services Engineer who maintains a VAX 7000-600 system should be able to:

- Recognize the sequence of actions performed by a VAX 7000-600 system during a machine check operation.
- Obtain and interpret an errorlog printout.
- Recognize VAX 7000-600 system error handling paths using the SCB.

Resources

VAX 7000 Pocket Service Guide	EK-7000A-PG
VAX 7000/10000 KA7AA CPU Technical Manual	EK-KA7AA-TM

VAX 7000/10000-600 System Processor Registers

- KA7AA Internal Processor Registers
- KA7AA Specific Registers
- KA7AA LSB Registers

Error Events

The VAX 7000/10000-600 system faults cause the following events.

- Console error halts
- Machine check (SCBB+04)
- Hard error (SCBB+60)
- Soft error (SCBB+54)
- IOP adapter error IPL17 (SCBB+DC)
- DWLMA (LAMB) adapter error IPL17 (SCBB+DC)
- Kernel-stack-not-valid exception (SCBB+08)

Console Error Halts

A console halt can occur for any of several reasons: power on, assertion of the external halt interrupt pin (HALT_H), or by certain microcode-detected double error conditions.

The reason for a console double-error halt is indicated by a display on the console terminal. The display is presented in a format similar to the console crash shown in Example 8–1.

Example 8–1 Console Mode Error Halt

```
**** BTAG STORE RAM MARCH TEST ****

CPU0: unexpected exception/interrupt, vector 60 (18)
process cpu_tst, pcb = 0015a720, pc: 0006cf82 psl: 00000000
Interrupt/Exception: hard error notification
LMERR: 00000000 LMODE: 00000028 LBER:  00001007 LLOCK: 00003a32
LDEV:  00008002 LCNR:  80000000 LBESR0:0000000c LBESR1:0000000c
LBESR2:0000000c LBESR3:0000000c LBECR0:0000ca60 LBECR1:00000040

BIU_CTL: afe09ff8 DIAG_CTL: 00000001 BC_TAG:    00003800 BIU_STAT: c00e1031
BIU_ADDR:00194c08 FILL_SYN: 00000000 FILL_ADDR: 000002a8
gprs:
0: 00194C20   3: 00194BC0   6: 00194C00   9: 00000000  12: 0015BDB8
1: 003FFFF8   4: AA800000   7: 0002174C  10: F7800100  13: 0015BDA0
2: 0001FFFF   5: 00194C00   8: 0004BDE8  11: 00194C00  14: 0006CF82
ksp: 0015BDA0 esp: 00000000 ssp: 00000000 usp: 00000000

CONSOLE CRASH: Type ;P to exit XDELTA and return to console

1 brk at 00077CAB
00077CAB/NOP

F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #
                        A   M   .   .   .   .   .   .   P   TYP
                        o   +   .   .   .   .   .   .   +   ST1
                        .   .   .   .   .   .   .   .   B   BPD
                        o   +   .   .   .   .   .   .   +   ST2
                        .   .   .   .   .   .   .   .   B   BPD
                        +   +   .   .   .   .   .   .   +   ST3
                        .   .   .   .   .   .   .   .   B   BPD

    .   .   .   .   .   .   .   .   .   .   .   .   .   .       C0
    .   .   .   .   .   .   .   .   .   .   .   .   .   .       C1
    .   .   .   .   .   .   .   .   .   .   .   .   .   .       C2
    .   .   .   .   .   .   .   .   .   .   .   .   .   .       C3

                            .   A0  .   .   .   .   .   .           ILV
                            .   64  .   .   .   .   .   .           64Mb

Firmware Rev = T1.0-2044      SROM Rev = V00.02      SYS SN = **************
EEPROM Image failed to verify

>>>
```

The DSE may use the console MCHK script to get information about the state of the processor.

The DSE may then use the console CRASH command to try to obtain an operating system crash dump. The CRASH command causes the console to boot and hand the operating system a dump argument. The operating system then attempts to perform a crash dump.

NVAX+ and LEVI Use of B Cache

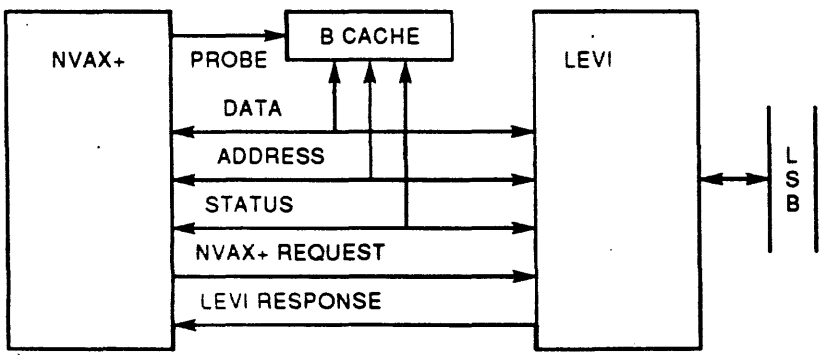
The NVAX+ and LEVI share use of the B cache, although the LEVI is responsible for B cache coherency. The NVAX+ probes B cache to see if a direct read or write can be accomplished. If the status is OK, then the NVAX+ reads or writes to the B cache without assistance. If the status is not OK, then the NVAX+ puts a request on cReq_h<2:0>.

- 001 Barrier (NVAX+ never issues)
- 010, 011 Fetch, FetchM (NVAX+ never prefetches)
- 100 READ_BLOCK (B cache miss)
- 101 WRITE_BLOCK (B cache miss)
- 110 LDxL (LOAD LOCK, start LOCK)
- 111 STxC (STORE CONDITION, clear LOCK)

The NVAX+ makes a request then is stalled waiting for the LEVI to release it. The LEVI releases it with the following responses on cAck_h<2:0>:

- 001 Hard error
- 010 Soft error
- 011 Store conditional failure
- 100 OK

Figure 8–1 NVAX+ and LEVI Cache Arrangement



Errorlog Entries

Table 8–1 VAX 7000-600 System Errorlog Entries

04	Machine check	Abort	Parameters on interrupt stack
60	Hard error	Interrupt	IPL is 1D (hex)
54	Soft error	Interrupt	IPL is 1A (hex)
DC	Device error	Interrupt	IPL is 17 (hex)

- MCHK errorlog entry
  - — Processor error (hard error detected by NVAX+)
    - * B cache probe
    - * B cache access
    - * Request to LEVI (LSB transaction)
  - — Possible subpackets
    - * DLIST
    - * LSB
    - * LMA
    - * I/O adapter (currently only IOP)
    - * P cache TAG
    - * P cache DATA
    - * VIC TAG
    - * VIC DATA
- Hard error INT60 errorlog entry
  - — System error (hard error not detected by NVAX+)
  - — Possible subpackets
    - * DLIST
    - * LSB
    - * LMA
    - * I/O adapter (currently only IOP)

- Soft error INT54 errorlog entry
  - — Soft error
    - * SBEs, tag or data error in P cache, VIC
    - * Some INT54 types are uncorrectable and will result in machine checks
  - — Possible subpackets
    - * DLIST
    - * LSB
    - * LMA
    - * P cache TAG
    - * P cache DATA
    - * VIC TAG
    - * VIC DATA
- CRD soft errorlog entry
- MEMSCAN errorlog entry
  - — Controller error detected while polling LMEM for CRDs.
- ADPERR (IOP, IPL17) errorlog entry
  - — Hard error on adapter (IOP interrupt)
  - — Possible subpackets
    - * Channel (LAMB)
- ADPERR (LAMB, IPL17) errorlog entry
  - — Hard error on adapter (LAMB Interrupt)
  - — Possible subpackets
    - * XMI
- LASTFAIL errorlog entry
- CONFIG errorlog entry
- POWER errorlog entry

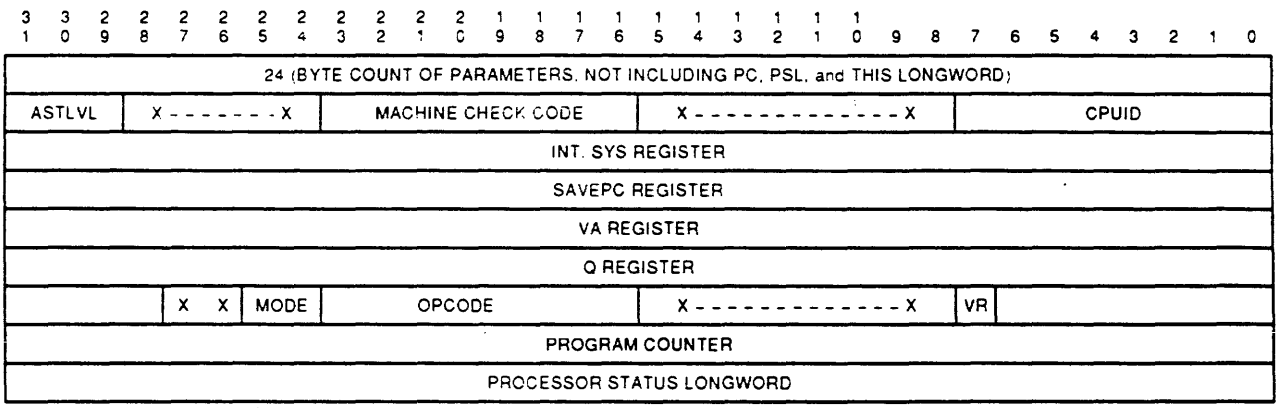
Error Packet and Subpacket Formats

The format of a packet and a subpacket are covered here. These packets are logged by operating system error handling code.

Machine Check (04)

The processor puts the machine check stack frame on the interrupt stack and then takes the jump through SCBB+4 to the error handling routine.

Figure 8–2 VAX 7000-600 Processor Machine Check Stack Frame



NEON_MCHK80

Hard Error (60)

The processor pushes the PC and PSL pushed on the interrupt stack and takes the jump through SCBB+60.

Soft Error (54)

The processor pushes the PC and PSL pushed on the interrupt stack and takes the jump through SCBB+54.

Hex Dump of Errorlog Entry

A hex dump of an errorlog entry can be obtained using the format shown in the following command sequence. The hex dump would only be used if the DSE suspects that ERF did not get all the data available.

Example 8–2 Hex Dump of Errorlog Entry

```
$SET DEF [SYSERR]
$ANA/ERR/BIN=146.BIN/ENTRY=(S:146,E:146) ERRORLOG.SYS
$DUMP/RECORD/OUT=146.DMP 146.BIN
```

Error Parse Trees and Descriptions

The parse trees for error types use register contents to diagnose the error. The DSE must be familiar with system operation and system registers to comprehend the logic of the parse trees.

A description is provided for each error diagnosis.

VAX 7000/10000-600 Errorcode Overview

The VAX/VMS SYSLOA facility contains, among other things, error handling and recovery routines for the VAX 7000/10000 systems. The file is SYSLOA1701.EXE, and it contains:

- MCHECK1701.MAR—MCHK (04), HERR (60), SERR (54)
- ERRSUB1701.MAR—cache, bugcheck, powerfail
- ADPLSBERR.MAR—LSB adapters
- SYSL_LMA_ROUTINES.MAR—LASER memory
- SYSL_CRD_ROUTINES.MAR—correctable errors
- SYSL_RDS_ROUTINES.MAR—uncorrectable errors
- ADPXMIERR.MAR—XMI device errors
- ADPBIERR.MAR—VAXBI device errors

VAX 7000/10000-600 Systems Fault Management Exercise

1. If a KA7AA module has its LBER<0> set, then its BUI_STAT<0> bit must be set.

   a. True

   b. False

2. Which one of the following statements is true?

   a. A hard error not detected by the NVAX+ chip but detected by the LEVI-A/Bs is reported as a Machine Check using vector 04.

   b. A hard error detected by the NVAX+ chip is reported as a Machine Check using vector 04.

   c. A soft error detected by the NVAX+ chip is processed using vector 660.

   d. A soft error detected by the LEVI-A/Bs is processed using vector 60.

   e. All of the above.

3. Which one of the following statements is true?

   a. A hard error detected by the IOP module is processed as an interrupt at IPL17.

   b. A hard error detected by the LAMB module is processed as an interrupt at IPL16.

   c. A hard error detected by the LMEM module is processed as an interrupt at IPL17.

   d. A hard error detected by a secondary processor is processed by an interrupt at IPL17 to the primary processor.

   e. None of the above.

4. The LASTFAIL packet in an errorlog entry tells which FRU was the most recent to fail and be replaced.

   a. True

   b. False

DEC 7000-600 and DEC 10000-600 Systems Differences

Introduction

This module highlights the differences between the VAX 7000/10000-600 systems and the DEC 7000/10000-600 systems. The main hardware differences are in the processor modules and support of I/O devices.

Objectives

A Digital Services Engineer who maintains a DEC 7000-600 or DEC 10000-600 system should be able to:

- Recognize the differences between the VAX 7000-600 and DEC 7000-600 systems.
- Recognize the differences between the VAX 10000-600 and DEC 10000-600 systems.
- Be familar with DEC 7000/10000-600 system maintenance features.

Resources

DEC 7000 AXP System Pocket Service Guide	EK-7700A-PG

DEC 7000-600—VAX 7000-600 Differences Overview

The main hardware difference between the DEC 7000-600 system and the VAX 7000-600 system is the processor module.

A system with one LEP processor module has a performance measurement of 150 SPECmarks, while a 4 processor system will perform up to 600 SPECmarks. The LEP processor puts proven CMOS 4 technology in a RISC processor using Alpha AXP architecture in the LASER platform.

The DEC 7000-600 system differs from the VAX 7000-600 system in several ways. Some of the main differences are listed in Table 9–1.

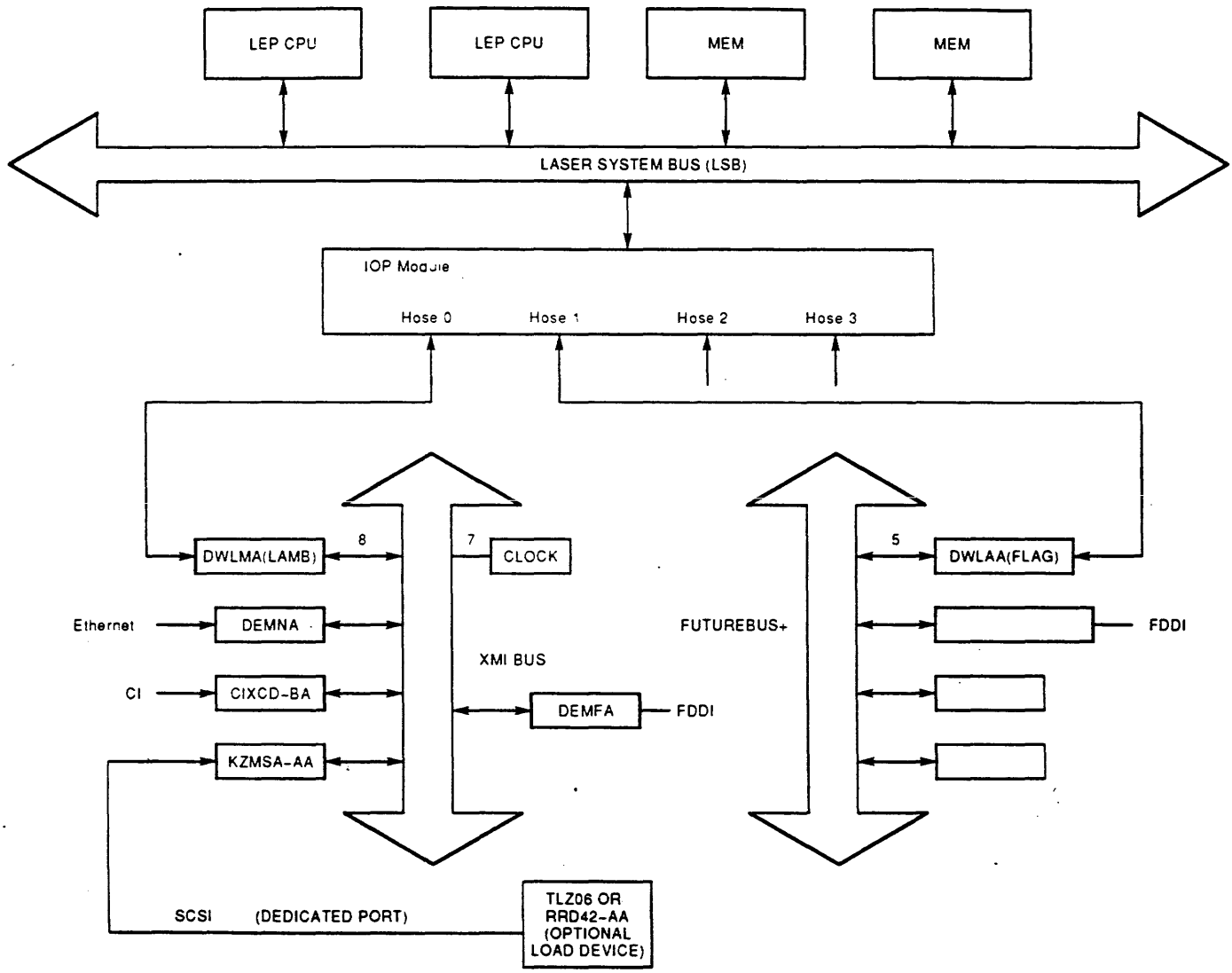
Table 9–1 VAX 7000-600—DEC 7000-600 Differences

Item	DEC 7000-600	VAX 7000-600
Processor modules	KN7AA-AA, KN7AA-YA (LEP, E2040-AA, E2040-YA)	KA7A-AA (LNP, E2045-AA)
Architecture	RISC (Alpha AXP)	CISC (VAX)
Operating systems	OpenVMS Alpha AXP OSF/1	OpenVMS VAX
Disk storage at FRS	KZMSA-AB (SCSI)	KFMSA-BA (DSSI)
Load device	TLZ06 or RRD42	TF85D
CPU module	LEP	LNP
Processor chip	EV4 chip (DC228)	NVAX+ chip (DC262)
16 MHz oscillator	Not used	For internal counter
Interval timer interrupt	Yes, Watch chip	Not used
GBus IPAL0	Alpha AXP interrupt scheme	VAX interrupt scheme
FRS disk storage	RZ73 (SCSI)	RF73 (DSSI)
Console memory cluster	About 2 MB	About 1 MB

System Functional Description

A typical DEC 7000-600 system functional diagram is shown in Figure 9–1

Figure 9–1 Typical DEC 7000-600 System Functional Diagram

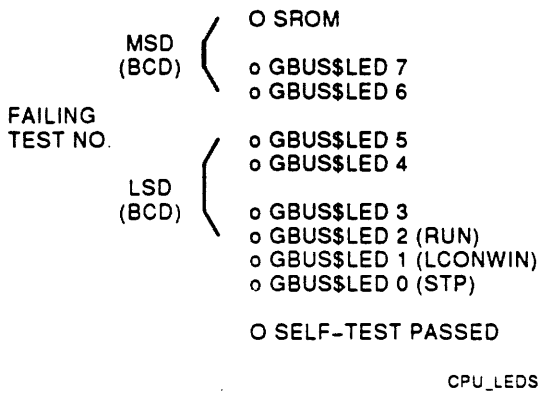


The DEC 7000-600 system is expected to support the FBUS+ in the future. The FBUS+ is an IEEE standard bus. Its bandwidth is similar to the XMI bus.

KN7AA Initialization Sequence

LEP processor modules each have a set of status LEDs visible at the outer edge of the module. The ten LEDs are shown in Figure 9–2.

Figure 9–2 Processor Module LEDs



If the CPU fails its self-test during the first two phases, SROM and GROM, the failing test is indicated in the CPU LEDs (MSD and LSD). A lit LED indicates a logical one.

- SROM—SROM clock.
- RUN—Reflects status of OCP run light.
- LCONWIN—Is turned on by primary CPU as it becomes the system console.
- STP—Self-Test Passed is lit when CPU successfully passes its self-test.

The failing test LEDs count up sequentially as power on tests are performed. The LEDs indicate 1–14 for LEP SROM tests and 15–55 for LEP GROM tests. The LEDs are lit to indicate the test number at the beginning of a test, and if the test fails, the LEDs remains lit.

The LEDs are useful when the processor executes its self-tests and contact with the console terminal has not been established. The status of the processor is indicated in the LEDs.

Eight of the LEDs are driven by bits <7:0> in the GBUS$LEDS register. This read/write register is accessible from the console at physical address 3 F700 0040.

LEP Processor Self-Test

There are two levels of processor self-tests: SROM and GBus Flash ROM. Tests contained in SROM are performed first and if successful they are followed by tests contained in GBus Flash ROM. These power on tests run with LDIAG.FIGN bit set which causes the module to ignore all LSB activity.

SROM Self-Test

The LEP processor has 14 SROM self-tests that run at power on only. The SROM self-tests verify:

- Processor chip
- Clocks
- Cache and cache control
- GBus devices
- LSB interface (self-directed transaction on LSB)

GBus Flash ROM Self-Test

The GBus Flash ROM (GROM) self-tests verify those parts of the processor module not tested by the SROM tests.

The LEP processor runs 41 GROM self-tests at power on or when initialized.

CPU/Memory Self-Test

The LEP processor runs 10 CPU/memory self-tests at initialization.

Multiprocessor Self-Test

The LEP processor has 7 multiprocessor (MP) tests. At least two processors must be present to run the MP tests.

I/O Port Self-Test

The LEP processor runs IOP and LAMB tests.

DEC 7000/10000-600 System Diagnostics and Exercises

Diagnostics

Module designations used in diagnostics are obtained using the SHOW CONFIGURATION command.

Example 9–1 DEC 7000-600 System SHOW CONFIGURATION Command

```
cs>show config

Name      Type     Rev    Mnemonic
LSB
0+     KN7AA     (8001)  0000  kn7aa0   ❶
6+     MS7AA     (4000)  0000  ms7aa0   ❷
7+     MS7AA     (4000)  0000  ms7aa1   ❸
8+     IOP       (2000)  0002  iop0     ❹

C0 XMI                           xmi0     ❺
       1+     DEMNA     (0C03)  060B  demna0   ❻
       3+     KZMSA     (0C36)  0003  kzmsa0   ❼
       8+     DWLMA     (102A)  0003  dwlma0   ❽
       E+     KZMSA     (0C36)  0003  kzmsa1   ❾

cs>show dev
dka0.0.0.3.0 DKA0  RZ73
dka1.1.0.3.0 DKA100  RZ73
dka2.2.0.3.0 DKA200          RZ73
dka3.3.0.3.0 DKA300  RZ73
dka4.4.0.3.0 DKA400  RZ73
dka5.5.0.3.0 DKA500  RZ73
dkc1.1.0.14.0 DKC100  RRD42
mkd2.2.1.14.0 MKD200  TLZ04
cs>
```

❶ LEP processor module is at LSB Node 0.

❷ LASER memory module, LMEM, is at LSB Node 6.

❸ LASER memory module, LMEM, is at LSB Node 7.

❶ I/O port module, IOP, is at LSB Node 8.

❺ IOP hose 0 (CO) is attached to XMI0.

❻ DEMNA in XMI bus Slot 1.

❼ KZMSA-AB in XMI bus Slot 3.

❽ DWLMA0 (LAMB) module is in XMI bus Slot 8.

❾ KZMSA-AB in XMI bus Slot E.

Example 9–2 Running the Diagnostics

```
cs>test -t 30 kn7aa0
 { processor 0 test runs }
cs>init
 { system init occurs }
cs>test -t 30 ms7aa0
 { memory 0 test runs }
cs>test -t 30 iop
 { iop test runs }
```

Exercisers

Exercisers can be run in the foreground or the background. Exercisers are run in console diagnostic mode as shown in Example 9–3 and Example 9–4.

Memory Exerciser

Example 9–3 Running the Memory Exerciser

```
cs>set mode diag
cs>mem_ex -p 5
 { memory exerciser runs 5 passes in foreground}
cs> mem_ex -p 0 &
 { memory exerciser runs in background}
cs> show status
15-JUN-145 139:35:44
ID  Program          Device      Pass Hard/Soft Bytes Written    Bytes Read
-------- -------- --------------- -------- --------- ------------- -------------
0000001a   mem_ex             mem      1204     0     0        9854976        9854976
cs>kill 1a
cs>
```

Disk Exerciser

A nondestructive read and read/compare is performed in Example 9–4. The disk area modifiers are *sb* (starting block), *eb* (ending block), and *bs* (block size). The action string designated by *-a* is defined in Table 9–2.

Table 9–2 Disk Exerciser Action Strings

Action Qualifier	Definition
c	Compare expected buffer with received buffer.
d	Fill expected buffer with default or specified pattern.
D	Fill received buffer with default or specified pattern.
n	Write without lock from expected buffer.
N	Write without lock from received buffer.
r	Read into expected buffer.
R	Read into received buffer.
w	Write from expected buffer.
W	Write from received buffer.
-	Seek to file offset prior to last read or write.
?	Seek to random block offset in the specified range of blocks.

Example 9–4 Running the Disk Exerciser

```
cs>set mode diag

cs>show dev
dka0.0.0.3.0 DKA0  RZ73
dka1.1.0.3.0 DKA100  RZ73
dka2.2.0.3.0 DKA200        RZ73
dka3.3.0.3.0 DKA300  RZ73
dka4.4.0.3.0 DKA400  RZ73
dka5.5.0.3.0 DKA500  RZ73
dkc1.1.0.14.0 DKC100  RRD42
mkd2.2.1.14.0 MKD200  TLZ04
cs>
cs>dsk_ex -a "?R-rc" -sb 0 -eb 20 -bs 2000 -p 0 dka1 &
cs>dsk_ex -a "?R-rc" -sb 0 -eb 20 -bs 2000 -p 0 dka2 &

cs>show status

ID       Program          Device     Pass Hard/Soft Bytes Written    Bytes Read
-------- -------- ---------------- -------- --------- ------------- -------------
00000014  dsk_ex   dka1.1.0.3.0     1429    0    0     0     112467968
00000017  dsk_ex   dka2.2.0.3.0     1429    0    0     0     112467968

cs>kill 14
cs>kill 17
cs>show status
ID       Program          Device     Pass Hard/Soft Bytes Written    Bytes Read
-------- -------- ---------------- -------- --------- ------------- -------------
cs>
```

Table 9–3 Device Types for DEC 7000-600 System XMI Modules

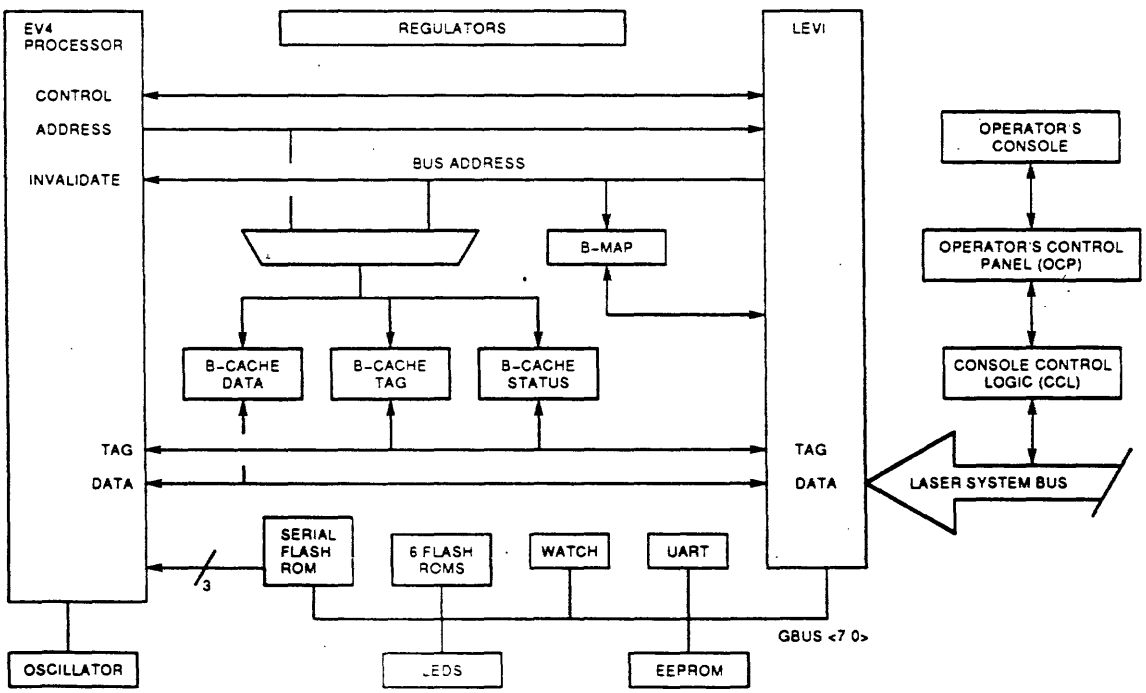
Module	Option	Code	Description
T2020	DEMNA	0C03	Ethernet/802 controller
T2080	CIXCD-AC	0C05	CI interface adapter
T2022, T2023	KDM70	0C22	DSA disk/tape controller (new ucode)
T2029-AB	KZMSA-AB	0C36	SCSI bus adapter
T2029-AC	KFMSB-AA	0C31	DSSI bus adapter
T2027	DEMFA	0823	XMI-to-FDDI adapter
T2028	DWLMA (LAMB)	102A	LASER-to-XMI board

LEP Processor Modules (KN7AA-AA, E2040-AA; KN7AA-YA, E2040-YA)

DEC 7000-600 uses LEP module E2040-AA, option KN7AA-AA (5.5 ns, 182 MHz).

DEC 10000-600 uses LEP module E2040-YA, option KN7AA-YA (5.0 ns, 200 MHz).

Figure 9–3 LEP Processor Module

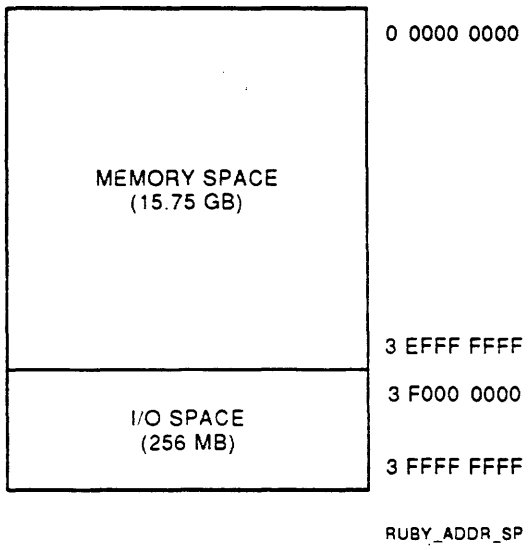


- CMOS-4 macropipelined chip
  - Implements Alpha AXP instructions (about 150)
  - 8 kB data cache
  - 8 kB virtual instruction cache (VIC)
  - Translation buffer
    - 32 entries for data
    - 8 entries for instructions
  - Supports DEC F and G floating point data types
  - Supports IEEE single and double floating data types
  - 6 ns cycle time
- 4 MB backup cache
- LASER/EVAX interface (LEVI)
- Privileged architecture library (PAL) code
- Console support hardware
- DC-to-DC power regulators (48 Vdc to 5.0 Vdc, 3.3 Vdc, 2.0 Vdc, and so on)

DEC 7000-600 System Physical Address Space

The DEC 7000-600 system supports 34-bit physical addresses. The 34-bit physical address space is shown in Figure 9–4.

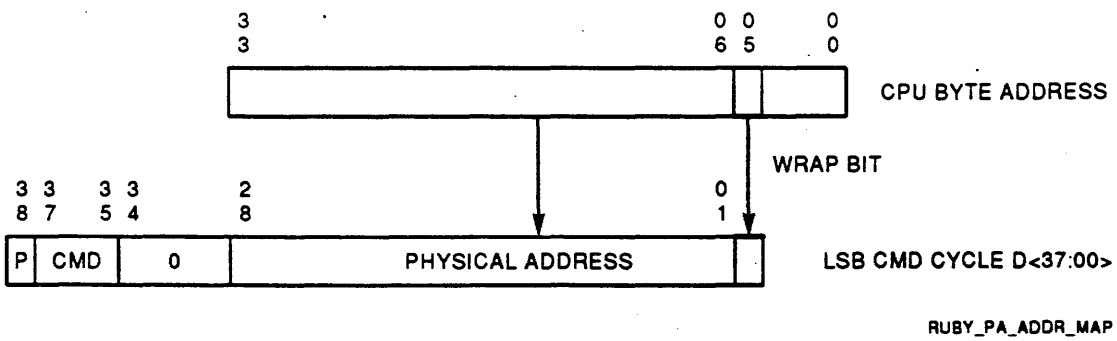
Figure 9–4 DEC 7000-600 System 34-Bit Address Space



Mapping DEC 7000-600 System Physical Addresses to LSB

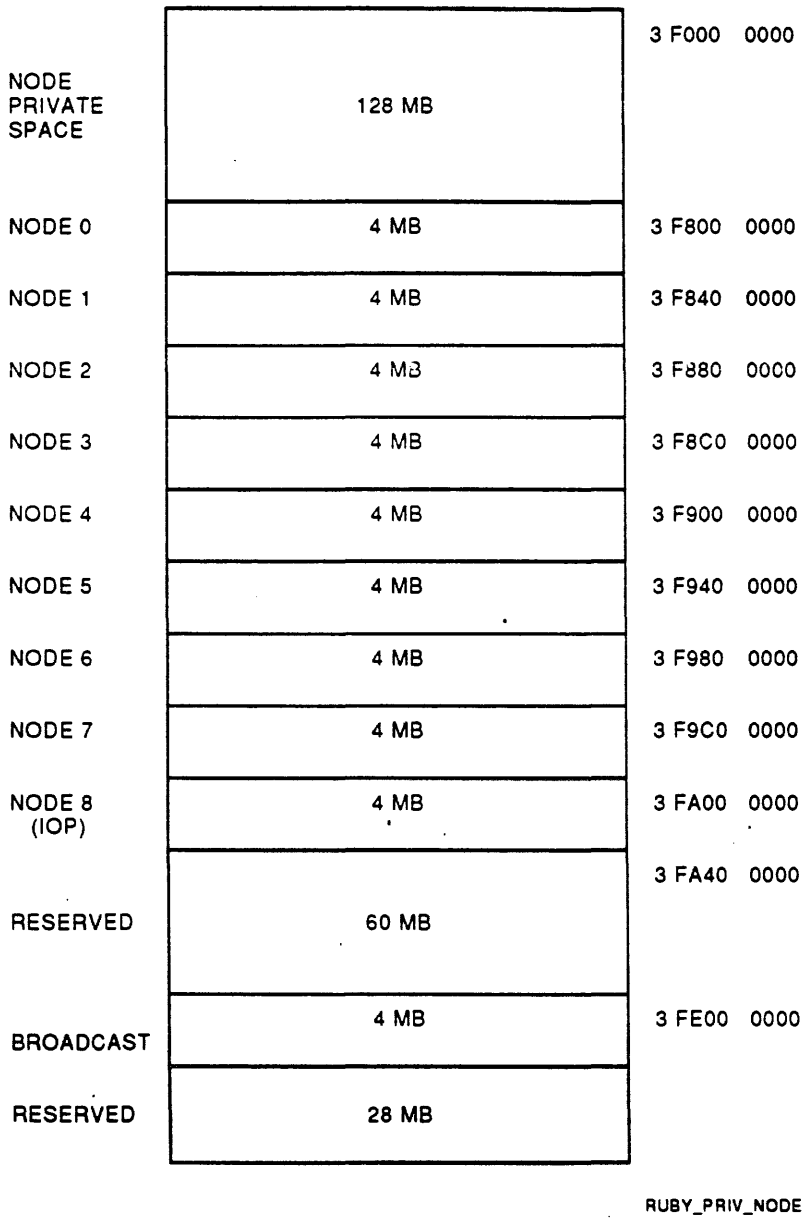
DEC 7000-600 system physical addresses are mapped onto the LSB as shown in Figure 9–5.

Figure 9–5 DEC 7000-600 System Memory Address Mapping to LSB



The address ranges allocated for node private space and node LSB CSRs are shown in Figure 9–6.

Figure 9–6 Node Private Space and CSR Space



LEP node private space is shown in Table 9–4.

Table 9–4 LEP Private Space

Address Range	Register Name
3 F000 0000 - 3 F37F FFFF	Seven Console Flash ROMs
3 F380 0000 - 3 F3FF FFFF	Console EEPROM
3 F400 0000 - 3 F57F FFFF	UART registers
3 F600 0000 - 3 F600 0FC0	WATCH registers
3 F700 0000	GBus$FWHAMI
3 F700 0040	GBus$LEDs
3 F700 0080	GBus$PMask
3 F700 00C0	GBus$Intr
3 F700 0100	GBus$Halt
3 F700 0140	GBus$LSBRST
3 F700 0180	GBus$Misc
3 F780 0000	GBus$RMode_ENA

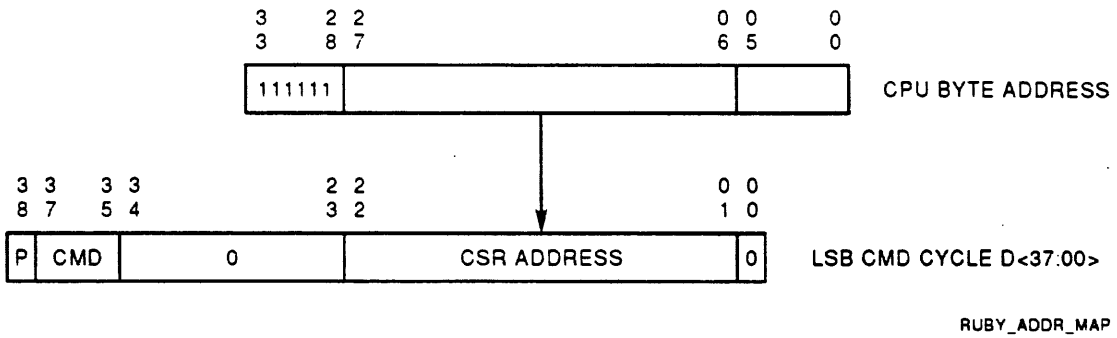
DEC 7000-600 System Register Addresses on the LSB

DEC 7000-600 system LSB nodes have control and status registers that can be accessed using the system console. The addresses of the registers are calculated using methods described in this section.

DEC 7000-600 System LSB Register Addresses

DEC 7000-600 system physical addresses that have bits <33:28> set, access LSB register space (example: 3 F880 0004). When a CSR read or write transaction occurs, address bits <33:06> are mapped to LSB D<36:00> as shown in Figure 9–7.

Figure 9–7 DEC 7000-600 System Register Address Mapping



The base addresses for each node and the broadcast space address is listed in Table 9–5.

Table 9–5 DEC 7000-600 System LSB Node Base Address (BB & BSB)

Node	Module/Node	DEC 7000-600 Base Physical Address	LSB Base C/A Bits 22:00
0	CPU 0	3 F800 0000	40 0000
1	CPU/MEM 1	3 F840 0000	42 0000
2	CPU/MEM 2	3 F880 0000	44 0000
3	CPU/MEM 3	3 F8C0 0000	46 0000
4	CPU/MEM 4	3 F900 0000	48 0000
5	CPU/MEM 5	3 F940 0000	4A 0000
6	CPU/MEM 6	3 F980 0000	4C 0000
7	CPU/MEM 7	3 F9C0 0000	4E 0000
8	IOP 8	3 FA00 0000	50 0000
Broadcast	BSB	3 FE00 0000	70 0000

DEC 7000-600 System Mailbox

All transfers through the IOP to or from remote buses (such as XMI bus) are accomplished using mailboxes. Only processors at nodes 0-3 can handle mailbox transfers.

The sequence of steps for a DEC 7000-600 system mailbox transfer follows:

1. The operating system creates a mailbox structure in memory.
2. The operating system uses a loop with a fail branch as shown in Example 9–5 to write to LMBRPn.

Example 9–5 LEP Write to LMBPRn

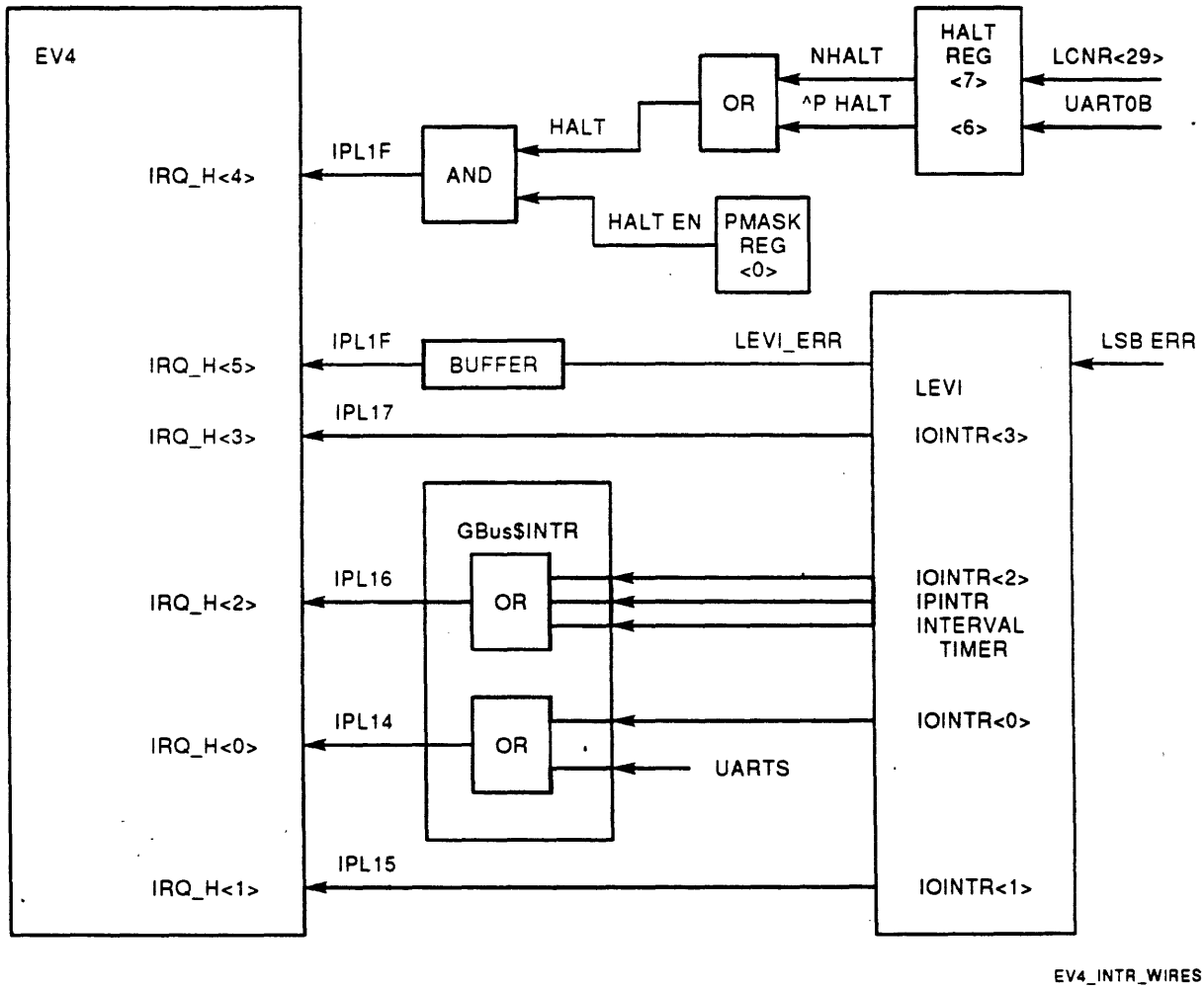
```
post_mbx:
        physical address of mbx --> R1
        virtual address of LMBPRn --> R0
        STQ_C R1,R0                         #if fail 0 --> R1
        BEQ R1, wait_post_mbx

        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

wait_post_mbx:  delay
        BR post_mbx

        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

3. After writing to LMBPRn, the operating system starts polling the DONE bit in the mailbox structure.
4. Meanwhile the IOP services the mailbox request by reading the mailbox and sending a packet to the LAMB.
5. Later the LAMB returns a mailbox status packet to the IOP. The IOP writes the status packet into the mailbox structure.
6. The operating system (polling) notices the set DONE bit in the mailbox structure and takes action depending on the status part of the mailbox.

DEC 7000-600 System Interrupts

The DEC 7000-600 system responds to interrupts through the system control block (SCB).

Figure 9–8 LEP Module Interrupt Connections



NOTE

The EV4 chip uses the Watch chip as an interval timer.

Systems Differences Exercises

1. Which one of the following statements is true?

   a. The KZMSA is used in the DEC 7000-600 system as the interface to the SCSI-2 bus.

   b. The KFMSA is never used on the DEC 7000-600 system.

   c. The KFMSA and the KZMSA are similar, but only the KZMSA is used on the DEC 7000-600 system.

   d. The T2029-AC variation of the KZMSA module is expected to be used on the DEC 7000-600 in the future to interface to the DSSI (KFMSB-AA).

   e. All of the above.

2. Which of the following statements is true?

   a. The VAX 7000-600 system supports a 34-bit physical address.

   b. The DEC 7000-600 system supports a 34-bit physical address.

   c. The VAX 7000-600 system only supports a 32-bit address.

   d. The VAX 7000-600 system only supports a 30-bit address.

   e. None of the above.

3. The DEC 7000-600 system uses a base physical address of 3FA000000 to communicate with the registers in the IOP module. How is this turned into a CSR transaction on the LSB?

   a. The EV4 chip notices that the 6 MSBs are 1s and turns the processor read/write to a CSR read/write.

   b. The CBox notices that the 6 MSBs are 1s and turns the processor read/write to a CSR read/write.

   c. The LEVI chips notice that the 6 MSBs are 1s and turns the processor read/write to a CSR read/write.

   d. The EV4 chip notices that the 6 MSBs are 1s and takes a trap to PALcode to perform the CSR read/write.

   e. None of the above.

4. The size of the console memory cluster for a DEC 7000-600 system is about 2 MB.

   a. True

   b. False

5. Which one of the following statements is true?

   a. The three DEC 7000-600 system UARTs interrupt at IPL 14.

   b. The three DEC 7000-600 system UARTs interrupt at IPL 15.

   c. The three DEC 7000-600 system UARTs interrupt at IPL 16.

   d. The three DEC 7000-600 system UARTs interrupt at IPL 17.

   e. None of the above.

6. The VAX 7000-600 system and the DEC 7000-600 system both access memory in 64-byte naturally aligned blocks.

   a. True

   b. False

7. Which one of the following statements is true?

   a. The OpenVMS Alpha AXP operating system can be booted from a CD–ROM on the DEC 7000-600 system.

   b. The OpenVMS VAX operating system can be booted from a CD–ROM on the DEC 7000-600 system.

   c. The OpenVMS VAX operating system can be booted from a CD–ROM on the DEC 10000-600 system.

   d. The OpenVMS VAX operating system can be booted from a CD–ROM on the VAX 7000-600 system.

   e. None of the above

DEC 7000-600 and DEC 10000-600 Systems Fault Management

Introduction

The sequence of actions performed by the DEC 7000-600 system during a machine check operation are described in this training module. The DSE is shown how to obtain and interpret an errorlog printout.

Objectives

A Digital Services Engineer who maintains a DEC 7000-600 or DEC 10000-600 system should be able to:

- Recognize the sequence of actions performed by a DEC 7000-600 system or DEC 10000-600 system during a machine check operation.
- Obtain and interpret an errorlog printout.
- Recognize DEC 7000-600 system error handling paths using the SCB.

Resources

DEC 7000 AXP System Pocket Service Guide	EK-7700A-PG

DEC 7000/10000-600 Registers

- KN7AA internal processor registers
- KN7AA specific registers
- KN7AA LSB registers

Error Events

The DEC 7000/10000-600 systems faults cause the following events:

- Console error halts
- Machine check (SCBB+670)
- Hard error (SCBB+660)
- Soft error (SCBB+630)
- IOP adapter error IPL17 (SCBB+[800-1FF0])
- DWLMA (LAMB) adapter error IPL17 (SCBB+[800-1FF0])

Console Error Halts

A console halt can occur for any of several reasons: power on, assertion of the external halt interrupt pin (HALT_H), or by certain microcode-detected double error conditions.

The reason for a console double-error halt is indicated by a display on the console terminal. The display is presented in a format similar to the console crash shown in Example 10–1.

Example 10–1 Console Mode Error Halt

```
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #
                            A   M   M   P   P   P   P   P   P   TYP
                            o   +   +   +   +   +   +   +   +   ST1
                            .   .   .   E   E   E   E   E   B   BPD
                            o   +   +   +   +   +   +   +   +   ST2
                            .   .   .   E   E   E   E   E   B   BPD
CPU 00: ic_diag timeout
                            +   +   +   +   +   +   +   +   +   ST3
                            .   .   .   E   E   E   E   E   B   BPD

    +   .   +   .   .   .   +   .   .   .   .   .   +   .       C0 XMI +
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   C1
    .   .   .   .   .   .   +   .   .   .   .   .   .   +       C2 XMI +
    +   .   .   .   .   .   +   .   .   .   .   +   +   .       C3 XMI +

                            .  A1  A0   .   .   .   .   .   .   ILV
                            . 256 256   .   .   .   .   .   .   512MB
CPU 1
unexpected exception/interrupt through vector 00000063
process lamb_tst, pcb = 001B6DA0

pc: 00000000 0003CB98  ps: 20000000 00000004
r2: 00000000 00142388  r5: 00000000 00001404
r3: 00000000 001ECC40  r6: 00000000 00000001
r4: 00000000 00000228  r7: 00000000 00000000

gprs:
0: 00000000 00000001  16: 00000000 00000000
1: 00000000 001C0D68  17: 00000000 001C0D68
2: 00000000 001164D0  18: 80000000 80000000
3: 00000000 001C0D40  19: FFFFFFFF FA000000
4: 00000000 00000000  20: FFFFFFFF FA000042
5: 00000000 00000000  21: 00000000 00000000
6: 00000000 00000001  22: 00000000 00000007
7: 00000000 00000000  23: 00000000 001C0D74
8: 00000000 001C0EC0  24: 00000000 001C0D40
9: 00000000 00000006  25: 00000000 00000000
10: 00000000 0002E3EA  26: 00000000 0003CC4C
11: FFFFFFFF 80000502  27: 00000000 00118890
12: 00000000 00000001  28: 00000000 0003CA98
13: 00000000 001C07E4  29: 00000000 001B7CA0
14: 00000000 00000000  30: 00000000 001B7CA0
15: 00000000 00000000
```

Example 10–1 Cont'd on next page

Example 10–1 (Continued) Console Mode Error Halt

```
fprs:
0: 40900000 00000000  16: 00000000 40000000
1: 00000000 40000000  17: FFFFFFFF FFFFFFFF
2: 00000000 00000000  18: FFFFFFFF FFFFFFFF
3: 00000000 00000000  19: FFFFFFFF FFFFFFFF
4: 00000000 00000000  20: FFFFFFFF FFFFFFFF
5: 00000000 00000000  21: FFFFFFFF FFFFFFFF
6: 00000000 00000000  22: FFFFFFFF FFFFFFFF
7: 00000000 00000000  23: FFFFFFFF FFFFFFFF
8: 00000000 00000000  24: FFFFFFFF FFFFFFFF
9: 00000000 00000000  25: FFFFFFFF FFFFFFFF
10: 40300000 00000000  26: FFFFFFFF FFFFFFFF
11: 00000010 00000000  27: FFFFFFFF FFFFFFFF
12: 3FA00000 00000001  28: FFFFFFFF FFFFFFFF
13: 00000000 40000000  29: FFFFFFFF FFFFFFFF
14: 40200000 00000000  30: FFFFFFFF FFFFFFFF
15: 00000000 40000000

Machine Check Logout - base: 00006300

flags:      00000000 00000000        byte_count:    80000000 000001D8
offsets:    000001A0 00000110        das_debug:     00E00555 000000DA
pt0:        00000001 00000100        pt1:           00000000 0000267C
pt2:        000004F8 00000004        pt3:           00000000 00000000
pt4:        00000000 00000000        pt5:           00000000 00000000
pt6:        00000000 001362E8        pt7:           00000000 00000000
pt8:        00000000 00000000        pt9:           00000000 00001404
pt10:       00000000 CCFE805B        pt11:          00000000 00000000
pt12:       00000000 00000001        pt13:          00000000 001C07E4
pt14:       00000000 00000000        pt15:          00000000 00000000
pt16:       00000000 00000000        pt17:          00000000 001C0D68
pt18:       80000000 80000000        pt19:          08020000 00404C80
pt20:       00000000 00000000        pt21:          00000000 001A719F
pt22:       00000000 001A7168        pt23:          00000020 00000440
pt24:       00000000 001B6DA0        pt25:          00000000 00050000
pt26:       02800000 00080004        pt27:          00000000 00000001
pt28:       00000000 00000001        pt29:          00000000 00000001
pt30:       00000000 001F0000        pt31:          00000000 00001000
exc_addr:   00000000 00142388        exc_sum:       00000000 00000000
iccsr:      00000000 009F0000        pal_base:      00000000 00008000
hier:       00000000 000018E0        hirr:          00000000 00000042
mm_csr:     00000000 000053A0        dc_stat:       00000000 00000007
dc_addr:    00000007 FFFFFFFF        abox_ctl:      00000000 0000050E
biu_stat:   00000000 00000250        biu_addr:      00000000 00006440
biu_ctl:    00000008 50007347

>>>
```

The DSE can then use the console CRASH command to try to obtain an operating system crash dump. The CRASH command causes the console to boot and hand the operating system a dump argument. The operating system then attempts to perform a crash dump.

The DSE can use the console MCHK script to get information about the state of the processor as shown in Example 10–2.

Example 10–2 CPU Status Console Mode Example

```
>>> mchk 0
pal                    V5.25-5/O1.14-2
pal_flags              04b4010860000005
PTBR ipr: 0000000A (  PTBR) 0000000000000000
SCBB ipr: 0000000B (  SCBB) 0000000000000000
PCBB ipr: 00000008 (  PCBB) 0000000000001000
exc_addr  pmem: 00006130 000000000003CFA0
iccsr   pmem: 00006148 000000000 9F0000
hirr   pmem: 00006160 0000000000000342
mm_csr   pmem: 00006168 0000000000005C50
dc_stat  pmem: 00006170 00000000000002E0
dc_addr  pmem: 00006178 00000007FFFFFFFF
biu_stat  pmem: 00006188 0000000000000250
biu_addr  pmem: 00006190 0000000000006140
biu_ctl  pmem: 00006198 0000000850007347
fill_syndrome  pmem: 000061A0 0000000000000000
fill_addr  pmem: 000061A8 0000000000006140
va   pmem: 000061B0 0000000000006190
lep_gbus  pmem: 000061C0 0020000000000B9
lber   pmem: 000061CC 00000021
lmerr  pmem: 000061D4 00000000
lbesr0 pmem: 000061D8 0000000C
lbesr1 pmem: 000061DC 0000000C
lbesr2 pmem: 000061E0 0000000C
lbesr3 pmem: 000061E4 0000000C
lbecr0 pmem: 000061E8 0000F366
lbecr1 pmem: 000061EC 00004000
vhit   pmem: F8000F80 00000000
tag    pmem: 00006008 00E005550000000E
dwlma0 XBE  xmi0: 60000004 0000000100000202
dwlma0 LERR xmi0: 6000004C 0000000100000000
>>>
```

EV4 and LEVI use of B Cache

The EV4 and LEVI share use of the B cache, although the LEVI is responsible for B cache coherency. The EV4 probes B cache to see if a direct read or write can be accomplished. If the status is OK, then the EV4 reads or writes to the B cache without assistance.

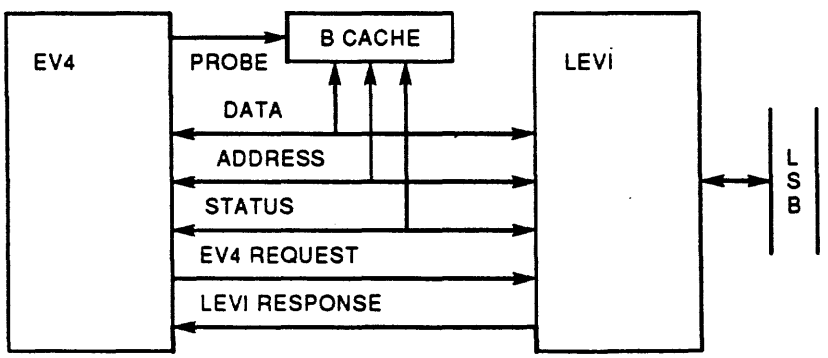
If the status is not OK, then the EV4 requests on cReq_h<2:0> and the LEVI releases the EV4 by responding on cAck_h<2:0>.

- 001 Barrier (LEVI simply acknowledges OK)
- 010,011 Fetch, FetchM (EV4 never prefetches)
- 100 READ_BLOCK (B cache miss)
- 101 WRITE_BLOCK (B cache miss)
- 110 LDxL (LOAD LOCK, start LOCK)
- 111 STxC (STORE CONDITION, clear LOCK)

The EV4 makes a request, then it is stalled waiting for the LEVI to release it. The LEVI releases it with the following responses on cAck_h<2:0>:

- 001 Hard error
- 010 Soft error
- 011 Store conditional failure
- 100 OK

Figure 10–1 EV4 and LEVI Cache Arrangement



Errorlog Entries

Table 10–1 DEC 7000-600 System Error SCB Entries

670	Machine check	Abort	IPL 1F, parameters on interrupt stack
660	Hard error	Interrupt	IPL is 1F (hex)
630	Soft error	Interrupt	IPL is 14 (hex)
800-1FF0	Device error	Interrupt	Target level

- 670 errorlog entry
  - — Processor Error (hard error detected by EV4)
    - * B cache probe
    - * B cache access
    - * Request to LEVI (LSB transaction)
  - — Possible subpackets
    - * DLIST
    - * LSB
    - * LMA
    - * I/O adapter (currently only IOP)
- 660 errorlog entry
  - — System error (hard error not detected by EV4)
  - — Possible subpackets
    - * DLIST
    - * LSB
    - * LMA
    - * I/O adapter (currently only IOP)

- 630 errorlog entry
  - — Soft error (SBE correctable using ECC)
    - * SBEs, tag or data error in P cache, VIC
    - * Some 630 types are uncorrectable and result in a 660 or a 670 entry
  - — Possible subpackets
    - * DLIST
    - * LSB
    - * LMA
- ADPERR (IOP, IPL17)
  - — Hard error on adapter (IOP interrupt)
  - — Possible subpackets
    - * Channel (LAMB)
- ADPERR (LAMB, IPL17)
  - — Hard error on adapter (LAMB interrupt)
  - — Possible subpackets
    - * XMI
- CRD soft errorlog
- MEMSCAN errorlog
  - — Controller error detected while polling LMEM for CRDs.
- CONFIG errorlog
- POWER errorlog

Error Packets and Subpackets Formats

Error packets and subpackets formats are described in detail in the *RUBY Fault Management Specification*.

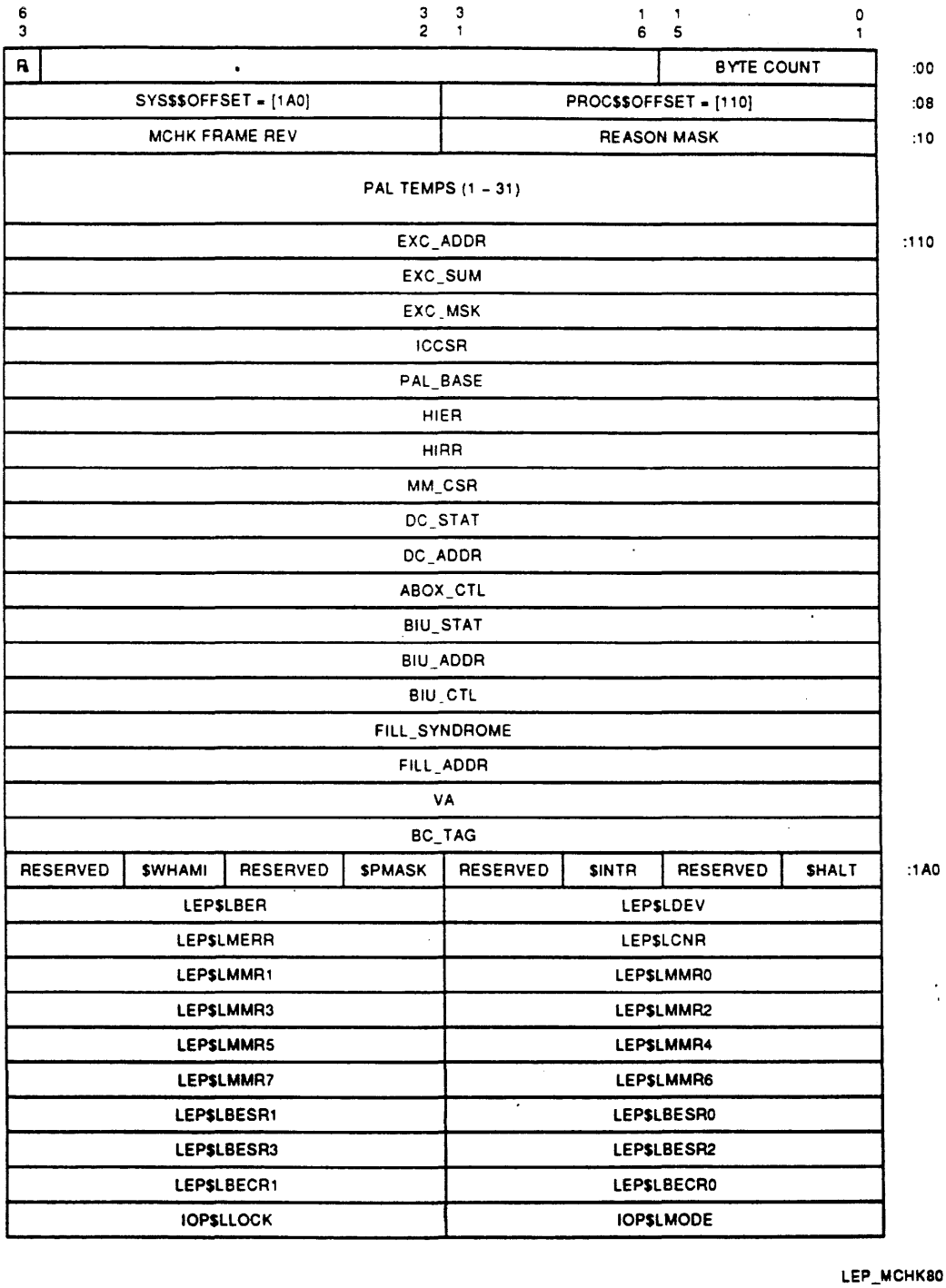
Processor Error 670

- Errorlog header
- Software error flags
  - — 670 error flags (95:00)
  - — Common error flags (127:96)
- Common LEP header area
- LEP machine check stack frame
- PALcode revision
- 670 error counters

Processor Error 660

- Errorlog header
- Software error flags
  - — 660 error flags (95:00)
  - — Common error flags (127:96)
- Common LEP header area
- LEP machine check stack frame
- PALcode revision
- 670 error counters

Figure 10–2 660/670 Error Stack Frame

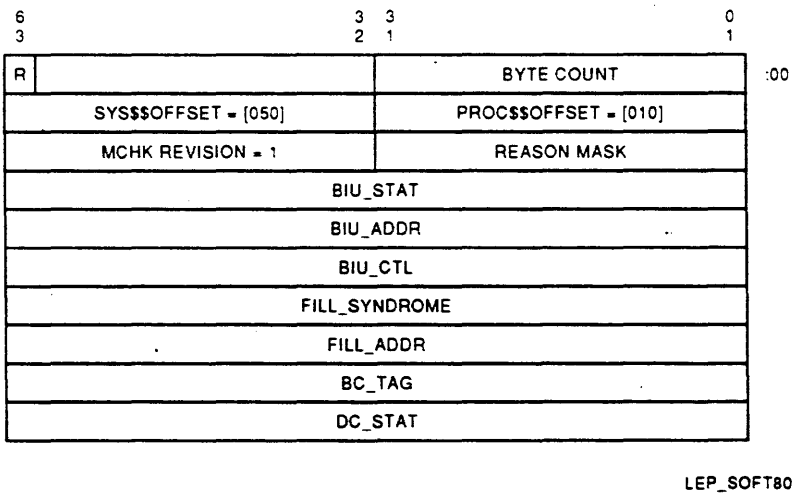


LEP_MCHK80

Soft Error (630)

- Errorlog header
- Software error flags
  - — 630 error flags (95:00)
  - — Common error flags (127:96)
- Common LEP header area
- LEP machine check stack frame
- PAL code revision
- WHAMI
- 630 error counters (96 bytes)

Figure 10–3 630 Error Stack Frame



ADPERR (IOP, IPL17)

ADPERR (LAMB, IPL17)

CRD Soft Errorlog

MEMSCAN Errorlog

CONFIG Errorlog

POWER Errorlog

Hex Dump of Errorlog Entry

A hex dump of an errorlog entry can be obtained using the format shown in the following command sequence. The hex dump is only used if the DSE suspects that ERF did not get all the data available.

Example 10–3 Hex Dump of BSTAT Parity Errorlog Entry

```
$SET DEF [SYSERR
$ANA/ERR/BIN=1380.BIN/ENTRY=(S:1380,E:1380) ERRORLOG.SYS
$DUMP/RECORD/OUT=1380.DMP 1380.BIN

Dump of file SYS$SYSROOT:[SYSERR]ERRLOG.SYS;1 on 10-SEP-1992 15:35:12.31
File ID (352,2,0)   End of file block 326 / Allocated 327

Record number 1380 (00000564), 968 (03C8) bytes

00000000 00000020 20203639 42555208 00000000 00020000 0003FFFB 00000000 ....0.............RUBY6   ....... 000000
00000104 00000000 34544462 302E3154 00042096 068159BD DD060002 40010000 ...@...Y½Y......T1.0-FT4........ 000020
00000000 00000000 00000001 80000000 00000000 00000010 00000000 00000001 ................................ 000040
00000000 00000000 00000000 00000000 00000000 00000001 00000000 00000000 ................................ 000060
00000000 00000108 7AFBFCFD 00000000 00000000 00000000 00000000 00000000 ....................yûüzø....... 000080
001B40F8 00000004 00000000 00000000 00000000 00000084 000001A0 00000110 ............................ø@.. 0000A0
FFFFFFFF FFE11510 FFFFFFFF 80520000 FFF17AFF 80503E40 00010000 00000000 ........@>\.......?.....0.a..... 0000C0
00000000 A4CD7E56 00000000 00000800 00000000 00000000 00000000 00000000 ........................V~Í¤.... 0000E0
FFFFFFFF 8454CC58 00000000 7FEBE004 FFFFFFFF 805FC840 00000000 00000000 ........@È_.....àë.....XÌT..... 000100
00000000 00000000 00000000 00000000 00000000 7FEBE8B4 FFFFFFFF 80516520 eQ.....´ëè..................... 000120
00000000 FFE6FFFE 00000007 FFE4FFFE 00000000 00000038 FFFFFFFF 80403200 .2@.....8.......p.æ.....p.æ..... 000140
00000000 7FF96000 00000000 00010001 FFFFFFFF 805CC000 00000000 00000008 ..........\.............`ù..... 000160
00000000 0098C000 00000002 00000000 00000000 00D6A000 00000000 00000000 .........Ö.............À..... 000180
00000000 00000000 00000000 00000000 FFFFFFFF 8008BFA2 00000000 00D6C080 .ÀÖ.....¢_..................... 0001A0
00000000 00000342 00000001 FFC01CE0 00000000 00008000 00000003 681F0000 ...h.............à.À.....B..... 0001C0
00000000 0000040E 00000007 FFFFFFFF 00000000 000002E0 00000000 00005210 .R......à..................... 0001E0
00000000 00000000 00000008 50006447 00000000 00800030 00000000 00000008 ..........0.......Gd.P..... 000200
00200009 00000381 00000000 00400816 00000000 00006190 00000000 00006120  a.......a.....@.....á..... . 000220
00000000 0000000C 0000000C 0000000C 00000020 00000000 00040001 00008001 ..... ..... ..... 000240
00000000 00000000 00080001 00010500 00018D6A 00010010 00008040 00040001 ....@.....j..... 000260
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 ..... 000280
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 ..... 0002A0
00000183 000000EC 00000000 00000000 00000000 00000000 00000000 00000000 .....ì..... 0002C0
00040001 00008001 00000000 000003FF 00000003 F8000000 00000003 F8000000 ...ø.....ø..... 0002E0
00008040 00040001 00000000 00000000 00000000 00000000 00000020 00000000 .... .....@.... 000300
00000000 00000001 00000001 00004000 00000000 000003F7 00000003 F8400000 ..@ø....÷.....@..... 000320
00000003 F9C00000 00000400 0000E098 00000000 00000000 00000000 00000000 .....à.....Aø.... 000340
00000000 00000000 00000000 00000001 00000001 00004000 00000000 000003F7 ÷.....@..... 000360
00000000 000003F7 00000003 FA000000 00000400 0000E098 00000000 00000000 .....à.....ú....÷..... 000380
0000000C 0000000C 0000000C 0000000C 00000000 00000001 00000001 80032000 . ..... 0003A0
                                                    00008060 00500036 6.P.`..... 0003C0
```

Error Parse Trees and Descriptions

DEC 7000/10000-600 Systems Fault Management Exercise

1. The EV4 microprocessor has 64-bit data paths, therefore all LEP module error registers have 64 bits.

   a. True

   b. False

2. Which one of the following statements is true for a DEC 7000 system?

   a. An LSB error not detected by the EV4 chip but detected by the LEVI-A/Bs is reported as a processor error using vector 670.

   b. A hard error detected by the EV4 chip is reported as a processor error using vector 04.

   c. A soft error detected by the EV4 chip is processed using vector 04.

   d. An LSB hard error detected by the LEVI-A/Bs is processed using vector 660.

   e. All of the above.

3. Bits in the common flags of the SW flags field will indicate which subpackets, if any, will be appended to an errorlog entry.

   a. True

   b. False

4. A B cache single-bit error (SBE) has propagated onto the LSB. The resulting errorlog entry is made by CPU 0, but CPU 0 did not cause the error. Which one of the following statements is true?

   a. The offending CPU node is indicated by its LBER alone.

   b. The offending CPU node is indicated by its LBER NSES bit and LMERR.

   c. The offending CPU node is indicated by its BIU_STAT.

   d. The offending CPU node is indicated by its WHAMI.

   e. None of the above.

Test

Questions

1. _____ Which of the following statements best describes the differences between a 7000 system and a 10000 system?

   a. The 10000 system has a UPS while the 7000 system cannot support a UPS.

   b. The 10000 system is always larger than a 7000 system.

   c. The 10000 system UPS can sustain the system for a longer duration than the 7000 system UPS.

   d. There is no difference between the systems.

   e. All of the above.

2. _____Which one of the following statements is most true?

   a. A VAX 10000-600 system NVAX+ microprocessor has a cycle time of 11 ns.

   b. A VAX 7000-600 system NVAX+ microprocessor has a cycle time of 11 ns.

   c. A DEC 10000-600 system EV4 microprocessor has a cycle time of 5.0 ns.

   d. A VAX 7000-600 system EV4 microprocessor has a cycle time of 5.5 ns.

   e. All of the above.

3. _____ Five LSB cycles are allocated for each LSB transaction.

   a. True

   b. False

4. _____ Which command would you use to find which device was last used to boot the system?

   a. >>>EXAMINE BOOT_DEV

   b. >>>SHOW EEPROM FIELD

   c. >>>SHOW BOOT_DEF

   d. >>>SHOW BOOTED_DEV

   e. None of the above

5. _____ Which one of the following statements is most true?

   a. The KZMSA is used in the DEC 7000-600 system as the interface to the SCSI-2 bus.

   b. The KFMSA is never used on the DEC 7000-600 system.

   c. The KFMSA and the KZMSA are similar but only the KZMSA is used on the DEC 7000-600 system.

   d. The T2029-AC variation of the KZMSA module is expected to be used on the DEC 7000-600 in the future to interface to the DSSI (KFMSB-AA)

   e. All of the above.

6. _____ Which one of the following statements is true for a DEC 7000 system?

   a. A hard error not detected by the EV4 chip but detected by the LEVI-A/Bs will be reported as a Processor Error using vector 670.

   b. A hard error detected by the EV4 chip will be reported as a Processor Error using vector 04.

   c. A soft error detected by the EV4 chip will be processed using vector 04.

   d. A hard error detected by the LEVI-A/Bs will be processed using vector 660.

   e. All of the above.

7. _____ At what priority does the IOP module arbitrate?

   a. 9 which is the highest

   b. 0 which is the highest

   c. 0, 5

   d. 0, 5, 9

   e. None of the above

8. _____ Which node register and bit is the primary indicator of an LSB error?

   a. LDEV register, bit 31

   b. XBER register, bit 00

   c. LBER register, bit 00

   d. BIU_STAT register, bit 00

   e. None of the above

9. _____ The LMEM module receives data with a double-bit error (DBE) from the LSB. The LMEM is unable to correct the DBE but will write the bad data into its DRAMs.

   a. True

   b. False

10. _____ Which of the following statements is most true?

   a. The LEVI treats the VICTIM buffer as an extension of P cache.

   b. The LEVI treats the VICTIM buffer as an extension of B cache.

   c. When a double-bit error occurs the VICTIM buffer is used to hold the DBE data.

   d. The VICTIM buffer is used to hold data received from the LSB in a transaction when the DIRTY line was asserted.

   e. None of the above.

11. _____ The LMEM module receives data with a single-bit error (SBE) from the LSB. The LMEM will correct the SBE and write the corrected data into its DRAMs.

   a. True

   b. False

12. _____ Which one of the following statements is most true?

   a. An LMEM module always has the best (latest) copy of data.

   b. The LEVI always knows where the best (latest) copy of data is located.

   c. The IOP module asserts the DIRTY line during a transaction if it notices that the transaction is to one of its active mailbox addresses. It then supplies the requested mailbox data.

   d. The NVAX+ always knows where the best (latest) copy of data is located.

   e. None of the above.

13. _____ When the LAMB module suffers a fatal error, it will stop communicating with the IOP module. How can a DSE tell if this has happened?

    a. If the error summary LED on the LAMB (fifth down of five) is lit (red)

    b. If there is no response to the console command >>>EX PMEM:FA000000 -L

    c. If there is no response to the console command >>>EX PMEM:FA000004 -L

    d. If the fatal error LED on the LAMB (fourth down of five) is lit (red)

    e. None of the above

14. _____ The LNP module B map is probed by the NVAX+ processor to check for valid entries in the B cache.

    a. True

    b. False

15. _____ Which one of the following statements about IOP participation in a mailbox operation is true?

    a. The IOP reads mailbox contents in memory and decodes the priority of the operation.

    b. The IOP reads mailbox contents but does not decode any of it. The IOP justs passes the data to the LAMB.

    c. The IOP reads mailbox contents and decodes it to find out which hose to send a packet down.

    d. The IOP does not read the mailbox contents without instructions from the LAMB. It sends the address of the mailbox to the LAMB and waits for instructions.

    e. None of the above.

16. _____ Which one of the following statements is most true for a VAX 7000-600 system?

    a. A hard error not detected by the NVAX+ chip but detected by the LEVI-A/Bs will always be reported as a Machine Check using vector 04.

    b. A hard error detected by the NVAX+ chip will be reported as a Machine Check using vector 04.

    c. A soft error detected by the NVAX+ chip will be processed using vector 660.

    d. A soft error detected by the LEVI-A/Bs will always be processed using vector 60.

    e. All of the above.

17. _____ The LASTFAIL packet in an errorlog entry tells which FRU was the most recent to fail and be replaced.

    a. True

    b. False

18. _____ Which one of the following statements is true?

    a. The OpenVMS Alpha AXP operating system can be booted from a CD–ROM on the DEC 7000-600 system.

    b. The OpenVMS VAX operating system can be booted from a CD–ROM on the DEC 7000-600 system.

    c. The OpenVMS VAX operating system can be booted from a CD–ROM on the DEC 10000-600 system.

    d. The OpenVMS VAX operating system can be booted from a CD–ROM on the VAX 7000-600 system.

    e. None of the above.

19. _____ Which one of the following statements is true?

    a. The DEC 7000-600 supports the system root argument in the -flags string.

    b. The DEC 7000-600 supports the shadow boot argument in the -flags string.

    c. The VAX 7000-600 supports the console baud argument in the -flags string.

    d. The VAX 7000-600 supports the background argument (&) in the -flags string.

    e. None of the above.

20. _____ If a KN7AA module has its LBER<0> set, then there must be a hardware fault on the LSB.

    a. True

    b. False

Answers

1. ____c____ Which of the following statements best describes the differences between a 7000 system and a 10000 system?

   a. The 10000 system has a UPS while the 7000 system cannot support a UPS.

   b. The 10000 system is always larger than a 7000 system.

   c. The 10000 system UPS can sustain the system for a longer duration than the 7000 system UPS.

   d. There is no difference between the systems.

   e. All of the above.

2. ____e____ Which one of the following statements is most true?

   a. A VAX 10000-600 system NVAX+ microprocessor has a cycle time of 11 ns.

   b. A VAX 7000-600 system NVAX+ microprocessor has a cycle time of 11 ns.

   c. A DEC 10000-600 system EV4 microprocessor has a cycle time of 5.0 ns.

   d. A VAX 7000-600 system EV4 microprocessor has a cycle time of 5.5 ns.

   e. All of the above.

3. ____a____ Five LSB cycles are allocated for each LSB transaction.

   a. True

   b. False

4. ____d____ Which command would you use to find which device was last used to boot the system?

   a. >>>EXAMINE BOOT_DEV

   b. >>>SHOW EEPROM FIELD

   c. >>>SHOW BOOT_DEF

   d. >>>SHOW BOOTED_DEV

   e. None of the above

5. ____e____ Which one of the following statements is most true?

   a. The KZMSA is used in the DEC 7000-600 system as the interface to the SCSI-2 bus.

   b. The KFMSA is never used on the DEC 7000-600 system.

   c. The KFMSA and the KZMSA are similar but only the KZMSA is used on the DEC 7000-600 system.

   d. The T2029-AC variation of the KZMSA module is expected to be used on the DEC 7000-600 in the future to interface to the DSSI (KFMSB-AA)

   e. All of the above.

6. ____d____ Which one of the following statements is true for a DEC 7000 system?

   a. A hard error not detected by the EV4 chip but detected by the LEVI-A/Bs will be reported as a Processor Error using vector 670.

   b. A hard error detected by the EV4 chip will be reported as a Processor Error using vector 04.

   c. A soft error detected by the EV4 chip will be processed using vector 04.

   d. A hard error detected by the LEVI-A/Bs will be processed using vector 660.

   e. All of the above.

7. ____c____ At what priority does the IOP module arbitrate?

   a. 9 which is the highest

   b. 0 which is the highest

   c. 0, 5

   d. 0, 5, 9

   e. None of the above

8. ___b___ Which node register and bit is the primary indicator of an LSB error?

   a. LDEV register, bit 31

   b. XBER register, bit 00

   c. LBER register, bit 00

   d. BIU_STAT register, bit 00

   e. None of the above

9. ___a___ The LMEM module receives data with a double-bit error (DBE) from the LSB. The LMEM is unable to correct the DBE but will write the bad data into its DRAMs.

   a. True

   b. False

10. ___b___ Which of the following statements is most true?

   a. The LEVI treats the VICTIM buffer as an extension of P cache.

   b. The LEVI treats the VICTIM buffer as an extension of B cache.

   c. When a double-bit error occurs the VICTIM buffer is used to hold the DBE data.

   d. The VICTIM buffer is used to hold data received from the LSB in a transaction when the DIRTY line was asserted.

   e. None of the above.

11. ___a___ The LMEM module receives data with a single-bit error (SBE) from the LSB. The LMEM will correct the SBE and write the corrected data into its DRAMs.

   a. True

   b. False

12. ___e___ Which one of the following statements is most true?

   a. An LMEM module always has the best (latest) copy of data.

   b. The LEVI always knows where the best (latest) copy of data is located.

   c. The IOP module asserts the DIRTY line during a transaction if it notices that the transaction is to one of its active mailbox addresses. It then supplies the requested mailbox data.

   d. The NVAX+ always knows where the best (latest) copy of data is located.

   e. None of the above.

13. ___d___ When the LAMB module suffers a fatal error it will stop communicating with the IOP module. How can a DSE tell if this has happened?

    a. If the error summary LED on the LAMB (fifth down of five) is lit (red)

    b. If there is no response to the console command >>>EX PMEM:FA000000 -L

    c. If there is no response to the console command >>>EX PMEM:FA000004 -L

    d. If the fatal error LED on the LAMB (fourth down of five) is lit (red)

    e. None of the above

14. ___b___ The LNP module B map is probed by the NVAX+ processor to check for valid entries in the B cache.

    a. True

    b. False

15. ___c___ Which one of the following statements about IOP participation in a mailbox operation is true?

    a. The IOP reads mailbox contents in memory and decodes the priority of the operation.

    b. The IOP reads mailbox contents but does not decode any of it. The IOP justs passes the data to the LAMB.

    c. The IOP reads mailbox contents and decodes it to find out which hose to send a packet down.

    d. The IOP does not read the mailbox contents without instructions from the LAMB. It sends the address of the mailbox to the LAMB and waits for instructions.

    e. None of the above.

16. ___b___ Which one of the following statements is most true for a VAX 7000-600 system?

    a. A hard error not detected by the NVAX+ chip but detected by the LEVI-A/Bs will always be reported as a Machine Check using vector 04.

    b. A hard error detected by the NVAX+ chip will be reported as a Machine Check using vector 04.

    c. A soft error detected by the NVAX+ chip will be processed using vector 660.

    d. A soft error detected by the LEVI-A/Bs will always be processed using vector 60.

    e. All of the above.

17. ____b____ The LASTFAIL packet in an errorlog entry tells which FRU was the most recent to fail and be replaced.

    a. True

    b. False

18. ____e____ Which one of the following statements is true?

    a. The OpenVMS Alpha AXP operating system can be booted from a CD–ROM on the DEC 7000-600 system.

    b. The OpenVMS VAX operating system can be booted from a CD–ROM on the DEC 7000-600 system.

    c. The OpenVMS VAX operating system can be booted from a CD–ROM on the DEC 10000-600 system.

    d. The OpenVMS VAX operating system can be booted from a CD–ROM on the VAX 7000-600 system.

    e. None of the above.

19. ____a____ Which one of the following statements is true?

    a. The DEC 7000-600 supports the system root argument in the -flags string.

    b. The DEC 7000-600 supports the shadow boot argument in the -flags string.

    c. The VAX 7000-600 supports the console baud argument in the -flags string.

    d. The VAX 7000-600 supports the background argument (&) in the -flags string.

    e. None of the above.

20. ____b____ If a KN7AA module has its LBER<0> set, then there must be a hardware fault on the LSB.

    a. True

    b. False

XMI I/O Modules and Error Registers

Introduction

This appendix is included as convenient reference material for the DSE and is not part of the course curriculum.

This appendix contains tables of useful information about XMI module device types, register lists, and address space allocation.

Similar information about the VAXBI bus is also included.

XMI Bus Operation

XMI and VAXBI Registers (Mailbox Operations)

The console can be used to examine XMI and VAXBI CSRs as shown in Example A–1. The console user must calculate the register address used in the command.

Example A–1 Examining XMI and VAXBI Registers

```
>>examine xmi0:61880004
>>e xmi0:61880004
```

The system console carries out the command using a mailbox in main memory. The sequence of steps in the operations follows:

1. The console constructs a mailbox in memory and puts in the pertinent data such as examine or deposit, which XMI, and the address.
2. The console notifies the IOP by setting bits in its mailbox pointer register (LMBPR).
   a. The console starts polling the DONE bit in the mailbox.
3. The IOP notices the set bits, retrieves the data from the mailbox, then performs the directed operation to the correct XMI.
4. If the operation is a write, the IOP sets the DONE bit in the mailbox.
   a. The console notices the set DONE bit in the mailbox.
5. If the operation is a read, the IOP waits. When the data comes back from the XMI, the IOP puts the data in the mailbox and sets the DONE bit.
   a. The console notices the set DONE bit in the mailbox and fetches the read data from the mailbox.

Calculating XMI CSR Addresses

The XMI bus supports a 40-bit address using data lines D57:48 and D29:00. The 7000/10000 system uses 32 of these address lines as it supports a 32-bit physical address on the XMI bus as shown in Table A–1.

Table A–1 XMI Node Space for 7000/10000 Systems

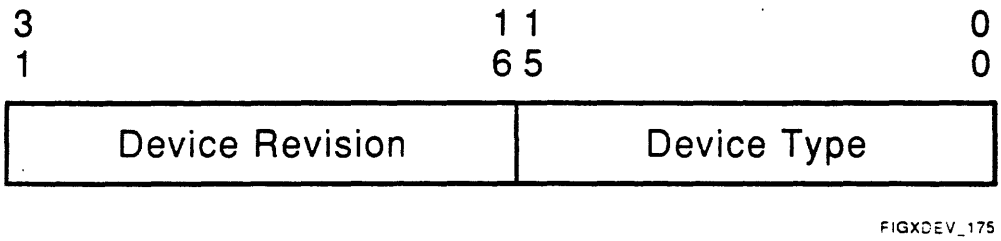
BB Address	Node Number	BB Address	Node Number
6180 0000	Node 0[1]	61C0 0000	Node 8[3]
6188 0000	Node 1	61C8 0000	Node 9
6190 0000	Node 2	61D0 0000	Node A
6198 0000	Node 3	61D8 0000	Node B
61A0 0000	Node 4	61E0 0000	Node C
61A8 0000	Node 5	61E8 0000	Node D
61B0 0000	Node 6	61F0 0000	Node E
61B8 0000	Node 7 [2]	61F8 0000	Node F[1]

[1]Not used.
[2]Clock module is in this slot.
[3]DWLMA (LAMB) module is in this slot.

All nodes on the XMI must have the following registers:

Device Type register (XDEV)	BB+00
Bus Error register (XBER)	BB+04
Failing Address register (XFADR)	BB+08

Figure A–1 XMI Device Type Register BB+00 (XDEV)



The device type contained in bits 15:00 is broken up into two fields:

- <15:08>—device class (DCLS)
- <07:00>—device ID (DEVID)

Table A–2 Device Types for 7000/10000 System XMI Modules

Module	Option	Code	Description
T2018	DWMBB/A[1]	2002	XMI-to-VAXBI adapter (XBI+, mapped)
T2018	DWMVE[1]	????	XMI-to-VME bus adapter
T2020	DEMNA	0C03	Ethernet/802 controller
T2080	CIXCD	0C05	CI interface adapter
T2022, T2023	KDM70	0C22	DSA disk and tape controller
T2036	KFMSA	0810	DSSI bus adapter
T2029-AA	KZMSA-AA[2]	0C36	SCSI-2 bus adapter
T2027	DEMFA	0823	XMI-to-FDDI adapter
T2028	DWLMA(LAMB)	102A	LASER-to-XMI board

[1]Planned for VAX 7000/10000 systems

[2]DEC 7000/10000 systems only

XMI Address Exercise

XMI Address Problem 1

You want to use the console to examine the FADR register in the CIXCD option shown in Figure A–2. Calculate the hexadecimal address of the FADR register.

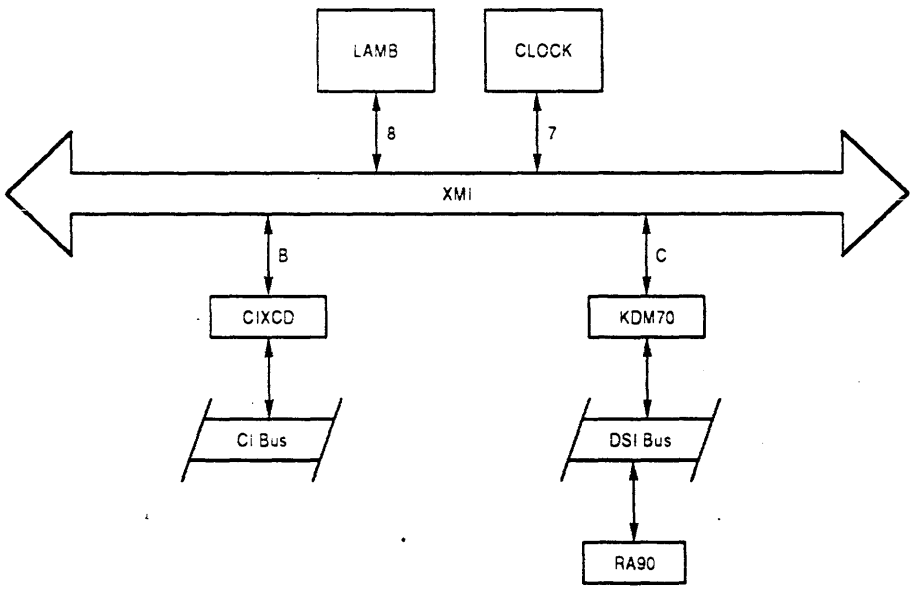
Node base address (BB) for node ____ is _____.

Offset for the FADR register is _____.

Address for node ____ FADR register is _____.

Console command _____.

Figure A–2 XMI Address Problem 1 and 2 Diagram



XMI Address Problem 2

Using the diagram in Figure A–2, calculate the hexadecimal address of the XBER register of the KDM70 node at slot C, as shown in Figure A–2.

Node base address (BB) for node ____ is _____.

Offset for the XBER register is _____.

Address for node ____ XBER register is _____.

Console command _____.

Table A–3 XMI I/O Area Address Allocation

Slot/Node	Node Space Base Address (BB)	I/O Window Space Prefix (bb)
1	6188 0000	62xx xxxx
2	6190 0000	64xx xxxx
3	6198 0000	66xx xxxx
4	61A0 0000	68xx xxxx
5	61A8 0000	6Axx xxxx
6	61B0 0000	6Cxx xxxx
7	61B8 0000	6Exx xxxx [1]
8	61C0 0000	70xx xxxx [2]
9	61C8 0000	72xx xxxx
A	61D0 0000	74xx xxxx
B	61D8 0000	76xx xxxx
C	61E0 0000	78xx xxxx
D	61E8 0000	7Axx xxxx
E	61F0 0000	7Cxx xxxx

[1]Clock module is in this slot.
[2]DWLMA module (LAMB) is in this slot.

The node space and window space addresses for the VAXBI are listed in Table A–4. There are no VAXBI bus adapters supported on the 7000/10000 systems at this time so the window space addresses should not be needed.

Table A–4 VAXBI Node Space and Window Space Address Allocation

VAXBI Node	Node Space Addresses		Window Space Addresses	
Number	Starting	Ending	Starting	Ending
0	xx00 0000	xx00 1FFF	xx40 0000	xx43 FFFF
1	xx00 2000	xx00 3FFF	xx44 0000	xx47 FFFF
2	xx00 4000	xx00 5FFF	xx48 0000	xx4B FFFF
3	xx00 6000	xx00 7FFF	xx4C 0000	xx4F FFFF
4	xx00 8000	xx00 9FFF	xx50 0000	xx53 FFFF
5	xx00 A000	xx00 BFFF	xx54 0000	xx57 FFFF
6	xx00 C000	xx00 DFFF	xx58 0000	xx5B FFFF
7	xx00 E000	xx00 FFFF	xx5C 0000	xx5F FFFF
8	xx01 0000	xx01 1FFF	xx60 0000	xx63 FFFF
9	xx01 2000	xx01 3FFF	xx64 0000	xx67 FFFF
A	xx01 4000	xx01 5FFF	xx68 0000	xx6B FFFF
B	xx01 6000	xx01 7FFF	xx6C 0000	xx6F FFFF
C	xx01 8000	xx01 9FFF	xx70 0000	xx73 FFFF
D	xx01 A000	xx01 BFFF	xx74 0000	xx77 FFFF
E	xx01 C000	xx01 DFFF	xx78 0000	xx7B FFFF
F	xx01 E000	xx01 FFFF	xx7C 0000	xx7F FFFF

Table A–5 VAXBI Registers

Name	Mnemonic	Address
Device Type Register	DTYPE	bb + 00
VAXBI Control and Status Register	VAXBICSR	bb + 04
Bus Error Register	BER	bb + 08
Error Interrupt Control Register	EINTRSCR	bb + 0C
Interrupt Destination Register	INTRDES	bb + 10
IPINTR Mask Register	IPINTRMSK	bb + 14
Force-Bit IPINTR/STOP Destination Register	FIPSDES	bb + 18
IPINTR Source Register	IPINTRSRC	bb + 1C
Starting Address Register	SADR	bb + 20
Ending Address Register	EADR	bb + 24
BCI Control and Status Register	BCICSR	bb + 28
Write Status Register	WSTAT	bb + 2C
Force-Bit IPINTR/STOP Command Register	FIPSCMD	bb + 30
User Interface Interrupt Control Register	UINTRCSR	bb + 40
General Purpose Register 0	GPR0	bb + F0
General Purpose Register 1	GPR1	bb + F4
General Purpose Register 2	GPR2	bb + F8
General Purpose Register 3	GRP3	bb + FC
Slave-Only Status Register	SOSR	bb + 100
Receive Console Data Register	RXCD	bb + 200

VAXBI Address Exercise

VAXBI Address Problem 1

Using the diagram in Figure A–3, calculate the hexadecimal address of the bus error register in the DMB32 option shown.

I/O Window Space prefix for XMI node ____ is _____.

VAXBI Nodespace Address for VAXBI node ____ is _____.

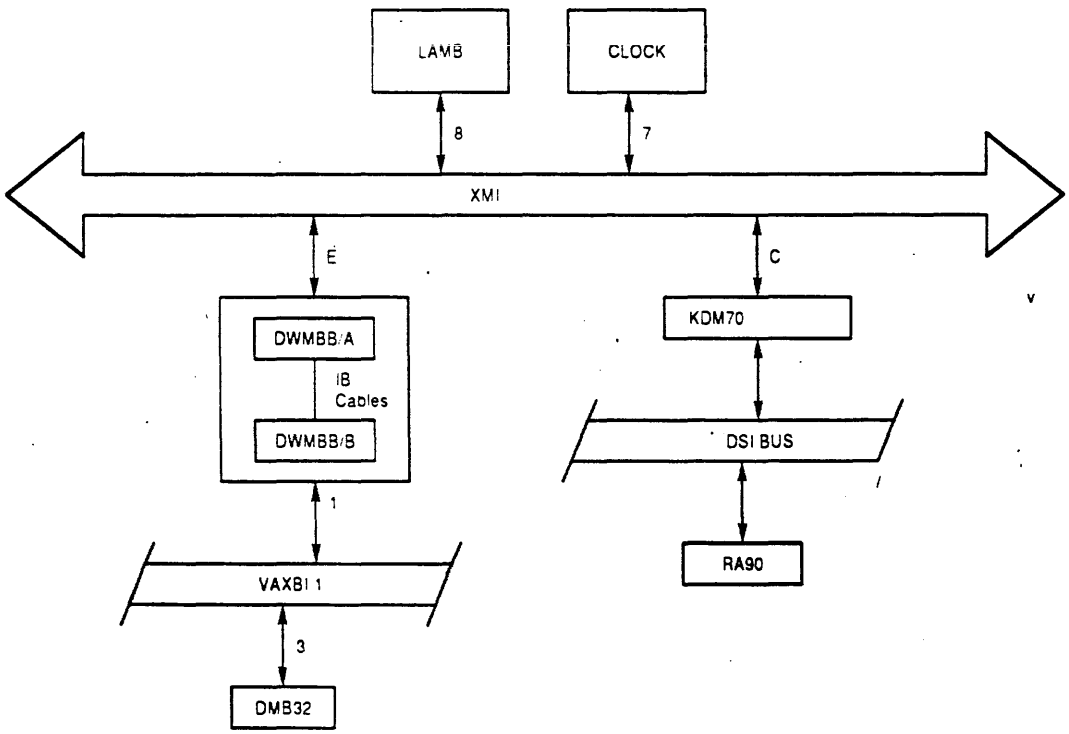
Calculate the bb Address to be _____.

Offset for the DMB32 bus error register is _____.

Address for bus error register in the DMB32 is _____

Console command _____.

Figure A–3 VAXBI Address Problem 1 Diagram



VAX 7000-600 Errorlog Examples

VAX 7000-600 Memory Error Example

Example B–1 VAX 7000-600 Memory Error

```
******************************** ENTRY     856. ********************************
ERROR SEQUENCE 1532.                          LOGGED ON:        SID 17000001
DATE/TIME 13-JAN-1993 12:15:59.93                              SYS_TYPE 01020001
SYSTEM UPTIME: 1 DAYS 23:05:01
SCS NODE: M6TEEN                                               VAX/VMS V5.5-2

MEMORY CONTROLLER ERROR KA7AA-AA  CPU FW REV# 1.  CONSOLE FW REV# 0.2
LMA Subpacket

Controller 1.
      PHYS ADDR       F9C00000
                      00000000
      VALID BITS      000FFFFF
                      00000000
      LDEV            00004000        Laser Memory Module
                                      Device Revision = 00(X)
      LBER            00040000
                                      NODE-SPECIFIC ERROR
      LCNR            00000000
                                      Self Test Passed
      LBESR0          00000000
                                      Syndrome 0 = 00(X)
      LBESR1          00000000
                                      Syndrome 1 = 00(X)
      LBESR2          00000000
                                      Syndrome 2 = 00(X)
      LBESR3          00000000
                                      Syndrome 3 = 00(X)
      LBECR0          000078A0
                                      Command Address =  00000078A0
      LBECR1          00000400
                                      Cmder ID REQ<3:0> = 8(X)
                                      Data Cycle Error = 0(X)
      MCR             00000004
                                      DRAM Type = 4 Mbit
                                      2 Strings
      AMR             0000038B
                                      Enable
                                      2 Way Interleave
                                      Interleave Address = 1(X)
                                      Address Width = C(X)
                                      2 Memory Banks per Module
                                      Module address = 00000000(X)
      MSTR0           00000000
      MSTR1           00000000
      FADR            0058213A
      MERA            00000020        Correctable ECC Error on MICB
                                      Failing String = 0(X)
      MSYNDA          00000000
                                      Syndrome A = 0(X)
      MERB            00000001
                                      Correctable Read Error
      MSYNDB          0000004A
                                      Syndrome B = A(X)
```

VAX 7000-600 System Soft Error Example (Vector 54)

Example B–2 VAX 7000-600 Soft Error

```
V A X / V M S        SYSTEM ERROR REPORT         COMPILED 18-FEB-1993 19:41:36
                                                                  PAGE  13.
******************************** ENTRY      5. ********************************
ERROR SEQUENCE 724.                              LOGGED ON:        SID 17000002
DATE/TIME 18-FEB-1993 04:24:16.03                                SYS_TYPE 01000001
SYSTEM UPTIME: 3 DAYS 19:10:24
SCS NODE: MARS                                                    VAX/VMS V5.5-2

INT54 ERROR KA7AA-AA  CPU FW REV# 2.  CONSOLE FW REV# 0.0
      PACKET REV      00000001
      RESERVED        00000000

      SW FLAGS        00000000
                      00000000
                      0000000C
                      00000000
      LOGGING CFF     00000000
                      00000000
                      00000000
                      00000000
      ACTIVE CPUS     00000007
      HW REVISION     00000000
      SYS SERIAL NUM  00000000
                      00000000
                          0000
                                       SYS SERIAL NUM = ..........
      SERIAL NUMBER   00000000
                      00000000
                          0000
                                       SERIAL NUMBER = ..........
      RESRC DISABLE   00000000

      PHYS ADDRESS    F8800000
                      00000000
      LDEV            00008002
                                       Laser NVAX Processor, 4M
                                       Device Revision = 00(X)
      LBER            00000000
      LCNR            00000001
                                       Enable Correctable Err Detection
                                       Self Test Passed

      LMERR           00000000
      LBESR0          0000000C
                                       Syndrome 0 = 0C(X)
      LBESR1          0000000C
                                       Syndrome 1 = 0C(X)
      LBESR2          0000000C
                                       Syndrome 2 = 0C(X)
      LBESR3          0000000C
                                       Syndrome 3 = 0C(X)
      LBECR0          00440036
                                       Command Address =  = 000440036
```

Example B–2 Cont'd on next page

Example B–2 (Continued) VAX 7000-600 Soft Error

```
LBECR1          00009060
                                CMD during error = Read CSR
                                CNF was asserted for this command
                                CMD/Address Parity = 1(X)
                                Cmder ID REQ<3:0> = 2(X)
                                Data Cycle Error = 0(X)
LMODE           000102A0
                                4M-Byte
                                Use PMAP lookup
                                NVAX+ 8k 2 sets I-Cache=2k VIC
                                Inhibit writes to STx_C
                                cnt 16 STx_C before LSB LOCKOUT
                                Pass 2 LEVI-A
LLOCK           00D4BC78
                                Address Invalid
GB HALT             003C
GB INTR             0000
GB PMASK            0001
                                halt enable
                                Select UART0A local terminal
GB WHAMI            0022
                                LSB Node ID = 2(X)
ECR             00120042
                                fbox enable
                                timeout clock
                                pmf pmux    = 01
                                pmf emux    = 02
VPSR            00000000
VAER            00000000
                                Vect Dest Reg Mask = 0000(X)
VMAC            00000000
BIU CTL         AFE09FF8
                                external cache enable
                                Generate/Expect ECC on check_h pins
                                output enable of cache rams
                                2-way associative
                                2X CPU Cycle
                                IO Map = 3(X)
                                4 Mbytes
BC TAG          19405800
                                tag_match
                                tag control V
                                tag control S
                                BC TAG = 0CA0(X)
BIU STAT        30080370
                                ECC ERR. ON EXT. CACHE FILL DATA
                                FILL ECC ERROR WAS CORRECTABLE
                                dread
                                Fill Quadword in Error = 0(X)
                                Bits 33,32 BIU Addr Reg = 3(X)
                                Bits 33,32 Fill Addr Reg = 0(X)
BIU ADDR        00000298
FILL SYN        00003800
                                L0 ECC Syn bits Low Longword = 0(X)
                                Hi ECC Syn bits High Longword = 0(X)
FILL ADDR       194B5318
```

Example B–2 Cont'd on next page

Example B–2 (Continued) VAX 7000-600 Soft Error

```
VMAR            000007E0
                                Sub Block Select = 0(X)
                                Row Index = 3F(X)
                                Error Address Field = 00000000(X)
VTAG            03C05486
VDATA           53D655D6
ICSR            00000001
                                enable VIC
PAMODE          00000000
                                30 bit physical address mode
MMEADR          C3800114
MMEPTE          00000000
MMESTS          10004005
                                MMESTS Lock Bits Not Set
TBADR           00000000
TBSTS           800001D0
                                TBSTS.LOCK<0> NOT SET
PCADR           FFFFFFF8
PCSTS           FFFFF8C0
                                PCSTS.LOCK<0> NOT SET
PCCTL           FFFFFC13
                                D-stream enabled
                                I-stream enabled
                                parity checking enabled
                                Performance Monitor Mode = 0(X)

COUNTERS        00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
```

VAX 7000-600 System Hard Error Examples (Vector 60)

Example B–3 VAX 7000-600 Hard Error (Example 1)

```
******************************** ENTRY       2. ********************************
ERROR SEQUENCE 941.                              LOGGED ON:        SID 17000002

DATE/TIME 18-NOV-1992 21:18:16.88                              SYS_TYPE 01000001

SYSTEM UPTIME: 2 DAYS 19:52:10
SCS NODE: BAMS4                                                 VAX/VMS V5.5-2

INT60 ERROR KA7AA-AA  CPU FW REV# 2.  CONSOLE FW REV# 0.0
      PACKET REV      00000001
      RESERVED        00000000

      SW FLAGS        00000000
                      00000000
                      00000000
                      00000000
      LOGGING OFF     00000000
                      00000000
                      00000000
                      00000000
      ACTIVE CPUS     00000003
      HW REVISION     00000000
      SYS SERIAL NUM  00000000
                      00000000
                          0000        SYS SERIAL NUM = ..........
      SERIAL NUMBER   00000000
                      00000000
                          0000        SERIAL NUMBER = ..........

      RESRC DISABLE   00000000

      PHYS ADDRESS    F8400000
                      00000000
      LDEV            00008002        Laser NVAX Processor, 4M
                                      Device Revision = 00(X)
      LBER            00040000
                                      NODE-SPECIFIC ERROR
      LCNR            00000001
                                      Enable Correctable Err Detection
                                      Self Test Passed
      LMERR           00000080
                                      B-CACHE DATA SINGLE BIT ERROR
      LBESR0          0000000C
                                      Syndrome 0 = 0C(X)
      LBESR1          0000000C
                                      Syndrome 1 = 0C(X)
      LBESR2          0000000C
                                      Syndrome 2 = 0C(X)
      LBESR3          0000000C
                                      Syndrome 3 = 0C(X)
```

Example B–3 Cont'd on next page

Example B–3 (Continued) VAX 7000-600 Hard Error (Example 1)

```
LBECR0          00420036
                                Command Address =  = 000420036
LBECR1          00008048
                                CMD during error = Write
                                CNF was asserted for this command
                                CMD/Address Parity = 1(X)
                                Cmder ID REQ<3:0> = 0(X)
                                Data Cycle Error = 0(X)
LMODE           000102A0
                                4M-Byte
                                Use PMAP lookup
                                NVAX+ 8k 2 sets I-Cache=2k VIC
                                Inhibit writes to STx_C
                                cnt 16 STx_C before LSB LOCKOUT
                                Pass 2 LEVI-A
LLOCK           00F735A4
                                Address Invalid
GB HALT             0038
GB INTR             0000
GB PMASK            0001        halt enable
                                Select UART0A local terminal
GB WHAMI            0021
                                LSB Node ID = 1(X)
ECR             00120042
                                fbox enable
                                timeout clock
                                pmf pmux    = 01
                                pmf emux    = 02
VPSR            00000000
VAER            00000000        Vect Dest Reg Mask = 0000(X)
VMAC            00000000
BIU CTL         AFE09FF8        external cache enable
                                Generate/Expect ECC on check_h pins
                                output enable of cache rams
                                2-way associative
                                2X CPU Cycle
                                IO Map = 3(X)
                                4 Mbytes
BC TAG          1F400800
                                tag_match
                                BC TAG = 0FA0(X)
BIU STAT        30091370
                                ECC ERR. ON EXT. CACHE FILL DATA
                                FILL ECC ERROR WAS CORRECTABLE
                                dread
                                Fill Quadword in Error = 1(X)
                                Bits 33,32 BIU Addr Reg = 3(X)
                                Bits 33,32 Fill Addr Reg = 0(X)
BIU ADDR        00000298
FILL SYN        00000070        L0 ECC Syn bits Low Longword = 0(X)
                                Hi ECC Syn bits High Longword = 0(X)
FILL ADDR       1F621BDC
VMAR            000007E0
                                Sub Block Select = 0(X)
                                Row Index = 3F(X)
                                Error Address Field = 00000000(X)
```

Example B–3 Cont'd on next page

Example B–3 (Continued) VAX 7000-600 Hard Error (Example 1)

```
VTAG            01C605C6
VDATA           53D655D6
ICSR            00000001
                                enable VIC
PAMODE          00000000
                                30 bit physical address mode
MMEADR          00007A00
MMEPTE          00000000
MMESTS          1C008000
                                MMESTS Lock Bits Not Set
TBADR           00000000
TBSTS           800001D0
                                TBSTS.LOCK<0> NOT SET
PCADR           FFFFFFF8
PCSTS           FFFFF800
                                PCSTS.LOCK<0> NOT SET
PCCTL           FFFFFC13
                                D-stream enabled
                                I-stream enabled
                                parity checking enabled
                                Performance Monitor Mode = 0(X)
COUNTERS        00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                6D120660
                00963D0F
                00000001
                00000000
                00000000
                00000000
                00000000
                A0000000
                00000000
                00000000
                00000000
                00000000
                00000003
                00000000
```

Example B–4 VAX 7000-600 Hard Error (Example 2)

```
V A X / V M S        SYSTEM ERROR REPORT         COMPILED 18-FEB-1993 19:41:35
                                                                  PAGE   1.
******************************** ENTRY       2. ********************************
ERROR SEQUENCE 721.                              LOGGED ON:        SID 17000002
DATE/TIME 18-FEB-1993 04:24:16.01                                SYS_TYPE 01000001
SYSTEM UPTIME: 3 DAYS 19:10:24
SCS NODE: MARS                                                    VAX/VMS V5.5-2

INT60 ERROR KA7AA-AA  CPU FW REV# 2.  CONSOLE FW REV# 0.0
      PACKET REV      00000001
      RESERVED        00000000

      SW FLAGS        00000000
                      00000000
                      00000000
                      00000400        LSB error log snapshot packet present

      LOGGING OFF     00000000
                      00000000
                      00000000
                      00000000
      ACTIVE CPUS     00000007
      HW REVISION     00000000
      SYS SERIAL NUM  00000000
                      00000000
                          0000        SYS SERIAL NUM = ..........

      SERIAL NUMBER   00000000
                      00000000
                          0000        SERIAL NUMBER = ..........

      RESRC DISABLE   00000000

      PHYS ADDRESS    F8000000
                      00000000
      LDEV            00008002        Laser NVAX Processor, 4M
                                      Device Revision = 00(X)

      LBER            00000009

                                      ERROR LINE ASSERTED
                                      CORRECTABLE DATA ERROR

      LCNR            00000001

                                      Enable Correctable Err Detection
                                      Self Test Passed

      LMERR           00000000
      LBESR0          00000000

                                      Syndrome 0 = 00(X)
      LBESR1          00000070
                                      Syndrome 1 = 70(X)
      LBESR2          00000000
                                      Syndrome 2 = 00(X)
      LBESR3          00000000
                                      Syndrome 3 = 00(X)
      LBECR0          00CA5A98
                                      Command Address =  = 000CA5A98
```

Example B–4 Cont'd on next page

Example B–4 (Continued) VAX 7000-600 Hard Error (Example 2)

```
LBECR1          00039000
                                CMD during error = Read
                                CNF was asserted for this command
                                Shared was asserted for this cmd.
                                Dirty was asserted for this cmd.
                                CMD/Address Parity = 0(X)
                                Cmder ID REQ<3:0> = 2(X)
                                Data Cycle Error = 0(X)
LMODE           000102A0
                                4M-Byte
                                Use PMAP lookup
                                NVAX+ 8k 2 sets I-Cache=2k VIC
                                Inhibit writes to STx_C
                                cnt 16 STx_C before LSB LOCKOUT
                                Pass 2 LEVI-A
LLOCK           00D4BC78
                                Address Invalid
GB HALT             003C
GB INTR             0000
GB PMASK            0009        halt enable
                                Select UART0A local terminal
GB WHAMI            0020
                                LSB Node ID = 0(X)
ECR             00000042
                                fbox enable
                                timeout clock
                                pmf pmux    = 00
                                pmf emux    = 00
VPSR            00000000
VAER            00000000
                                Vect Dest Reg Mask = 0000(X)
VMAC            00000000
BIU CTL         AFE09FF8        external cache enable
                                Generate/Expect ECC on check_h pins
                                output enable of cache rams
                                2-way associative
                                2X CPU Cycle
                                IO Map = 3(X)
                                4 Mbytes
BC TAG          1F803800
                                tag_match
                                tag control V
                                tag control D
                                BC TAG = 0FC0(X)
BIU STAT        F00E3070
                                Bits 33,32 BIU Addr Reg = 3(X)
                                Bits 33,32 Fill Addr Reg = 3(X)
BIU ADDR        00000298
FILL SYN        00000000        L0 ECC Syn bits Low Longword = 0(X)
                                Hi ECC Syn bits High Longword = 0(X)
FILL ADDR       000002A8
VMAR            000007E0
                                Sub Block Select = 0(X)
                                Row Index = 3F(X)
                                Error Address Field = 00000000(X)
```

Example B–4 Cont'd on next page

Example B–4 (Continued) VAX 7000-600 Hard Error (Example 2)

```
VTAG            03C05486

VDATA           53D655D6
ICSR            00000001
                                enable VIC
PAMODE          00000000
                                30 bit physical address mode
MMEADR          C3800114
MMEPTE          00000000
MMESTS          10004005
                                MMESTS Lock Bits Not Set
TBADR           00000000
TBSTS           800001D0
                                TBSTS.LOCK<0> NOT SET
PCADR           FFFFFFF8
PCSTS           FFFFF800
                                PCSTS.LOCK<0> NOT SET
PCCTL           FFFFFE13
                                D-stream enabled
                                I-stream enabled
                                parity checking enabled
                                Performance Monitor Mode = 0(X)
COUNTERS        00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
```

Example B–4 Cont'd on next page

Example B–4 (Continued) VAX 7000-600 Hard Error (Example 2)

```
LSB SUBPACKET
      PHYS ADDR       F8000000
                      00000000
Node Present = 0.
LSB CPU SUBPACKET
      PHYS ADDR       F8000000
                      00000000
      LDEV            00008002        Laser NVAX Processor, 4M
                                      Device Revision = 00(X)
      LBER            00000009
                                      ERROR LINE ASSERTED
                                      CORRECTABLE DATA ERROR
      LCNR            00000001
                                      Enable Correctable Err Detection
                                      Self Test Passed
      LMERR           00000000
      LBESR0          00000000
                                      Syndrome 0 = 00(X)
      LBESR1          00000070
                                      Syndrome 1 = 70(X)
      LBESR2          00000000
                                      Syndrome 2 = 00(X)
      LBESR3          00000000
                                      Syndrome 3 = 00(X)
      LBECR0          00CA5A98
                                      Command Address =  = 000CA5A98
      LBECR1          00039000
                                      CMD during error = Read
                                      CNF was asserted for this command
                                      Shared was asserted for this cmd.
                                      Dirty was asserted for this cmd.
                                      CMD/Address Parity = 0(X)
                                      Cmder ID REQ<3:0> = 2(X)
                                      Data Cycle Error = 0(X)
Node Present = 1.
LSB CPU SUBPACKET
      PHYS ADDR       F8400000
                      00000000
      LDEV            00008002        Laser NVAX Processor, 4M
                                      Device Revision = 00(X)
      LBER            00040209
                                      ERROR LINE ASSERTED
                                      CORRECTABLE DATA ERROR
                                      TRANSMITTER DURING ERROR
                                      NODE-SPECIFIC ERROR
      LCNR            00000001
                                      Enable Correctable Err Detection
                                      Self Test Passed
      LMERR           00000080
                                      B-CACHE DATA SINGLE BIT ERROR
      LBESR0          00000000
                                      Syndrome 0 = 00(X)
      LBESR1          00000070
                                      Syndrome 1 = 70(X)
      LBESR2          00000000
                                      Syndrome 2 = 00(X)
      LBESR3          00000000
                                      Syndrome 3 = 00(X)
```

Example B–4 Cont'd on next page

Example B–4 (Continued) VAX 7000-600 Hard Error (Example 2)

```
      LBECR0          00CA5A98
                                      Command Address =  = 000CA5A98
      LBECR1          00039000
                                      CMD during error = Read
                                      CNF was asserted for this command
                                      Shared was asserted for this cmd.
                                      Dirty was asserted for this cmd.
                                      CMD/Address Parity = 0(X)
                                      Cmder ID REQ<3:0> = 2(X)
                                      Data Cycle Error = 0(X)
Node Present = 2.
LSB CPU SUBPACKET
      PHYS ADDR       F8800000
                      00000000
      LDEV            00008002        Laser NVAX Processor, 4M
                                      Device Revision = 00(X)
      LBER            00000009
                                      ERROR LINE ASSERTED
                                      CORRECTABLE DATA ERROR
      LCNR            00000001
                                      Enable Correctable Err Detection
                                      Self Test Passed
      LMERR           00000000
      LBESR0          00000000
                                      Syndrome 0 = 00(X)
      LBESR1          00000070
                                      Syndrome 1 = 70(X)
      LBESR2          00000000
                                      Syndrome 2 = 00(X)
      LBESR3          00000000
                                      Syndrome 3 = 00(X)
      LBECR0          00CA5A98
                                      Command Address =  = 000CA5A98
      LBECR1          00039000
                                      CMD during error = Read
                                      CNF was asserted for this command
                                      Shared was asserted for this cmd.
                                      Dirty was asserted for this cmd.
                                      CMD/Address Parity = 0(X)
                                      Cmder ID REQ<3:0> = 2(X)
                                      Data Cycle Error = 0(X)
Node Present = 7.
LSB MEMORY SUBPACKET
      PHYS ADDR       F9C00000
                      00000000
      LDEV            00004000        Laser Memory Module
                                      Device Revision = 00(X)
      LBER            00000001
                                      ERROR LINE ASSERTED
      LCNR            00000000
                                      Self Test Passed
      LBESR0          0000000C
                                      Syndrome 0 = 0C(X)
      LBESR1          0000000C
                                      Syndrome 1 = 0C(X)
      LBESR2          0000000C
                                      Syndrome 2 = 0C(X)
      LBESR3          0000000C
                                      Syndrome 3 = 0C(X)
```

Example B–4 Cont'd on next page

Example B–4 (Continued) VAX 7000-600 Hard Error (Example 2)

```
      LBECR0          00007354
                                      Command Address =  0000007354
      LBECR1          00000400
                                      Cmder ID REQ<3:0> = 8(X)
                                      Data Cycle Error = 0(X)

Node Present = 8.
LSB IOP SUBPACKET
      PHYS ADDR       FA000000
                      00000000
      LDEV            00042000        Laser IO Module
                                      Device Revision = 04(X)
      LBER            00000001
                                      ERROR LINE ASSERTED
      LCNR            80000000
                                      Self Test Passed
      LBESR0          0000000C
                                      Syndrome 0 = 0C(X)
      LBESR1          0000000C
                                      Syndrome 1 = 0C(X)
      LBESR2          0000000C
                                      Syndrome 2 = 0C(X)
      LBESR3          0000000C
                                      Syndrome 3 = 0C(X)
      LBECR0          00500036
                                      Command Address =  6000500036
      LBECR1          00008060
                                      CNF WAS ASSERTED FOR THIS COMMAND
                                      Cmder ID REQ<3:0> = 0(X)
```

Example B–5 VAX 7000-600 Hard Error (Example 3)

```
********************************* ENTRY       3. *********************************
ERROR SEQUENCE 722.                              LOGGED ON:        SID 17000002
DATE/TIME 18-FEB-1993 04:24:16.03                                  SYS_TYPE 01000001
SYSTEM UPTIME: 3 DAYS 19:10:24
SCS NODE: MARS                                                     VAX/VMS V5.5-2

INT60 ERROR KA7AA-AA  CPU FW REV# 2.  CONSOLE FW REV# 0.0
      PACKET REV      00000001
      RESERVED        00000000

      SW FLAGS        00000000
                      00000000
                      00000000
                      00000000
      LOGGING OFF     00000000
                      00000000
                      00000000
                      00000000
      ACTIVE CPUS     00000007
      HW REVISION     00000000
      SYS SERIAL NUM  00000000
                      00000000
                          0000
                                        SYS SERIAL NUM = ..........
      SERIAL NUMBER   00000000
                      00000000
                          0000
                                        SERIAL NUMBER = ..........
      RESRC DISABLE   00000000

      PHYS ADDRESS    F8400000
                      00000000
      LDEV            00008002
                                        Laser NVAX Processor, 4M
                                        Device Revision = 00(X)
      LBER            00040000
                                        NODE-SPECIFIC ERROR
      LCNR            00000001
                                        Enable Correctable Err Detection
                                        Self Test Passed
      LMERR           00000080
                                        B-CACHE DATA SINGLE BIT ERROR
      LBESR0          0000000C
                                        Syndrome 0 = 0C(X)
      LBESR1          0000000C
                                        Syndrome 1 = 0C(X)
      LBESR2          0000000C
                                        Syndrome 2 = 0C(X)
      LBESR3          0000000C
                                        Syndrome 3 = 0C(X)
      LBECR0          00420036
                                        Command Address =  = 000420036
```

Example B–5 Cont'd on next page

Example B–5 (Continued) VAX 7000-600 Hard Error (Example 3)

```
LBECR1          00008860
                                CMD during error = Read CSR
                                CNF was asserted for this command
                                CMD/Address Parity = 1(X)
                                Cmder ID REQ<3:0> = 1(X)
                                Data Cycle Error = 0(X)
LMODE           000102A0
                                4M-Byte
                                Use PMAP lookup
                                NVAX+ 8k 2 sets I-Cache=2k VIC
                                Inhibit writes to STx_C
                                cnt 16 STx_C before LSB LOCKOUT
                                Pass 2 LEVI-A
LLOCK           00E72114
                                Address Invalid
GB HALT             003C
GB INTR             0000
GB PMASK            0001
                                halt enable
                                Select UART0A local terminal
GB WHAMI            0021
                                LSB Node ID = 1(X)
ECR             00120002
                                fbox enable
                                pmf pmux    = 01
                                pmf emux    = 02
VPSR            00000000
VAER            00000000
                                Vect Dest Reg Mask = 0000(X)
VMAC            00000000
BIU CTL         AFE09FF8
                                external cache enable
                                Generate/Expect ECC on check_h pins
                                output enable of cache rams
                                2-way associative
                                2X CPU Cycle
                                IO Map = 3(X)
                                4 Mbytes
BC TAG          1A803800
                                tag_match
                                tag control V
                                tag control D
                                BC TAG = 0D40(X)
BIU STAT        F00E3070
                                Bits 33,32 BIU Addr Reg = 3(X)
                                Bits 33,32 Fill Addr Reg = 3(X)
BIU ADDR        00000298
FILL SYN        00000000
                                L0 ECC Syn bits Low Longword = 0(X)
                                Hi ECC Syn bits High Longword = 0(X)
FILL ADDR       000002A8
VMAR            000007E0
                                Sub Block Select = 0(X)
                                Row Index = 3F(X)
                                Error Address Field = 00000000(X)
```

Example B–5 Cont'd on next page

Example B–5 (Continued) VAX 7000-600 Hard Error (Example 3)

```
VTAG            03C05486
VDATA           53D655D6
ICSR            00000001
                                enable VIC
PAMODE          00000000
                                30 bit physical address mode
MMEADR          001E3C80
MMEPTE          00000000
MMESTS          18008000
                                MMESTS Lock Bits Not Set
TBADR           00000000
TBSTS           800001D0
                                TBSTS.LOCK<0> NOT SET
PCADR           FFFFFFF8
PCSTS           FFFFF800
                                PCSTS.LOCK<0> NOT SET
PCCTL           FFFFFC13
                                D-stream enabled
                                I-stream enabled
                                parity checking enabled
                                Performance Monitor Mode = 0(X)

COUNTERS        00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
                00000000
```

VAX 7000-600 System IOP Error Example (INT 17, Vector DC)

Example B–6 VAX 7000-600 System IOP Error

```
******************************** ENTRY     880. ********************************
ERROR SEQUENCE 35.                          LOGGED ON:        SID 17000002
DATE/TIME  5-JAN-1993 16:20:00.20                          SYS_TYPE 01000001
SYSTEM UPTIME: 0 DAYS 04:29:51
SCS NODE: PHV069                                            VAX/VMS V5.5-2

ADAPTER ERROR KA7AA-AA  CPU FW REV# 2.  CONSOLE FW REV# 0.0
      PKT REV          00000000
      SW FLAGS         00000000
                       00000001
                       00000000
                       00008000
                                        Channel subpacket present
IOP Subpacket

      PHYS ADDR        FA000000
                       00000000
      LDEV             00042000
                                        Laser IO Module
                                        Device Revision = 04(X)
      LBER             00040000
                                        NODE-SPECIFIC ERROR
      LCNR             80000000
                                        Self Test Passed
      LBESR0           0000000C
                                        Syndrome 0 = 0C(X)
      LBESR1           0000000C
                                        Syndrome 1 = 0C(X)
      LBESR2           0000000C
                                        Syndrome 2 = 0C(X)
      LBESR3           0000000C
                                        Syndrome 3 = 0C(X)
      LBECR0           00500036
                                        Command Address =  6000500036
      LBECR1           00008060
                                        CNF WAS ASSERTED FOR THIS COMMAND
                                        Cmder ID REQ<3:0> = 0(X)
                                        Data Cycle Error = 0(X)
      IPCNSE           80000100
                                        Uphose #0 packet error
                                        INTERUPT ON NSES
      IPCVR            00000050
                                        IOP Interrupt Vector = 50(X)
      IPCMSR           00000000
                                        arbs 6x @ high 2x @ low
      IPCHST           00000006
                                        HOSE #0 STATUS:
                                                  Hose connected & power ok
                                                  Hose Power OK
                                        HOSE #1 STATUS:
                                                  Hose Not Connected
                                                  No Hose Power
```

Example B–6 Cont'd on next page

Example B–6 (Continued) VAX 7000-600 System IOP Error

```
                                    HOSE #2 STATUS:
                                              Hose Not Connected
                                              No Hose Power
                                    HOSE #3 STATUS:
                                              Hose Not Connected
                                              No Hose Power
      SW FLAGS          00010001
      Chan cnt =        00000001

 Channel #1
      PCA ADDR          C2000000
      XDEV              0106102A
                                    MEMORY DEVICE = 42.
                                    DEVICE REV = 262.
      XBER              00000202
                                    Enable More Protocol
      XFADR             00010008
                                    Failing Addr = 00000010008(X)
                                    Failing length = Hexaword
      XFAER             90000000
                                    Transaction Byte Mask = 0000
                                    IDENT CMD
      LDIAG             81000000
                                    6 free Q loc left after ARB sup asserts
                                    Correct Parity Asserted
                                    ARB Suppress Asserted
                                    LAMB Node ID  = 8.
      IMSK              0BFFA070
                                    Interrupt On Mailbox Errors
                                    Interrupt on Read Buffer DPE
                                    Interrupt on DATA FIFO Data Field PE
                                    Interrupt on Transaction Timeout
                                    Interrupt on Command NOACK
                                    Interrupt on Read Error Response
                                    Interrupt on Read Sequence Error
                                    Interrupt on No Read Response
                                    Interrupt on Corrected Read Data
                                    Interrupt on Write Data NOACK
                                    Interrupt on Read/IDENT Data NOACK
                                    Interrupt on Write sequence error
                                    Interrupt on XMI Parity Error
                                    Interrupt on XMI Inconsistent PE
                                    Interrupt on Write Error Interrupt
                                    Intrpt on Corrected Confirmation Err
      LEVR              000001E0
                                    Vector  = 01E0
      LERR              00018000
      LGPR              00000000
      IPR1              00000000
      IPR2              00000000
      IIPR              00000003
                                    Intrupt in progress from Node 3 @IPL 14
```

VAX 7000-600 System XMI Bus Errors

Example B–7 VAX 7000-600 System XMI Error (Example 1)

```
V A X */ V M S        SYSTEM ERROR REPORT         COMPILED 14-JAN-1993 11:51:52
                                                                  PAGE   1.

******************************** ENTRY     718. ********************************
ERROR SEQUENCE 859.                          LOGGED ON:        SID 17000001
DATE/TIME  9-JAN-1993 10:41:52.04                           SYS_TYPE 01020001
SYSTEM UPTIME: 1 DAYS 19:08:00
SCS NODE: M6TEEN                                             VAX/VMS V5.5-2

ADAPTER ERROR KA7AA-AA  CPU FW REV# 1.  CONSOLE FW REV# 0.2

DWLMA Subpacket
      PKT REV          00000000
      RESERVED         00000000
      SW FLAGS         00004402
                       00000000
                       00000000
                       00010000
                                       XMI present
      PCA ADDR         C2000000
      XDEV             0105102A
                                       MEMORY DEVICE = 42.
                                       DEVICE REV = 261.
                                       MEMORY DEVICE = DWLMA
      XBER             90042202
                                       Enable More Protocol
                                       Transaction Timeout
                                       Nc Read Response
                                       XMI Bad
                                       Error Summary
                                       Commander ID = NODE #08(X)
      XFADR            61E80108
                                       Failing Addr = 00021E80108(X)
                                       Failing length = Longword
      XFAER            1000000F
                                       Transaction Byte Mask = 000F
                                       READ
      LDIAG            81000000
                                       6 free Q loc left after ARB sup asserts
                                       Correct Parity Asserted
                                       ARB Suppress Asserted
                                       LAMB Node ID  = 8.
      IMSK             0BFFA070
                                       Interrupt On Mailbox Errors
                                       Interrupt on Read Buffer DPE
                                       Interrupt on DATA FIFO Data Field PE
                                       Interrupt on Transaction Timeout
                                       Interrupt on Command NOACK
                                       Interrupt on Read Error Response
                                       Interrupt on Read Sequence Error
```

Example B–7 Cont'd on next page

Example B–7 (Continued) VAX 7000-600 System XMI Error (Example 1)

```
                                    Interrupt on No Read Response
                                    Interrupt on Corrected Read Data
                                    Interrupt on Write Data NOACK
                                    Interrupt on Read/IDENT Data NOACK
                                    Interrupt on Write sequence error
                                    Interrupt on XMI Parity Error
                                    Interrupt on XMI Inconsistent PE
                                    Interrupt on Write Error Interrupt
                                    Intrpt on Corrected Confirmation Err

      LEVR          000001E0
                                    Vector  = 01E0
      LERR          00028000
      LGPR          00000000
      IPR1          00010000
                                    Interrupt Pending, node 5, IPL 14
      IPR2          00000000
      IIPR          00000002
                                    Intrupt in progress from Node 2 @IPL 14

XMI NODE DATA
      PHYS ADDR     C2000000
      XDEV VALID        3126
      XBE VALID         3126
      XFADR VALID       3000
      XFAER VALID       3000
      NODE PRESENT      3126

    XMI NODE #1.
      XDEV          08020C03
                                    DEMNA
                                    HW REV = H
                                    EEPROM FW REV = 2.
      XBE           00000046
                                    COMMANDER ID = NODE #01(X)
                                    ENABLE MORE PROTOCOL
                                    DISABLE XMI TIMEOUTS

    XMI NODE #2.
      XDEV          A4A60810
                                    KFMSA
                                    HW REV = A4
                                    FW REV = V5.6
      XBE           00000080
                                    COMMANDER ID = NODE #02(X)

    XMI NODE #5.
      XDEV          46110C05
                                    CIXCD
                                    HW REV = A1
                                    FW REV = V2.6
      XBE           00000149
                                    COMMANDER ID = NODE #05(X)
                                    ENABLE HEXAWORD WRITES
```

Example B–7 Cont'd on next page

Example B–7 (Continued) VAX 7000-600 System XMI Error (Example 1)

```
XMI NODE #8.
  XDEV              0105102A
                                  MEMORY DEVICE = 42.
                                  DEVICE REV = 261.
  XBE               10000202
                                  COMMANDER ID = NODE #08(X)
                                  XMI BAD
                                  ENABLE MORE PROTOCOL

XMI NODE #12
  XDEV              00FD0823
                                  IO DEVICE = 35.
                                  DEVICE REV = 253.
  XBE               C0000400
                                  COMMANDER ID = NODE #00(X)
                                  SELF TEST FAILED
                                  NODE RESET
                                  ERROR DETECTED
  XFADR             00000000
                                  Failing Addr = 00000000000(X)
                                  Failing length = Hexaword
  XFAER             00000000
                                  Transaction Byte Mask = 0000
                                  RSRVD

XMI NODE #13
  XDEV              66FD0823
                                  IO DEVICE = 35.
                                  DEVICE REV = 9981.
  XBE               C0000400
                                  COMMANDER ID = NODE #00(X)
                                  SELF TEST FAILED
                                  NODE RESET
                                  ERROR DETECTED
  XFADR             00000000
                                  Failing Addr = 00000000000(X)
                                  Failing length = Hexaword
  XFAER             00000000
                                  Transaction Byte Mask = 0000
                                  RSRVD
```

The system configuration shown here is associated with system error report 14.

Example B–8 VAX 7000-600 System Configuration

```
P00>>> show conf

        Name      Type     Rev   Mnemonic
  LSB
  0+    KA7AA     (8002)   000C  ka7aa0
  1+    KA7AA     (8002)   000C  ka7aa1
  7+    MS7AA     (4000)   0000  ms7aa0
  8+    IOP       (2000)   0006  iop0

  C0 XMI                         xmi0
  2+    KDM70     (0C22)   1E11  kdm700
  8+    DWLMA     (102A)   0106  dwlma0
  B+    DEMFA     (0823)   9114  demfa0
  C+    DEMFA     (0823)   0514  demfa1
  D+    DEMNA     (0C03)   0802  demna0
  E+    KFMSA     (0810)   A4A6  kfmsa0

  C1 XMI                         xmi1
  1+    KFMSA     (0810)   A4A6  kfmsa1
  2+    DEMNA     (0C03)   0802  demna1
  3+    DEMFA     (0823)   0514  demfa2
  4+    DEMFA     (0823)   9114  demfa3
  8+    DWLMA     (102A)   0106  dwlma1
```

Example B–9 VAX 7000-600 System XMI Error (Example 2)

```
V A X / V M S        SYSTEM ERROR REPORT         COMPILED 12-FEB-1993 06:41:55
                                                                  PAGE   1.

******************************** ENTRY      14. ********************************
ERROR SEQUENCE 1175.                          LOGGED ON:        SID 17000002
DATE/TIME 12-FEB-1993 05:42:36.55                             SYS_TYPE 01020001
SYSTEM UPTIME: 0 DAYS 05:09:37
SCS NODE: SUVA03                                               VAX/VMS V5.5-2

ADAPTER ERROR KA7AA-AA  CPU FW REV# 2.  CONSOLE FW REV# 0.2

DWLMA Subpacket
      PKT REV          00000000
      RESERVED         00000000
      SW FLAGS         00004402
                       00000000
                       00000000
                       00010000
```

Example B–9 Cont'd on next page

Example B–9 (Continued) VAX 7000-600 System XMI Error (Example 2)

```
                                XMI present
PCA ADDR        C6000000
XDEV            0106102A
                                MEMORY DEVICE = 42.
                                DEVICE REV = 262.
                                MEMORY DEVICE = DWLMA
XBER            90042202
                                Enable More Protocol
                                Transaction Timeout
                                No Read Response
                                XMI Bad
                                Error Summary
                                Commander ID = NODE #08(X)
XFADR           61A00108
                                Failing Addr = 00021A00108(X)
                                Failing length = Longword
XFAER           1000000F
                                Transaction Byte Mask = 000F
                                READ
LDIAG           81000000
                                6 free Q loc left after ARB sup asserts
                                Correct Parity Asserted
                                ARB Suppress Asserted
                                LAMB Node ID  = 8.
IMSK            0BFFA070
                                Interrupt On Mailbox Errors
                                Interrupt on Read Buffer DPE
                                Interrupt on DATA FIFO Data Field PE
                                Interrupt on Transaction Timeout
                                Interrupt on Command NOACK
                                Interrupt on Read Error Response
                                Interrupt on Read Sequence Error
                                Interrupt on No Read Response
                                Interrupt on Corrected Read Data
                                Interrupt on Write Data NOACK
                                Interrupt on Read/IDENT Data NOACK
                                Interrupt on Write sequence error
                                Interrupt on XMI Parity Error
                                Interrupt on XMI Inconsistent PE
                                Interrupt on Write Error Interrupt
                                Intrpt on Corrected Confirmation Err
LEVR            000003E0
                                Vector  = 03E0
LERR            00000000
LGPR            00000000
IPR1            00000000
IPR2            00000000
IIPR            00000000
```

Example B–9 Cont'd on next page

Example B–9 (Continued) VAX 7000-600 System XMI Error (Example 2)

```
XMI NODE DATA
      PHYS ADDR        C6000000
      XDEV VALID           7904
      XBE VALID            7904
      XFADR VALID          1800
      XFAER VALID          1800
      NODE PRESENT         7904

    XMI NODE #2.
      XDEV             1E110C22
                                    KDM70
                                    DEVICE REV = 7697.
      XBE              10000080
                                    COMMANDER ID = NODE #02(X)
                                    XMI BAD

    XMI NODE #8.
      XDEV             0106102A
                                    MEMORY DEVICE = 42.
                                    DEVICE REV = 262.
      XBE              10000202
                                    COMMANDER ID = NODE #08(X)
                                    XMI BAD
                                    ENABLE MORE PROTOCOL

    XMI NODE #11
      XDEV             91140823
                                    IO DEVICE = 35.
                                    DEVICE REV = 4372.
      XBE              C0000400
                                    COMMANDER ID = NODE #00(X)
                                    SELF TEST FAILED
                                    NODE RESET
                                    ERROR DETECTED
      XFADR            00000000
                                    Failing Addr = 00000000000(X)
                                    Failing length = Hexaword
      XFAER            00000000
                                    Transaction Byte Mask = 0000
                                    RSRVD

    XMI NODE #12
      XDEV             05140823
                                    IO DEVICE = 35.
                                    DEVICE REV = 1300.
      XBE              C0000400
                                    COMMANDER ID = NODE #00(X)
                                    SELF TEST FAILED
                                    NODE RESET
                                    ERROR DETECTED
      XFADR            00000000
                                    Failing Addr = 00000000000(X)
                                    Failing length = Hexaword
      XFAER            00000000
                                    Transaction Byte Mask = 0000
                                    RSRVD
```

Example B–9 Cont'd on next page

Example B–9 (Continued) VAX 7000-600 System XMI Error (Example 2)

```
    XMI NODE #13
      XDEV            08020C03
                                    DEMNA
                                    HW REV = H
                                    EEPROM FW REV = 2.
      XBE             00000346
                                    COMMANDER ID = NODE #0D(X)
                                    ENABLE MORE PROTOCOL
                                    DISABLE XMI TIMEOUTS

    XMI NODE #14
      XDEV            A4A60810
                                    KFMSA
                                    HW REV = A4
                                    FW REV = V5.6
      XBE             00000380
                                    COMMANDER ID = NODE #0E(X)

******************************** ENTRY      15. ********************************
ERROR SEQUENCE 1176.                          LOGGED ON:        SID 17000002
DATE/TIME 12-FEB-1993 05:42:36.55                             SYS_TYPE 01020001
SYSTEM UPTIME: 0 DAYS 05:09:37
SCS NODE: SUVA03                                               VAX/VMS V5.5-2

FATAL BUGCHECK KA7AA-AA  CPU FW REV# 2.  CONSOLE FW REV# 0.2

INCONSTATE, Inconsistent I/O data base

      PROCESS NAME    VET

      PROCESS ID      00010011

      ERROR PC        814EFEF1
      ERROR PSL       04150000
                                    INTERRUPT PRIORITY LEVEL = 21.
                                    PREVIOUS MODE = KERNEL
                                    CURRENT MODE = KERNEL
                                    INTERRUPT STACK
                                    FIRST PART DONE CLEAR

STACK POINTERS

KSP 7FFE7800  ESP 7FFE9800  SSP 7FFECA48  USP 7FEF3708  ISP 818DB0DC

GENERAL REGISTERS

R0  00000054  R1  0006072B  R2  00000108  R3  8039EC80  R4  80A131A0
R5  80A12EE0  R6  000C0AA0  R7  000BEC20  R8  000BEC64  R9  000BED14
R10 0000060C  R11 7FEF570A  AP  818DB1A4  FP  818DB184  SP  818DB17C
```

Example B–9 Cont'd on next page

Example B–9 (Continued) VAX 7000-600 System XMI Error (Example 2)

```
SYSTEM REGISTERS
      P0BR            81A0FC00
                                    P0 PTE BASE (VIRT ADDRS)
      P0LR            0000081C
                                    TOTAL P0 PAGES
      P1BR            812D7A00
                                    P1 PTE BASE (VIRT ADDRS)
      P1LR            001FF77D
                                    TOTAL NON-EXISTENT P1 PAGES
      SBR             07DB2400
                                    SYSTEM PTE BASE (PHYS ADDRS)
      SLR             00093680
                                    TOTAL PAGES "SYSTEM" VIRT MEM
      PCBB            06B36420
                                    PCB BASE (PHYS ADDRS)
      SCBB            07DAD200
                                    SCB BASE (PHYS ADDRS)
      ASTLVL          00000004
                                    NO AST'S PENDING
      SISR            00000000
                                    INTERRUPT REQUEST ACTIVE = 0.
      ICCS            00000000
```

DEC 7000-600 Errorlog Examples

DEC 7000-600 System Hard Error Examples (Vector 660)

Example C–1 DEC 7000-600 Hard Error (Example 1)

```
******************************* ENTRY     225. *******************************
ERROR SEQUENCE 19.                          LOGGED ON:  CPU_TYPE 00000002
DATE/TIME  2-FEB-1993 18:39:48.97                        SYS_TYPE 00000003
SYSTEM UPTIME: 0 DAYS 00:10:05
SCS NODE: CLYP01                                          VMS T1.5-FT3

INT60 ERROR KN7AA DEC 7000 MODEL 640
REVISION        00000001
SW FLAGS        10000001
20800000
00000000
00000001
Bystander CE
IOP LSB Inconsistent state
Inconsistent 660 Error
LSB error log snapshot packet present
LOGGING OFF     00000000
00000000
00000000
00000000
ACTIVE CPUS     0000000F
HW REVISION     00000000
SYS SERIAL NUM  00000000
00000000
0000
SYS SERIAL NUM = ..........
SERIAL NUMBER   00000000
00000000
0000
SERIAL NUMBER = ..........
RESRC DISABLE   00000000

660 MACHINE CHECK FRAME
RETRY/BYTE CNT  80000000 000001D8
BYTE COUNT = 000001D8(X)
CAN'T RETRY
PAL ERROR CODE  00000100
PAL REVISION    00000001
PAL ERROR CODE  00000100
PAL REVISION    00000001
PALTEMP1        00000000 7FF78060
PALTEMP2        001744F8 00000004
PALTEMP3        00000000 00000000
PALTEMP4        00000000 000311E0
PALTEMP5        00000000 000310E0
PALTEMP6        00000000 001120B8
PALTEMP7        00000000 00000000
PALTEMP8        00000000 00000000
PALTEMP9        00000000 0000001B
PALTEMP10       000000AE E9117D8D
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
PALTEMP11        00000000 00000000
PALTEMP12        00000000 00100024
PALTEMP13        00000000 00000040
PALTEMP14        00000000 00000100
PALTEMP15        00000000 000310E0
PALTEMP16        00000000 00000030
PALTEMP17        00000000 00000040
PALTEMP18        00000000 00000100
PALTEMP19        00000000 00000000
PALTEMP20        00000000 00000023
PALTEMP21        FFFFFFFF 80649FB6
PALTEMP22        00000000 00000000
PALTEMP23        00000000 00000000
PALTEMP24        FFFFFFFF 80649B00
PALTEMP25        00000000 00090000
PALTEMP26        00000000 7FF96000
PALTEMP27        00000000 00000003
PALTEMP28        00000000 124C0000
PALTEMP29        00000002 00000000
PALTEMP30        00000000 009C0000
PALTEMP31        00000000 022A0080
EXCP ADDR REG    00000000 000E15D8
Not PALmode instruction
EXECPTION PC   = 0000000000038576(X)
EXCP SUM REG     00000000 00000000
EXCP MASK REG    00000000 00000000
ICCS REG         00000002 E89F0000
PAL BASE         00000000 00008000
PAL BASE PA = 000008000(X)
HW INTR EN REG   00000001 FFFFDCE0
CRD ERROR INT. DISABLE
HARDWARE INT. ENABLED ON PIN 3
HARDWARE INT. ENABLED ON PIN 4
HARDWARE INT. ENABLED ON PIN 5
PC1 INT. DISABLED
PC0 INT. DISABLED
HARDWARE INT. ENABLED ON PIN 0
HARDWARE INT. ENABLED ON PIN 1
HARDWARE INT. ENABLED ON PIN 2
SLU INT. DISABLE
SOFTWARE INT. LEVEL 1 ENABLED
SOFTWARE INT. LEVEL 2 ENABLED
SOFTWARE INT. LEVEL 3 ENABLED
SOFTWARE INT. LEVEL 4 ENABLED
SOFTWARE INT. LEVEL 5 ENABLED
SOFTWARE INT. LEVEL 6 ENABLED
SOFTWARE INT. LEVEL 7 ENABLED
SOFTWARE INT. LEVEL 8 ENABLED
SOFTWARE INT. LEVEL 9 ENABLED
SOFTWARE INT. LEVEL 10 ENABLED
SOFTWARE INT. LEVEL 11 ENABLED
SOFTWARE INT. LEVEL 12 ENABLED
SOFTWARE INT. LEVEL 13 ENABLED
SOFTWARE INT. LEVEL 14 ENABLED
SOFTWARE INT. LEVEL 15 ENABLED
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
KERNAL MODE AST INT. ENABLED
EXEC. MODE AST INT. ENABLED
SUPER. MODE AST INT. ENABLED
USER MODE AST INT. ENABLED
HW INTR REQ REG 00000000 00000342
HW INTR. REQ
CPU INTR REQ. on pin 4
PC1 INTR REQ.
PC0 INTR REQ.
MEM MGMT ER/DTB 00000000 00005210
Integer Reg. used is R = 01(X)
OP code = 29(X)
D-CACHE STA REG 00000000 000002E0
D-CACHE ADD REG 00000007 FFFFFFFF
ABOX CTL REG    00000000 0000040E
MCHECK ENABLED for UNCOR. ERR
CRD INTR. ENABLE
ICACHE STREAM BUFFER ENABLED
DCACHE ENABLED
BIU STAT        00000000 00000050
DCACHE FILL ERROR
Bits 33,32 BIU Addr Reg = 0(X)
Bits 33,32 Fill Addr Reg = 0(X)
BIU ADD REG     00000000 000069E0
BIU CTL REG     00000008 50006447
External Cache Enable
ECC Checking
Output Enable of Cache RAMs
BCache Read Speed in cycles = 5(X)
BCache Write Speed in cycles = 5(X)
ECC SYNDROMES   00000000 00000000
FILL ADDR REG   00000000 00006A20
MACHINE CHK VA  00000000 00006A90
B-CACHE TAG REG 180A8060 C05C0032
TAG Control P
TAG Control V
B-Cache TAG  = 6001(X)
GB HALT             0038
GB INTR             0000
GB PMASK            0001
halt enable
Select UART0A local terminal
GB WHAMI            0023
LSB Node ID = 3(X)
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
lMERR           00000000
STARTING ADDRESS  = 7FE6F378(X)
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          0000000C
Syndrome 1 = 0C(X)
LBESR2          0000000C
Syndrome 2 = 0C(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00440030
Command Address =  0000440030
LBECR1          00009060
CNF WAS ASSERTED FOR THIS COMMAND
Cmder ID REQ<3:0> = 2(X)
Data Cycle Error = 0(X)
LMODE           00010010
4M-Byte
Use PMAP lookup
EV4 8k 1 set I-Cache=8/16k VIC
Lockout is off (ignored)
Pass 2 LEVI-A
LOCK ADDR REG   0015B8E8
Lock Address = 0015B8E8(X)
ENDING ADDRESS  = 7FE6F378(X)
PAL REV         FFFFFFFF 00010270

ERROR COUNTERS

Bystander CE          1
IOP LSB Incon state   1
Incon 660             2

LSB SUBPACKET
NODES PRESENT   000001FF
PHYS ADDR       00000003 F8C00000
Node Present = 0.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8000000
LDEV            00008001
Laser EV4 Processor, 4M
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Correctable Err Detection
Self Test Passed
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
LBESR0          0000000E
Syndrome 0 = 0E(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          00000000
Syndrome 2 = 00(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          002E61F8
LBECR1          00019040
Node Present = 1.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8400000
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          0000000C
Syndrome 0 = 0C(X)
LBESR1          0000000C
Syndrome 1 = 0C(X)
LBESR2          0000000C
Syndrome 2 = 0C(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          000006F2
LBECR1          00008020
Node Present = 2.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8800000
LDEV            000B8001
EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          0000000C
Syndrome 0 = 0C(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000000C
Syndrome 2 = 0C(X)
LBESR3          0000000C
Syndrome 3 = 0C(X)
LBECR0          00420062
LBECR1          00018800
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
Node Present = 3.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8C00000
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          0000000C
Syndrome 1 = 0C(X)
LBESR2          0000000C
Syndrome 2 = 0C(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00440030
LBECR1          00009060
Node Present = 4.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9000000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000040
Syndrome 0 = 40(X)
LBESR1          00000040
Syndrome 1 = 40(X)
LBESR2          00000040
Syndrome 2 = 40(X)
LBESR3          00000040
Syndrome 3 = 40(X)
LBECR0          0000E06A
LBECR1          00000440
Node Present = 5.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9400000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
Self Test Passed
LBESR0          0000000C
Syndrome 0 = 0C(X)
LBESR1          0000000C
Syndrome 1 = 0C(X)
LBESR2          0000000C
Syndrome 2 = 0C(X)
LBESR3          0000000C
Syndrome 3 = 0C(X)
LBECR0          0000E06A
LBECR1          00000440
Node Present = 6.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9800000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000040
Syndrome 0 = 40(X)
LBESR1          00000040
Syndrome 1 = 40(X)
LBESR2          00000040
Syndrome 2 = 40(X)
LBESR3          00000040
Syndrome 3 = 40(X)
LBECR0          0000E06A
LBECR1          00000440
Node Present = 7.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9C00000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          0000007F
Syndrome 0 = 7F(X)
LBESR1          0000007F
Syndrome 1 = 7F(X)
LBESR2          0000007F
Syndrome 2 = 7F(X)
LBESR3          0000007F
Syndrome 3 = 7F(X)
LBECR0          0000E06A
LBECR1          00000440
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
Node Present = 8.
LSB IOP SUBPACKET
PHYS ADDR       00000003 FA000000
LDEV            00022000
Laser IO Module
Device Revision = 02(X)
LBER            00000001
ERROR LINE ASSERTED
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
1MERR           00000000
**Register contents not valid**
LBESR0          0000000C
Syndrome 0 = 0C(X)
LBESR1          0000000C
Syndrome 1 = 0C(X)
LBESR2          0000000C
Syndrome 2 = 0C(X)
LBESR3          0000000C
Syndrome 3 = 0C(X)
LBECR0          00500036
LBECR1          000081E0

********************************* ENTRY     226. *********************************
ERROR SEQUENCE 20.                                LOGGED ON:  CPU_TYPE 00000002
DATE/TIME  2-FEB-1993 18:39:48.97                              SYS_TYPE 00000003
SYSTEM UPTIME: 0 DAYS 00:10:05
SCS NODE: CLYP01                                               VMS T1.5-FT3

FATAL BUGCHECK KN7AA

MACHINECHK, Machine check while in kernel mode

PROCESS NAME    UPL_FFT2D00000
PROCESS ID      00010035
ERROR PC        FFFFFFFF 800422A0

Process Status = 00000000 00001F04, SW = 00, Previous Mode = KERNEL
System State = 01, Current Mode = KERNEL
VMM = 00 IPL = 31, SP Alignment = 0

STACK POINTERS

KSP 00000000 7FF95E40  ESP 00000000 7FF9A000  SSP 00000000 7FFA0500
USP 00000000 7FE6F9C0

GENERAL REGISTERS

R0  00000000 00000001  R1  00000000 00000001  R2  00000000 000001D8
R3  FFFFFFFF 804284C8  R4  00000000 00000001  R5  FFFFFFFF 804249F0
R6  FFFFFFFF FFFFFFFF  R7  FFFFFFFF 80425800  R8  00000000 000001D8
R9  FFFFFFFF 804257F8  R10 00000000 00000000  R11 00000000 00000000
R12 00000000 00000000  R13 00000000 00000001  R14 00000000 00000210
R15 00000000 00000014  R16 00000000 00000214  R17 00000000 00000001
R18 FFFFFFFF 870664F0  R19 00000000 00000000  R20 00000000 00000000
R21 FFFFFFFF 87066300  R22 FFFFFFFF 804249F0  R23 FFFFFFFF 804257F0
R24 FFFFFFFF 804257F0  R25 00000000 00000001  R26 FFFFFFFF 80041CA8
R27 00008020 000006F2  R28 00051104 00010010  FP  00000000 7FF95E40
SP  00000000 7FF95E40  PC  FFFFFFFF 800422A0  PS  00000000 00001F04
```

Example C–1 Cont'd on next page

Example C–1 (Continued) DEC 7000-600 Hard Error (Example 1)

```
SYSTEM REGISTERS

PTBR            00000000 000016F9
Page Table Base Register
PCBB            00000000 05406080
Privileged Context Block Base
PRBR            FFFFFFFF 80648E80
Processor Base Register
SCBB            00000002 00000000
System Control Block Base
SISR            00000000 000004E0
Software Interrupt Summary Register
ASN             00000000 00000000
Address Space Number
ASTSR_ASTEN     00000000 00000007
AST Summary/AST Enable
AT              00000000 0000000F
Absolute Time
FEN             00000000 00000001
Floating-Point Enable
IPL             00000000 0000001F
Interrupt Priority Level
MCES            00000000 00000008
Machine Check Error Summary
```

Example C–2 DEC 7000-600 Hard Error (Example 2)

```
******************************** ENTRY     235. ********************************
ERROR SEQUENCE 51.                              LOGGED ON:  CPU_TYPE 00000002
DATE/TIME  2-FEB-1993 22:36:18.73                           SYS_TYPE 00000003
SYSTEM UPTIME: 0 DAYS 03:54:07
SCS NODE: CLYP01                                             VMS T1.5-FT3

INT60 ERROR KN7AA DEC 7000 MODEL 640
REVISION         00000001
SW FLAGS         10000001
20000000
00000000
00000001
Bystander CE
Inconsistent 660 Error
LSB error log snapshct packet present
LOGGING OFF      00000000
00000000
00000000
00000000
ACTIVE CPUS      0000000F
HW REVISION      00000000
SYS SERIAL NUM   00000000
00000000
0000
SYS SERIAL NUM = ..........
SERIAL NUMBER    00000000
00000000
0000
SERIAL NUMBER = ..........
RESRC DISABLE    00000000

660 MACHINE CHECK FRAME
RETRY/BYTE CNT   80000000 000001D8
BYTE COUNT = 000001D8(X)
CAN'T RETRY
PAL ERROR CODE   00000100
PAL REVISION     00000001
PAL ERROR CODE   00000100
PAL REVISION     00000001
PALTEMP1         00000000 00000200
PALTEMP2         000000F8 00000004
PALTEMP3         00000000 00000000
PALTEMP4         00000000 000013BE
PALTEMP5         00000000 00000002
PALTEMP6         FFFFFFFF 86F1A000
PALTEMP7         00000000 00000000
PALTEMP8         00000000 00000000
PALTEMP9         00000000 00000301
PALTEMP10        0000030A 3DF6A83E
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
PALTEMP11       00000000 00000000
PALTEMP12       00000000 00000000
PALTEMP13       FFFFFFFF 80433BD0
PALTEMP14       FFFFFFFF 80403400
PALTEMP15       FFFFFFFF 8052B7C0
PALTEMP16       00000000 00000001
PALTEMP17       0000FFF0 00007E00
PALTEMP18       FFFFFFFF 80403400
PALTEMP19       00000000 00000000
PALTEMP20       00000000 00000001
PALTEMP21       00000000 00000002
PALTEMP22       FFFFFFFF 80403400
PALTEMP23       00000000 00000041
PALTEMP24       FFFFFFFF 80610000
PALTEMP25       00000000 00090000
PALTEMP26       FFFFFFFF 86E46000
PALTEMP27       00000000 00000000
PALTEMP28       00000000 001F4000
PALTEMP29       00000002 00000000
PALTEMP30       00000000 009C0000
PALTEMP31       00000000 01110080
EXCP ADDR REG   FFFFFFFF 8007875C
Not PALmode instruction
EXECPTION PC  = FFFFFFFC2001E1D7(X)
EXCP SUM REG    00000000 00000000
EXCP MASK REG   00000000 00000000
ICCS REG        00000000 001F0000
PAL BASE        00000000 00008000
PAL BASE PA = 000008000(X)
HW INTR EN REG  00000000 1FFE1CE0
CRD ERROR INT. DISABLE
HARDWARE INT. ENABLED ON PIN 3
HARDWARE INT. ENABLED ON PIN 4
HARDWARE INT. ENABLED ON PIN 5
PC1 INT. DISABLED
PC0 INT. DISABLED
HARDWARE INT. ENABLED ON PIN 0
HARDWARE INT. ENABLED ON PIN 1
HARDWARE INT. ENABLED ON PIN 2
SLU INT. DISABLE
SOFTWARE INT. LEVEL 4 ENABLED
SOFTWARE INT. LEVEL 5 ENABLED
SOFTWARE INT. LEVEL 6 ENABLED
SOFTWARE INT. LEVEL 7 ENABLED
SOFTWARE INT. LEVEL 8 ENABLED
SOFTWARE INT. LEVEL 9 ENABLED
SOFTWARE INT. LEVEL 10 ENABLED
SOFTWARE INT. LEVEL 11 ENABLED
SOFTWARE INT. LEVEL 12 ENABLED
SOFTWARE INT. LEVEL 13 ENABLED
SOFTWARE INT. LEVEL 14 ENABLED
SOFTWARE INT. LEVEL 15 ENABLED
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
HW INTR REQ REG 00000000 00000342
HW INTR. REQ
CPU INTR REQ. on pin 4
PC1 INTR REQ.
PC0 INTR REQ.
MEM MGMT ER/DTB 00000000 00005BF1
D-stream ref WRITE ERR
Integer Reg. used is R = 1F(X)
OP code = 2D(X)
D-CACHE STA REG 00000000 000002E0
D-CACHE ADD REG 00000007 FFFFFFFF
ABOX CTL REG    00000000 0000040E
MCHECK ENABLED for UNCOR. ERR
CRD INTR. ENABLE
ICACHE STREAM BUFFER ENABLED
DCACHE ENABLED
BIU STAT        00000000 00000250
FILL ECC ERROR WAS CORRECTABLE
DCACHE FILL ERROR
CMD = UNKNOWN CMD
Bits 33,32 BIU Addr Reg = 0(X)
33,32 Fill Addr Reg = 0(X)
BIU ADD REG     00000000 000060E0
BIU CTL REG     00000008 50006447
External Cache Enable
ECC Checking
Output Enable of Cache RAMs
BCache Read Speed in cycles = 5(X)
BCache Write Speed in cycles = 5(X)
ECC SYNDROMES   00000000 00000000
FILL ADDR REG   00000000 00006120
MACHINE CHK VA  00000000 00006190
B-CACHE TAG REG 02400A24 00804000
B-Cache TAG  = 0200(X)
GB HALT             0038
GB INTR             0000
GB PMASK            0009
halt enable
Ctrl P Enable
Select UART0A local terminal
GB WHAMI            0020
LSB Node ID = 0(X)
LDEV            00008001
Laser EV4 Processor, 4M
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
1MERR           00000000
STARTING ADDRESS  = 7FE6F378(X)
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
Command Address =  0000135F90
LBECR1          00049818
CNF WAS ASSERTED FOR THIS COMMAND
Cmder ID REQ<3:0> = 3(X)
Data Cycle Error = 1(X)
LMODE           00010010
4M-Byte
Use PMAP lookup
EV4 8k 1 set I-Cache=8/16k VIC
Lockout is off (ignored)
Pass 2 LEVI-A
LOCK ADDR REG   000510E4
Lock Address = 000510E4(X)
ENDING ADDRESS  = 7FE6F378(X)
PAL REV         FFFFFFFF 00010270

ERROR COUNTERS

Bystander CE          1
Incon 660             1

LSB SUBPACKET
NODES PRESENT   000001FF
PHYS ADDR       00000003 F8000000
Node Present = 0.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8000000
LDEV            00008001
Laser EV4 Processor, 4M
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00049818
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
Node Present = 1.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8400000
LDEV            000B8001
Laser EV4 Processor, 4M .
Device Revision = 0B(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
IBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00049818
Node Present = 2.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8800000
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00049818
Node Present = 3.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8C00000
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00040209
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
TRANSMITTER DURING ERROR
NODE-SPECIFIC ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00049818
Node Present = 4.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9000000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00048198
Node Present = 5.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9400000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00048198
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
Node Present = 6.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9800000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER              00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00048198
Node Present = 7.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9C00000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00048198
Node Present = 8.
LSB IOP SUBPACKET
PHYS ADDR       00000003 FA000000
LDEV            00022000
Laser IO Module
Device Revision = 02(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
IMERR           00000000
**Register contents not valid**
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00135F90
LBECR1          00048198

******************************** ENTRY     236. ********************************
ERROR SEQUENCE 52.                              LOGGED ON:  CPU_TYPE 00000002
DATE/TIME  2-FEB-1993 22:36:18.73                            SYS_TYPE 00000003
SYSTEM UPTIME: 0 DAYS 03:54:07
SCS NODE: CLYP01                                              VMS T1.5-FT3

FATAL BUGCHECK KN/AA

MACHINECHK, Machine check while in kernel mode

PROCESS NAME    WHET00000
PROCESS ID      0001001E

ERROR PC        FFFFFFFF 800422A0

Process Status = 00000000 00001F04, SW = 00, Previous Mode = KERNEL
System State = 01, Current Mode = KERNEL
VMM = 00 IPL = 31, SP Alignment = 0

STACK POINTERS

KSP 00000000 7FF95E40  ESP 00000000 7FF9A000  SSP 00000000 7FFA0500
USP 00000000 7FE6FB40

GENERAL REGISTERS

R0  00000000 00000001  R1  00000000 00000001  R2  00000000 000001D8
R3  FFFFFFFF 804284C8  R4  00000000 00000003  R5  FFFFFFFF 80424DF0
R6  FFFFFFFF FFFFFFFF  R7  FFFFFFFF 80425800  R8  00000000 000001D8
R9  FFFFFFFF 804257F8  R10 00000000 00000000  R11 00000000 00000000
R12 00000000 00000000  R13 FFFFFFFF 86F0F4C0  R14 00000000 00000210
R15 00000000 7FF3FC5C  R16 00000000 00000214  R17 00000000 00000001
R18 FFFFFFFF 87086AF0  R19 00000000 00000000  R20 00000000 00000000
R21 FFFFFFFF 87086900  R22 FFFFFFFF 80424DF0  R23 FFFFFFFF 804257F0
R24 FFFFFFFF 804257F0  R25 00000000 00000001  R26 FFFFFFFF 80041CA8
R27 00049818 00135F90  R28 00051108 00010010  FP  00000000 7FF95E40
SP  00000000 7FF95E40  PC  FFFFFFFF 800422A0  PS  00000000 00001F04
```

Example C–2 Cont'd on next page

Example C–2 (Continued) DEC 7000-600 Hard Error (Example 2)

```
SYSTEM REGISTERS

PTBR            00000000 000012FE
Page Table Base Register
PCBB            00000000 025EE080
Privileged Context Block Base
PRBR            FFFFFFFF 80649B00
Processor Base Register
SCBB            00000002 00000000
System Control Block Base
SISR            00000000 000004E0
Software Interrupt Summary Register
ASN             00000000 00000000
Address Space Number
ASTSR_ASTEN     00000000 00000011
AST Summary/AST Enable
AT              00000000 0000000F
Absolute Time
FEN             00000000 00000000
Floating-Point Enable
IPL             00000000 0000001F
Interrupt Priority Level
MCES            00000000 00000008
Machine Check Error Summary
```

Example C–3 DEC 7000-600 Hard Error (Example 3)

```
******************************** ENTRY     245. ********************************
ERROR SEQUENCE 83.                              LOGGED ON:  CPU_TYPE 00000002
DATE/TIME  3-FEB-1993 02:37:04.62                           SYS_TYPE 00000003
SYSTEM UPTIME: 0 DAYS 03:59:26
SCS NODE: CLYP01                                            VMS T1.5-FT3

INT60 ERROR KN7AA DEC 7000 MODEL 640
REVISION        00000001
SW FLAGS        80010001
00000000
00000000
00000001
LSB Error
LSB error log snapshot packet present
LOGGING OFF     00000000
00000000
00000000
00000000
ACTIVE CPUS     0000000F
HW REVISION     00000000
SYS SERIAL NUM  00000000
00000000
0000
SYS SERIAL NUM = ..........
SERIAL NUMBER   00000000
00000000
0000
SERIAL NUMBER = ..........
RESRC DISABLE   00000000

660 MACHINE CHECK FRAME
RETRY/BYTE CNT  80000000 000001D8
BYTE COUNT = 000001D8(X)
CAN'T RETRY
PAL ERROR CODE  00000100
PAL REVISION    00000001
PAL ERROR CODE  00000100
PAL REVISION    00000001
PALTEMP1        00000000 7FF76510
PALTEMP2        0001C0F8 00000004
PALTEMP3        00000000 00000000
PALTEMP4        FFFFFFFF 806D75C0
PALTEMP5        00000000 00000064
PALTEMP6        00000000 7FF7A004
PALTEMP7        00000000 00000000
PALTEMP8        00000000 00000000
PALTEMP9        00000000 00000201
PALTEMP10       00000563 7E40F2CE
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
PALTEMP11       00000000 00000000
PALTEMP12       00000000 00000001
PALTEMP13       FFFFFFFF 8041D0C0
PALTEMP14       00000000 00000000
PALTEMP15       FFFFFFFF 8040C800
PALTEMP16       00000000 7FF57049
PALTEMP17       00000000 00000029
PALTEMP18       00000000 7FF57042
PALTEMP19       00000006 00000010
PALTEMP20       FFFFFFFF FFFFFFFC
PALTEMP21       00000000 00000000
PALTEMP22       FFFFFFFF 80403B68
PALTEMP23       00000000 00000000
PALTEMP24       FFFFFFFF 80649B00
PALTEMP25       00000000 00090000
PALTEMP26       00000C00 7FF96000
PALTEMP27       00000000 00000003
PALTEMP28       00000000 023FE000
PALTEMP29       00000002 00000000
PALTEMP30       00000000 009C0000
PALTEMP31       00000000 026C2080
EXCP ADDR REG   FFFFFFFF 80006AE8
Not PALmode instruction
EXECPTION PC  = FFFFFFFC20001ABA(X)
EXCP SUM REG    00000000 00000000
EXCP MASK REG   00000000 00000000
ICCS REG        00000000 381F0000
PAL BASE        00000000 00008000
PAL BASE PA = 000008000(X)
HW INTR EN REG  00000001 FFFF1CE0
CRD ERROR INT. DISABLE
HARDWARE INT. ENABLED ON PIN 3
HARDWARE INT. ENABLED ON PIN 4
HARDWARE INT. ENABLED ON PIN 5
PC1 INT. DISABLED
PC0 INT. DISABLED
HARDWARE INT. ENABLED ON PIN 0
HARDWARE INT. ENABLED ON PIN 1
HARDWARE INT. ENABLED ON PIN 2
SLU INT. DISABLE
SOFTWARE INT. LEVEL 3 ENABLED
SOFTWARE INT. LEVEL 4 ENABLED
SOFTWARE INT. LEVEL 5 ENABLED
SOFTWARE INT. LEVEL 6 ENABLED
SOFTWARE INT. LEVEL 7 ENABLED
SOFTWARE INT. LEVEL 8 ENABLED
SOFTWARE INT. LEVEL 9 ENABLED
SOFTWARE INT. LEVEL 10 ENABLED
SOFTWARE INT. LEVEL 11 ENABLED
SOFTWARE INT. LEVEL 12 ENABLED
SOFTWARE INT. LEVEL 13 ENABLED
SOFTWARE INT. LEVEL 14 ENABLED
SOFTWARE INT. LEVEL 15 ENABLED
KERNAL MODE AST INT. ENABLED
EXEC. MODE AST INT. ENABLED
SUPER. MODE AST INT. ENABLED
USER MODE AST INT. ENABLED
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
HW INTR REQ REG 00000000 00000342
HW INTR. REQ
CPU INTR REQ. on pin 4
PC1 INTR REQ.
PC0 INTR REQ.
MEM MGMT ER/DTB 00000000 00001F21
D-stream ref WRITE ERR
Integer Reg. used is R = 12(X)
OP code = 0F(X)
D-CACHE STA REG 00000000 000002E0
D-CACHE ADD REG 00000007 FFFFFFFF
ABOX CTL REG    00000000 0000040E
MCHECK ENABLED for UNCOR. ERR
CRD INTR. ENABLE
ICACHE STREAM BUFFER ENABLED
DCACHE ENABLED
BIU STAT        00000000 00000050
DCACHE FILL ERROR
Bits 33,32 BIU Addr Reg = 0(X)
Bits 33,32 Fill Addr Reg = 0(X)
BIU ADD REG     00000000 000069E0
BIU CTL REG     00000008 50006447
External Cache Enable
ECC Checking
Output Enable of Cache RAMs
BCache Read Speed in cycles = 5(X)
BCache Write Speed in cycles = 5(X)
ECC SYNDROMES   00000000 00000000
FILL ADDR REG   00000000 00006A20
MACHINE CHK VA  00000000 00006A90
B-CACHE TAG REG 0060C200 C1804000
B-Cache TAG  = 0200(X)
GB HALT             0038
GB INTR             0000
GB PMASK            0001
halt enable
Select UART0A local terminal
GB WHAMI            0023
LSB Node ID = 3(X)
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00040209
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
TRANSMITTER DURING ERROR
NODE-SPECIFIC ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
1MERR           00000080
B-CACHE DATA SINGLE BIT ERROR
STARTING ADDRESS  = 7FE6F378(X)
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
Command Address =  0000155F90
LBECR1          00049818
CNF WAS ASSERTED FOR THIS COMMAND
Cmder ID REQ<3:0> = 3(X)
Data Cycle Error = 1(X)
LMODE           00010010
4M-Byte
Use PMAP lookup
EV4 8k 1 set I-Cache=8/16k VIC
Lockout is off (ignored)
Pass 2 LEVI-A
LOCK ADDR REG   00051108
Lock Address = 00051108(X)
ENDING ADDRESS  = 7FE6F378(X)
PAL REV         FFFFFFFF 0001027C

ERROR COUNTERS

LSB Err             1

LSB SUBPACKET
NODES PRESENT   000001FF
PHYS ADDR       00000003 F8C00000
Node Present = 0.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8000000
LDEV            00008001
Laser EV4 Processor, 4M
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00049818
Node Present = 1.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8400000
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00000009
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00049818
Node Present = 2.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8800000
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00049818
Node Present = 3.
LSB CPU SUBPACKET
PHYS ADDR       00000003 F8C00000
LDEV            000B8001
Laser EV4 Processor, 4M
Device Revision = 0B(X)
LBER            00040209
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
TRANSMITTER DURING ERROR
NODE-SPECIFIC ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00049818
Node Present = 4.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9000000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00048198
Node Present = 5.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9400000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00048198
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
Node Present = 6.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9800000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00048198
Node Present = 7.
LSB MEMORY SUBPACKET
PHYS ADDR       00000003 F9C00000
LDEV            00004000
Laser Memory Module
Device Revision = 00(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
Self Test Passed
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00048198
Node Present = 8.
LSB IOP SUBPACKET
PHYS ADDR       00000003 FA000000
LDEV            00022000
Laser IO Module
Device Revision = 02(X)
LBER            00000009
ERROR LINE ASSERTED
CORRECTABLE DATA ERROR
LCNR            00000001
Enable Correctable Err Detection
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
Self Test Passed
lMERR           00000000
**Register contents not valid**
LBESR0          00000000
Syndrome 0 = 00(X)
LBESR1          00000000
Syndrome 1 = 00(X)
LBESR2          0000004F
Syndrome 2 = 4F(X)
LBESR3          00000000
Syndrome 3 = 00(X)
LBECR0          00155F90
LBECR1          00048198

******************************** ENTRY     246. ********************************
ERROR SEQUENCE 84.                              LOGGED ON:  CPU_TYPE 00000002
DATE/TIME  3-FEB-1993 02:37:04.62                           SYS_TYPE 00000003
SYSTEM UPTIME: 0 DAYS 03:59:26
SCS NODE: CLYP01                                             VMS T1.5-FT3

FATAL BUGCHECK KN7AA

MACHINECHK, Machine check while in kernel mode

PROCESS NAME    NULL
PROCESS ID      00010000

ERROR PC        FFFFFFFF 800422A0

Process Status = 00000000 00001F04, SW = 00, Previous Mode = KERNEL
System State = 01, Current Mode = KERNEL
VMM = 00 IPL = 31, SP Alignment = 0

STACK POINTERS

KSP FFFFFFFF 87071DC0  ESP FFFFFFFF 8706D000  SSP FFFFFFFF 87069000
USP FFFFFFFF 87069000

GENERAL REGISTERS

R0  00000000 00000001  R1  00000000 00000001  R2  00000000 000001D8
R3  FFFFFFFF 804284C8  R4  00000000 00000002  R5  FFFFFFFF 80424BF0
R6  FFFFFFFF FFFFFFFF  R7  FFFFFFFF 80425800  R8  00000000 000001D8
R9  FFFFFFFF 804257F8  R10 00000000 00000000  R11 00000000 00000000
R12 00000000 00000000  R13 FFFFFFFF 80433BD0  R14 00000000 00000210
R15 00000000 59535B08  R16 00000000 00000214  R17 00000000 00000001
R18 FFFFFFFF 870767F0  R19 00000000 00000000  R20 00000000 00000000
R21 FFFFFFFF 87076600  R22 FFFFFFFF 80424BF0  R23 FFFFFFFF 804257F0
R24 FFFFFFFF 804257F0  R25 00000000 00000001  R26 FFFFFFFF 80041CA8
R27 00049818 00155F90  R28 00051104 00010010  FP  FFFFFFFF 87071DC0
SP  FFFFFFFF 87071DC0  PC  FFFFFFFF 800422A0  PS  00000000 00001F04
```

Example C–3 Cont'd on next page

Example C–3 (Continued) DEC 7000-600 Hard Error (Example 3)

```
SYSTEM REGISTERS

PTBR            00000000 000000FA
Page Table Base Register
PCBB            00000000 01149580
Privileged Context Block Base
PRBR            FFFFFFFF 80649500
Processor Base Register
SCBB            00000002 00000000
System Control Block Base
SISR            00000000 000004E0
Software Interrupt Summary Register
ASN             00000000 00000000
Address Space Number
ASTSR_ASTEN     00000000 00000000
AST Summary/AST Enable
AT              00000000 00000000
Absolute Time
FEN             00000000 00000000
Floating-Point Enable
IPL             00000000 0000001F
Interrupt Priority Level
MCES            00000000 00000008
Machine Check Error Summary
```

DEC 7000-600 System Power Event Example

Example C–4 DEC 7000-600 System Power Event

```
******************************** ENTRY       9. ********************************
ERROR SEQUENCE 7.                              LOGGED ON:  CPU_TYPE 00000002
DATE/TIME 27-JAN-1993 16:31:51.12                          SYS_TYPE 00000003
SYSTEM UPTIME: 0 DAYS 00:05:00
SCS NODE:                                                   VMS T1.5-FT3

POWER KN7AA DEC 7000 MODEL 610
PWR REG SUMMARY 00000003
REVISION        35302E31 58202020
PWR EVENT TYPE        02
History Packet
Center Cabinet Regulator A
IDENTIFICATION        41
A
RANGE                 4C
L
REVISION        31313331
1311
PEAK AC         30303030
0000
DC BULK V       30303030
0000
48V DC BUS V    30333830
0830
48V DC BUS C    30333130
0130
48V BAT PACK V  30303030
0000
24V BAT PACK V  30303030
0000
BAT PACK CHR C  30303030

AMBIENT TEMP    39313530
0519
ELPS RUN TIME   39303430
0409
REM BAT CAP         3030
00
BAT CUTOFF CNTR   303030
000
BAT CONFIG            30
0
HEATSINK STS          30
0
BAT PAK STATE         30
0
TEST STATUS           30
0
PWR SUP STATE         34
4
CHECKSUM            3232
```

Example C–4 Cont'd on next page