digital

DIGITAL Semiconductor AlphaPC 164LX Motherboard

Technical Reference Manual

Order Number: EC–R46WB–TE

Revision/Update Information: This is a revised, preliminary document. It supersedes the *DIGITAL Semiconductor AlphaPC 164LX Motherboard Technical Reference Manual* (EC–R46WA–TE).

Preliminary

Digital Equipment Corporation
Maynard, Massachusetts

http://www.digital.com/semiconductor

January 1998

Contents

Preface

1 Introduction

1.1 System Components and Features 1–1
1.1.1 DIGITAL Semiconductor 21174 Core Logic Chip 1–3
1.1.2 Memory Subsystem 1–3
1.1.3 L3 Bcache Subsystem Overview 1–4
1.1.4 PCI Interface Overview 1–4
1.1.5 ISA Interface Overview 1–4
1.1.6 Miscellaneous Logic 1–4
1.2 Software Support 1–5
1.2.1 AlphaBIOS Windows NT Firmware 1–5
1.2.2 Alpha SRM Console Firmware 1–5
1.2.3 Motherboard Software Developer's Kit (SDK) 1–6
1.3 Hardware Design Support 1–6

2 System Configuration and Connectors

2.1 AlphaPC 164LX Configuration Jumpers 2–4
2.2 CPU Speed Selection 2–5
2.3 Bcache Size Jumpers (CF1 and CF2) 2–5
2.4 Password Bypass Jumper (CF3) 2–5
2.5 Boot Option Jumper (CF7) 2–5
2.6 Flash ROM Update Jumper (J28) 2–6
2.7 AlphaPC 164LX Connector Pinouts 2–6
2.7.1 PCI Bus Connector Pinouts 2–6
2.7.2 ISA Expansion Bus Connector Pinouts 2–8
2.7.3 SDRAM DIMM Connector Pinouts 2–9
2.7.4 EIDE Drive Bus Connector Pinouts 2–10
2.7.5 Diskette Drive Bus Connector Pinouts 2–11
2.7.6 Parallel Bus Connector Pinouts 2–11

2.7.7 COM1/COM2 Serial Line Connector Pinouts 2–12
2.7.8 Keyboard/Mouse Connector Pinouts 2–12
2.7.9 SROM Test Data Input Connector Pinouts 2–13
2.7.10 Input Power Connector Pinouts 2–13
2.7.11 Enclosure Fan Power Connector Pinouts 2–13
2.7.12 Speaker Connector Pinouts 2–14
2.7.13 Microprocessor Fan Power Connector Pinouts 2–14
2.7.14 Power LED Connector Pinouts 2–14
2.7.15 IDE Drive LED Connector Pinouts 2–15
2.7.16 Reset Button Connector Pinouts 2–15
2.7.17 Halt Button Connector Pinouts 2–15
2.7.18 Soft Power Connector Pinouts 2–15

3 Power and Environmental Requirements

3.1 Power Requirements 3–1
3.2 Environmental Requirements 3–2
3.3 Board Dimensions 3–2
3.3.1 ATX Hole Specification 3–3
3.3.2 ATX I/O Shield Requirements 3–4

4 Functional Description

4.1 AlphaPC 164LX Bcache Interface 4–2
4.2 DIGITAL Semiconductor 21174 Core Logic Chip 4–3
4.2.1 21174 Chip Overview 4–4
4.2.2 Main Memory Interface 4–4
4.2.3 PCI Devices 4–5
4.2.4 Saturn-IO (SIO) Chip 4–6
4.2.5 PCI Expansion Slots 4–7
4.3 ISA Bus Devices 4–7
4.3.1 Combination Controller 4–7
4.3.2 Utility Bus Memory Device 4–9
4.3.3 ISA Expansion Slots 4–9
4.3.4 ISA I/O Address Map 4–9
4.3.5 Flash ROM Address Map 4–10
4.4 Interrupts 4–10
4.4.1 Interrupt PLD Function 4–14
4.5 System Clocks 4–15
4.6 Reset and Initialization 4–17
4.7 Serial ROM 4–18
4.8 DC Power Distribution 4–19

5 Upgrading the AlphaPC 164LX

5.1 Configuring SDRAM Memory 5–1
5.2 Upgrading SDRAM Memory 5–2
5.3 Increasing Microprocessor Speed. 5–2
5.3.1 Preparatory Information 5–3
5.3.2 Required Tools 5–3
5.3.3 Removing the 21164 Microprocessor 5–3
5.3.4 Installing the 21164 Microprocessor 5–4

A System Address Space

A.1 Address Map A–1
A.2 PCI Address Space. A–6
A.3 21164 Address Space. A–7
A.3.1 System Address Map. A–10
A.4 21164 Byte/Word PCI Space A–12
A.4.1 21164 Size Field A–14
A.5 Cacheable Memory Space A–15
A.6 PCI Dense Memory Space A–15
A.7 PCI Sparse Memory Space A–17
A.7.1 Hardware Extension Register (HAE_MEM). A–18
A.7.2 Memory Access Rules and Operation A–18
A.8 PCI Sparse I/O Space. A–23
A.8.1 Hardware Extension Register (HAE_IO) A–23
A.8.2 PCI Sparse I/O Space Access Operation A–23
A.9 PCI Configuration Space A–26
A.10 PCI Special/Interrupt Cycles A–31
A.11 Hardware-Specific and Miscellaneous Register Space A–31
A.12 PCI to Physical Memory Address A–32
A.13 Direct-Mapped Addressing A–37
A.14 Scatter-Gather Addressing A–38
A.15 Scatter-Gather TLB. A–40
A.15.1 Scatter-Gather TLB Hit Process A–42
A.15.2 Scatter-Gather TLB Miss Process A–42
A.16 Suggested Use of a PCI Window A–44
A.16.1 Peripheral Component Architecture Compatibility Addressing and Holes. . . A–45
A.16.2 Memory Chip Select Signal mem_cs_l A–45

B Supporting Products

B.1 Memory B–1
B.2 Thermal Products B–3
B.3 Power Supply B–3

B.4 Enclosure B–3

C Support, Products, and Documentation

Index

Figures

1–1	AlphaPC 164LX Functional Block Diagram	1–2
2–1	AlphaPC 164LX Jumper/Connector/Component Location	2–2
2–2	AlphaPC 164LX Configuration Jumpers	2–4
3–1	ATX Hole Specification	3–3
3–2	ATX I/O Shield Dimensions	3–4
4–1	AlphaPC 164LX L3 Bcache Array	4–2
4–2	Main Memory Interface	4–3
4–3	AlphaPC 164LX PCI Bus Devices	4–5
4–4	AlphaPC 164LX ISA Bus Devices	4–8
4–5	Interrupt Logic	4–11
4–6	Interrupt/Interrupt Mask Registers	4–14
4–7	AlphaPC 164LX System Clocks	4–16
4–8	System Reset and Initialization	4–18
4–9	Serial ROM	4–19
4–10	AlphaPC 164LX Power Distribution	4–21
5–1	Fan/Heat-Sink Assembly	5–5
A–1	Address Space Overview	A–5
A–2	Memory Remapping	A–6
A–3	21164 Address Space Configuration	A–8
A–4	21164 and DMA Read and Write Transactions	A–9
A–5	System Address Map	A–11
A–6	21174 CSR Space	A–12
A–7	Byte/Word PCI Space	A–13
A–8	Dense-Space Address Generation	A–17
A–9	PCI Memory Sparse-Space Address Generation – Region 1	A–21
A–10	PCI Memory Sparse-Space Address Generation – Region 2	A–22
A–11	PCI Memory Sparse-Space Address Generation – Region 3	A–22
A–12	PCI Sparse I/O Space Address Translation (Region A, Lower 32MB)	A–25
A–13	PCI Sparse I/O Space Address Translation (Region B, Higher Area)	A–25
A–14	PCI Configuration Space Definition (Sparse)	A–27
A–15	PCI Configuration Space Definition (Dense)	A–27
A–16	PCI Bus Hierarchy	A–30
A–17	PCI DMA Addressing Example	A–35
A–18	PCI Target Window Compare	A–36
A–19	Scatter-Gather PTE Format	A–39
A–20	Scatter-Gather Associative TLB	A–41
A–21	Scatter-Gather Map Translation	A–43
A–22	Default PCI Window Allocation	A–44
A–23	mem_cs_l Decode Area	A–46
A–24	mem_cs_l Logic	A–47

Tables

1–1	AlphaPC 164LX SDRAM Memory Configurations	1–3
2–1	AlphaPC 164LX Jumper/Connector/Component List	2–3
2–2	PCI Bus Connector Pinouts	2–6
2–3	ISA Expansion Bus Connector Pinouts (J30, J31)	2–8
2–4	SDRAM DIMM Connector Pinouts (J8 through J11)	2–9
2–5	EIDE Drive Bus Connector Pinouts (J6, J7)	2–10
2–6	Diskette (Floppy) Drive Bus Connector Pinouts (J15)	2–11
2–7	Parallel Bus Connector Pinouts (J13)	2–11
2–8	COM1/COM2 Serial Line Connector Pinouts (J4)	2–12
2–9	Keyboard/Mouse Connector Pinouts (J5)	2–12
2–10	SROM Test Data Input Connector Pinouts (J29)	2–13
2–11	Input Power Connector Pinouts (J3)	2–13
2–12	Enclosure Fan (+12 V dc) Power Connector Pinouts (J2, J19)	2–13
2–13	Speaker Connector Pinouts (J20)	2–14
2–14	Microprocessor Fan Power Connector Pinouts (J18)	2–14
2–15	Power LED Connector Pinouts (J24)	2–14
2–16	IDE Drive LED Connector Pinouts (J25)	2–15
2–17	Reset Button Connector Pinouts (J21)	2–15
2–18	Halt Button Connector Pinouts (J22)	2–15
2–19	Soft Power Connector Pinouts (J1)	2–15
3–1	Power Supply DC Current Requirements	3–1
3–2	AlphaPC 164LX Motherboard Environmental Requirements	3–2
4–1	ISA I/O Address Map	4–9
4–2	AlphaPC 164LX System Interrupts	4–12
4–3	ISA Interrupts	4–13
5–1	AlphaPC 164LX SDRAM Memory Configurations	5–1
A–1	Physical Address Map (Byte/Word Mode Disabled)	A–1
A–2	Physical Address Map (Byte/Word Mode Enabled)	A–2
A–3	21164 Byte/Word Addressing	A–14
A–4	21164 Byte/Word Translation Values	A–14
A–5	Int4_valid and 21164 Address Relationship	A–19
A–6	PCI Memory Sparse-Space Read/Write Encodings	A–20
A–7	PCI Address Mapping	A–21
A–8	PCI Sparse I/O Space Read/Write Encodings	A–24
A–9	CPU Address to IDSEL Conversion	A–28
A–10	PCI Configuration Space Read/Write Encodings	A–29
A–11	Hardware and Miscellaneous Address Map	A–31
A–12	PCI Target Window Mask Register Fields	A–33
A–13	Direct-Mapped PCI Target Address Translation	A–37
A–14	Scatter-Gather Mapped PCI Target Address Translation	A–39
A–15	PCI Window Power-Up Configuration	A–45

Preface

Overview

This manual describes the DIGITAL AlphaPC 164LX motherboard, a module for computing systems based on the DIGITAL Semiconductor Alpha 21164 microprocessor and the DIGITAL Semiconductor 21174 core logic chip.

Audience

This manual is intended for system designers and others who use the AlphaPC 164LX motherboard to design or evaluate computer systems based on the DIGITAL Semiconductor Alpha 21164 microprocessor and the DIGITAL Semiconductor 21174 core logic chip.

Scope

This manual describes the features, configuration, functional operation, and interfaces of the AlphaPC 164LX motherboard. This manual does not include specific bus specifications (for example, PCI or ISA buses). Additional information is available in the AlphaPC 164LX schematics, program source files, and the appropriate vendor and IEEE specifications. See Appendix C for information on how to order related documentation and obtain additional technical support.

Manual Organization

As outlined on the next page, this manual includes the following chapters, appendixes, and an index.

- Chapter 1, Introduction, is an overview of the AlphaPC 164LX motherboard, including its components, features, and uses.
- Chapter 2, System Configuration and Connectors, describes the user-environment configuration, board connectors and functions, and jumper functions. It also identifies jumper and connector locations.
- Chapter 3, Power and Environmental Requirements, describes the AlphaPC 164LX power and environmental requirements and provides board dimensions.
- Chapter 4, Functional Description, provides a functional description of the AlphaPC 164LX motherboard, including the 21174 core logic chip, L3 backup cache (Bcache) and memory subsystems, system interrupts, clock and power subsystems, and peripheral component interconnect (PCI) and Industry Standard Architecture (ISA) devices.
- Chapter 5, Upgrading the AlphaPC 164LX, describes how to upgrade the AlphaPC 164LX motherboard's DRAM memory and microprocessor speed.
- Appendix A, System Address Space, describes the mapping of the 40-bit processor address space into memory and I/O space addresses. It also lists the physical PCI address spaces and regions, including the 21174 operating registers and PCI/ISA device registers.
- Appendix B, Supporting Products, lists sources for components and accessories not included with the AlphaPC 164LX motherboard.
- Appendix C, Support, Products, and Documentation, describes how to obtain DIGITAL Semiconductor information and technical support, and how to order DIGITAL Semiconductor products and associated literature.

Conventions

This section defines product-specific terminology, abbreviations, and other conventions used throughout this manual.

Abbreviations

- Register Access

The following list describes the register bit and field abbreviations:

Bit/Field Abbreviation	Description
RO (read only)	Bits and fields specified as RO can be read but not written.
RW (read/write)	Bits and fields specified as RW can be read and written.
WO (write only)	Bits and fields specified as WO can be written but not read.

- Binary Multiples

  The abbreviations K, M, and G (kilo, mega, and giga) represent binary multiples and have the following values.

  K = 2^{10} (1024)
  M = 2^{20} (1,048,576)
  G = 2^{30} (1,073,741,824)

  For example:

  2KB = 2 kilobytes = 2×2^{10} bytes
  4MB = 4 megabytes = 4×2^{20} bytes
  8GB = 8 gigabytes = 8×2^{30} bytes

Addresses

Unless otherwise noted, all addresses and offsets are hexadecimal.

Bit Notation

Multiple-bit fields can include contiguous and noncontiguous bits contained in angle brackets (< >). Multiple contiguous bits are indicated by a pair of numbers separated by a colon (:). For example, <9:7,5,2:0> specifies bits 9,8,7,5,2,1, and 0. Similarly, single bits are frequently indicated with angle brackets. For example, <27> specifies bit 27.

Caution

Cautions indicate potential damage to equipment, software, or data.

Data Field Size

The term INT*nn*, where *nn* is one of 2, 4, 8, 16, 32, or 64, refers to a data field of *nn* contiguous NATURALLY ALIGNED bytes. For example, INT4 refers to a NATURALLY ALIGNED longword.

Data Units

The following data-unit terminology is used throughout this manual.

Term	Words	Bytes	Bits	Other
Byte	½	1	8	—
Word	1	2	16	—
Longword/Dword	2	4	32	Longword
Quadword	4	8	64	2 Longwords
Octaword	8	16	128	2 Quadwords
Hexword	16	32	256	2 Octawords

Note

Notes emphasize particularly important information.

Numbering

All numbers are decimal or hexadecimal unless otherwise indicated. The prefix 0x indicates a hexadecimal number. For example, 19 is decimal, but 0x19 and 0x19A are hexadecimal (also see Addresses). Otherwise, the base is indicated by a subscript; for example, 100_2 is a binary number.

Ranges and Extents

Ranges are specified by a pair of numbers separated by two periods (..) and are inclusive. For example, a range of integers 0..4 includes the integers 0, 1, 2, 3, and 4.

Extents are specified by a pair of numbers in angle brackets (< >) separated by a colon (:) and are inclusive. Bit fields are often specified as extents. For example, bits <7:3> specifies bits 7, 6, 5, 4, and 3.

Register and Memory Figures

Register figures have bit and field position numbering starting at the right (low order) and increasing to the left (high order).

Memory figures have addresses starting at the top and increasing toward the bottom.

Schematic References

Logic schematics are included in the AlphaPC 164LX design package. In this manual, references to schematic pages are printed in *italics*. For example, the following specifies schematic page 4:

“. . . the configuration jumpers (*pc164lx.4*) provide . . .”

Signal Names

All signal names are printed in boldface type. Signal names that originate in an industry-standard specification, such as PCI or IDE, are printed in the case as found in the specification (usually uppercase). Active-high signals are indicated by the **_h** suffix. Active-low signals have the **_l** suffix, a pound sign "#" appended, or a "not" overscore bar. Signals with no suffix are considered high-asserted signals. For example, signals **data_h<127:0>** and **cia_int** are active-high signals. Signals **mem_ack_l**, **FRAME#**, and **$\overline{\text{RESET}}$** are active-low signals.

UNPREDICTABLE and UNDEFINED

Throughout this manual the terms UNPREDICTABLE and UNDEFINED are used. Their meanings are quite different and must be carefully distinguished.

In particular, only privileged software (that is, software running in kernel mode) can trigger UNDEFINED operations. Unprivileged software cannot trigger UNDEFINED operations. However, either privileged or unprivileged software can trigger UNPREDICTABLE results or occurrences.

UNPREDICTABLE results or occurrences do not disrupt the basic operation of the processor. The processor continues to execute instructions in its normal manner. In contrast, UNDEFINED operations can halt the processor or cause it to lose information.

The terms UNPREDICTABLE and UNDEFINED can be further described as follows:

- UNPREDICTABLE
  - Results or occurrences specified as UNPREDICTABLE might vary from moment to moment, implementation to implementation, and instruction to instruction within implementations. Software can never depend on results specified as UNPREDICTABLE.
  - An UNPREDICTABLE result might acquire an arbitrary value that is subject to a few constraints. Such a result might be an arbitrary function of the input operands or of any state information that is accessible to the process in its current access mode. UNPREDICTABLE results may be unchanged from their previous values.

    Operations that produce UNPREDICTABLE results might also produce exceptions.

- An occurrence specified as UNPREDICTABLE may or may not happen based on an arbitrary choice function. The choice function is subject to the same constraints as are UNPREDICTABLE results and must not constitute a security hole.

  Specifically, UNPREDICTABLE results must not depend upon, or be a function of, the contents of memory locations or registers that are inaccessible to the current process in the current access mode.

  Also, operations that might produce UNPREDICTABLE results must not write or modify the contents of memory locations or registers to which the current process in the current access mode does not have access. They must also not halt or hang the system or any of its components.

  For example, a security hole would exist if some UNPREDICTABLE result depended on the value of a register in another process, on the contents of processor temporary registers left behind by some previously running process, or on a sequence of actions of different processes.

• UNDEFINED

  - Operations specified as UNDEFINED can vary from moment to moment, implementation to implementation, and instruction to instruction within implementations. The operation can vary in effect from nothing, to stopping system operation.
  - UNDEFINED operations can halt the processor or cause it to lose information. However, UNDEFINED operations must not cause the processor to hang, that is, reach an unhalted state from which there is no transition to a normal state in which the machine executes instructions. Only privileged software (that is, software running in kernel mode) can trigger UNDEFINED operations.

1
Introduction

This chapter provides an overview of DIGITAL Semiconductor's AlphaPC 164LX motherboard, including its components, features, and uses. The motherboard is a module for computing systems based on the DIGITAL Semiconductor 21174 core logic chip.

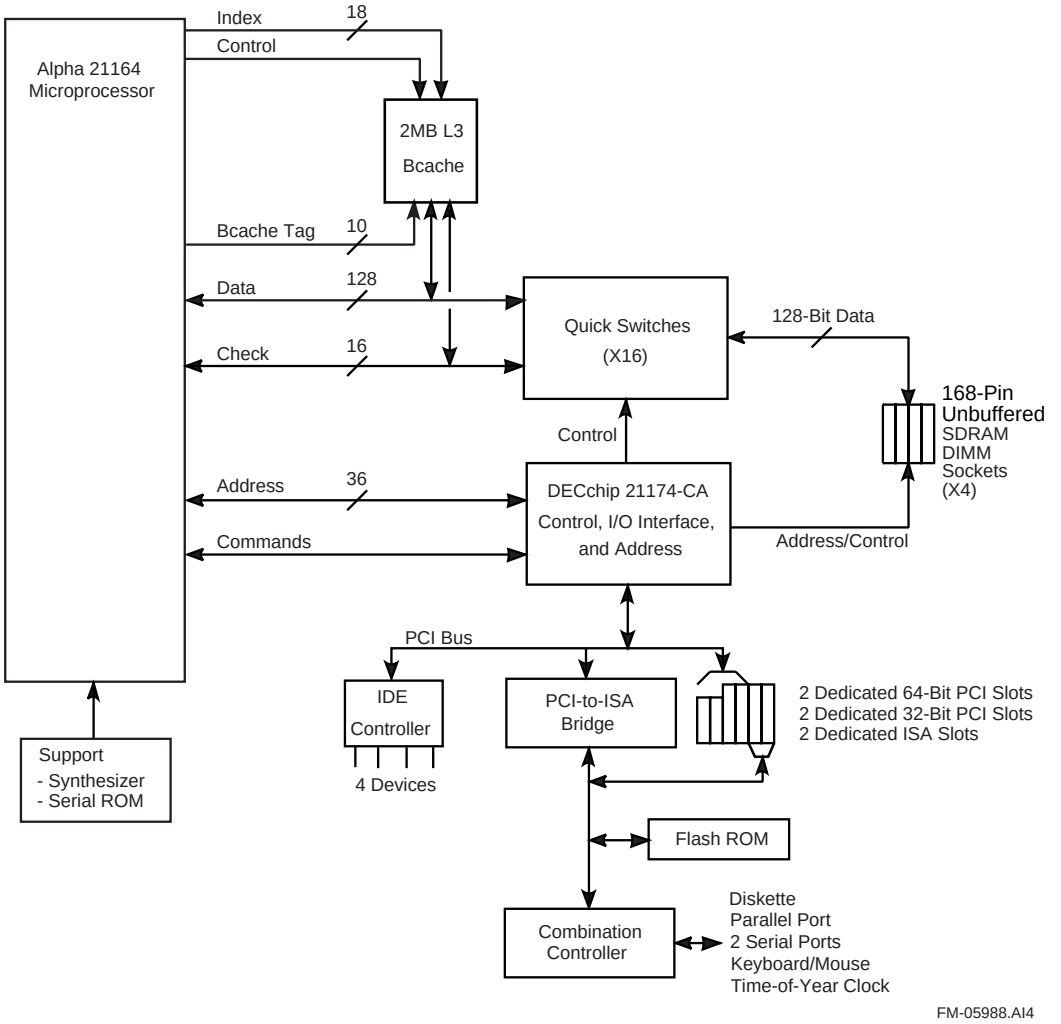
The AlphaPC 164LX provides a single-board hardware and software development platform for the design, integration, and analysis of supporting logic and subsystems. The board also provides a platform for PCI I/O device hardware and software development.

1.1 System Components and Features

The AlphaPC 164LX is implemented in industry-standard parts and uses a DIGITAL Semiconductor Alpha 21164 microprocessor running at 466, 533, and 600 MHz.

Figure 1–1 shows the board's functional components.

Figure 1–1 AlphaPC 164LX Functional Block Diagram



1.1.1 DIGITAL Semiconductor 21174 Core Logic Chip

The Alpha 21164 microprocessor is supported by the 21174 core logic chip, which provides an interface between three units—memory, the PCI bus, and the 21164 (along with flash ROM). This core logic chip is the interface between the 21164 microprocessor, main memory (addressing and control), and the PCI bus.

Sixteen quick switches provide the memory interface data path.

The 21174 includes the majority of functions necessary to develop a high-performance PC or workstation, requiring minimum discrete logic on the module. It provides flexible and generic functions to allow its use in a wide range of systems.

1.1.2 Memory Subsystem

The synchronous dynamic random-access memory (SDRAM) is contained in two banks of dual inline memory modules (DIMMs). Single- or double-sided DIMMs may be used. Each DIMM is 72 bits wide, with 64 data bits and 8 check bits, with 100 MHz or faster speed. Two DIMMs provide 32Mb to 256MB of memory, while four DIMMs provide up to 512MB. Table 1–1 lists the DIMM sizes tested and the corresponding main memory size for 128-bit arrays.

Table 1–1 AlphaPC 164LX SDRAM Memory Configurations

Total Memory	Bank 0 (J8 and J9)	Bank 1 (J10 and J11)
32MB	2Mb x 72	—
64MB	2Mb x 72	2Mb x 72
	4Mb x 72	—
96MB	4Mb x 72	2Mb x 72
128MB	4Mb x 72	4Mb x 72
	8Mb x 72	—
160MB	8Mb x 72	2Mb x 72
192MB	8Mb x 72	4Mb x 72
256MB	8Mb x 72	8Mb x 72
	16Mb x 72	—
512MB	16Mb x 72	16Mb x 72

1.1.3 L3 Bcache Subsystem Overview

The AlphaPC 164LX board-level L3 backup cache (Bcache) is a 2MB, direct-mapped, synchronous SRAM with a 128-bit data path. The board is capable of handling an L3 cache size of 4MB. See Section 2.3 for more information about the Bcache.

1.1.4 PCI Interface Overview

The AlphaPC 164LX PCI interface is the main I/O bus for the majority of functions (SCSI interface, graphics accelerator, and so on). The PCI interface has a 33-MHz data transfer rate. PCI-IDE support is provided by an onboard controller chip (CMD646). An onboard PCI-to-ISA bridge is provided through an Intel 82378ZB Saturn-I/O (SIO) chip.

The PCI bus has four dedicated PCI expansion slots (two 64-bit and two 32-bit).

1.1.5 ISA Interface Overview

The ISA bus provides the following system support functions:

- Two expansion slots.
- An SMC FDC37C935 combination controller chip that provides:
  - A mouse and keyboard controller
  - A diskette controller
  - Two universal asynchronous receiver-transmitters (UARTs) with full modem control
  - A bidirectional parallel port
  - A time-of-year (TOY) clock
- Operating system support—through a 1MB flash ROM that contains supporting firmware.

1.1.6 Miscellaneous Logic

The AlphaPC 164LX contains the following miscellaneous components:

- Synthesizer for clocks:
  - A clock synthesizer (MC12439) provides a programmable 466-, 533-, and 600-MHz clock source to the 21164 microprocessor. The microprocessor supplies a clock to the system PLL/clock buffer for the 21174.

  - The 21174 core logic chip provides the SDRAM and PCI clocks.
  - A 14.318-MHz crystal and frequency generator provide a clock source for the FDC37C935 ISA device controller. The controller's onchip generator then provides other clocks as needed.
  - A 32-kHz crystal provides the TOY clock source.
- Serial ROM – A Xilinx XC17128 serial ROM (SROM) contains initial code that is loaded into the 21164 instruction cache (Icache) on power-up. A serial line interface is also provided to allow direct connection to a terminal line for debugging purposes.
- Two AMD MACH210-15 programmable logic devices (PLDs) for interrupts and PCI bus arbitration.
- Altera EPM7032-7 for DMA boundary issue.

1.2 Software Support

The support elements described in this section are either included with the AlphaPC 164LX or are available separately.

1.2.1 AlphaBIOS Windows NT Firmware

The AlphaPC 164LX motherboard ships with AlphaBIOS firmware and online documentation that describes how to configure the firmware for Windows NT. This firmware initializes the system and enables you to install and boot the Windows NT operating system. The AlphaBIOS firmware resides in the flash ROM on the 21A04-C0 variation of the AlphaPC 164LX motherboard. Binary images of the AlphaBIOS firmware are included in the motherboard Software Developer's Kit (SDK), along with a license describing the terms for use and distribution.

1.2.2 Alpha SRM Console Firmware

The Alpha SRM Console firmware is required to install and boot DIGITAL UNIX on the AlphaPC 164LX. This DIGITAL Semiconductor firmware comes factory installed in the 21A04-C1 variation of the AlphaPC 164LX. When installed, this firmware occupies the flash blocks reserved for the primary firmware. Binary images of the Alpha SRM Console firmware are included in the SDK and Firmware Update compact disk, along with a license describing the terms for use and distribution.

1.2.3 Motherboard Software Developer's Kit (SDK)

The SDK and Firmware Update is designed to provide an environment for developing software for Alpha motherboard products. It is also specially suited for low-level software development and hardware debug for other Alpha microprocessor-based designs.

The following list includes some of the components of the SDK:

- The Alpha Motherboard Debug Monitor firmware with source code.
- Power-up initialization SROM and SROM Mini-Debugger with source code.
- Sample PALcode sources modeled after DIGITAL UNIX with source code.
- Fail-safe booter with source code.
- Various additional tools with source code.

The following development platforms are supported by the SDK:

- DIGITAL UNIX with the C Developer's Extensions.
- Windows NT (Alpha) with the Microsoft Visual C++ Development System for DIGITAL Alpha.
- Windows NT (Intel) with the Microsoft Visual C++ Development System and Tools provide limited support. This environment is currently useful for SROM and PALcode development only.

1.3 Hardware Design Support

The full design database, including schematics and source files, is supplied. User documentation is also included. The database allows designers with no previous Alpha architecture experience to successfully develop a working Alpha system with minimal assistance.

2
System Configuration and Connectors

This chapter describes the AlphaPC 164LX configuration, board connectors and functions, and jumper functions. It also identifies jumper and connector locations.

The AlphaPC 164LX uses jumpers to implement configuration parameters such as system speed and boot parameters. These jumpers must be configured for the user's environment. Onboard connectors are provided for the I/O interfaces, DIMMs, and serial and parallel peripheral ports.

Figure 2–1 shows the board outlines and identifies the location of jumpers, connectors, and major components. Table 2–1 lists and defines these items.

Figure 2–1 AlphaPC 164LX Jumper/Connector/Component Location

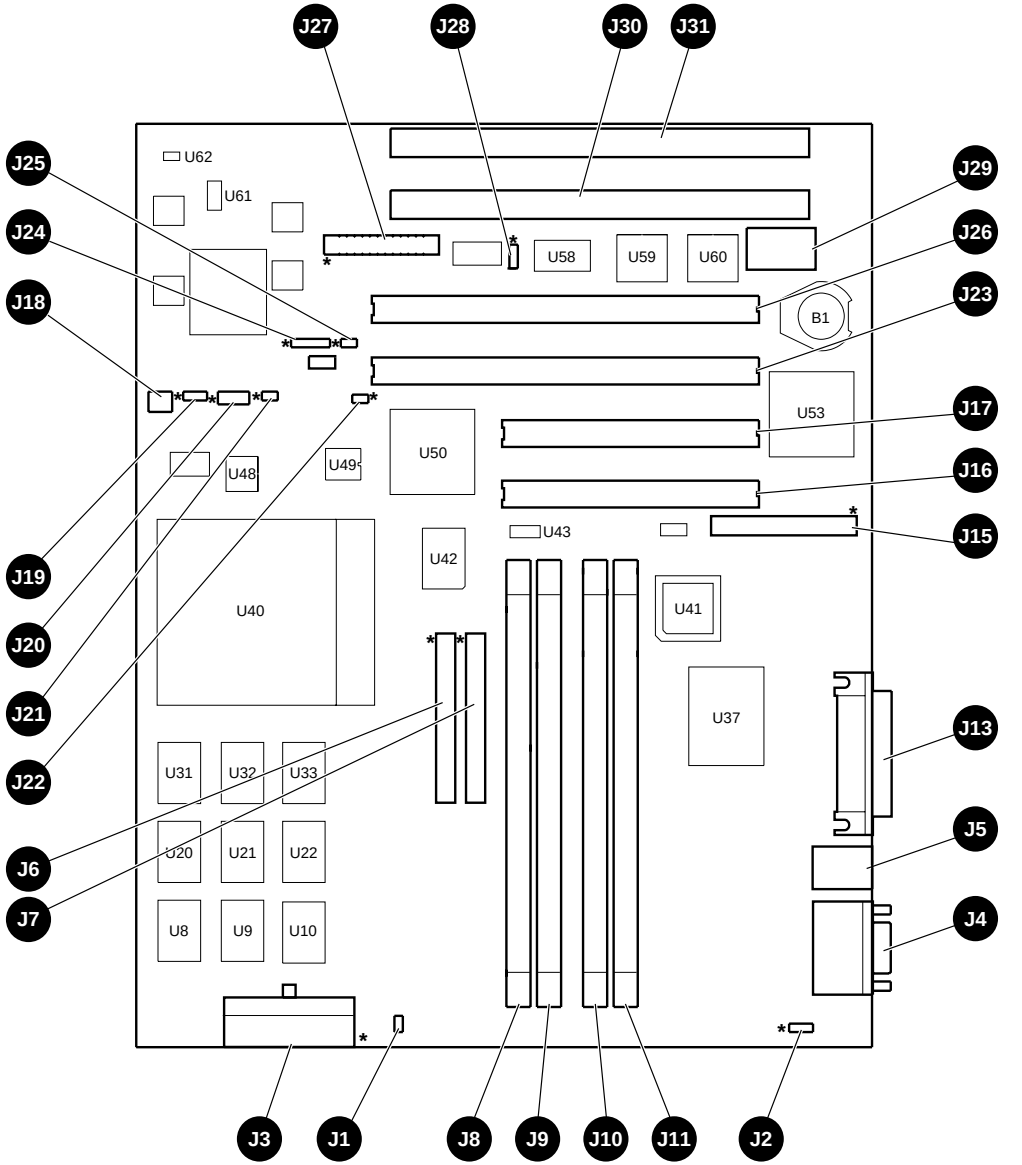


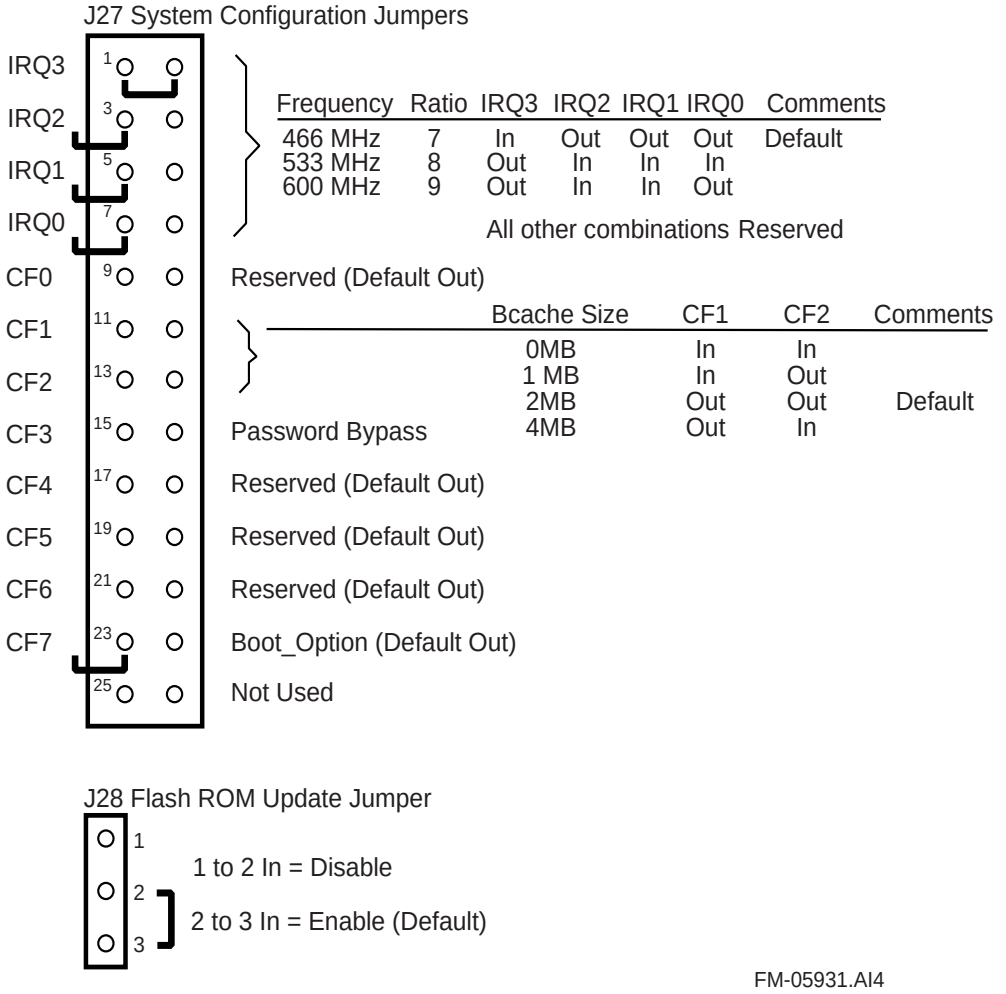
Table 2–1 AlphaPC 164LX Jumper/Connector/Component List

Item No.	Description	Item No.	Description
B1	RTC battery (CR2032)	J25	Hard-drive LED connector
J1	Soft power connector	J26	PCI slot 0 (64-bit)
J2	Fan power, enclosure (+12V)	J27	System configuration jumpers
J3	Power (+3V, +5V, -5V, +12V, -12V)	J28	Flash ROM update enable/disable jumper
J4	COM1/COM2 (DB9) connectors	J29	SROM test port connector
J5	Keyboard/mouse connectors	J30	ISA slot 1
J6	EIDE drive 2/3 connector	J31	ISA slot 0
J7	EIDE drive 0/1 connector	U8 to U10	Cache SRAM (L3)
J8	SDRAM DIMM 0 <0:63> connector	U20 to U22	Cache SRAM (L3)
J9	SDRAM DIMM 1 <64:128> connector	U31 to U33	Cache SRAM (L3)
J10	SDRAM DIMM 2 <0:63> connector	U37	I/O interface and address control (DSC 21174-CA)
J11	SDRAM DIMM 3 <64:128> connector	U40	Microprocessor, socket (DSC Alpha 21164)
J12	Reserved	U41	Patch 8K PAL
J13	Parallel I/O connector	U42	EIDE controller
J14	Reserved	U43	System clock PLL (CY2308)
J15	Diskette (floppy) drive connector	U48	Microprocessor clock synthesizer (MC12439)
J16	PCI slot 3 (32-bit)	U49	Serial ROM, socketed (Xilinx XC17128D)
J17	PCI slot 2 (32-bit)	U50	PCI-to-ISA bridge (Intel 82378ZB)
J18	Microprocessor fan/fan sense connector	U53	Combination controller, Super I/O (SMC FDC37C935)
J19	Enclosure fan +12V power connector	U58	Flash ROM (1MB)
J20	Speaker connector	U59	PCI arbiter PAL
J21	Reset button connector	U60	PCI interrupt request PAL
J22	Halt button connector	U61	Power controller
J23	PCI slot 1 (64-bit)	U62	Power sense
J24	Power LED connector		

2.1 AlphaPC 164LX Configuration Jumpers

The AlphaPC 164LX motherboard has two groups of jumpers located at J27 and J28, as shown previously in Figure 2–1. These jumpers set the hardware configuration and boot options. Figure 2–2 shows the jumper functions for each group.

Figure 2–2 AlphaPC 164LX Configuration Jumpers



2.2 CPU Speed Selection

The clock synthesizer at location U47 makes it possible to change the frequency of the microprocessor's system clock output without having to change the clock crystal. Simply set the system clock divisor jumpers to adjust the frequency of the microprocessor's system clock output. These system clock divisor jumpers are located at J27–1/2 (IRQ3), J27–3/4 (IRQ2), J27–5/6 (IRQ1), and J27–7/8 (IRQ0). The jumper configuration is set in IRQ3 through IRQ0. These four jumpers set the speed at power-up as listed in Figure 2–2. The microprocessor frequency divided by the ratio determines the system clock frequency.

2.3 Bcache Size Jumpers (CF1 and CF2)

The Bcache size jumpers are located at J27–11/12 (CF1) and J27–13/14 (CF2), as shown in Figure 2–2. The AlphaPC 164LX is configured with 2MB of Bcache during production; the other jumpers shown in Figure 2–2 (0, 1, and 4) are for other implementations.

Note: The standard motherboard is manufactured with 128K X 18 data SSRAMs (21A04-C0 for Windows NT and 21A04-C1 for UNIX). An OEM, however, can create a 4MB L3 cache variation using 256K X 18 data SSRAMs.

2.4 Password Bypass Jumper (CF3)

AlphaBIOS provides password protection. However, password bypass is provided for system setup or startup when the AlphaBIOS password is unavailable.

Password bypass is enabled by inserting jumper CF3 in the J27 system configuration jumper block. This disables the AlphaBIOS password verification and enables the user to perform system setup and startup without the AlphaBIOS password. Password bypass also clears the password.

2.5 Boot Option Jumper (CF7)

The boot option jumper is located at J27–23/24 (CF7). The default position for this jumper is out (Figure 2–2). This jumper selects the image to be loaded into memory from the system flash ROM. With the jumper out, the AlphaBIOS firmware is loaded. With the jumper in, the fail-safe booter is loaded. For more information about the fail-safe booter, refer to the *AlphaPC 164LX Motherboard Windows NT User's Manual.*

2.6 Flash ROM Update Jumper (J28)

When J28–2/3 are jumpered together (default), the flash ROM is write-enabled. When J28–1/2 are jumpered together, the flash ROM is write-protected.

2.7 AlphaPC 164LX Connector Pinouts

This section lists the pinouts of all AlphaPC 164LX connectors. See Figure 2–1 for connector locations.

2.7.1 PCI Bus Connector Pinouts

Table 2–2 shows the PCI bus connector pinouts.

Table 2–2 PCI Bus Connector Pinouts *(Sheet 1 of 2)*

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
32-Bit and 64-Bit PCI Connectors (J16, J17, J23, J26)							
A1	**TRST#**	A2	**+12V**	A3	**TMS**	A4	**TDI**
A5	**Vdd**	A6	**INTA**	A7	**INTC**	A8	**Vdd**
A9	—	A10	**Vdd**	A11	—	A12	**Gnd**
A13	**Gnd**	A14	—	A15	**RST#**	A16	**Vdd**
A17	**GNT#**	A18	**Gnd**	A19	—	A20	**AD<30>**
A21	**+3V**	A22	**AD<28>**	A23	**AD<26>**	A24	**Gnd**
A25	**AD<24>**	A26	**IDSEL**	A27	**+3V**	A28	**AD<22>**
A29	**AD<20>**	A30	**Gnd**	A31	**AD<18>**	A32	**AD<16>**
A33	**+3V**	A34	**FRAME#**	A35	**Gnd**	A36	**TRDY#**
A37	**STOP#**	A38	**STOP#**	A39	**+3V**	A40	**SDONE**
A41	**SBO#**	A42	**Gnd**	A43	**PAR**	A44	**AD<15>**
A45	**+3V**	A46	**AD<13>**	A47	**AD<11>**	A48	**Gnd**
A49	**AD<09>**	A50	Not used	A51	Not used	A52	**C/BE#<0>**
A53	**+3V**	A54	**AD<06>**	A55	**AD<04>**	A56	**Gnd**
A57	**AD<02>**	A58	**AD<00>**	A59	**Vdd**	A60	**REQ64#**
A61	**Vdd**	A62	**Vdd**	B1	**–12V**	B2	**TCK**
B3	**Gnd**	B4	**TDO**	B5	**Vdd**	B6	**Vdd**
B7	**INTB**	B8	**INTD**	B9	**PRSNT1#**	B10	—
B11	**PRSNT2#**	B12	**Gnd**	B13	**Gnd**	B14	—
B15	**Gnd**	B16	**CLK**	B17	**Gnd**	B18	**REQ#**
B19	**Vdd**	B20	**AD<31>**	B21	**AD<29>**	B22	**Gnd**
B23	**AD<27>**	B24	**AD<25>**	B25	**+3V**	B26	**C/BE#<3>**

Table 2–2 PCI Bus Connector Pinouts *(Sheet 2 of 2)*

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
B27	**AD<23>**	B28	**Gnd**	B29	**AD<21>**	B30	**AD<19>**
B31	**+3V**	B32	**AD<17>**	B33	**C/BE#<2>**	B34	**Gnd**
B35	**IRDY#**	B36	**+3V**	B37	**DEVSEL#**	B38	**Gnd**
B39	**LOCK#**	B40	**PERR#**	B41	**+3V**	B42	**SERR#**
B43	**+3V**	B44	**C/BE#<1>**	B45	**AD<14>**	B46	**Gnd**
B47	**AD<12>**	B48	**AD<10>**	B49	**Gnd**	B50	Not used
B51	Not used	B52	**AD<08>**	B53	**AD<07>**	B54	**+3V**
B55	**AD<05>**	B56	**AD<03>**	B57	**Gnd**	B58	**AD<01>**
B59	**Vdd**	B60	**ACK64#**	B61	**Vdd**	B62	**Vdd**
64-Bit PCI Connectors Only (J23, J26)							
A63	**Gnd**	A64	**C/BE#<7>**	A65	**C/BE#<5>**	A66	**Vdd**
A67	**PAR64**	A68	**D<62>**	A69	**Gnd**	A70	**D<60>**
A71	**D<58>**	A72	**Gnd**	A73	**D<56>**	A74	**D<54>**
A75	**Vdd**	A76	**D<52>**	A77	**D<50>**	A78	**Gnd**
A79	**D<48>**	A80	**D<46>**	A81	**Gnd**	A82	**D<44>**
A83	**D<42>**	A84	**Vdd**	A85	**D<40>**	A86	**D<38>**
A87	**Gnd**	A88	**D<36>**	A89	**D<34>**	A90	**Gnd**
A91	**D<32>**	A92	—	A93	**Gnd**	A94	—
B63	—	B64	**Gnd**	B65	**C/BE#<6>**	B66	**C/BE#<4>**
B67	**Gnd**	B68	**D<63>**	B69	**D<61>**	B70	**Vdd**
B71	**D<59>**	B72	**D<57>**	B73	**Gnd**	B74	**D<55>**
B75	**D<53>**	B76	**Gnd**	B77	**D<51>**	B78	**D<49>**
B79	**Vdd**	B80	**D<47>**	B81	**D<45>**	B82	**Gnd**
B83	**D<43>**	B84	**D<41>**	B85	**Gnd**	B86	**D<39>**
B87	**D<37>**	B88	**Vdd**	B89	**D<35>**	B90	**D<33>**
B91	**Gnd**	B92	—	B93	—	B94	**Gnd**

2.7.2 ISA Expansion Bus Connector Pinouts

Table 2–3 shows the ISA expansion bus connector pinouts.

Table 2–3 ISA Expansion Bus Connector Pinouts (J30, J31)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	**Gnd**	2	**IOCHCK#**	3	**RSTDRV**	4	**SD7**
5	**Vdd**	6	**SD6**	7	**IRQ9**	8	**SD5**
9	**–5V**	10	**SD4**	11	**DRQ2**	12	**SD3**
13	**–12V**	14	**SD2**	15	**ZEROWS#**	16	**SD1**
17	**+12V**	18	**SD0**	19	**Gnd**	20	**IOCHRDY**
21	**SMEMW#**	22	**AEN**	23	**SMEMR#**	24	**SA19**
25	**IOW#**	26	**SA18**	27	**IOR#**	28	**SA17**
29	**DACK3#**	30	**SA16**	31	**DRQ3**	32	**SA15**
33	**DACK1#**	34	**SA14**	35	**DRQ1**	36	**SA13**
37	**REFRESH#**	38	**SA12**	39	**SYSCLK**	40	**SA11**
41	**IRQ7**	42	**SA10**	43	**IRQ6**	44	**SA9**
45	**IRQ5**	46	**SA8**	47	**IRQ4**	48	**SA7**
49	**IRQ3**	50	**SA6**	51	**DACK2#**	52	**SA5**
53	**TC**	54	**SA4**	55	**BALE**	56	**SA3**
57	**Vdd**	58	**SA2**	59	**OSC**	60	**SA1**
61	**Gnd**	62	**SA0**	63	**MEMCS16#**	64	**SBHE#**
65	**IOCS16#**	66	**LA23**	67	**IRQ10**	68	**LA22**
69	**IRQ11**	70	**LA21**	71	**IRQ12**	72	**LA20**
73	**IRQ15**	74	**LA19**	75	**IRQ14**	76	**LA18**
77	**DACK0#**	78	**LA17**	79	**DRQ0**	80	**MEMR#**
81	**DACK5#**	82	**MEMW#**	83	**DRQ5**	84	**SD8**
85	**DACK6#**	86	**SD9**	87	**DRQ6**	88	**SD10**
89	**DACK7#**	90	**SD11**	91	**DRQ7**	92	**SD12**
93	**Vdd**	94	**SD13**	95	**MASTER#**	96	**SD14**
97	**Gnd**	98	**SD15**	—	—	—	—

2.7.3 SDRAM DIMM Connector Pinouts

Table 2–4 shows the SDRAM DIMM connector pinouts.

Table 2–4 SDRAM DIMM Connector Pinouts (J8 through J11)[1] *(Sheet 1 of 2)*

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	**Gnd**	2	**DQ0**	3	**DQ1**	4	**DQ2**
5	**DQ3**	6	**+3V**	7	**DQ4**	8	**DQ5**
9	**DQ6**	10	**DQ7**	11	**DQ8**	12	**Gnd**
13	**DQ9**	14	**DQ10**	15	**DQ11**	16	**DQ12**
17	**DQ13**	18	**+3V**	19	**DQ14**	20	**DQ15**
21	**CB0**	22	**CB1**	23	**Gnd**	24	NC
25	NC	26	**+3V**	27	**$\overline{\text{WE}}$**	28	**DQMB0**
29	**DQMB1**	30	**$\overline{\text{S0}}$**	31	NC	32	**Gnd**
33	**A0**	34	**A2**	35	**A4**	36	**A6**
37	**A8**	38	**A10**	39	**A12**	40	**+3V**
41	**+3V**	42	**CK0**	43	**Gnd**	44	NC
45	**$\overline{\text{S2}}$**	46	**DQMB2**	47	**DQMB3**	48	NC
49	**+3V**	50	NC	51	NC	52	**CB2**
53	**CB3**	54	**Gnd**	55	**DQ16**	56	**DQ17**
57	**DQ18**	58	**DQ19**	59	**+3V**	60	**DQ20**
61	NC	62	NC	63	**CKE1**	64	**Gnd**
65	**DQ21**	66	**DQ22**	67	**DQ23**	68	**Gnd**
69	**DQ24**	70	**DQ25**	71	**DQ26**	72	**DQ27**
73	**+3V**	74	**DQ28**	75	**DQ29**	76	**DQ30**
77	**DQ31**	78	**Gnd**	79	**CK2**	80	NC
81	NC	82	**SDA**	83	**SCL**	84	**+3V**
85	**Gnd**	86	**DQ32**	87	**DQ33**	88	**DQ34**
89	**DQ35**	90	**+3V**	91	**DQ36**	92	**DQ37**
93	**DQ38**	94	**DQ39**	95	**DQ40**	96	**Gnd**
97	**DQ41**	98	**DQ42**	99	**DQ43**	100	**DQ44**
101	**DQ45**	102	**+3V**	103	**DQ46**	104	**DQ47**
105	**CB4**	106	**CB5**	107	**Gnd**	108	NC
109	NC	110	**+3V**	111	**$\overline{\text{CAS}}$**	112	**DQMB4**
113	**DQMB5**	114	**S1**	115	**$\overline{\text{RAS}}$**	116	**Gnd**
117	**A1**	118	**A3**	119	**A5**	120	**A7**
121	**A9**	122	**BA0**	123	**A13**	124	**+3V**

Table 2–4 SDRAM DIMM Connector Pinouts (J8 through J11)[1] *(Sheet 2 of 2)*

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
125	**CK1**	126	**BA1**[2]	127	**Gnd**	128	**CKE0**
129	**$\overline{S3}$**	130	**DQMB6**	131	**DQMB7**	132	**PD**[3]
133	**+3V**	134	NC	135	NC	136	**CB6**
137	**CB7**	138	**Gnd**	139	**DQ48**	140	**DQ49**
141	**DQ50**	142	**DQ51**	143	**+3V**	144	**DQ52**
145	NC	146	NC	147	**PD**	148	**Gnd**
149	**DQ53**	150	**DQ54**	151	**DQ55**	152	**Gnd**
153	**DQ56**	154	**DQ57**	155	**DQ58**	156	**DQ59**
157	**+3V**	158	**DQ60**	159	**DQ61**	160	**DQ62**
161	**DQ63**	162	**Gnd**	163	**CK3**	164	NC
165	**SA0**	166	**SA1**	167	**SA2**	168	**+3V**

[1] Pins 1 through 84 are on the front side and pins 85 through 168 are on the back side.
[2] The AlphaPC 164LX uses **BA1** as both **BA1** and **ADDR12**. Therefore, four-bank DIMMs using **ADDR<11:0>** are the maximum size. (Two-bank DIMMs can use **ADDR<12:0>**.)
[3] Pull-down.

2.7.4 EIDE Drive Bus Connector Pinouts

Table 2–5 shows the EIDE drive bus connector pinouts.

Table 2–5 EIDE Drive Bus Connector Pinouts (J6, J7)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	**$\overline{RESET}$**	2	**Gnd**	3	**IDE_D7**	4	**IDE_D8**
5	**IDE_D6**	6	**IDE_D9**	7	**IDE_D5**	8	**IDE_D10**
9	**IDE_D4**	10	**IDE_D11**	11	**IDE_D3**	12	**IDE_D12**
13	**IDE_D2**	14	**IDE_D13**	15	**IDE_D1**	16	**IDE_D14**
17	**IDE_D0**	18	**IDE_D15**	19	**Gnd**	20	NC (key pin)
21	**MARQ**	22	**Gnd**	23	**$\overline{IOW}$**	24	**Gnd**
25	**$\overline{IOR}$**	26	**Gnd**	27	**CHRDY**	28	**BALE**
29	**MACK**	30	**Gnd**	31	**IRQ**	32	**$\overline{IOCS16}$**
33	**ADDR1**	34	NC	35	**ADDR0**	36	**ADDR2**
37	**$\overline{CS0}$**	38	**$\overline{CS1}$**	39	**$\overline{ACT}$**	40	**Gnd**

2.7.5 Diskette Drive Bus Connector Pinouts

Table 2–6 shows the diskette (floppy) drive bus connector pinouts.

Table 2–6 Diskette (Floppy) Drive Bus Connector Pinouts (J15)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	**Gnd**	2	**DEN0**	3	**Gnd**	4	NC
5	**Gnd**	6	**DEN1**	7	**Gnd**	8	**INDEX**
9	**Gnd**	10	**MTR0**	11	**Gnd**	12	**DR1**
13	**Gnd**	14	**DR0**	15	**Gnd**	16	**MTR1**
17	**Gnd**	18	**DIR**	19	**Gnd**	20	**STEP**
21	**Gnd**	22	**WDATA**	23	**Gnd**	24	**WGATE**
25	**Gnd**	26	**TRK0**	27	**Gnd**	28	**WRTPRT**
29	**ID0**	30	**RDATA**	31	**Gnd**	32	**HDSEL**
33	**ID1**	34	**DSKCHG**	—	—	—	—

2.7.6 Parallel Bus Connector Pinouts

Table 2–7 shows the parallel bus connector pinouts.

Table 2–7 Parallel Bus Connector Pinouts (J13)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	**STB**	2	**PD0**	3	**PD1**	4	**PD2**
5	**PD3**	6	**PD4**	7	**PD5**	8	**PD6**
9	**PD7**	10	**ACK**	11	**BUSY**	12	**PE**
13	**SLCT**	14	**AFD**	15	**ERR**	16	**INIT**
17	**SLIN**	18	**Gnd**	19	**Gnd**	20	**Gnd**
21	**Gnd**	22	**Gnd**	23	**Gnd**	24	**Gnd**
25	**Gnd**	—	—	—	—	—	—

2.7.7 COM1/COM2 Serial Line Connector Pinouts

Table 2–8 shows the COM1/COM2 serial line connector pinouts.

Table 2–8 COM1/COM2 Serial Line Connector Pinouts (J4)

COM1 Pin (Top)	COM1 Signal	COM2 Pin (Bottom)	COM2 Signal
1	**DCD1**	1	**DCD2**
2	**RxD1**	2	**RxD2**
3	**TxD1**	3	**TxD2**
4	**DTR1**	4	**DTR2**
5	**SG1**	5	**SG2**
6	**DSR1**	6	**DSR2**
7	**RTS1**	7	**RTS2**
8	**CTS1**	8	**CTS2**
9	**RI1**	9	**RI2**

2.7.8 Keyboard/Mouse Connector Pinouts

Table 2–9 shows the keyboard/mouse connector pinouts.

Table 2–9 Keyboard/Mouse Connector Pinouts (J5)

Keyboard Pin (Top)	Keyboard Signal	Mouse Pin (Bottom)	Mouse Signal
1	**KBDATA**	1	**MSDATA**
2	NC	2	NC
3	**Gnd**	3	**Gnd**
4	**Vdd**	4	**Vdd**
5	**KBCLK**	5	**MSCLK**
6	NC	6	NC

2.7.9 SROM Test Data Input Connector Pinouts

Table 2–10 shows the SROM test data input connector pinouts.

Table 2–10 SROM Test Data Input Connector Pinouts (J29)

Pin	Signal	Name
1	NC	—
2	**SROM_CLK_L**	Clock out
3	**Gnd**	—
4	NC	—
5	**TEST_SROM_D_L**	SROM serial data in
6	NC	—

2.7.10 Input Power Connector Pinouts

Table 2–11 shows the input power connector pinouts.

Table 2–11 Input Power Connector Pinouts (J3)[1]

Pin	Voltage	Pin	Voltage	Pin	Voltage	Pin	Voltage
1	**+3.3 V dc**	2	**+3.3 V dc**	3	**Gnd**	4	**+5 V dc**
5	**Gnd**	6	**+5 V dc**	7	**Gnd**	8	**P_DCOK**
9	**5 V SB**	10	**+12 V dc**	11	**+3.3 V dc**	12	**–12 V dc**
13	**Gnd**	14	**PS_ON**	15	**Gnd**	16	**Gnd**
17	**Gnd**	18	**–5 V dc**	19	**+5 V dc**	20	**+5 V dc**

[1] This pinout is ATX-compliant.

2.7.11 Enclosure Fan Power Connector Pinouts

Table 2–12 shows the enclosure fan power connector pinouts.

Table 2–12 Enclosure Fan (+12 V dc) Power Connector Pinouts (J2, J19)

Pin	Voltage
1	**Gnd**
2	**+12 V dc**
3	**Gnd**

2.7.12 Speaker Connector Pinouts

Table 2–13 shows the speaker connector pinouts.

Table 2–13 Speaker Connector Pinouts (J20)

Pin	Signal	Name
1	**SPKR**	Speaker output
2	**Gnd**	—
3	**Gnd**	—
4	**Gnd**	—

2.7.13 Microprocessor Fan Power Connector Pinouts

Table 2–14 shows the microprocessor fan power connector pinouts.

Table 2–14 Microprocessor Fan Power Connector Pinouts (J18)

Pin	Signal	Name
1	**+12V**	—
2	**FAN_CONN_L**	Fan connected
3	**Gnd**	—

2.7.14 Power LED Connector Pinouts

Table 2–15 shows the power LED connector pinouts.

Table 2–15 Power LED Connector Pinouts (J24)

Pin	Signal	Name
1	**POWER_LED_L**	Power LED input
2	**Gnd**	—
3	NC	—
4	NC	—
5	NC	—

2.7.15 IDE Drive LED Connector Pinouts

Table 2–16 shows the IDE drive LED connector pinouts.

Table 2–16 IDE Drive LED Connector Pinouts (J25)

Pin	Signal	Name
1	**HD_ACT_L**	Hard drive active
2	**HD_LED_L**	Hard drive LED input

2.7.16 Reset Button Connector Pinouts

Table 2–17 shows the reset button connector pinouts.

Table 2–17 Reset Button Connector Pinouts (J21)

Pin	Signal	Name
1	**RESET_BUTTON**	Reset system
2	**Gnd**	—

2.7.17 Halt Button Connector Pinouts

Table 2–18 shows the halt button connector pinouts.

Table 2–18 Halt Button Connector Pinouts (J22)

Pin	Signal	Name
1	**HALT_BUTTON**	Halt system
2	**Gnd**	—

Note: The Halt button is not used with the Windows NT operating system.

2.7.18 Soft Power Connector Pinouts

Table 2–19 shows the soft power connector pinouts.

Table 2–19 Soft Power Connector Pinouts (J1)

Pin	Signal	Name
1	**Input**	System power on/off
2	**Gnd**	—

3
Power and Environmental Requirements

This chapter describes the AlphaPC 164LX power and environmental requirements, and physical board parameters.

3.1 Power Requirements

The AlphaPC 164LX derives its main dc power from a user-supplied power supply. The board has a total power dissipation of 100 W, excluding any plug-in PCI and ISA devices. An onboard +5-V to +2.5-V dc-to-dc converter is designed to handle 24 A of current. Table 3–1 lists the power requirement for each dc supply voltage.

The power supply must supply a **dcok** signal to the system reset logic. Refer to Section 4.6, and schematic pages *pc164lx.29* and *pc164lx.30* for additional information.

Table 3–1 Power Supply DC Current Requirements

Voltage/Tolerance	Current[1]
+3.3 V dc, ±5%	5.0 A
+5 V dc, ±5%	14.0 A
–5 V dc, ±5%	0 A
+12 V dc, ±5%	1.0 A
–12 V dc, ±5%	100.0 mA

[1] Values indicated are for an AlphaPC 164LX motherboard with an Alpha 21164 microprocessor operating at 600 MHz, with 64MB SDRAM, excluding adapter cards and disk drives.

Caution: **Fan Sensor Required**. The 21164 microprocessor cooling fan *must* have a built-in sensor that will drive a signal if the airflow stops. The sensor is connected to AlphaPC 164LX board connector J18. When the signal is generated, it resets the system.

3.2 Environmental Requirements

The 21164 microprocessor is cooled by a small fan blowing directly into the chip's heat sink. The AlphaPC 164LX motherboard is designed to run efficiently by using only this fan. Additional fans may be necessary depending upon cabinetry and the requirements of plug-in cards.

The AlphaPC 164LX motherboard is specified to run within the environment listed in Table 3–2.

Table 3–2 AlphaPC 164LX Motherboard Environmental Requirements

Parameter	Specification
Operating temperature	10°C to 40°C (50°F to 104°F)
Storage temperature	–55°C to 125°C (–67°F to 257°F)
Relative humidity	10% to 90% with maximum wet bulb temperature 28°C (82°F) and minimum dew point 2°C (36°F)
Rate of (dry bulb) temperature change	11°C/hour ±2°C/hour (20°F/hour ±4°F/hour)

3.3 Board Dimensions

The AlphaPC 164LX is an ATX-size printed-wiring board (PWB) with the following dimensions:

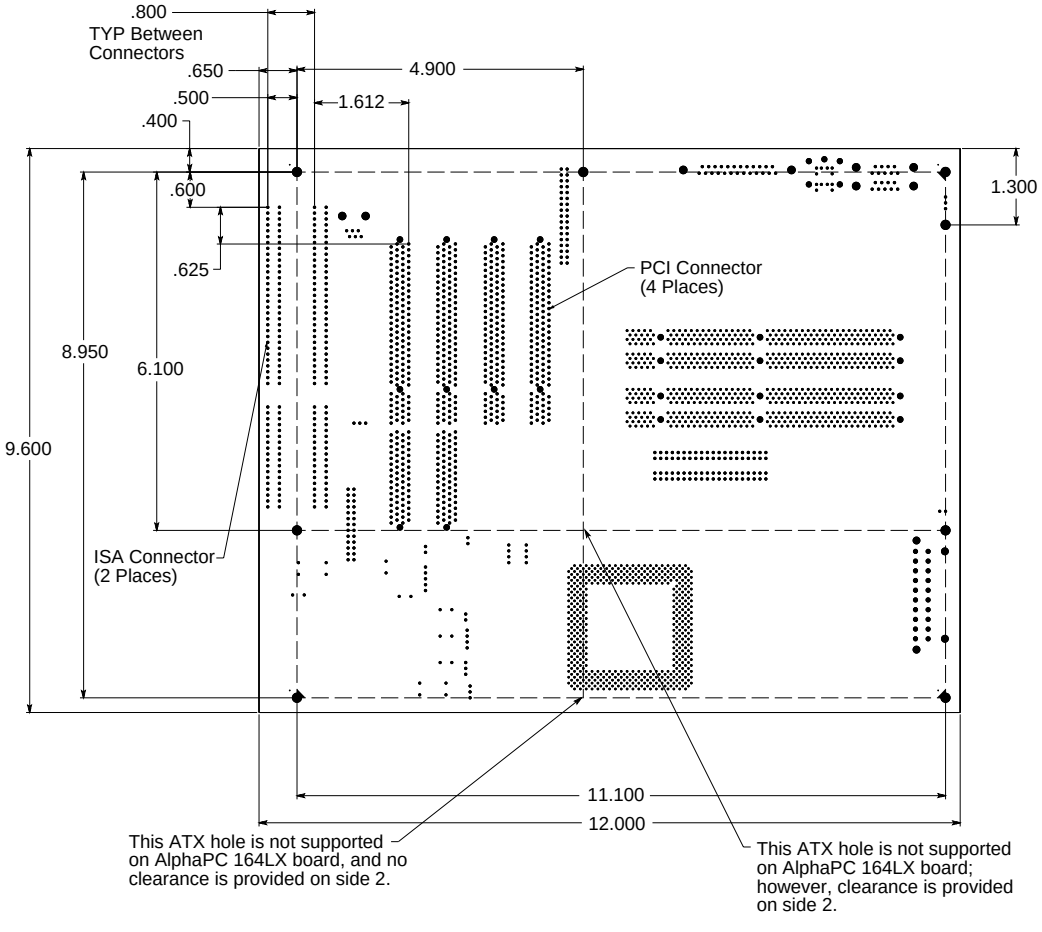
- Length: 30.48 cm (12.0 in ±0.0005 in)
- Width: 24.38 cm (9.6 in ±0.0005 in)
- Height: 6.86 cm (2.7 in)

The board can be used in certain desktop and deskside systems that have adequate clearance for the 21164 heat sink and its cooling fan. All ISA and PCI expansion slots are usable in standard desktop or deskside enclosures.

3.3.1 ATX Hole Specification

Figure 3–1 shows the ATX hole specification for the AlphaPC 164LX.

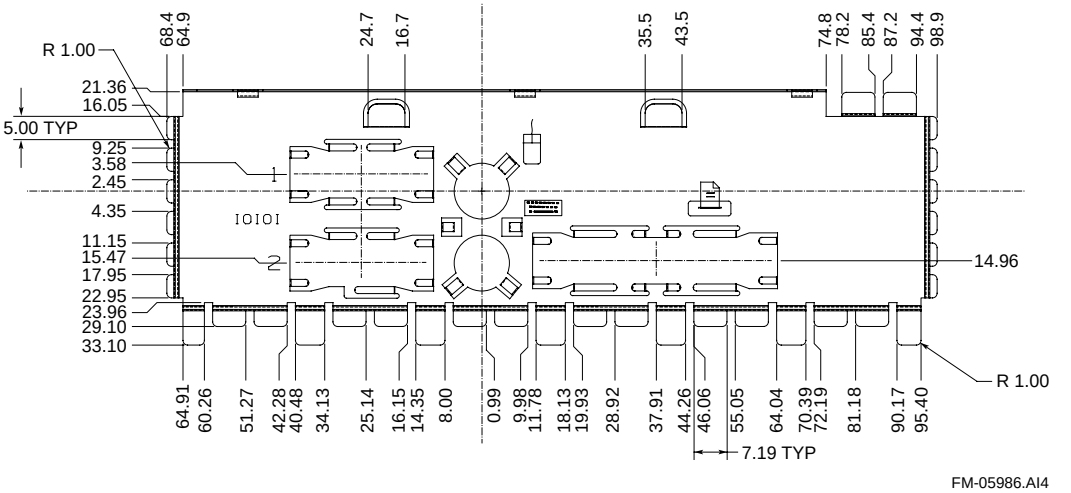
Figure 3–1 ATX Hole Specification



3.3.2 ATX I/O Shield Requirements

Figure 3–2 shows the ATX I/O shield dimensions for the AlphaPC 164LX.

Figure 3–2 ATX I/O Shield Dimensions



4
Functional Description

This chapter describes the functional operation of the AlphaPC 164LX. The description introduces the DIGITAL Semiconductor 21174 core logic chip and describes its implementation with the 21164 microprocessor, its supporting memory, and I/O devices. Figure 1–1 shows the AlphaPC 164LX major functional components.

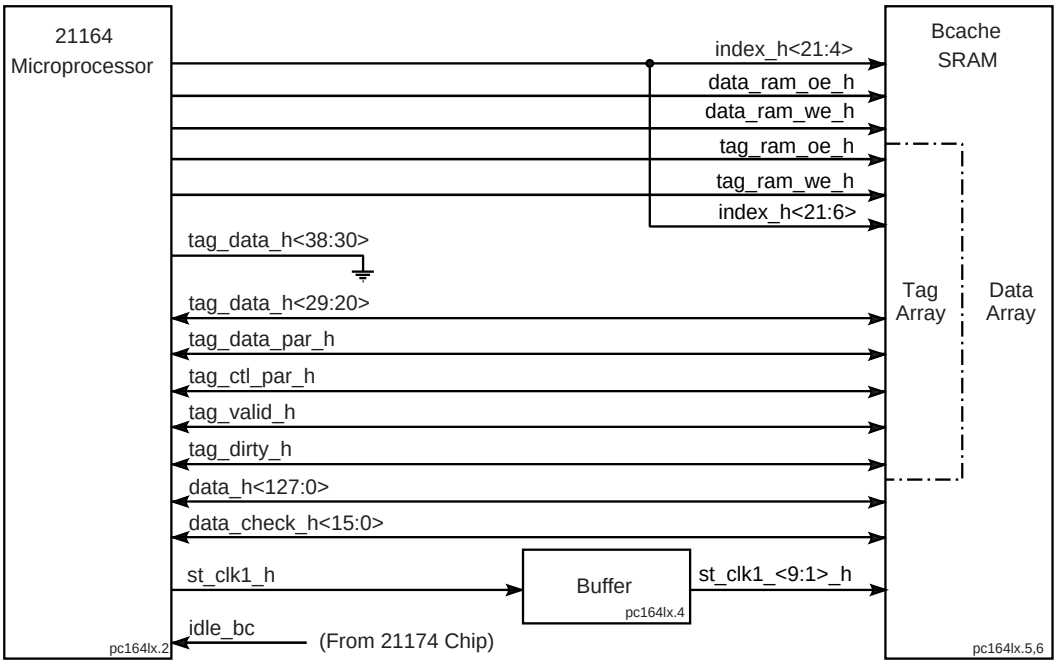
Bus timing and protocol information found in other data sheets and reference documentation is not duplicated. See Appendix C for a list of supporting documents and order numbers.

Note: For detailed descriptions of bus transactions, chip logic, and operation, refer to the *21164 Alpha Microprocessor Hardware Reference Manual* and the *DIGITAL Semiconductor 21174 Core Logic Chip Technical Reference Manual.* For details of the PCI interface, refer to the *PCI System Design Guide*.

4.1 AlphaPC 164LX Bcache Interface

The 21164 microprocessor controls the board-level L3 backup cache (Bcache) array (see Figure 4–1). The data bus (**data_h<127:0>**), check bus (**data_check_h<15:0>**), **tag_dirty_h**, and **tag_ctl_par_h** signals are shared with the system interface.

Figure 4–1 AlphaPC 164LX L3 Bcache Array



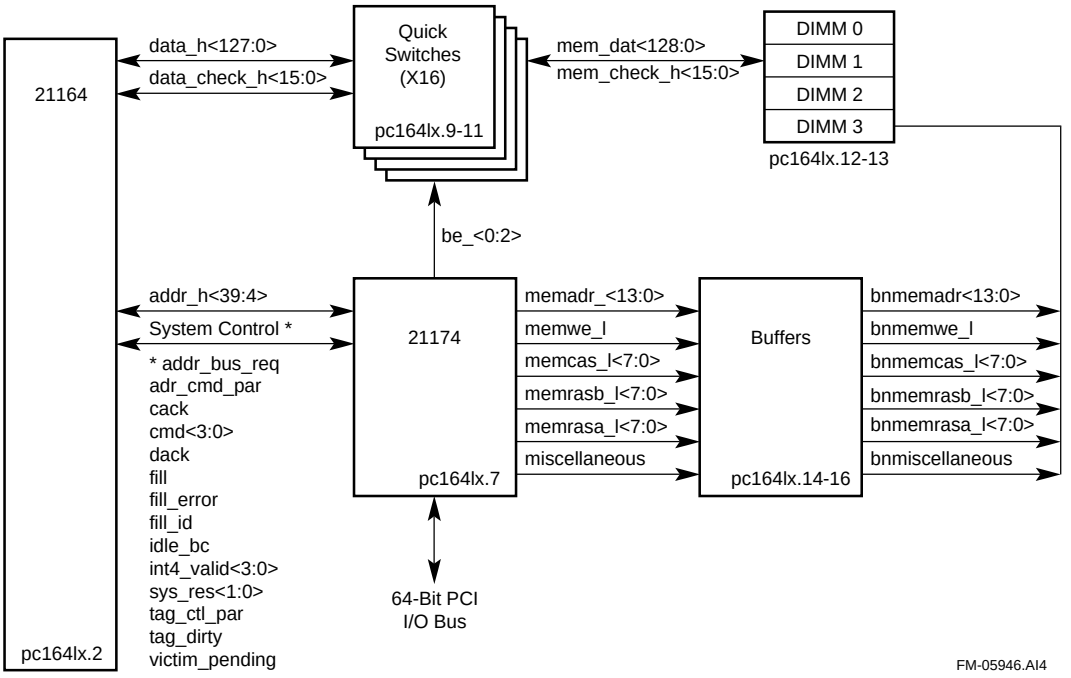
The Bcache is a 2MB, direct-mapped, synchronous SRAM (SSRAM) with a 128-bit data path. It is populated with a quantity of eight 9 ns, 128K x 18 SSRAMs for data store, and one 9 ns, 64K x 18 SSRAM for the tag store. In most cases, wave-pipelined accesses can decrease the cache loop times by one CPU cycle. The Bcache supports 64-byte transfers to and from memory.

4.2 DIGITAL Semiconductor 21174 Core Logic Chip

The 21174 core logic chip provides a cost-competitive solution for designers using the 21164 microprocessor to develop uniprocessor systems. The chip provides a 128-bit memory interface and a PCI I/O interface, and includes the DIGITAL Semiconductor 21174-CA chip packaged in a 474-pin plastic ball grid array (PBGA).

Figure 4–2 shows the AlphaPC 164LX implementation of the 21174 core logic chip.

Figure 4–2 Main Memory Interface



4.2.1 21174 Chip Overview

The 21174 application-specific integrated circuit (ASIC) accepts addresses and commands from the 21164 microprocessor and drives the main memory array with the address, control, and clock signals. It also provides an interface to the 64-bit PCI I/O bus.

The 21174 chip provides the following functions:

- Serves as the interface between the 21164 microprocessor, main memory (addressing and control), and the PCI bus. A three-entry CPU instruction queue is implemented to capture commands should the memory or I/O port be busy.
- Provides control to the Quick Switch chips to isolate the L3 cache from the main memory bus during private reads and writes.
- Generates the clocks, row, and column addresses for the SDRAM DIMMs, as well as all of the memory control signals (RAS, CAS, WE). All of the required SDRAM refresh control is contained in the 21174.
- Provides all the logic to map 21164 noncacheable addresses to PCI address space, as well as all the translation logic to map PCI DMA addresses to system memory.

Two DMA conversion methods are supported:

- Direct mapping, in which a base offset is concatenated with the PCI address.
- Scatter-gather mapping, which maps an 8KB PCI page to any 8KB memory page. The 21174 contains an eight-entry scatter-gather translation lookaside buffer (TLB), where each entry holds four consecutive page table entries (PTEs).

Refer to Appendix A for additional details on PCI and DMA address mapping.

4.2.2 Main Memory Interface

Sixteen Quick Switches provide the interface between the 21164/L3 cache (**data_h<127:0>**, **check_h<15:0>**) and the memory/21174 (**mem_data_h<127:0>**, **mem_check_h<15:0>**). The AlphaPC 164LX supports four 168-pin unbuffered 72-bit SDRAM DIMM modules. Quadword ECC is supported on the SDRAM and CPU buses. Even parity is generated on the PCI bus.

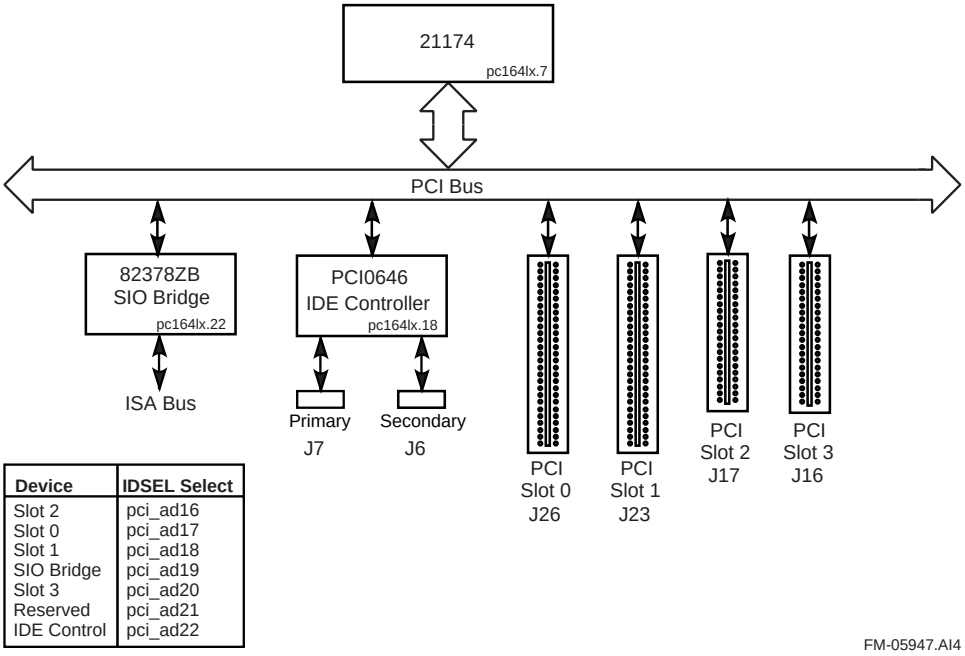
The AlphaPC 164LX supports a maximum of 512MB of main memory. The memory is organized as two banks. Table 1–1 lists total memory options along with the corresponding DIMM sizes required. All CPU cacheable memory accesses and PCI DMA accesses are controlled and routed to main memory by the 21174 core logic chip.

The AlphaPC 164LX implements the alternate memory mode for SDRAM RAS and CAS control signals. Alternate memory mode is explained in the *DIGITAL Semiconductor 21174 Core Logic Chip Technical Reference Manual*.

4.2.3 PCI Devices

The AlphaPC 164LX uses the PCI bus as the main I/O bus for the majority of peripheral functions. As Figure 4–3 shows, the board implements the ISA bus as an expansion bus for system support functions and for relatively slow peripheral devices.

Figure 4–3 AlphaPC 164LX PCI Bus Devices



Device	IDSEL Select
Slot 2	pci_ad16
Slot 0	pci_ad17
Slot 1	pci_ad18
SIO Bridge	pci_ad19
Slot 3	pci_ad20
Reserved	pci_ad21
IDE Control	pci_ad22

The PCI bus supports multiplexed, burst mode, read and write transfers. It supports synchronous operation of 33 MHz. It also supports either a 32-bit or 64-bit data path with 32-bit device support in the 64-bit configuration. Depending upon

the configuration and operating frequencies, the PCI bus supports up to 264-MB/s (33 MHz, 64-bit) peak throughput. The PCI provides parity on address and data cycles. Three physical address spaces are supported:

- 32-bit memory space
- 32-bit I/O space
- 256-byte-per-agent configuration space

The bridge from the 21164 system bus to the 64-bit PCI bus is provided by the 21174 chip. It generates the required 32-bit PCI address for 21164 I/O accesses directed to the PCI. It also accepts 64-bit double address cycles and 32-bit single address cycles. However, the 64-bit address support is subject to some constraints. Refer to Appendix A for more information on 64-bit addressing constraints.

4.2.4 Saturn-IO (SIO) Chip

The 82378ZB SIO chip provides the bridge between the PCI bus and the ISA bus. The SIO incorporates the logic for the following:

- A PCI interface (master and slave)
- An ISA interface (master and slave)
- Enhanced 7-channel DMA controller that supports fast DMA transfers and scatter-gather, and data buffers to isolate the PCI bus from the ISA bus
- PCI and ISA arbitration
- A 14-level interrupt controller
- A 16-bit basic input/output system (BIOS) timer
- Three programmable timer counters
- Nonmaskable interrupt (NMI) control logic
- Decoding and control for utility bus peripheral devices
- Speaker driver

Refer to Intel document *82420/82430 PCIset ISA and EISA Bridges* for additional information.

4.2.5 PCI Expansion Slots

Four dedicated PCI expansion slots are provided on the AlphaPC 164LX. This allows the system user to add additional 32-bit or 64-bit PCI options. While both the 32-bit and the 64-bit slots use the standard 5-V PCI connector and pinout, +3.3 V is supplied for those boards that require it. The SIO chip provides the interface to the ISA expansion I/O bus.

4.3 ISA Bus Devices

Figure 4–4 shows the AlphaPC 164LX ISA bus implementation with peripheral devices and connectors. Two dedicated ISA expansion slots are provided. System support features such as serial lines, parallel port, diskette controller, keyboard/ mouse control, and time-of-year clock are embedded on the module by means of an FDC37C935 combination controller chip. Also shown is the utility bus (Ubus) with its system support devices.

4.3.1 Combination Controller

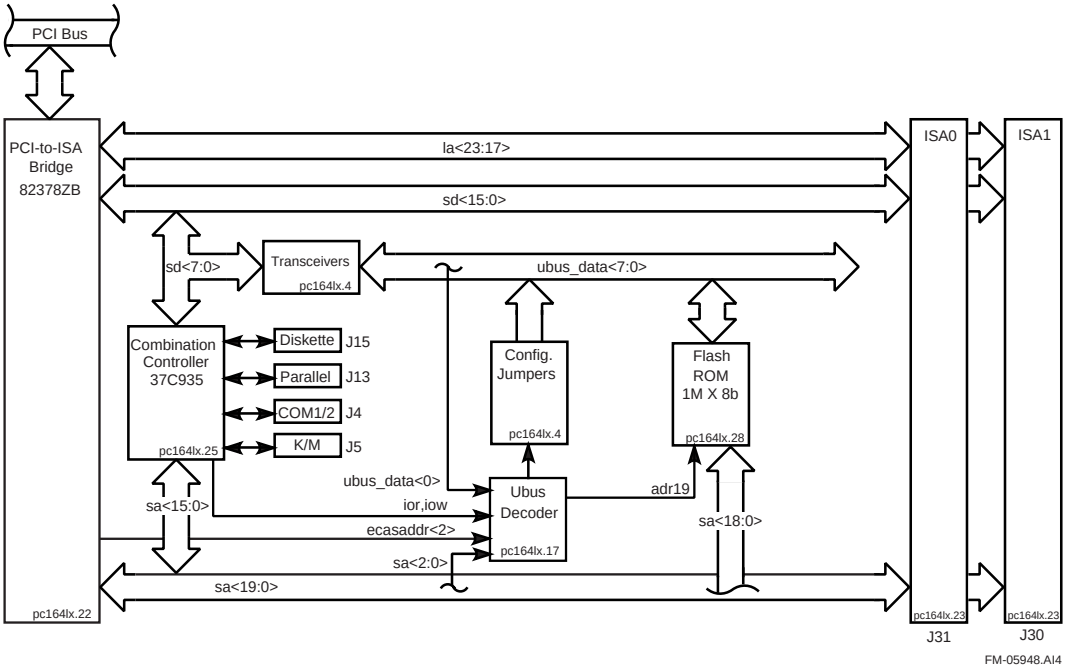
The AlphaPC 164LX uses the Standard Microsystems Corporation FDC37C935 Ultra I/O combination controller chip (see Figure 4–4). It is packaged in a 160-pin PQFP configuration. The chip provides the following ISA peripheral functions:

- **Diskette controller**–Software compatible to the Intel N82077 FDC. Integrates the functions of the formatter/controller, digital data separator, write precompensation, and data-rate selection logic requiring no external filter components. Supports the 2.88MB drive format and other standard diskette drives used with 5.25-inch and 3.5-inch media. FDC data and control lines are brought out to a standard 34-pin connector (J15). A ribbon cable interfaces the connector to one or two diskette drives.
- **Serial ports**–Two UARTs with full modem control, compatible with NS16450 or PC16550 devices, are brought out to two separate onboard, 9-pin D-subminiature connectors (J4).
- **Parallel port**–The bidirectional parallel port is brought out to an onboard 25-pin connector (J13). It can be brought out through a 25-pin female D-subminiature connector on the bulkhead of a standard PC enclosure.

- **Keyboard/mouse**–An 8042-compatible interface is brought out to separate 6-pin DIN connectors (J5).
- **Time-of-year clock**–A DS1287-compatible clock is backed up by a replaceable battery.

An onboard clock generator chip supplies a 14.3-MHz reference clock for the diskette data separator and serial ports.

Figure 4–4 AlphaPC 164LX ISA Bus Devices



4.3.2 Utility Bus Memory Device

The AlphaPC 164LX Ubus drives a flash ROM memory device. The flash ROM chip provides 1MB of flash memory for operating system support.

Flash data is accessed through 20 address inputs. The low-order 19 address bits are driven by ISA bus **sa<18:0>**. The high-order 20th bit (**flash_adr19**) is driven by the Ubus decode PLA. Address bit **flash_adr19** can be changed by writing to ISA I/O port x800.

The +12 V is applied to the flash ROM by means of jumper J28 so that code updates can be accomplished, if desired.

4.3.3 ISA Expansion Slots

Two ISA expansion slots are provided for plug-in ISA peripherals (J30 and J31).

4.3.4 ISA I/O Address Map

Table 4–1 lists the AlphaPC 164LX ISA I/O space address mapping.

Table 4–1 ISA I/O Address Map *(Sheet 1 of 2)*

Range (hex)	Usage
000-00F	8237 DMA #1
020-021	8259 PIC #1
040-043	8253 timer
060-061	Ubus IRQ12 and NMI control
070	CMOS RAM address and NMI mask register
080-08F	DMA page registers
0A0-0A1	8259 PIC #2
0C0-0DF	8237 DMA #2
2F8-2FF	Serial port—COM2
370-377	Secondary diskette (floppy)
3BC-3BF	Parallel port—LPT1
3F0-3F7	Primary diskette (floppy)
3F8-3FF	Serial port—COM1

Table 4–1 ISA I/O Address Map *(Sheet 2 of 2)*

Range (hex)	Usage
800	FLASH_ADR19 register
801	AlphaPC 164LX configuration register
804-806	PCI interrupt registers

4.3.5 Flash ROM Address Map

The address range for the flash ROM is FFF8.0000–FFFF.FFFF. Flash space of 1MB is obtained by double mapping this 512KB space. FLASH_ADR19 register at I/O location 800_{16} provides this function. Writing a 0 to this location enables the lower 512KB of flash. Writing a 1 to this location enables the upper 512KB of flash.

4.4 Interrupts

This section describes the AlphaPC 164LX interrupt logic. PCI-, ISA-, and 21174-generated interrupts are described. Figure 4–5 shows the interrupt logic.

The PCI-to-ISA SIO bridge chip provides the functionality of two 8259 interrupt control devices. These ISA-compatible interrupt controllers are cascaded so that 14 external and 2 internal interrupts are available. The PCI interrupt acknowledge command should be used to read the interrupt request vector from the SIO.

However, the AlphaPC 164LX system has more external interrupts than the SIO can handle. Therefore, all the ISA interrupts are sent to the SIO except for the two 21174 interrupts, the TOY interrupt, the IDE controller interrupt, and the 16 PCI interrupts. They are sent to an external interrupt programmable logic array (PLA). This PLA takes these interrupts, as well as an OR of the nonexistent memory (NMI) and error signals from the SIO, and generates **pci_isa_irq**. During reset, **cpu_irq<3:0>** convey the system clocking ratios and delays, which are set by jumpers on J27.

Table 4–2 lists each system interrupt, its fixed interrupt priority level (IPL), and its AlphaPC 164LX implementation. Table 4–3 lists each ISA bus interrupt and its AlphaPC 164LX implementation.

Figure 4–5 Interrupt Logic

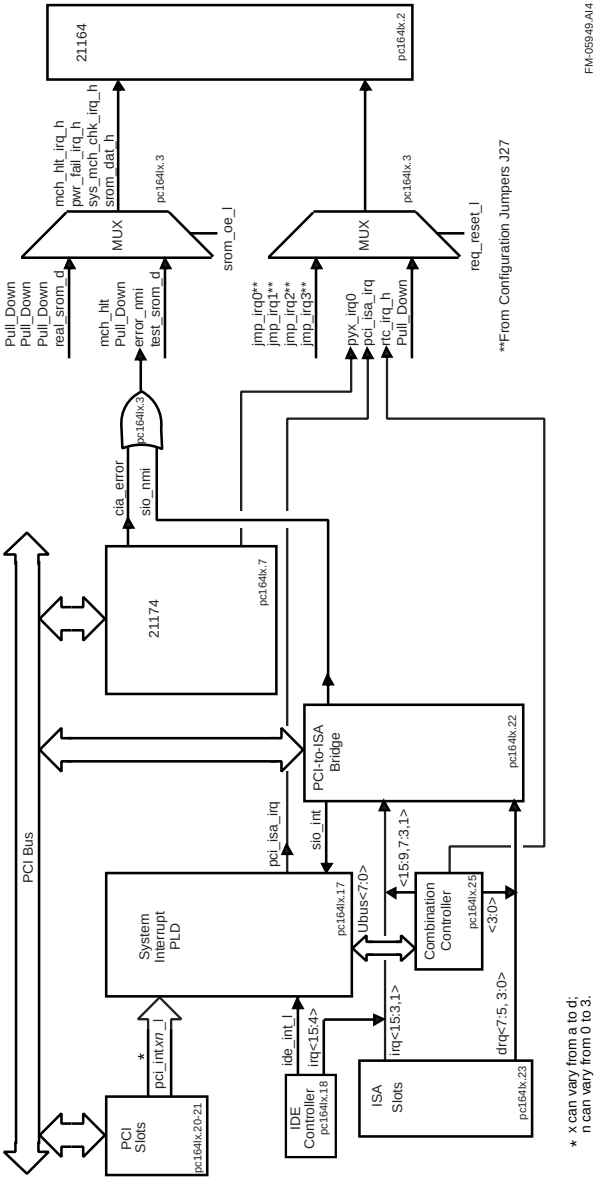


Table 4–2 AlphaPC 164LX System Interrupts

21164 Interrupt	IPL[1]	Suggested Usage	AlphaPC 164LX Usage
cpu_irq<0>	20	Corrected system error	Corrected ECC error and sparse space reserved encodings detected by the 21174
cpu_irq<1>	21	—	PCI and ISA interrupts
cpu_irq<2>	22	Interprocessor and timer interrupts	TOY clock interrupt
cpu_irq<3>	23	—	Reserved
pwr_fail_irq	30	Powerfail interrupt	Reserved
sys_mch_chk_irq	31	System machine check interrupt	SIO NMI and 21174 errors
mch_hlt_irq	—	Halt	Reserved

[1] IPL = interrupt priority level (fixed).

Table 4–3 ISA Interrupts

Interrupt Number	Interrupt Source
IRQ0	Internal timer
IRQ1	Keyboard
IRQ2	Interrupt from controller 2
IRQ3	COM2
IRQ4	COM1
IRQ5	Available
IRQ6	Diskette (floppy)
IRQ7	Parallel port
IRQ8#[1]	Reserved
IRQ9	Available
IRQ10	Available
IRQ11	Available
IRQ12	Mouse
IRQ13	Available
IRQ14	IDE
IRQ15	IDE

[1] The # symbol indicates an active low signal.

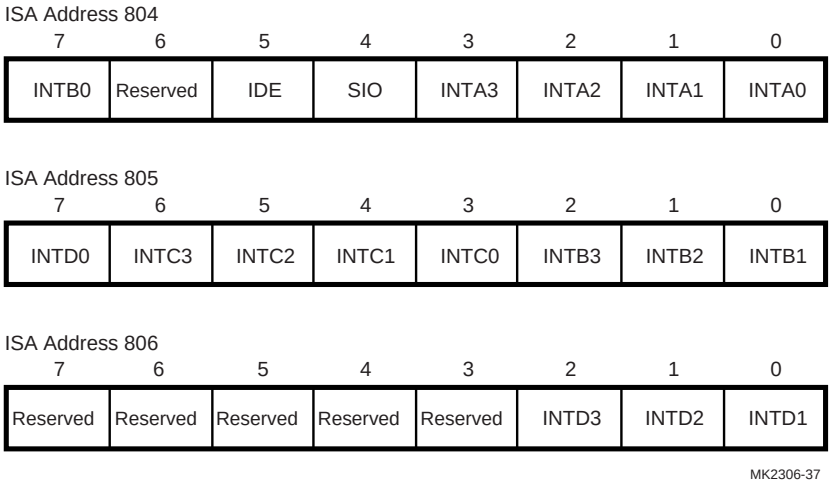
4.4.1 Interrupt PLD Function

The MACH210A PLD is an 8-bit I/O slave on the ISA bus at hex addresses 804, 805, and 806. This is accomplished by a decode of the three ISA address bits **sa<2:0>** and the three **ecas_addr<2:0>** bits.

Each interrupt can be individually masked by setting the appropriate bit in the mask register (see Figure 4–6). An interrupt is disabled by writing a 1 to the desired position in the mask register. An interrupt is enabled by writing a 0. For example, bit <1> set in interrupt mask register 1 indicates that the INTB2 interrupt is disabled. There are three mask registers located at ISA addresses 804, 805, and 806.

An I/O read transaction at ISA addresses 804, 805, and 806 returns the state of the 18 PCI interrupts rather than the state of the masked interrupts. On read transactions, a 1 means that the interrupt source has asserted its interrupt. The mask register can be updated by writing addresses 804, 805, or 806. The mask register is write-only.

Figure 4–6 Interrupt/Interrupt Mask Registers



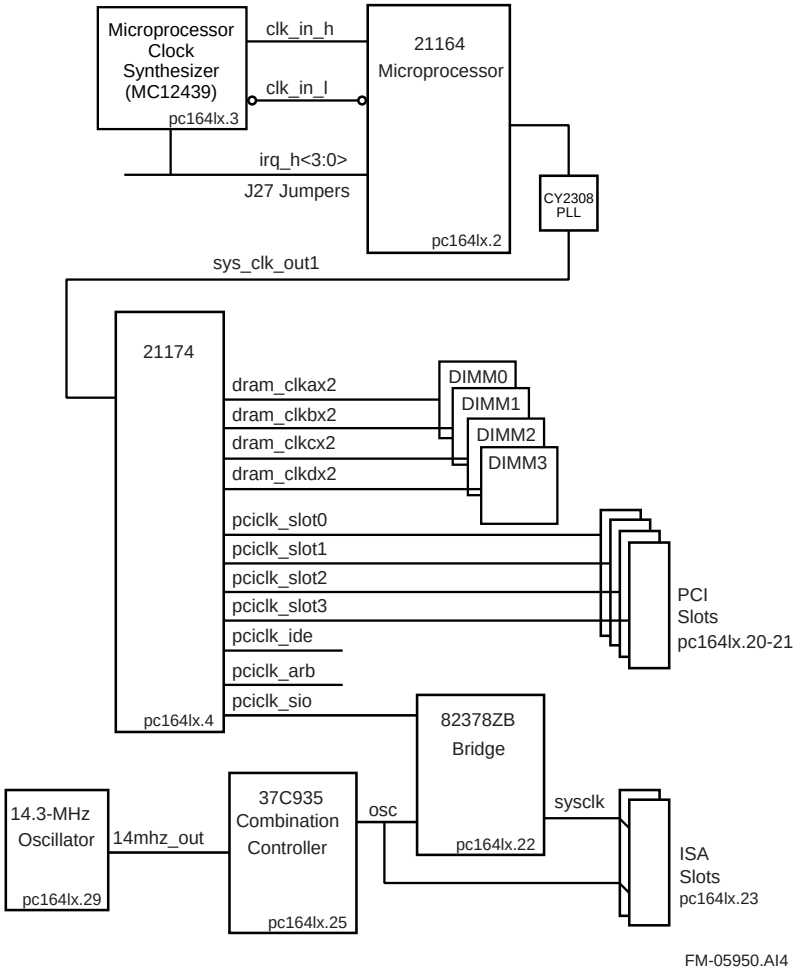
4.5 System Clocks

Figure 4–7 shows the AlphaPC 164LX clock generation and distribution scheme.

The AlphaPC 164LX system includes input clocks to the microprocessor as well as clock distribution for the various system memory and I/O devices. There are other miscellaneous clocks for ISA bus support. System clocking can be divided into the following three main areas:

- **Microprocessor input clock** — The input clock runs at the operating frequency of the 21164 microprocessor. The AlphaPC 164LX supports cycle times from 2.15 ns to 1.67 ns. This implies input clock frequencies from 466 MHz to 600 MHz. The clock is provided by using a synthesizer. The synthesizer's output is used as the input clock for the 21164.
- **Clock distribution** — Clock distribution includes the distribution of system clocks from the 21164 microprocessor to the system logic. The AlphaPC 164LX clock distribution scheme is flexible enough to allow the majority of cycle-time combinations to be supported. Because the PCI is synchronous to the system clock generated by the 21164 microprocessor, the PCI cycle time is a multiple of the 21164 cycle time. This distribution scheme supports a PCI operation of 33 MHz.
- **Miscellaneous clocks** — The miscellaneous clocks include those needed for ISA and the combination controller. These clocks are provided by a crystal and a frequency generator with fixed scaling.

Figure 4–7 AlphaPC 164LX System Clocks



At system reset, the 21164 microprocessor's **irq_h<3:0>** pins are driven by the clock divisor values set by four jumpers on J27. During normal operation, these signals are used for interrupt requests. The pins are either switched to ground or pulled up in a specific combination to set the 21164 microprocessor's internal divider.

The 21164 microprocessor produces the divided clock output signal **sys_clk_out1** that drives the CY2308 PLL clock-driver chip. This clock provides the references to synchronize the 21164 microprocessor and the 21174 chip. The 21174 provides the system memory and I/O (PCI) clock references. It also provides system-level clocking to DIMMs, PCI slots, the PCI-ISA bridge, the PCI ID controller, and the PCI arbiter.

A 14.3-MHz crystal produces the signal **14mhz_out**. This signal is delivered to the 37C935 combination controller for the diskette data separator and other I/O clocks. The combination controller produces output clock **osc**, which is then delivered to the two ISA slots and the PCI-to-ISA bridge for synchronization.

4.6 Reset and Initialization

A TL7702B power monitor senses the +3.3-V rail to ensure that it is stable before +2.5 V is applied to the 21164 microprocessor. In normal operation, should the +3.3-V rail fall below 2.5 V, the power monitor enables **shdn_l**, which turns off the +2.5-V regulator (*pc164lx.32*).

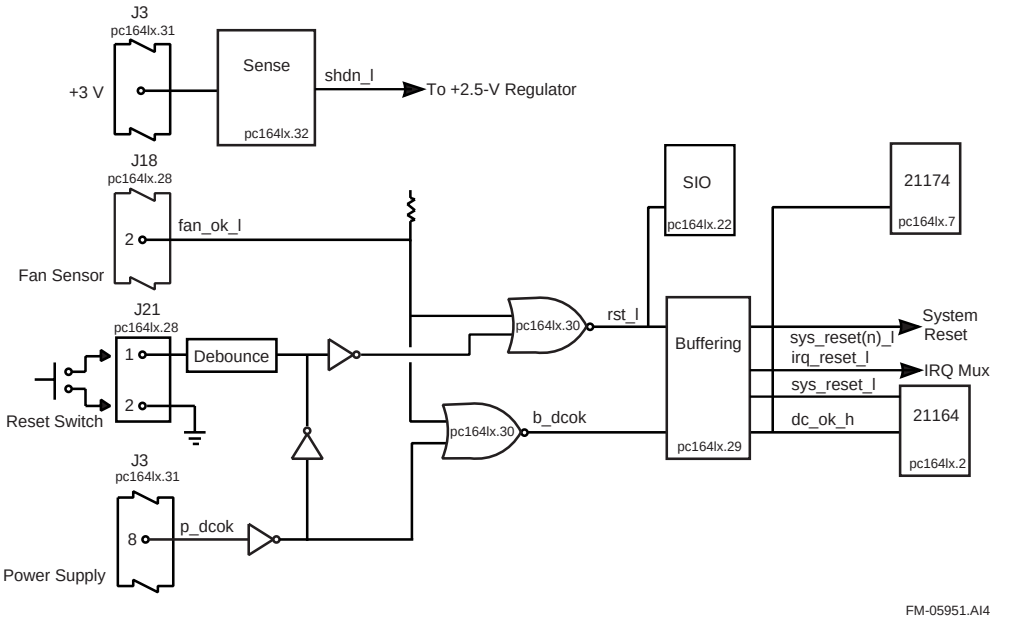
An external reset switch can be connected to J21 (*pc164lx.28*). The reset function initializes the 21164 microprocessor and the system logic. The **p_dcok** signal provides a full system initialization, equivalent to a power-down and power-up cycle.

In addition, the fan sense signal (**fan_ok_l**) is logically ORed with the reset switch output and, when enabled, drives **rst_l**, indicating a fan failure.

The **rst_l** signal is buffered and drives a set of **sys_reset** signals to reset the remainder of the system, including PCI and ISA devices through the 21174 chip.

Figure 4–8 shows the system reset and initialization logic.

Figure 4–8 System Reset and Initialization



4.7 Serial ROM

The serial ROM code is contained in the Xilinx XC17128 serial configuration ROM. This code is executed by the 21164 microprocessor when system power is turned on. The serial ROM code initializes the system, then transfers control to either the Mini-Debugger or the selected firmware, depending upon the setting of the configuration jumper CF7.

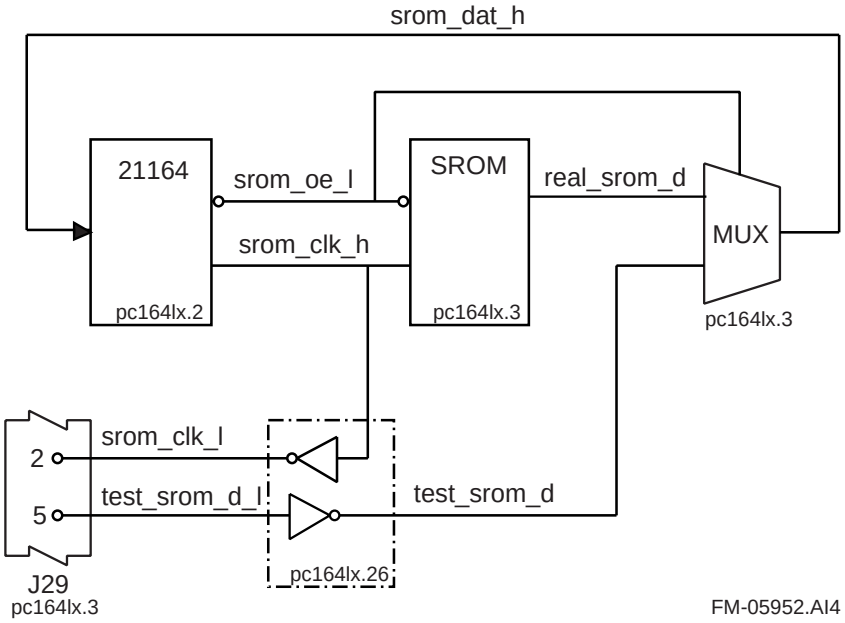
Figure 4–9 is a simplified diagram of the SROM and serial port logic.

Signal **srom_oe_l** selects the input to a multiplexer (*pc164lx.3*). The multiplexer selects either the output of the Xilinx XC17128 SROM (**real_srom_d**) or a user-supplied input through the test SROM port (**test_srom_d**). The multiplexer output (**srom_dat_h**) provides data input to the 21164 microprocessor.

After the initial SROM code has been read into the 21164 microprocessor's Icache, the test SROM port can be used as a software-controlled serial port. This serial port can be used for diagnosing system problems when the only working devices are the microprocessor, the SROM, and the circuits needed for the direct support of the

microprocessor and SROM (such as the clock). Connector J29 supports an RS-232 or RS-422 terminal connection to this port by using 1488 and 1489 line driver and receiver components. Additional external logic is not required.

Figure 4–9 Serial ROM



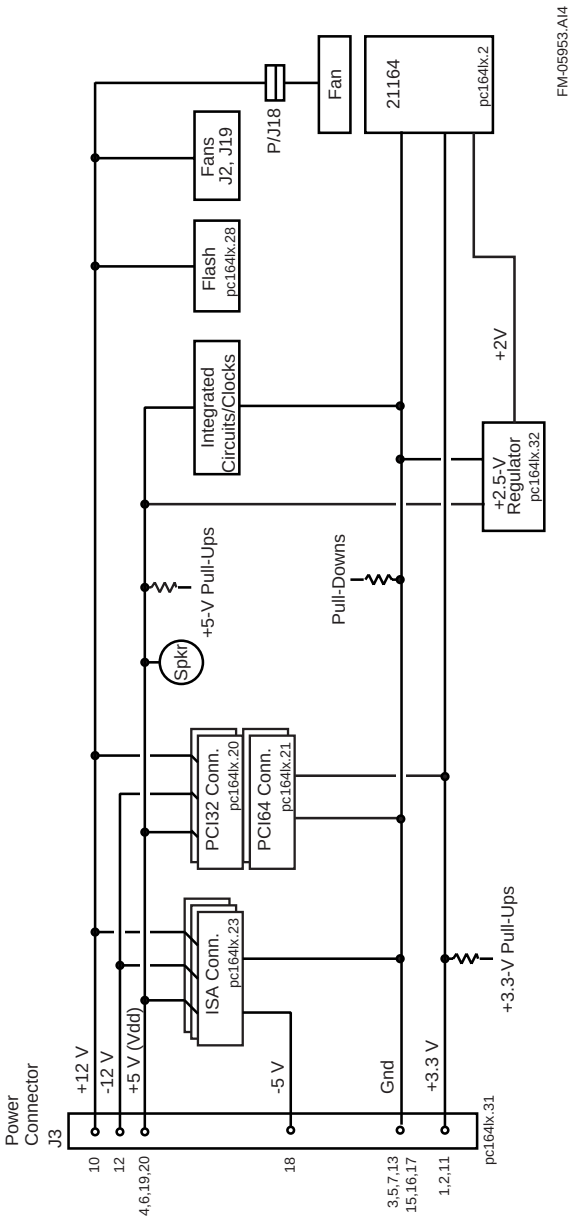
4.8 DC Power Distribution

The AlphaPC 164LX derives its system power from a user-supplied PC power supply. The power supply must provide +12 V dc and -12 V dc, -5 V dc, +3 V dc, and +5 V dc (**Vdd**). The dc power is supplied through power connector J3 (*pc164lx.31*), as shown in Figure 4–10. Power is distributed to the board logic through dedicated power planes within the eight-layer board structure.

Figure 4–10 shows that the +12 V dc, -12 V dc, and -5 V dc are supplied to ISA connectors J30 and J31 (*pc164lx.23*). The +12 V dc and -12 V dc are supplied to ISA connectors and PCI32 connectors J16 and J17 (*pc164lx.20*). The +12 V dc is also supplied to the CPU fan connector J18 (*pc164lx.28*), auxiliary fan connectors J2 and J19 (*pc164lx.28*), and to the flash ROM write-enable connector J28 (*pc164lx.28*).

Vdd (+5.0 V) is supplied to ISA connectors, PCI32 connectors, and most of the board's integrated circuits. **Vdd** also drives the +2.5-V regulator, which supplies the 21164 microprocessor.

Figure 4–10 AlphaPC 164LX Power Distribution



5
Upgrading the AlphaPC 164LX

For higher system speed or greater throughput, you can upgrade SDRAM memory by replacing DIMMs with those of greater size.

When configuring or upgrading SDRAM, observe the following rules:

- Each DIMM must be a 168-pin unbuffered version and have a frequency of 100 MHz.
- All DIMMs must be of equal size if they are in the same bank.

5.1 Configuring SDRAM Memory

Although not an exhaustive list, Table 5–1 lists the tested SDRAM memory configurations available.

For a list of vendors who supply components and accessories for the AlphaPC 164LX, see Appendix B.

Refer to Figure 2–1 for DIMM connector locations.

Note: 1Mb x 72 is not supported.

Table 5–1 AlphaPC 164LX SDRAM Memory Configurations *(Sheet 1 of 2)*

Total Memory	Bank 0 (J8 and J9)	Bank 1 (J10 and J11)
32MB	2Mb x 72	—
64MB	2Mb x 72	2Mb x 72
	4Mb x 72	—
96MB	4Mb x 72	2Mb x 72
128MB	4Mb x 72	4Mb x 72
	8Mb x 72	—

Table 5–1 AlphaPC 164LX SDRAM Memory Configurations *(Sheet 2 of 2)*

Total Memory	Bank 0 (J8 and J9)	Bank 1 (J10 and J11)
160MB	8Mb x 72	2Mb x 72
192MB	8Mb x 72	4Mb x 72
256MB	8Mb x 72	8Mb x 72
	16Mb x 72	—
512MB	16Mb x 72	16Mb x 72

5.2 Upgrading SDRAM Memory

You can upgrade memory in the AlphaPC 164LX by adding more DIMMs or replacing the ones that you have with a greater size.

Use the following general guidelines:

1. *Observe antistatic precautions.* Handle DIMMs only at the edges to prevent damage.
2. Remove power from the system.
3. Open levers and align the DIMM.
4. Firmly push the module into the connector. Ensure that the DIMM snaps into the plastic locking levers on both ends.
5. Restore power to the system.

5.3 Increasing Microprocessor Speed

This section describes how to complete the following actions to increase microprocessor speed:

- Replace the DIGITAL Semiconductor 21164 microprocessor with an Alpha chip that has a higher speed rating.
- Reconfigure the clock divisor jumpers.

5.3.1 Preparatory Information

Caution: Static-Sensitive Component – Due to the sensitive nature of electronic components to static electricity, anyone handling the microprocessor *must* wear a properly grounded antistatic wriststrap. Use of antistatic mats, ESD approved workstations, or exercising other good ESD practices is recommended.

A DIGITAL Semiconductor 21164 microprocessor with a higher speed rating is available from your local distributor. See Appendix B for information about supporting products.

When replacing the microprocessor chip, also replace the thermal conducting GRAFOIL pad. See Appendix B for information about the parts kit, which includes the heat sink, GRAFOIL pad, two hex nuts, heat-sink clips, 60-mm fan, fan guard, and four screws.

5.3.2 Required Tools

The following tools are required when replacing the microprocessor chip:

A TS30 manual nut/torque driver (or equivalent) with the following attachments is required to affix the heat sink and fan to the microprocessor's IPGA package:

- 1/4-inch hex bit
- 7/16-inch socket with 1/4-inch hex drive
- #2 Phillips-head screwdriver bit

5.3.3 Removing the 21164 Microprocessor

Remove the microprocessor currently in place at location U40 by performing the following steps:

1. Unplug the fan power/sensor cable from connector J18 (see Figure 2–1).
2. Remove the four 6-32 X 0.875-inch screws that secure the fan and fan guard to the heat sink.
3. Remove the fan and fan guard.
4. If the sink/chip/fan clip is used, remove it by unhooking its ends from around the ZIF socket retainers.

5. Using a 7/16-inch socket, remove the two nuts securing the heat sink to the microprocessor studs.
6. Remove the heat sink by gently lifting it off the microprocessor.
7. Remove and discard the GRAFOIL heat conduction pad.
8. Thoroughly clean the bottom surface of the heat sink before affixing it to the new microprocessor.
9. Lift the ZIF socket actuator handle to a full 90° angle.
10. Remove the microprocessor chip by lifting it straight out of the socket.

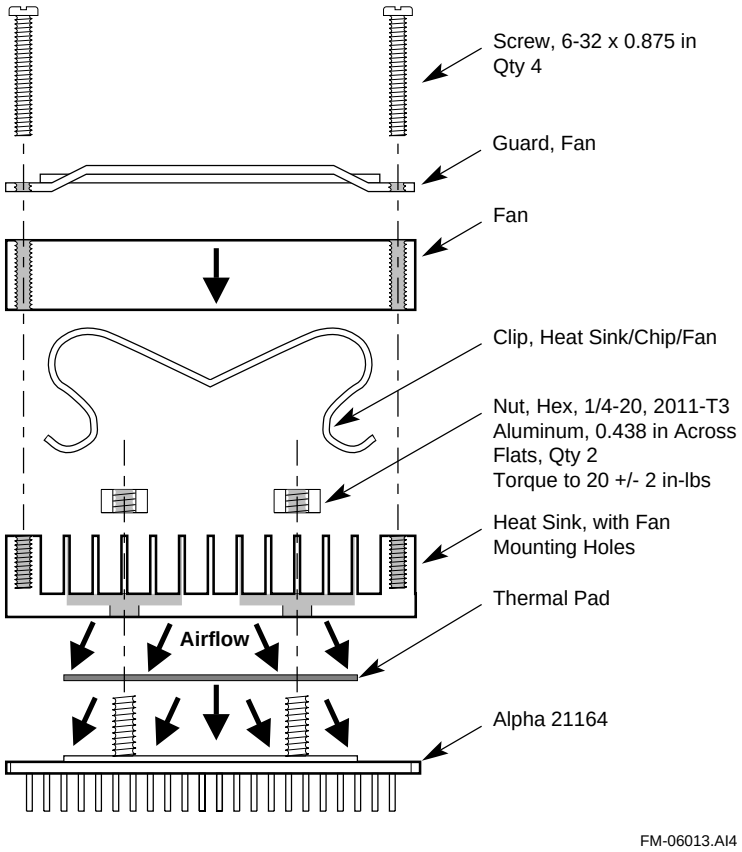
5.3.4 Installing the 21164 Microprocessor

Install the new microprocessor in location U40 by performing the following steps:

Note: Install the heat sink only after the microprocessor has been assembled to the ZIF socket.

1. Observe antistatic precautions.
2. Lift the ZIF socket actuator handle to a full 90° angle.
3. Ensure that all the pins on the microprocessor package are straight.
4. The ZIF socket and microprocessor are keyed to allow for proper installation. Align the microprocessor, with its missing AD01 pin, with the corresponding plugged AD01 position on the ZIF socket. Gently lower into position.
5. Close the ZIF socket actuator handle to its locked position.
6. Install the heat sink and heat-sink fan as directed in the following steps. A heat-sink/fan kit is available from the vendor listed in Appendix B. Refer to Figure 5–1 for heat-sink and fan assembly details.

Figure 5–1 Fan/Heat-Sink Assembly



a. Put the GRAFOIL thermal pad in place. The GRAFOIL pad is used to improve the thermal conductivity between the chip package and the heat sink by replacing micro air pockets with a less insulative material. Perform the following steps to position the GRAFOIL pad:

1. Perform a visual inspection of the package slug to ensure that it is free of contamination.
2. Wearing clean gloves, pick up the GRAFOIL pad. *Do not* perform this with bare hands because skin oils can be transferred to the pad.
3. Place the GRAFOIL pad on the gold-plated slug surface and align it with the threaded studs.

b. Attach the microprocessor heat sink. The heat-sink material is clear anodized, hot-water-sealed, 6061-T6 aluminum. The nut material is 2011-T3 aluminum (this grade is critical). Perform the following steps to attach the heat sink:

1. Observe antistatic precautions.
2. Align the heat-sink holes with the threaded studs on the ceramic package.
3. Handle the heat sink by the edges and lower it onto the chip package, taking care not to damage the stud threads.
4. Set a calibrated torque driver to 20 in-lbs, ±2 in-lbs (2.3 Nm, ±0.2 Nm). The torque driver should have a mounted 7/16-inch socket.
5. Insert a nut into the 7/16-inch socket, place on one of the studs, and tighten to the specified torque. Repeat for the second nut.
6. If the sink/chip/fan clip is used, properly install it by positioning it over the assembly and hooking its ends around the ZIF socket retainers.

c. Attach the heat-sink fan assembly:

1. Place the fan assembly on top of the heat sink, aligning the fan mounting holes with the corresponding threaded heat-sink holes. Align the fan so that the fan power/sensor wires exit the fan closest to connector J18 (see Figure 2–1). Fan airflow must be directed into the heat sink (fan label facing down toward the heat sink).
2. Place the fan guard on top of the fan. Orient the guard so that the corner mounting areas lay flush against the heat sink.
3. Secure the fan and fan guard to the heat sink with four 6-32 X 0.875-inch screws.
4. Plug the fan power/sensor cable into connector J18.

Important: When installing the microprocessor, you must change the frequency of its clock output by setting the system clock divisor jumpers, as described in Section 2.2.

A
System Address Space

This appendix describes the mapping of 21164 40-bit physical addresses to memory and I/O space addresses. It also describes the translation of a 21164-initiated address (**addr_h<39:4>**) into a PCI address (**ad<63:0>**) and the translation of a PCI-initiated address into a physical memory address.

PCI addressing topics include dense and sparse address space and scatter-gather address translation for DMA operations.

A.1 Address Map

The system address mapping operates with byte/word transactions enabled or disabled. Byte/word operation is controlled by PYXIS_CTRL1<0> (IOA_BEN). Table A–1 shows system address mapping operations when IOA_BEN equals 0 (byte/word operation disabled).

Table A–1 Physical Address Map (Byte/Word Mode Disabled) *(Sheet 1 of 2)*

21164 Address[1]	Size (GB)	Selection
00.000.0000 – 01.FFFF.FFFF	8.00	Main memory
E.0000.0000 – E.FFFF.FFFF	4.00	Dummy memory region
80.0000.0000 – 83.FFFF.FFFF	16.00	PCI sparse memory region 0, 512MB
84.0000.0000 – 84.FFFF.FFFF	4.00	PCI sparse memory region 1, 128MB
85.0000.0000 – 85.7FFF.FFFF	2.00	PCI sparse memory region 2, 64MB
85.8000.0000 – 85.BFFF.FFFF	1.00	PCI sparse I/O space region A, 32MB
85.C000.0000 – 85.FFFF.FFFF	1.00	PCI sparse I/O space region B, 32MB
86.0000.0000 – 86.FFFF.FFFF	4.00	PCI dense memory
87.0000.0000 – 87.1FFF.FFFF	0.50	PCI sparse configuration space

Table A–1 Physical Address Map (Byte/Word Mode Disabled) *(Sheet 2 of 2)*

21164 Address[1]	Size (GB)	Selection
87.2000.0000 – 87.3FFF.FFFF	0.50	PCI special/interrupt acknowledge
87.4000.0000 – 87.4FFF.FFFF	0.25	21174 main CSRs
87.5000.0000 – 87.5FFF.FFFF	0.25	21174 memory control CSRs
87.6000.0000 – 87.6FFF.FFFF	0.25	21174 PCI address translation
87.7000.0000 – 87.7FFF.FFFF	0.25	Reserved
87.8000.0000 – 87.8FFF.FFFF	0.25	21174 miscellaneous CSRs
87.9000.0000 – 87.9FFF.FFFF	0.25	21174 power management CSRs
87.A000.0000 – 87.AFFF.FFFF	0.25	21174 interrupt control CSRs
87.B000.0000 – 87.FFFF.FFFF	1.25	Reserved

[1] All addresses in the range of 80.0000.0000 and 8F.FFFF.FFFF are aliased. Address bits 36 through 38 are ignored in the address.

Table A–2 shows system address mapping operations when IOA_BEN equals 1 (byte/word operation enabled).

Table A–2 Physical Address Map (Byte/Word Mode Enabled) *(Sheet 1 of 2)*

21164 Address	Size (GB)	Selection
00.000.0000 – 01.FFFF.FFFF	8.00	Main memory
E.0000.0000 – E.FFFF.FFFF	4.00	Dummy memory region
80.0000.0000 – 83.FFFF.FFFF	16.00	PCI sparse memory region 0, 512MB
84.0000.0000 – 84.FFFF.FFFF	4.00	PCI sparse memory region 1, 128MB
85.0000.0000 – 85.7FFF.FFFF	2.00	PCI sparse memory region 2, 64MB
85.8000.0000 – 85.BFFF.FFFF	1.00	PCI sparse I/O space region A, 32MB
85.C000.0000 – 85.FFFF.FFFF	1.00	PCI sparse I/O space region B, 32MB
86.0000.0000 – 86.FFFF.FFFF	4.00	PCI dense memory
87.0000.0000 – 87.1FFF.FFFF	0.50	PCI sparse configuration space
87.2000.0000 – 87.3FFF.FFFF	0.50	PCI special/interrupt acknowledge
87.4000.0000 – 87.4FFF.FFFF	0.25	21174 main CSRs
87.5000.0000 – 87.5FFF.FFFF	0.25	21174 memory control CSRs

Table A–2 Physical Address Map (Byte/Word Mode Enabled) *(Sheet 2 of 2)*

21164 Address	Size (GB)	Selection
87.6000.0000 – 87.6FFF.FFFF	0.25	21174 PCI address translation
87.7000.0000 – 87.7FFF.FFFF	0.25	Reserved
87.8000.0000 – 87.8FFF.FFFF	0.25	21174 miscellaneous CSRs
87.9000.0000 – 87.9FFF.FFFF	0.25	21174 power management CSRs
87.A000.0000 – 87.AFFF.FFFF	0.25	21174 interrupt control CSRs
87.B000.0000 – 87.BFFF.FFFF	0.25	Reserved
88.0000.0000 – 88.FFFF.FFFF	4.00	PCI memory space INT8
98.0000.0000 – 98.FFFF.FFFF[1]	4.00	PCI memory space INT4
A8.0000.0000 – A8.FFFF.FFFF[1]	4.00	PCI memory space INT2
B8.0000.0000 – B8.FFFF.FFFF[1]	4.00	PCI memory space INT1
89.0000.0000 – 89.FFFF.FFFF	4.00	PCI I/O space INT8
99.0000.0000 – 99.FFFF.FFFF[1]	4.00	PCI I/O space INT4
A9.0000.0000 – A9.FFFF.FFFF[1]	4.00	PCI I/O space INT2
B9.0000.0000 – B9.FFFF.FFFF[1]	4.00	PCI I/O space INT1
8A.0000.0000 – 8A.FFFF.FFFF	4.00	PCI configuration space, type 0, INT8
9A.0000.0000 – 9A.FFFF.FFFF[1]	4.00	PCI configuration space, type 0, INT4
AA.0000.0000 – AA.FFFF.FFFF[1]	4.00	PCI configuration space, type 0, INT2
BA.0000.0000 – BA.FFFF.FFFF[1]	4.00	PCI configuration space, type 0, INT1
8B.0000.0000 – 8B.FFFF.FFFF	4.00	PCI configuration space, type 1, INT8
9B.0000.0000 – 9B.FFFF.FFFF[1]	4.00	PCI configuration space, type 1, INT4
AB.0000.0000 – AB.FFFF.FFFF[1]	4.00	PCI configuration space, type 1, INT2
BB.0000.0000 – BB.FFFF.FFFF[1]	4.00	PCI configuration space, type 1, INT1
C7.C000.0000 – C7.FFFF.FFFF[2]	1.00	Flash ROM read/write space

[1] Address bits 37 and 38 are generated by the 21164 and not by software. These address bits are used by the 21164 to indicate to external hardware that this transaction is a byte, word, longword, or quadword operation.

[2] Read/write transactions to flash ROM must be done with byte transactions to address range 87.C000.0000 through 87.FFFF.FFFF. All other transaction types will produce UNDEFINED results.

The 21164 address space is divided into two regions using physical address <39>:

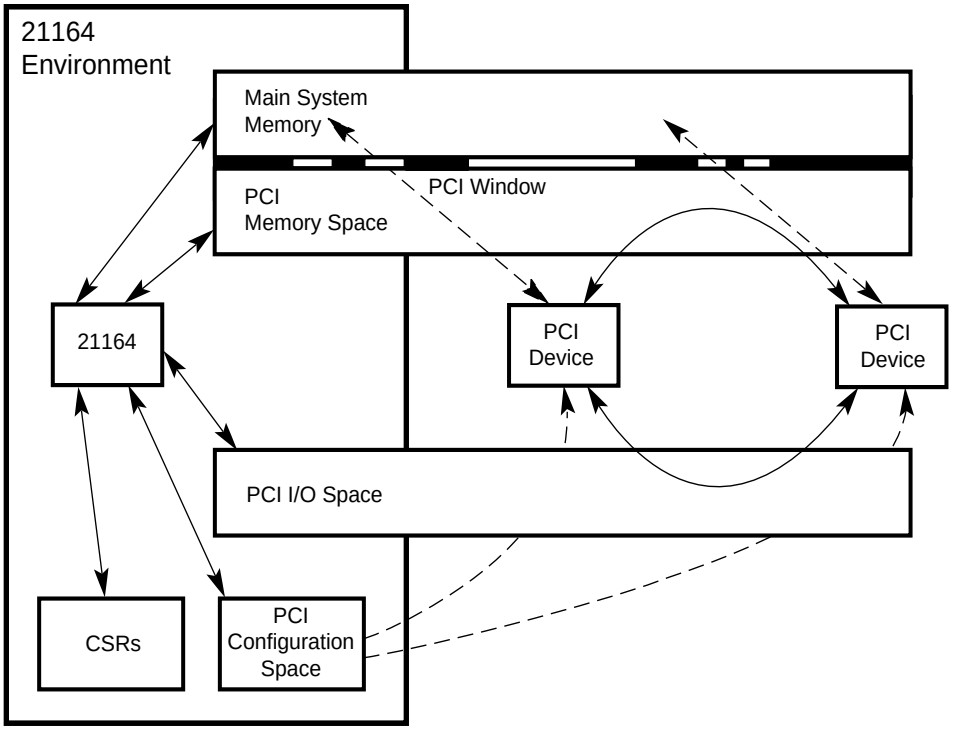
- 0 – 21164 access is to the cached memory space.
- 1 – 21164 access is to noncached space. This noncached space is used to access memory-mapped I/O devices. Mailboxes are not supported.

The noncached space contains the CSRs, noncached memory space (for diagnostics), and the PCI address space. The PCI defines three physical address spaces: a 64-bit PCI memory space, a 4GB PCI I/O space, and a 256 byte-per-device PCI configuration space. In addition to these three address spaces on the PCI, the 21164's noncached space is also used to generate PCI interrupt acknowledge and special cycles.

The 21164 has visibility to the complete address space. It can access the cached memory region, the CSR region, the PCI memory region, the PCI I/O region, and the configuration regions (see Figure A–1).

The PCI devices have a restricted view of the address space. They can access any PCI device through the PCI memory space or the PCI I/O space; but they have no access to the PCI configuration space. The system restricts access to the system memory (for DMA operations) to the use of five programmable windows in the PCI memory space (see Figure A–1).

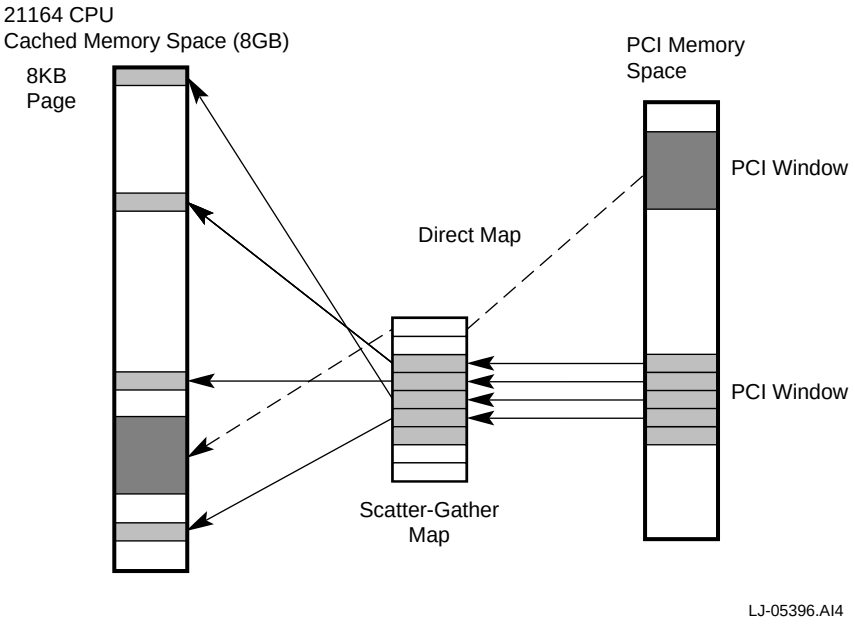
Figure A–1 Address Space Overview



DMA access to the system memory is achieved using windows in one of the following three ways:

- Directly, using the "Monster Window" with dual-address cycles (DAC), where **ad<33:0>** equals **addr_h<33:0>**.
- Directly-mapped, by concatenating an offset to a portion of the PCI address.
- Virtually, through a scatter-gather translation map. The scatter-gather map allows any 8KB page of PCI memory address region to be redirected to any 8KB cached memory page, as shown in Figure A–2.

Figure A–2 Memory Remapping



A.2 PCI Address Space

The system generates 32-bit PCI addresses but accepts both 64-bit address (DAC[1]) cycles and 32-bit PCI address (SAC[2]) cycles. Accessing main memory is as follows:

- Window 4, the "Monster Window," provides full access to main memory. It is accessed by DAC only with **ad<40>** equal to 1. Memory address **addr_h<33:0>** equals PCI address **ad<33:0>**.
- Window 3 can be either DAC or SAC, but not both. If DAC, **ad<63:40>** must be zero, **ad<39:32>** must match the DAC register, and **ad<31:0>** must hit in window 3.
- Windows 0, 1, and 2 are SAC-only.

---

1 Dual-address cycle (PCI 64-bit address transfer) requires that address bits <63:32> contain a nonzero value.

2 Single-address cycle (PCI 32-bit address transfer) requires that address bits <63:32> contain a value of zero.

A.3 21164 Address Space

Figure A–3 shows an overview of the 21164 address space. Figure A–4 shows how the 21164 address map translates to the PCI address space and how PCI devices access the 21164 memory space using DMA transactions. The PCI memory space is double mapped via dense and sparse space.

The 21164 I/O address map has the following characteristics:

- Provides 4GB of dense[3] address space to completely map the 32-bit PCI memory space.
- Provides abundant PCI sparse[3] memory address space because sparse-space regions have byte granularity and is the safest memory space to use (that is, no prefetching). Furthermore, the larger the space the less likely software will need to dynamically relocate the sparse-space segments. The main problem with sparse space is that it wastes 21164 address space (for example, 16GB of 21164 address space maps to 512MB of PCI sparse space).

  The system provides three PCI sparse-space memory regions, allowing 704MB of total sparse-space memory. The three regions are relocatable using the HAE_MEM CSR. The simplest configuration allows for 704MB of contiguous memory space.

  – 512MB region, which may be located in any naturally aligned 512MB segment of the PCI memory space. Software programmers may find this region sufficient for their needs and can ignore the remaining two regions.

  – 128MB regions, which may be located on any naturally aligned 128MB segment of the PCI memory space.

  – 64MB region, which may be located on any naturally aligned 64MB segment of the PCI memory space.
- Limits the PCI I/O space to sparse space. Although the PCI I/O space can handle 4GB, most PCI devices will not exceed 64KB for the foreseeable future. The system provides 64MB of sparse I/O space because address decoding is faster.
- Provides two PCI I/O sparse-space regions: region A, which is 32MB and is fixed in PCI segment 0–32MB; and region B, which is also 32MB, but is relocatable using the HAE_IO register.

---

[3] Dense and sparse space address space are described later in this chapter.

Figure A–3 21164 Address Space Configuration

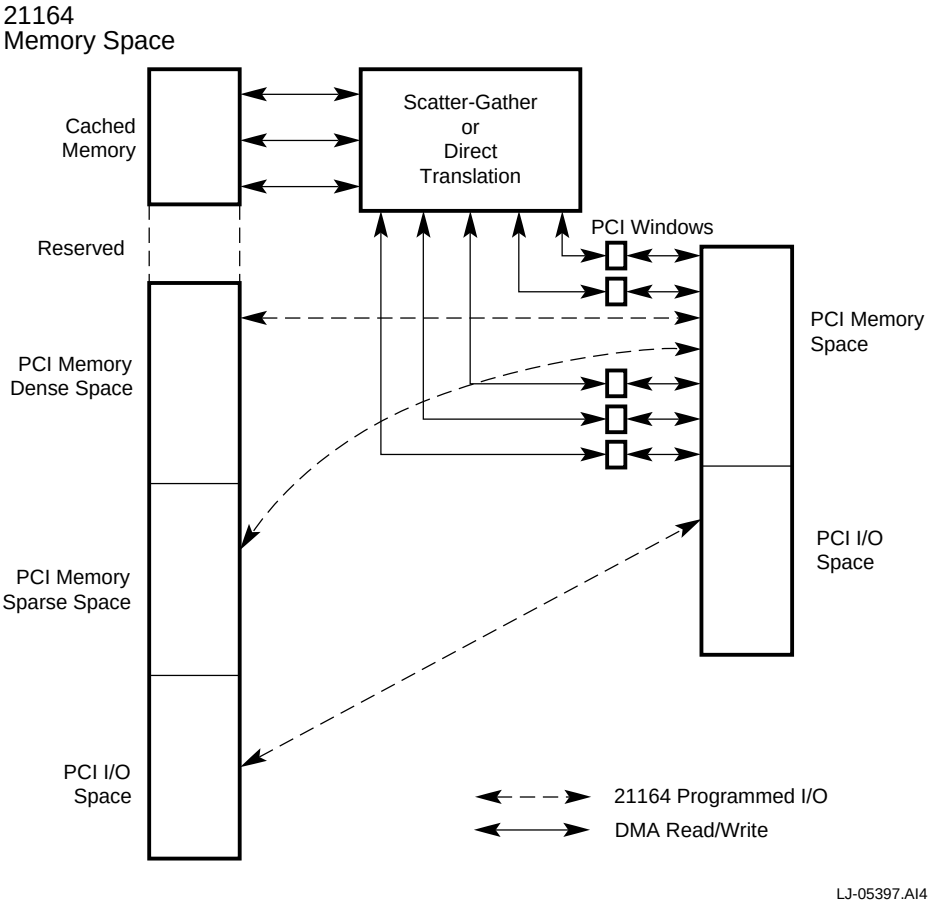
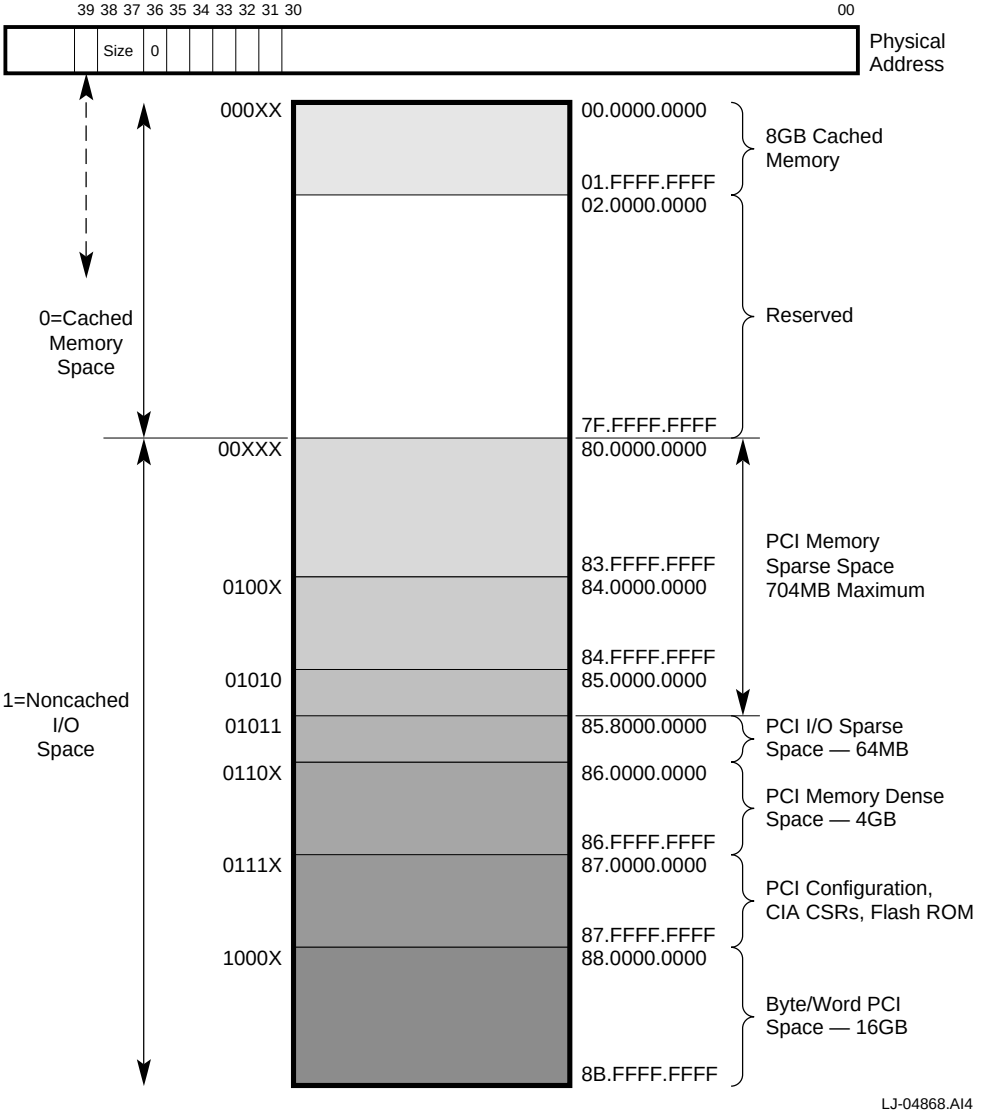


Figure A–4 21164 and DMA Read and Write Transactions



A.3.1 System Address Map

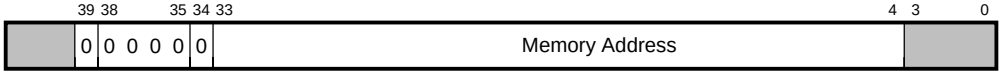
Figure A–5 shows the following system address regions:

- Main memory address space contains 8GB. All transactions contain 64 bytes, are cache-block aligned, and are placed in cache by the 21164. Both Istream and Dstream transactions access this address space.
- PCI sparse-space memory region 1 contains 512MB. Noncached 21164 read/write transactions are allowed, including byte, word, tribyte, longword (LW), and quadword (QW) types. There is no read prefetching.
- PCI sparse-space memory region 2 contains 128MB.
- PCI sparse-space memory region 3 contains 64MB.
- PCI I/O sparse-space memory region A contains 32MB and is not relocatable.
- PCI I/O sparse-space memory region B contains 32MB and is relocatable by way of the HAE_IO register.
- PCI dense memory space contains 4GB for 21164 noncached 21164 transactions. It is used for devices with access granularity greater or equal to a LW. Read prefetching is allowed, and thus read transactions can have no side effects.
- The PCI configuration space is used for noncached 21164 access. Sparse-space read/write transactions are allowed, including byte, word, tribyte, LW, and QW types. Prefetching of read data is not allowed.

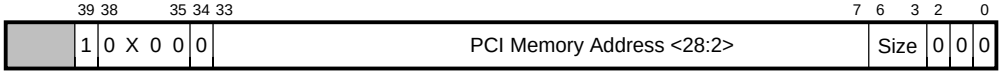
  Figure A–6 shows a detailed view of PCI configuration space that includes 21174 CSRs. The 21174 CSR address space is chosen for hardware convenience.

Figure A–5 System Address Map

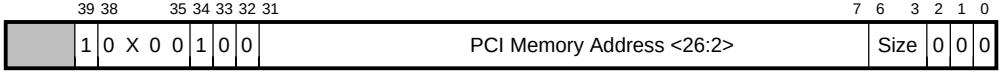
Main Memory — 8GB



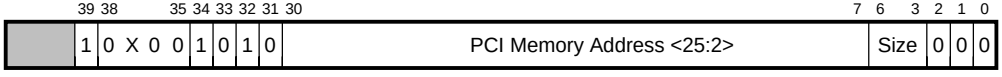
PCI Sparse Memory Space — 512MB Region 1



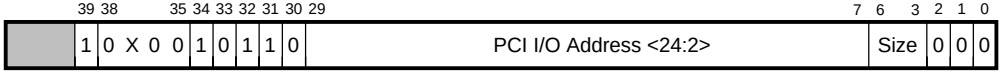
PCI Sparse Memory Space — 128MB Region 2



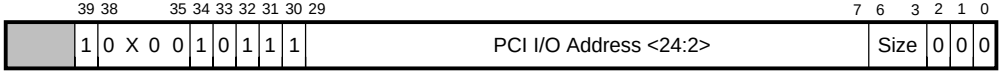
PCI Sparse Memory Space — 64MB Region 3



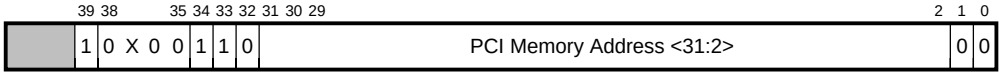
PCI I/O Sparse Space — 32MB Region A



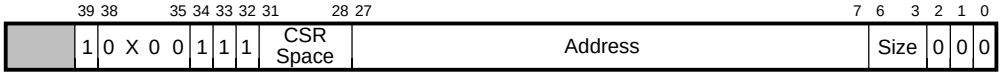
PCI I/O Sparse Space — 32MB Region B



PCI Memory Dense Space — 4GB

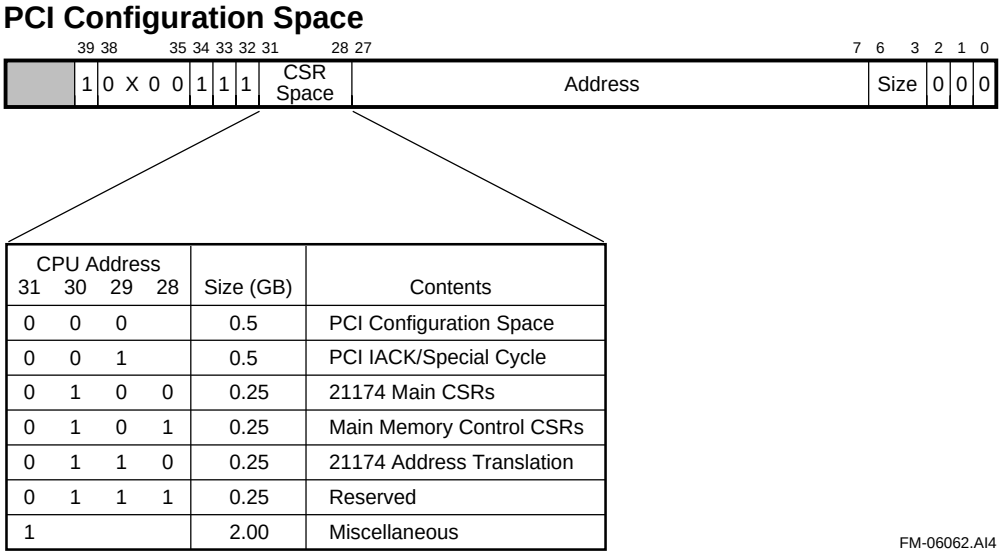


PCI Configuration Space



LJ-05398.AI4

Figure A–6 21174 CSR Space



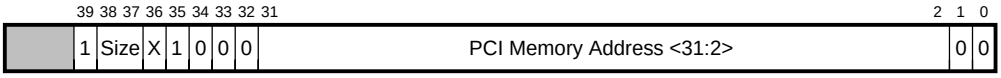
CPU Address 31 30 29 28	Size (GB)	Contents
0 0 0	0.5	PCI Configuration Space
0 0 1	0.5	PCI IACK/Special Cycle
0 1 0 0	0.25	21174 Main CSRs
0 1 0 1	0.25	Main Memory Control CSRs
0 1 1 0	0.25	21174 Address Translation
0 1 1 1	0.25	Reserved
1	2.00	Miscellaneous

A.4 21164 Byte/Word PCI Space

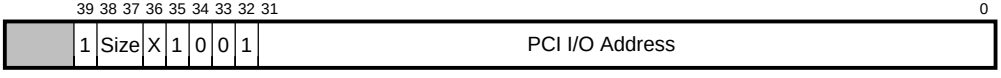
The 21164 supports byte/word instructions that allow software to perform byte granularity transactions to and from I/O space without using sparse address space. This space is divided into four regions: memory, I/O, configuration – type 0, and configuration – type 1, as shown in Figure A–7.

Figure A–7 Byte/Word PCI Space

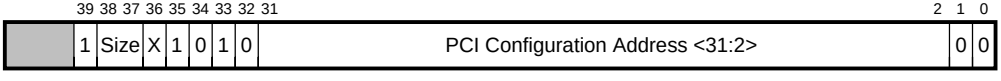
PCI Memory Space — 4GB



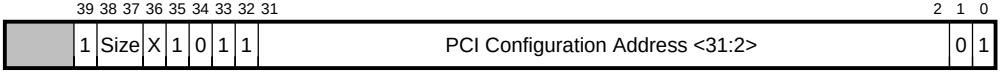
PCI I/O Space — 4GB



PCI Type 0 Configuration Space — 4GB



PCI Type 1 Configuration Space — 4GB



LJ-05399.AI4

Operations are the same for the four regions. The 21164 will issue a single byte/word read or write transaction for PCI byte and word instructions. The 21164 will not pack longword load instructions. The 21164 can pack up to eight longword store instructions for a single 32-byte block into one transaction. Up to four quadword instructions can also be packed to the same 32-byte block. Byte/word support is enabled when 21164 IPR register ICSR<17> equals 1 and when 21174 CSR register PYXIS_CTRL1<0> also equals 1.

Table A–3 shows noncached 21164 addresses when byte/word support is enabled.

Table A–3 21164 Byte/Word Addressing

Instruction	addr_h <38:37>	int4_valid <3>	<2>	<1>	<0>
LDQ	00	INT8	—	—	—
LDL	01	**addr_h<3:2>**	—	Undefined	—
LDWU	10	**addr_h<3:1>**	—	—	Undefined
LDBU	11	**addr_h<3:0>**	—	—	—
STQ	00	INT4 Mask	—	—	—
STL	01	INT4 Mask	—	—	—
STW	10	**addr_h<3:1>**	—	—	Undefined
STB	11	**addr_h<3:0>**	—	—	—

A.4.1 21164 Size Field

Table A–4 shows the calculation of the 21164 size field.

Table A–4 21164 Byte/Word Translation Values

Size<38:37>	Data Size
00	INT8 (Quadword — 8 bytes, 64 bits)
01	INT4 (Longword — 4 bytes, 32 bits)
10	INT2 (Word — 2 bytes, 16 bits)
11	INT1 (Byte — 1 byte, 8 bits)

The following transactions use single data transfers on the PCI:

- INT1 and INT2 read and write transactions
- INT4 read transactions

The following transactions have multiple data transfers on the PCI:

- INT4 write transactions
- INT8 read and write transactions

A.5 Cacheable Memory Space

Cacheable memory space is located in the range 00.0000.0000 to 01.FFFF.FFFF. The 21174 recognizes the first 8GB to be in cacheable memory space. The block size is fixed at 64 bytes. Read and flush commands to the 21164 caches occur for DMA traffic.

A.6 PCI Dense Memory Space

PCI dense memory address space is located in the range 86.0000.0000 to 86.FFFF.FFFF. This address space is typically used for memory-like data buffers such as a video frame buffer or a nonvolatile RAM (NVRAM). Dense space does not allow byte or word access, but has the following advantages over sparse space:

- Contiguous locations — Some software, such as the default graphics routines of the Windows NT operating system, requires memory-like transactions. These routines cannot use sparse-space addresses, because they require transactions on the PCI bus to be at adjacent 21164 addresses, instead of being widely separated as in sparse space. As a result, if the user-mode driver manipulates its frame buffer in sparse space, it cannot hand over the buffer to the common Windows NT operating system graphics code.
- Higher bus bandwidth — PCI bus burst transfers are not usable in sparse space except for a 2-longword burst for quadword write transactions. Dense space is defined to allow both burst read and write transactions.
- Efficient read/write buffering — In sparse space, separate transactions use separate read or write buffer entries. Dense space allows separate transactions to be collapsed in read and write buffers (as the 21164 does).
- Few memory barriers (MBs) — In general, sparse-space transactions are separated by MB instructions to avoid read/write buffer collapsing. Dense-space transactions only require barriers when explicit ordering is required by the software.

Dense space is provided for the 21164 to access PCI memory space, not for access to PCI I/O space. Dense space has the following characteristics:

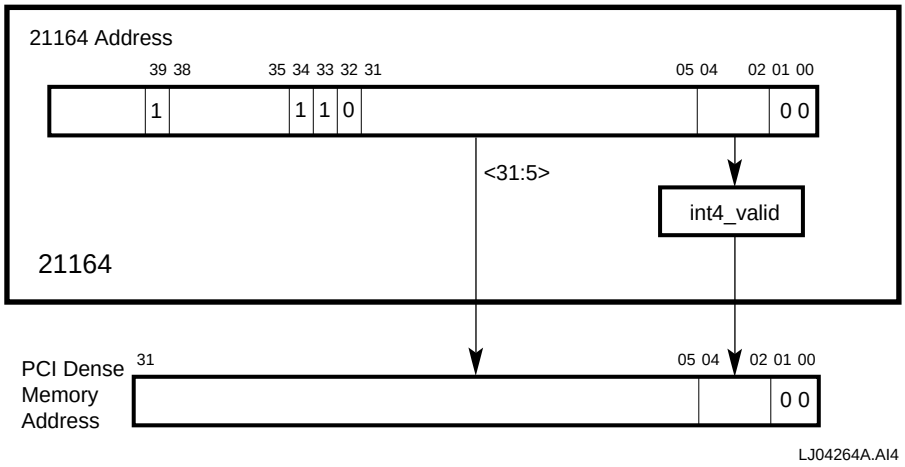
- It holds a one-to-one mapping between 21164 addresses and PCI addresses. A longword address from the 21164 will map to a longword on the PCI with no shifting of the address field. Hence, the term dense space. Sparse space, on the other hand, maps a large piece of 21164 memory space (32 bytes) to a small piece (such as a byte) on the PCI.

- The concept of dense space (and sparse space) is applicable only to a 21164-generated address. There is no such thing as dense space (or sparse space) for a PCI generated address.
- Byte or word transactions are not possible in dense space. The minimum access granularity is a longword on write transactions and a quadword on read transactions. The maximum transfer length is 32 bytes (performed as a burst of eight longwords on the PCI). Any combination of longwords may be valid on write transactions. Valid longwords surrounding an invalid longword(s) (called a hole) are required to be handled correctly by all PCI devices. The 21174 will allow such holes to be issued.
- Read transactions will always be performed as a burst of two or more longwords on the PCI because the minimum granularity is a quadword. The 21164 can request a longword but the 21174 will always fetch a quadword, thus prefetching a second longword. Therefore, this space cannot be used for devices that have read side effects. Although a longword may be prefetched, the prefetch buffer is not treated as a cache and so coherency is not an issue. A quadword read transaction is not atomic on the PCI; that is, the target device is at liberty to force a retry after the first longword of data is sent, and then to allow another PCI device to take control of the PCI bus[4].
- The 21164 merges noncached reads of up to 32 bytes maximum. The largest dense-space read transaction is 32 bytes from the PCI bus.
- Write transactions to dense space are buffered in the 21164 chip. The 21174 supports a burst length of 8 on the PCI, corresponding to 32 bytes of data. Also, the 21174 provides four 32-byte write buffers to maximize I/O write transaction performance. These four buffers are strictly ordered. Write transactions are sent out on the bus in the order that they were received from the 21164. Avoid write buffer merging and use memory barrier (MB) and write memory barrier (WMB) instructions carefully.

[4] The 21174 does not drive the PCI lock signal and this cannot ensure atomicity. This is true of all current Alpha microprocessors.

Figure A–8 shows dense-space address generation.

Figure A–8 Dense-Space Address Generation



The following list describes address generation in dense space:

- **addr_h<31:5>** value is sent directly out on **ad<31:5>**.
- **addr_h<4:2>** is not sent out by the 21164 and instead is inferred from the **int4_valid<3:0>**.
- **ad<4:3>** is a copy of **addr_h<4:3>**.
- **ad<2>** differs for read and write transactions as follows:
  - For a read transaction, **ad<2>** is zero (that is, the minimum read transaction resolution in noncached space is a quadword).
  - For a write transaction, **ad<2>** equals **addr_h<2>**.

A.7 PCI Sparse Memory Space

The system provides three regions of contiguous 21164 address space that maps to PCI sparse memory space. The total 21164 range is from 80.0000.0000 to 85.7FFF.FFFF.

A.7.1 Hardware Extension Register (HAE_MEM)

In sparse space, **addr_h<7:3>** are used to encode byte enable bits, size bits and the low-order PCI address, **ad<2:0>**. This means that there are now five fewer address bits available to generate the PCI physical address.

The system provides three sparse-space PCI memory regions and allows all three sparse-space regions to be relocated by way of bits in the HAE_MEM register. This provides software with great flexibility.

A.7.2 Memory Access Rules and Operation

The Alpha instruction set can express only aligned longword and quadword data references. The PCI bus requires the ability to express byte, word, tribyte, longword (double word), and quadword references. Intel processors are capable of generating unaligned references, so the 21174 should be able to emulate the resulting PCI transactions to ensure compatibility with PCI devices designed for Intel systems.

The size of the data transfer (byte, word, tribyte, longword, or quadword) and the byte enables are encoded in the 21164 address. The 21164 signals **addr_h<6:3>** are used for this purpose, leaving the remaining **addr_h<31:7>** signals to generate a PCI longword address <26:3>[5]. This loss of address bits has resulted in a 21164 22GB sparse 32-bit address space that maps to only 704MB of address space on the PCI.

The rules for accessing sparse space are as follows:

- Sparse space supports all the byte encodings that may be generated in an Intel system to ensure compatibility with PCI devices/drivers. The results of some references are not explicitly defined. These are the missing entries in Table A–6 (that is, word size with address<6:5> = 11). The hardware will complete the reference, but the reference is not required to produce any particular result, nor will the system report an error.
- Software must use longword load or store instructions (LDVSTL) to perform a reference of longword length or less on the PCI bus. The bytes to be transferred must be positioned within the longword in the correct byte lanes as indicated by the PCI byte enable bits. The hardware does not shift bytes within the longword. Quadword load and store instructions must be used only to perform quadword transfers. Use of STQ/LDQ instructions for any other references will produce UNPREDICTABLE results.

---

[5] Quadword encoding is provided by way of 21164 address bits <6:3>. In this case, 21164 address bit <7> is treated as zero by the hardware.

- Hardware does not perform read-ahead (prefetch) transactions in sparse space because read-ahead transactions may have detrimental side effects.
- Programmers are required to insert memory barrier (MB) instructions between sparse-space transactions to prevent collapsing in the 21164 write buffer. However, this is not always necessary. For example, consecutive sparse-space addresses will be separated by 32 bytes (and will not be collapsed by the 21164).
- Programmers are required to insert MB instructions if the sparse-space address ordering/coherency to a dense-space address is to be maintained.
- Table A–6 shows encoding of the 21164 address for sparse-space read transactions to PCI space. An important point to note is that signals **addr_h<33:5>** are directly available from the 21164 pins. On read transactions, the 21164 sends out **addr_h<2:0>** indirectly on the **int4_valid** pins. Signals **addr_h<2:0>** are required to be zero. Transactions with **addr_h<2:0>** not equal to zero will produce UNPREDICTABLE results.
- Table A–5 shows the relation between **int4_valid<3:0>** and **addr_h<4:3>** for a sparse-space write transaction. Unlisted **int4_valid** patterns will produce UNPREDICTABLE results (that is, as a result of collapsing in the 21164 write buffer; or by issuing a STQ instruction when a STL instruction is required).

Table A–5 Int4_valid and 21164 Address Relationship

EV5 Data Cycle	Int4_valid<3:0>[1]	Address<4:3>
First	00 01	0 0
	00 10	0 0
	01 00	0 1
	10 00	0 1
Second	00 01	1 0
	00 10	1 0
	01 00	1 1
	10 00	1 1
	11 00 (STQ)[2]	1 1

[1] All other **int4_valid** patterns result in UNPREDICTABLE results.
[2] Only one valid STQ case is allowed.

Table A–6 defines the low-order PCI sparse memory address bits. Signals **addr_h<7:3>** are used to generate the length of the PCI transaction in bytes, the byte enable bits, and **ad<2:0>**. The 21164 signals **addr_h<30:8>** correspond to the quad-word PCI address and are sent out on **ad<25:3>**.

Table A–6 PCI Memory Sparse-Space Read/Write Encodings

Size	addr_h<4:3>	Byte Offset addr_h <6:5>	21164 Instruction Allowed	ad<2:0>	PCI Byte Enable[1]	Data-In Register Byte Lanes 63.....32 31.....0
		00		A<7>[2],00[3]	1110	OOOX
		01		A<7>,00	1101	OOXO
Byte	00	10	LDL,STL	A<7>,00	1011	OXOO
		11		A<7>,00	0111	XOOO
		00		A<7>,00	1100	OOXX
Word[4]	01	01	LDL,STL	A<7>,00	1001	OXXO
		10		A<7>,00	0011	XXOO
		00		A<7>,00	1000	OXXX
Tribyte	10	01	LDL,STL	A<7>,00	0001	XXXO
Longword	11	00	LDL,STL	A<7>,00	0000	XXXX
Quadword	11	11	LDQ,STQ	000	0000	XXXX XXXX

[1] Byte enable set to 0 indicates that byte lane carries meaningful data.
[2] A<7> = **addr_h<7>.**
[3] In PCI sparse memory space, **ad<1:0>** is always zero.
[4] Missing entries (for example, word size with 21164 address = 11) enjoy UNPREDICTABLE results.

The high-order **ad<31:26>** are obtained from either the hardware extension register (HAE_MEM) or the 21164 address depending on sparse-space regions, as shown in Table A–7. See the *DIGITAL Semiconductor 21174 Core Logic Chip Technical Reference Manual* for more information about the 21174 HAE_MEM CSR.

Table A–7 PCI Address Mapping

21164 Address	Region	ad					
		<31>	<30>	<29>	<28>	<27>	<26>
80.0000.0000 to 83.FFFF.FFFF	1	HAE_MEM <31>	HAE_MEM <30>	HAE_MEM <29>	CPU<33>	CPU<32>	CPU<31>
84.0000.0000 to 84.FFFF.FFFF	2	HAE_MEM <15>	HAE_MEM <14>	HAE_MEM <13>	HAE_MEM <12>	HAE_MEM <11>	CPU<31>
85.0000.0000 to 85.FFFF.FFFF	3	HAE_MEM <7>	HAE_MEM <6>	HAE_MEM <5>	HAE_MEM <4>	HAE_MEM <3>	HAE_MEM <2>

Figure A–9 shows the mapping for region 1.

Figure A–9 PCI Memory Sparse-Space Address Generation – Region 1

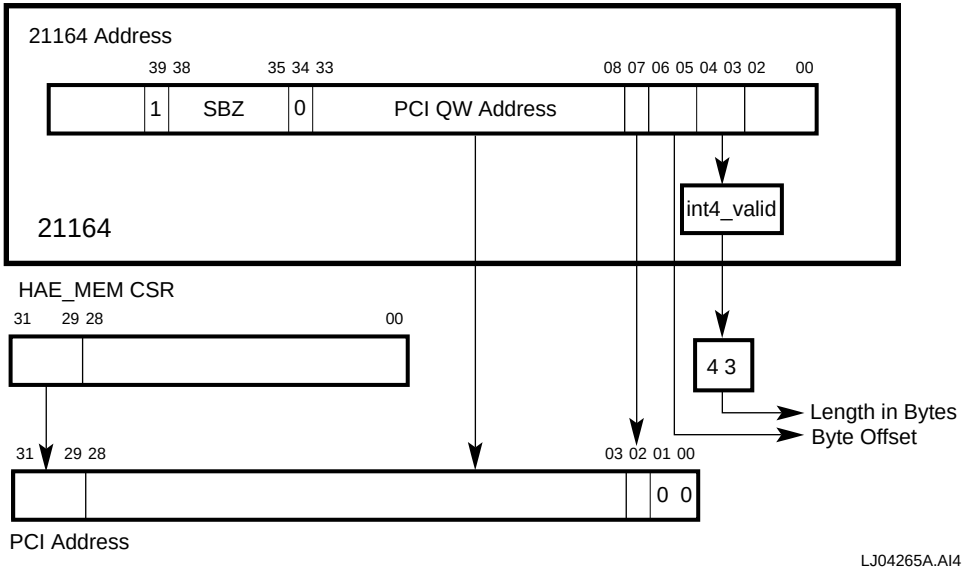


Figure A–10 shows the mapping for region 2.

Figure A–10 PCI Memory Sparse-Space Address Generation – Region 2

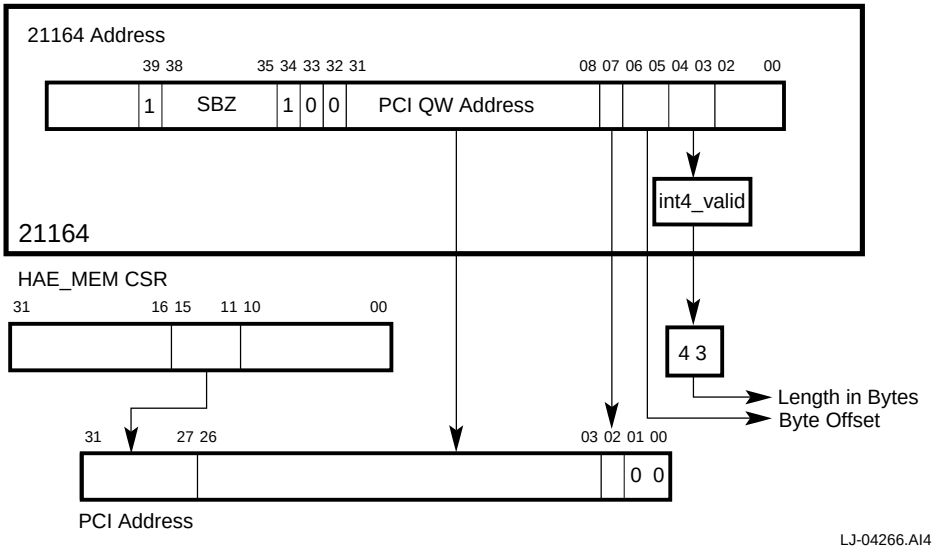
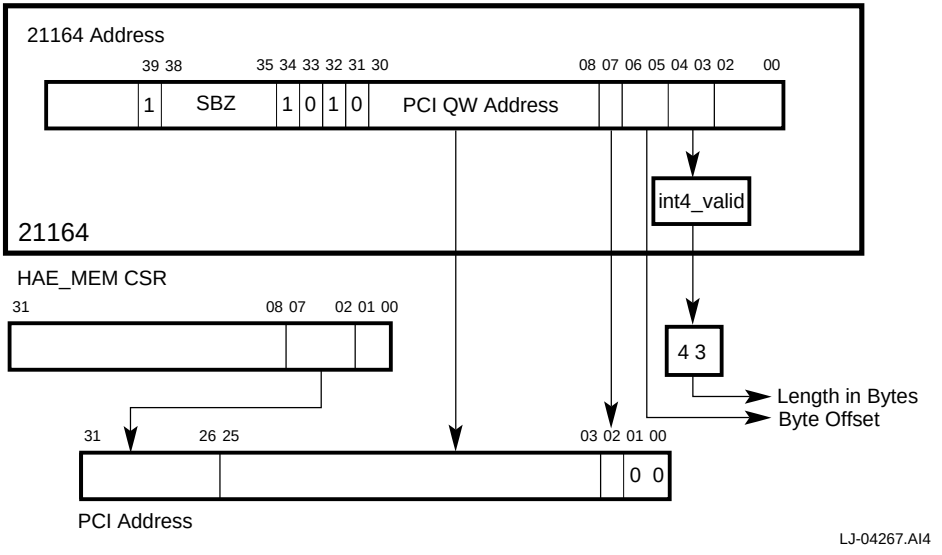


Figure A–11 shows the mapping for region 3.

Figure A–11 PCI Memory Sparse-Space Address Generation – Region 3



A.8 PCI Sparse I/O Space

The PCI sparse I/O space is divided into two regions — region A and region B. Region A addresses the lower 32MB of PCI I/O space and is never relocated. This region will be used to address the (E)ISA devices. Region B is used to address a further 32MB of PCI I/O space and is relocatable using the HAE_IO register.

A.8.1 Hardware Extension Register (HAE_IO)

In sparse space, the 21164 address bits <7:3> are used to encode byte enable bits, size bits, and the low-order **ad<2:0>**. This means that there are now five fewer address bits available to generate the PCI physical address.

The system provides two PCI sparse I/O space regions and allows one region to be relocated by way of bits in the HAE_IO register.

A.8.2 PCI Sparse I/O Space Access Operation

The PCI sparse I/O space is located in the range 85.8000.0000 to 85.FFFF.FFFF. This space has characteristics similar to the PCI sparse memory space. This 2GB 21164 address segment maps to two 32MB regions of PCI I/O address space. A read or write transaction to this space causes a PCI I/O read or write command. The high-order PCI address bits are handled as follows:

- Region A: This region has **addr_h<34:30>** = 10110 and addresses the lower 32MB of PCI sparse I/O space. Signals **ad<31:25>** are asserted at zero by the hardware (see Figure A–12). Region A is used to address (E)ISA address space (the EISA 64KB I/O space cannot be relocated). Figure A–12 shows PCI sparse I/O space address translation in Region A.
- Region B: This region has **addr_h<34:30>** = 10111 and addresses a relocatable 32MB of PCI sparse I/O space. This 32MB segment is relocated by assigning **ad<31:25>** to equal HAE_IO<31:25>. Figure A–13 shows PCI sparse I/O space address translation in Region B.

The remainder of the PCI I/O address is formed in the same way for both regions:

- **ad<24:3>** are derived from **addr_h<29:8>**.
- **ad<2:0>** are defined in Table A–8.

Table A–8 contains the PCI sparse I/O space read/write encodings.

Table A–8 PCI Sparse I/O Space Read/Write Encodings

Size		Byte Offset addr_h <6:5>	21164 Instruction Allowed	ad<2:0>	PCI Byte Enable[1]	Data-In Register Byte Lanes 63.....32 31.....0
addr_h<4:3>						
		00		A<7>[2],00	1110	OOOX
		01		A<7>,00	1101	OOXO
Byte	00	10	LDL,STL	A<7>,00	1011	OXOO
		11		A<7>,00	0111	XOOO
		00		A<7>,00	1100	OOXX
Word[3]	01	01	LDL,STL	A<7>,00	1001	OXXO
		10		A<7>,00	0011	XXOO
		00		A<7>,00	1000	OXXX
Tribyte	10	01	LDL,STL	A<7>,00	0001	XXXO
Longword	11	00	LDL,STL	A<7>,00	0000	XXXX
Quadword	11	11	LDQ,STQ	000	0000	XXXX XXXX

[1] Byte enable set to 0 indicates that byte lane carries meaningful data.
[2] A<7> = **addr_h<7>.**
[3] Missing entries (for example, word size with 21164 address = 11) enjoy UNPREDICTABLE results.

Figure A–12 PCI Sparse I/O Space Address Translation (Region A, Lower 32MB)

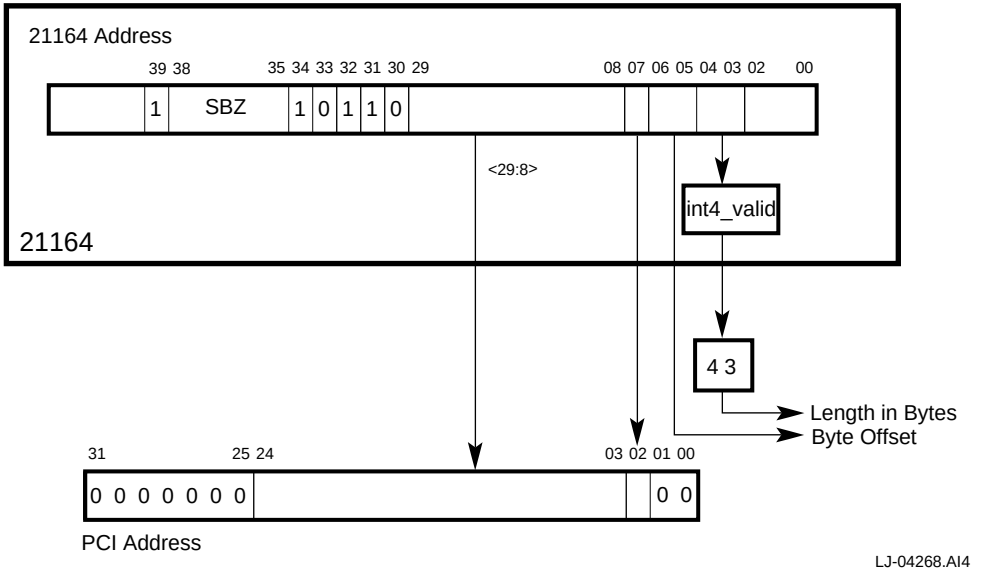
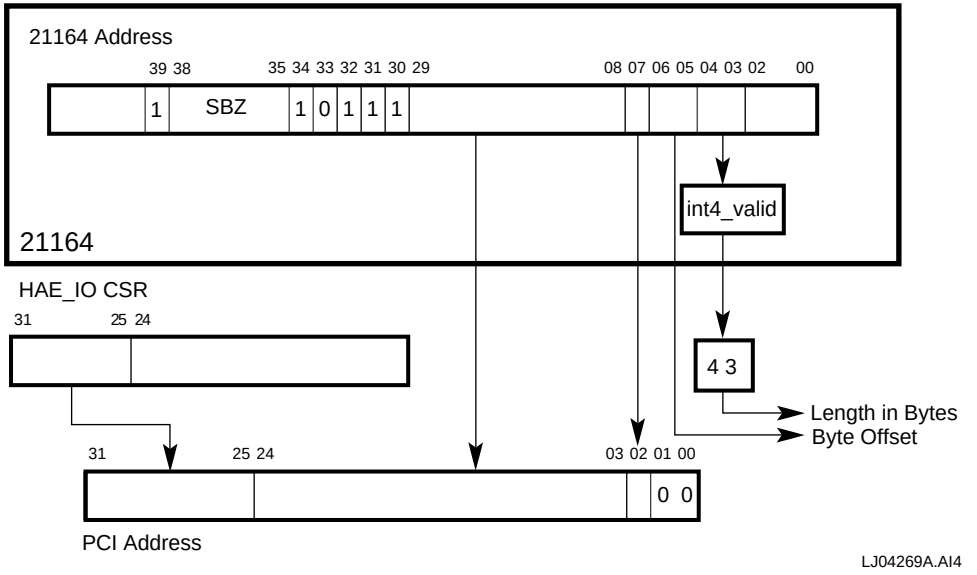


Figure A–13 PCI Sparse I/O Space Address Translation (Region B, Higher Area)



A.9 PCI Configuration Space

The PCI configuration space is located in the range 87.0000.0000 to 87.1FFF.FFFF. Software is advised to clear PYXIS_CTRL<FILL_ERR_EN> when probing for PCI devices by way of configuration space read transactions. This will prevent the 21174 from generating an ECC error if no device responds to the configuration cycle (and random data is picked up on the PCI bus).

A read or write transaction to this space causes a configuration read or write cycle on the PCI. There are two classes of targets that are selected, based on the value of the CFG register.

- Type 0 — These are targets on the primary 64-bit PCI bus. These targets are selected by making CFG<1:0> = 0.
- Type 1 — These are targets on the secondary 32-bit PCI bus (that is, behind a PCI-to-PCI bridge). These targets are selected by making CFG<1:0> = 1.

Note: CFG<1:0> = 10 or 11 are reserved (by the PCI specification).

Software must program the CFG register before running a configuration cycle. Sparse address decoding is used. Signals **addr_h<6:3>** are used to generate both the length of the PCI transaction in bytes and the byte enable bits. Signals **ad<1:0>** are obtained from CFG<1:0>. Signals **addr_h<28:7>** correspond to **ad<23:2>** and provide the configuration command information (such as which device to select). The high-order **ad<31:24>** are always zero.

Figure A–14 depicts PCI configuration space (sparse). Figure A–15 shows PCI configuration space (dense).

Figure A–14 PCI Configuration Space Definition (Sparse)

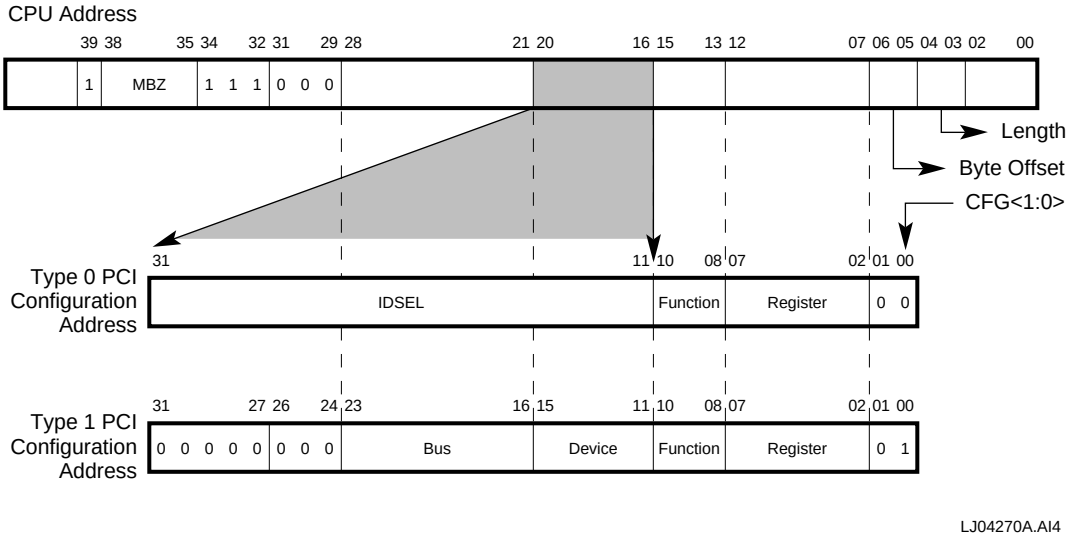
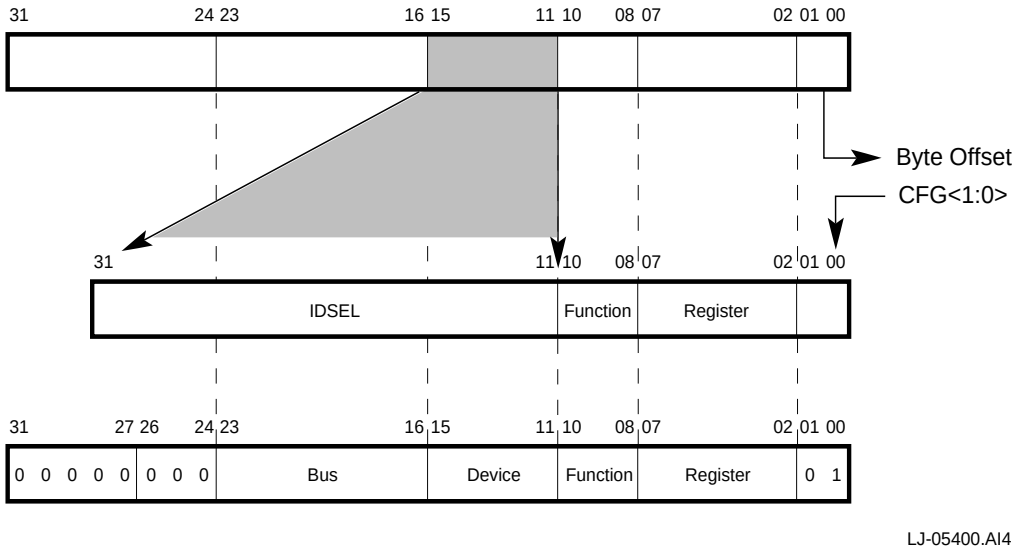


Figure A–15 PCI Configuration Space Definition (Dense)



Peripherals are selected during a PCI configuration cycle if the following three conditions are met:

1. Their IDSEL pin is asserted.
2. The PCI bus command indicates a configuration read or write.
3. Address bits <1:0> are 00.

Address bits <7:2> select a Dword (longword) register in the peripheral's 256-byte configuration address space. Transactions can use byte masks.

Peripherals that integrate multiple functional units (for example, SCSI and Ethernet) can provide configuration space for each function. Address bits <10:8> can be decoded by the peripheral to select one of eight functional units.

Signals **ad<31:11>** are available to generate the IDSEL bits (note that IDSEL bits behind a PCI-to-PCI bridge are determined from the device field encoding of a type 1 access). The IDSEL pin of each device is connected to a unique PCI address bit from **ad<31:11>**. The binary value of **addr_h<20:16>** is used to select which **ad<31:11>** is asserted, as shown in Table A–9.

Table A–9 CPU Address to IDSEL Conversion

CPU Address <20:16>	ad<31:11> – IDSEL
00000	0000 0000 0000 0000 0000 1
00001	0000 0000 0000 0000 0001 0
00010	0000 0000 0000 0000 0010 0
00011	0000 0000 0000 0000 0100 0
.....	....
.....	....
10011	0100 0000 0000 0000 0000 0
10100	1000 0000 0000 0000 0000 0
10101	0000 0000 0000 0000 0000 0
.....	...(No device selected)
.....	—
11111	0000 0000 0000 0000 0000 0

Note: If a quadword access is specified for the configuration cycle, then the least significant bit of the register number field (such as **ad<2>**) must be zero. Quadword transactions must access quadword aligned registers.

If the PCI cycle is a configuration read or write cycle but the **ad<1:0>** are 01 (that is, a type 1 transfer), then a device on a hierarchical bus is being selected via a PCI-to-PCI bridge. This cycle is accepted by the PCI-to-PCI bridge for propagation to its secondary PCI bus. During this cycle, <23:16> selects a unique bus number, and address <15:8> selects a device on that bus (typically decoded by the PCI-to-PCI bridge to generate the secondary PCI address pattern for IDSEL). In addition, address <7:2> selects a Dword (longword) in the device's configuration space.

Table A–10 contains the PCI configuration space read/write encodings.

Table A–10 PCI Configuration Space Read/Write Encodings

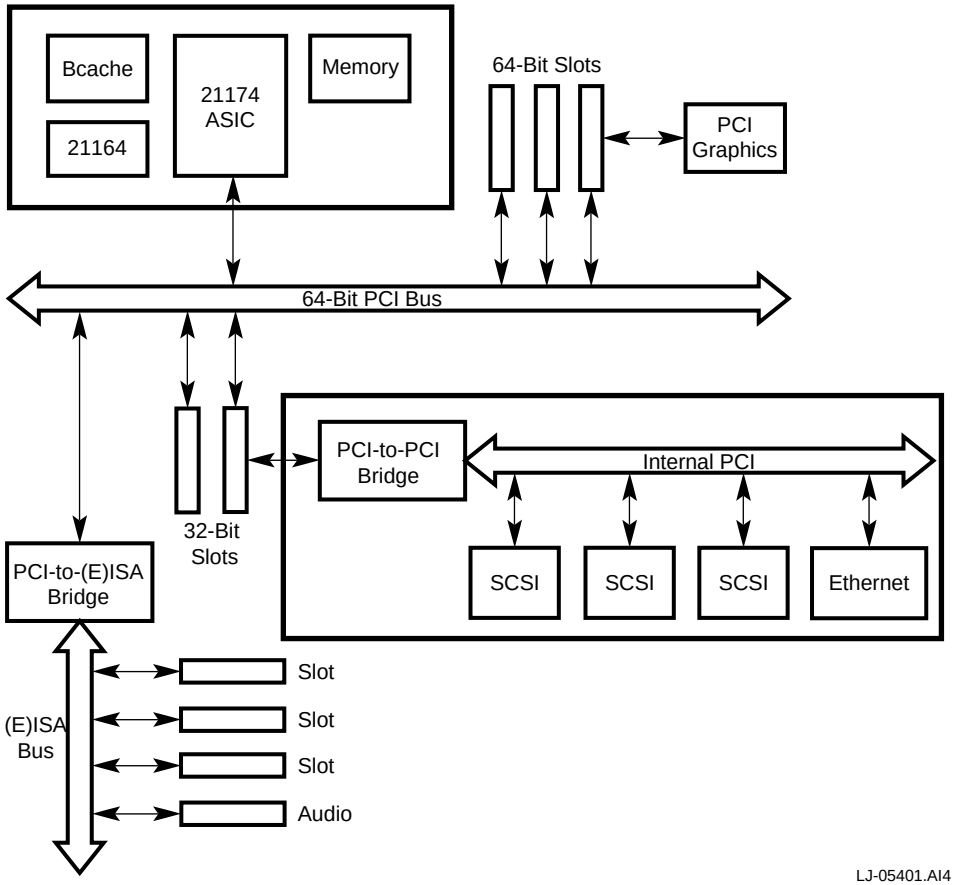
Size	addr_h<4:3>	Byte Offset addr_h <6:5>	21164 Instruction Allowed	ad<2:0>	PCI Byte Enable[1]	Data-In Register Byte Lanes 63.....32 31.....0
		00		A<7>[2],00	1110	OOOX
		01		A<7>,00	1101	OOXO
Byte	00	10	LDL,STL	A<7>,00	1011	OXOO
		11		A<7>,00	0111	XOOO
		00		A<7>,00	1100	OOXX
Word[3]	01	01	LDL,STL	A<7>,00	1001	OXXO
		10		A<7>,00	0011	XXOO
		00		A<7>,00	1000	OXXX
Tribyte	10	01	LDL,STL	A<7>,00	0001	XXXO
Longword	11	00	LDL,STL	A<7>,00	0000	XXXX
Quadword	11	11	LDQ,STQ	000	0000	XXXX XXXX

[1] Byte enable set to 0 indicates that byte lane carries meaningful data.
[2] A<7> = **addr_h<7>.**
[3] Missing entries (for example, word size with **addr_h<6:5>** = 11) generate UNPREDICTABLE results.

Each PCI-to-PCI bridge can be configured via PCI configuration cycles on its primary PCI interface. Configuration parameters in the PCI-to-PCI bridge will identify the bus number for its secondary PCI interface and a range of bus numbers that may exist hier-

archically behind it. If the bus number of the configuration cycle matches the bus number of the bridge chip's secondary PCI interface, it will accept the configuration cycle, decode it, and generate a PCI configuration cycle with **ad<1:0>** = 00 on its secondary PCI interface. If the bus number is within the range of bus numbers that may exist hierarchically behind its secondary PCI interface, the bridge chip passes the PCI configuration cycle on unmodified (**ad<1:0>** = 01). It will be accepted by a bridge further downstream. Figure A–16 shows a typical PCI hierarchy. This is only one example of how the 21174 can be used in a system design.

Figure A–16 PCI Bus Hierarchy



A.10 PCI Special/Interrupt Cycles

PCI special/interrupt cycles are located in the range 87.2000.0000 to 87.3FFF.FFFF.

The Special cycle command provides a simple message broadcasting mechanism on the PCI. The Intel processor uses this cycle to broadcast processor status; but in general it may be used for logical sideband signaling between PCI agents. The special cycle contains no explicit destination address, but is broadcast to all agents. Each receiving agent must determine if the message contained in the data field is applicable to it.

A write access in the range 87.2000.0000 to 87.3FFF.FFFF causes a special cycle on the PCI. The 21164's write data will be passed unmodified to the PCI. Software must write the data in longword 0 of the hexword with the following fields:

- Bytes 0 and 1 contain the encoded message.
- Bytes 2 and 3 are message dependent (optional) data fields.

A read of the same address range will result in an Interrupt Acknowledge cycle on the PCI and return the vector data provided by the PCI-EISA bridge to the 21164.

A.11 Hardware-Specific and Miscellaneous Register Space

These registers are located in the range 87.4000.0000 to 87.FFFF.FFFF.

Table A–11 lists the address map for the hardware-specific registers.

Table A–11 Hardware and Miscellaneous Address Map

CPU Address <39:28>	Selected Region
1000 0111 0100	General control, diagnostic, performance monitoring, and error logging registers
1000 0111 0101	Memory control registers
1000 0111 0110	PCI address translation (scatter-gather, windows, and so on)
1000 0111 0111	Reserved
1000 0111 1000	Miscellaneous registers
1000 0111 1001	Power management registers
1000 0111 1010	Interrupt controller registers
1000 0111 11xx	Flash ROM read/write space – for programming

The address space here is a hardware-specific variant of sparse-space encoding. For the CSRs, **addr_h<27:6>** specifies a longword address where **addr_h<5:0>** must be zero. All the 21174 registers are accessed with a LW granularity. For more specific details on the 21174 CSRs, see the *DIGITAL Semiconductor 21174 Core Logic Chip Technical Reference Manual*. For the flash ROM, **addr_h<30:6>** defines a byte address. The fetched byte is always returned in the first byte lane (bits <7:0>).

A.12 PCI to Physical Memory Address

Incoming PCI addresses (32-bit or 64-bit) have to be mapped to the 21164 cached memory space (8GB). The 21174 provides five programmable address windows that control access of PCI peripherals to system memory.

The mapping from the PCI address to the physical address can be direct, direct mapped (physical mapping with an address offset), or scatter-gather mapped (virtual mapping). These five address windows are referred to as the PCI target windows.

Window 4 maps directly, using the "Monster Window" with dual-address cycles (DAC), where **ad<33:0>** equals **addr_h<33:0>**.

The following three registers are associated with windows <3:0>:

- Window base (W_BASE) register
- Window mask (W_MASK) register
- Translated base (T_BASE) register

In addition, there is an extra register associated with window 3 only. This is the window DAC register and is used for PCI 64-bit addressing (that is, the DAC mode). The following text applies only to windows <3:0>.

The window mask register provides a mask corresponding to **ad<31:20>** of an incoming PCI address. The size of each window can be programmed to be from 1MB to 4GB in powers of two, by masking bits of the incoming PCI address using the window mask register, as shown in Table A–12. (Note that the mask field pattern was chosen to speed up timing-critical logic circuits.)

Table A–12 shows the PCI target window mask fields.

Table A–12 PCI Target Window Mask Register Fields[1]

PCI_MASK<31:20>	Size of Window	Value of n
0000 0000 0000	1MB	20
0000 0000 0001	2MB	21
0000 0000 0011	4MB	22
0000 0000 0111	8MB	23
0000 0000 1111	16MB	24
0000 0001 1111	32MB	25
0000 0011 1111	64MB	26
0000 0111 1111	128MB	27
0000 1111 1111	256MB	28
0001 1111 1111	512MB	29
0011 1111 1111	1GB	30
0111 1111 1111	2GB	31
1111 1111 1111	4GB	32
Otherwise	UNPREDICTABLE	—

[1] Only the incoming **ad<31:n>** are compared with <31:*n*> of the window base register, as shown in Figure A–18. If *n*=32, no comparison is performed.

Based on the value of the window mask register, the unmasked bits of the incoming PCI address are compared with the corresponding bits of each window's window base register. If one of the window base registers and the incoming PCI address match, then the PCI address has hit the PCI target window. Otherwise, the PCI address has missed the window. A window enable bit, W_EN, is provided in each window's window base register to allow windows to be independently enabled (W_EN = 1) or disabled (W_EN = 0).

If a hit occurs in any of the four windows that are enabled, then the 21174 will respond to the PCI cycle by asserting the signal **devsel**. The PCI target windows must be programmed so that their address ranges do not overlap; otherwise, the results are UNDEFINED.

The window base address must be on a naturally aligned boundary address depending on the size of the window[6]. This rule is not particularly difficult to obey, because the address space of any PCI device can be located anywhere in the PCI's 4GB memory space, and this scheme is compatible with the PCI specification:

> A PCI device specifies the amount of memory space it requires via the Base registers in its configuration space. The Base Address registers are implemented so that the address space consumed by the device is a power of two in size, and is naturally aligned on the size of the space consumed.

A PCI device need not use all the address range it consumes (that is, the size of the PCI address window defined by the base address) and it does not need to respond to unused portions of the address space. The one exception to this is a PCI bridge that requires two additional registers (the base and limit address registers). These registers accurately specify the address space that the bridge device will respond to[7] and are programmed by the power-on self-test (POST) code. The 21174, as a PCI host-bridge device, does not have base and limit registers[8], but does respond to all the addresses defined by the window base register (that is, all addresses within a window).

Figure A–17 shows how the DMA address ranges of a number of PCI devices are accepted by the PCI-window ranges. PCI devices are allowed to have multiple DMA address ranges, as shown for device 2. The example also shows that the window can be larger than the corresponding device's DMA address range, as shown for device 0. Device 1 and device 2 have address ranges that are accepted by one window. Each window determines whether direct mapping or scatter-gather mapping is used to access physical memory.

---

[6] For example, a 4MB window cannot begin at address 1MB. It must start at addresses 4MB, 8MB, 12MB, ...

[7] A PCI bridge device responds to all addresses in the range: base ≤ address < limit.

[8] Host-bridge devices, because they are under system control, are free to violate the rules.

Figure A–17 PCI DMA Addressing Example

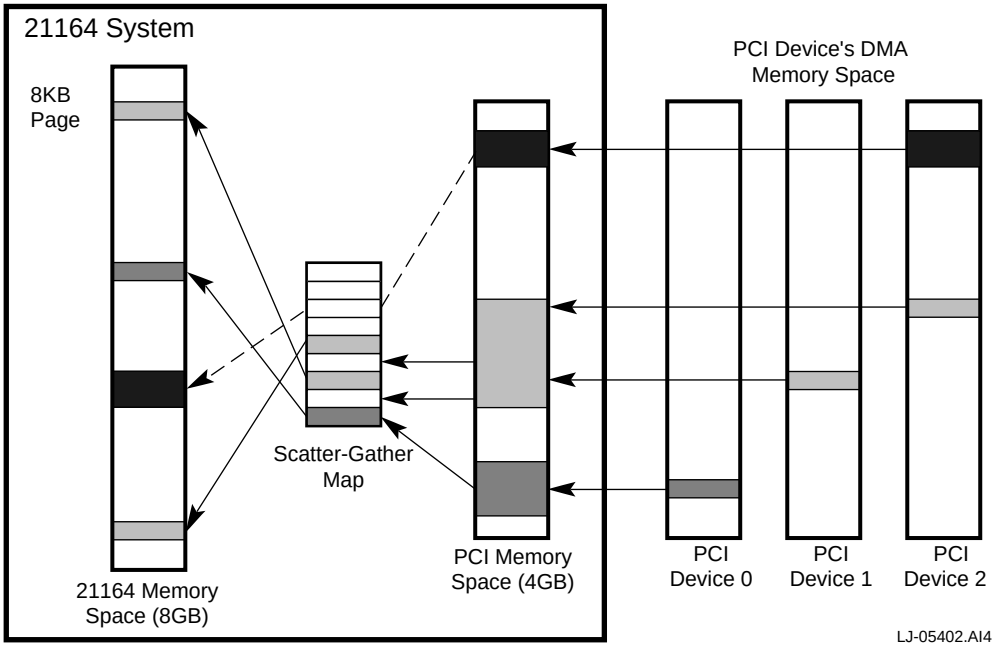


Figure A–18 shows the PCI window logic. The comparison logic associated with **ad<63:32>** is only used for DAC[9] mode; and only if enabled by a bit in the window base register for window 3. This logic is only applicable to window 3. The remaining windows only recognize 32-bit PCI addresses (that is, SAC[10] cycles).

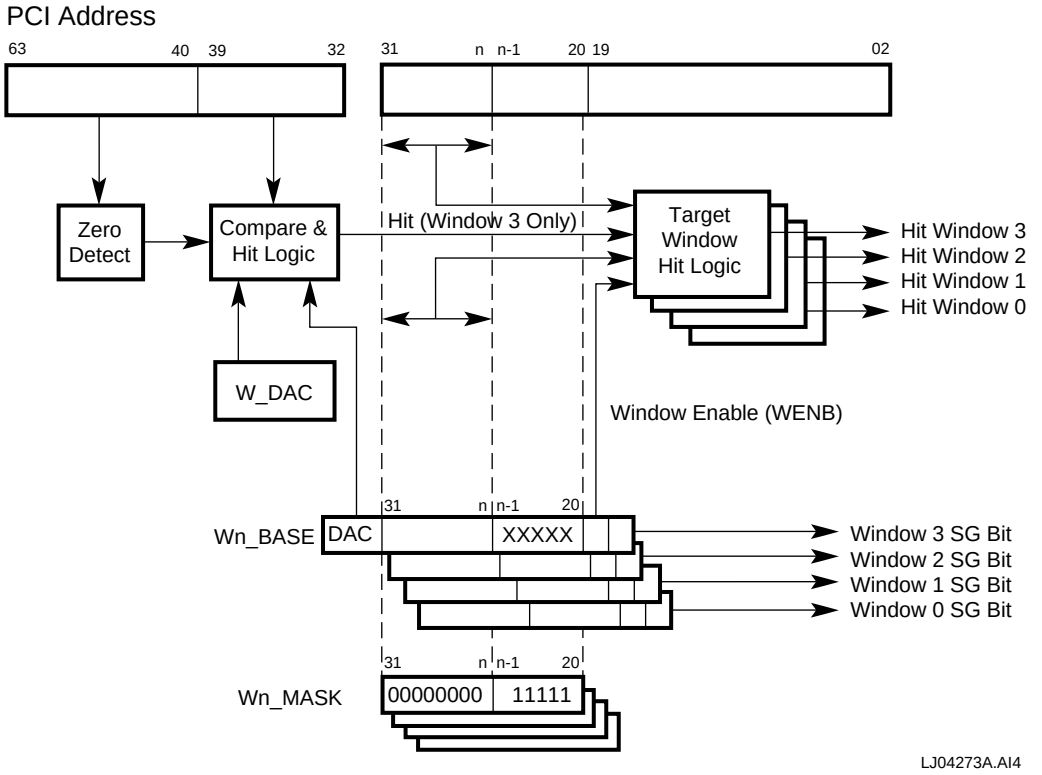
For a hit to occur in a DAC address, **ad<63:40>** must be zero, **ad<39:32>** must match the window DAC base register, and **ad<31:20>** must also have a compare hit. This scheme allows a naturally aligned, 1MB–4GB PCI window to be placed anywhere in the first 1TB of a 64-bit PCI address. When an address match occurs with a PCI target window, the 21174 translates the 32-bit PCI address to **addr_h<33:0>**.

---

[9] Dual-address cycle (DAC) — only issued if <63:32> are nonzero for a 64-bit address.

[10] Single-address cycle (SAC) — all 32-bit addresses. A PCI device must use SAC if <63:32> equals 0.

Figure A–18 PCI Target Window Compare



A.13 Direct-Mapped Addressing

The target address is translated by direct mapping or scatter-gather mapping as determined by the Wx_BASE_SG (scatter-gather) bit of the window's PCI base register. If the Wx_BASE_SG bit is clear, the DMA address is direct mapped, and the translated address is generated by concatenating bits from the matching window's translated base register (T_BASE) with bits from the incoming PCI address. The bits involved in the concatenation are defined by the window mask register as shown in Table A–13. The unused bits of the translated base register (also in Table A–13) must be cleared (that is, the hardware performs an AND-OR operation to accomplish the concatenation). Because memory is located in the lower 8GB of the 21164 address space, the 21174 ensures (implicitly) that address bits <39:33> are always zero.

Because the translated base is simply concatenated to the PCI address, then the direct mapping is to a naturally aligned memory region. For example, a 4MB direct-mapped window will map to any 4MB region in main memory that falls on a 4MB boundary (for instance, it is not possible to map a 4MB region to the main memory region 1MB–5MB).

Table A–13 lists direct-mapped PCI target address translations.

Table A–13 Direct-Mapped PCI Target Address Translation *(Sheet 1 of 2)*

W_MASK<31:20>	Size of Window	Translated Address <32:2>
0000 0000 0000	1MB	Translated Base<33:20> : **ad<19:2>**
0000 0000 0001	2MB	Translated Base<33:21> : **ad<20:2>**
0000 0000 0011	4MB	Translated Base<33:22> : **ad<21:2>**
0000 0000 0111	8MB	Translated Base<33:23> : **ad<22:2>**
0000 0000 1111	16MB	Translated Base<33:24> : **ad<23:2>**
0000 0001 1111	32MB	Translated Base<33:25> : **ad<24:2>**
0000 0011 1111	64MB	Translated Base<33:26> : **ad<25:2>**
0000 0111 1111	128MB	Translated Base<33:27> : **ad<26:2>**
0000 1111 1111	256MB	Translated Base<33:28> : **ad<27:2>**
0001 1111 1111	512MB	Translated Base<33:29> : **ad<28:2>**
0011 1111 1111	1GB	Translated Base<33:30> : **ad<29:2>**

Table A–13 Direct-Mapped PCI Target Address Translation *(Sheet 2 of 2)*

W_MASK<31:20>	Size of Window	Translated Address <32:2>
0111 1111 1111	2GB	Translated Base<33:31> : **ad<30:2>**
1111 1111 1111	4GB	Translated Base<33:32> : **ad<31:2>**
Otherwise	Not supported	—

A.14 Scatter-Gather Addressing

If the Wx_BASE_SG bit of the PCI base register is set, then the translated address is generated by a lookup table. This table is called a scatter-gather map. Figure A–20 shows the scatter-gather addressing scheme — full details of this scheme are provided later in Section A.15, but for now a quick description is provided. The incoming PCI address is compared to the PCI window addresses looking for a hit. The translated base register, associated with the PCI window that is hit, is used to specify the starting address of the scatter-gather map table in memory. Bits of the incoming PCI address are used as an offset from this starting address, to access the scatter-gather PTE. This PTE, in conjunction with the remaining, least-significant PCI address bits, forms the required memory address.

Each scatter-gather map entry maps an 8KB page of PCI address space into an 8KB page of the 21164 address space. This offers a number of advantages to software:

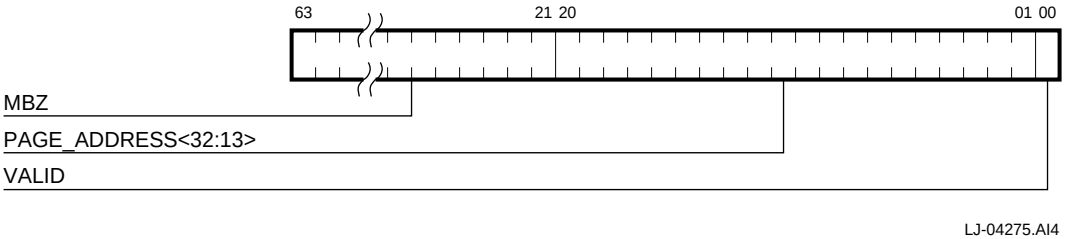
- Performance: ISA devices map to the lower 16MB of memory. The Windows NT operating system currently copies data from here to user space. The scatter-gather map eliminates the need for this copy operation.
- User I/O buffers might not be physically contiguous or contained within a page. With scatter-gather mapping, software does not have to manage the scattered nature of the user buffer by copying data.

In the personal computer (PC) world, scatter-gather mapping is not an address translation scheme but is used to signify a DMA transfer list. An element in this transfer list contains the DMA address and the number of data items to transfer. The DMA device fetches each item of the list until the list is empty. Many of the PCI devices (such as an EISA bridge) support this form of scatter-gather mapping.

Each scatter-gather map page table entry (PTE) is a quadword and has a valid bit in bit position 0, as shown in Figure A–19. Address bit 13 is at bit position 1 of the map entry. Because the 21174 implements valid memory addresses up to 16GB, then bits <63:22> of the scatter-gather map entry must be programmed to 0. Bits <21:1> of the scatter-gather map entry are used to generate the physical page address. The physical page address is appended to **ad<12:5>** of the incoming PCI address to generate the memory address.

System implementations may support less than 16GB of physical addressing; however, any unused address bits must be forced to zero. Otherwise, behavior will be UNPREDICTABLE.

Figure A–19 Scatter-Gather PTE Format



The size of the scatter-gather map table is determined by the size of the PCI target window as defined by the window mask register shown in Table A–14. The number of entries in the table equals the window size divided by the page size (8KB). The size of the table is simply the number of entries multiplied by 8 bytes.

The scatter-gather map table address is obtained from the translated base register and the PCI address as shown in Table A–14.

Table A–14 Scatter-Gather Mapped PCI Target Address Translation *(Sheet 1 of 2)*

W_MASK<31:20>	Size of SG Map Table	Translated Address <32:2>
0000 0000 0000	1KB	Translated Base<33:10>[1] : **ad<19:13>**
0000 0000 0001	2KB	Translated Base<33:11> : **ad<20:13>**
0000 0000 0011	4KB	Translated Base<33:12> : **ad<21:13>**
0000 0000 0111	8KB	Translated Base<33:13> : **ad<22:13>**
0000 0000 1111	16KB	Translated Base<33:14> : **ad<23:13>**

Table A–14 Scatter-Gather Mapped PCI Target Address Translation *(Sheet 2 of 2)*

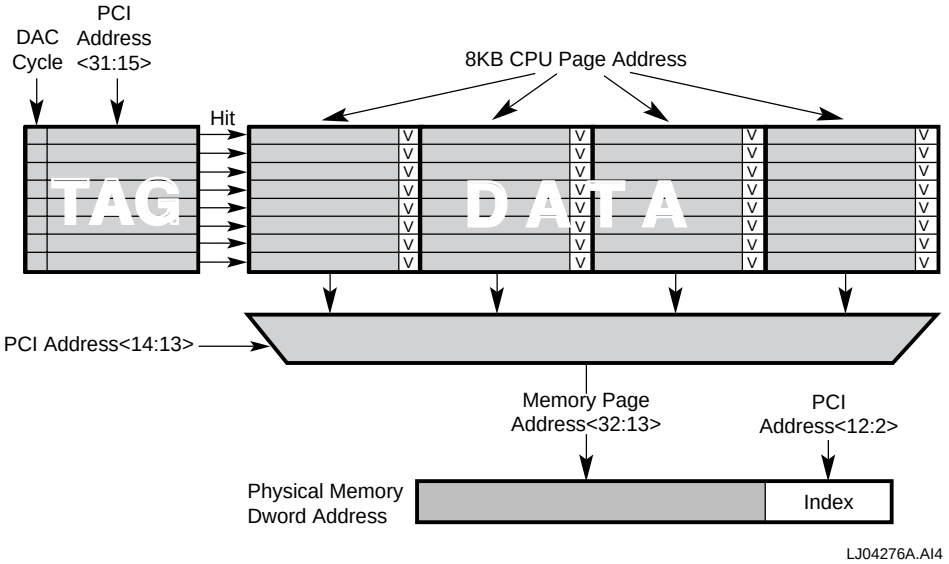
W_MASK<31:20>	Size of SG Map Table	Translated Address <32:2>
0000 0001 1111	32KB	Translated Base<33:15> : **ad<24:13>**
0000 0011 1111	64KB	Translated Base<33:16> : **ad<25:13>**
0000 0111 1111	128KB	Translated Base<33:17> : **ad<26:13>**
0000 1111 1111	256KB	Translated Base<33:18> : **ad<27:13>**
0001 1111 1111	512KB	Translated Base<33:19> : **ad<28:13>**
0011 1111 1111	1MB	Translated Base<33:20> : **ad<29:13>**
0111 1111 1111	2MB	Translated Base<33:21> : **ad<30:13>**
1111 1111 1111	4MB	Translated Base<33:22> : **ad<31:13>**

[1] Unused bits of the Translated Base Register must be zero for correct operation.

A.15 Scatter-Gather TLB

An eight-entry translation lookaside buffer (TLB) is provided in the 21174 for scatter-gather map entries. The TLB is a fully associative cache and holds the eight most-recent scatter-gather map lookup PTEs. Four of these entries can be locked to prevent their being displaced by the hardware TLB-miss handler. Each of the eight TLB entries holds a PCI address for the tag and four consecutive 8KB 21164 page addresses as the TLB data, as shown in Figure A–20.

Figure A–20 Scatter-Gather Associative TLB



Each time an incoming PCI address hits in a PCI target window that has scatter-gather translation enabled, **ad<31:15>** are compared with the 32KB PCI page address in the TLB tag. If a match is found, the required 21164 page address is one of the four items provided by the data of the matching TLB entry. PCI address **ad<14:13>** selects the correct 8KB 21164 page from the four pages fetched.

A TLB hit avoids having to look up the scatter-gather map PTEs in memory, resulting in improved system performance. If no match is found in the TLB, the scatter-gather map lookup is performed and four PTE entries are fetched and written over an existing entry in the TLB.

The TLB entry to be replaced is determined by a round-robin algorithm on the unlocked entries. Coherency of the TLB is maintained by software write transactions to the SG_TBIA (scatter-gather translation buffer invalidate all) register.

The tag portion contains a DAC flag to indicate that the PCI tag address <31:15> corresponds to a 64-bit DAC address. Only one bit is required instead of the high-order PCI address bits <39:32> because only one window is assigned to a DAC cycle, and the window-hit logic has already performed a comparison of the high-order bits with the PCI DAC base register. Figure A–21 shows the entire translation from PCI address to physical address on a window that implements scatter-gather

mapping. Both paths are indicated — the right side shows the path for a TLB hit, while the left side shows the path for a TLB miss. The scatter-gather TLB is shown in a slightly simplified, but functionally equivalent form.

A.15.1 Scatter-Gather TLB Hit Process

The process for a scatter-gather TLB hit is as follows:

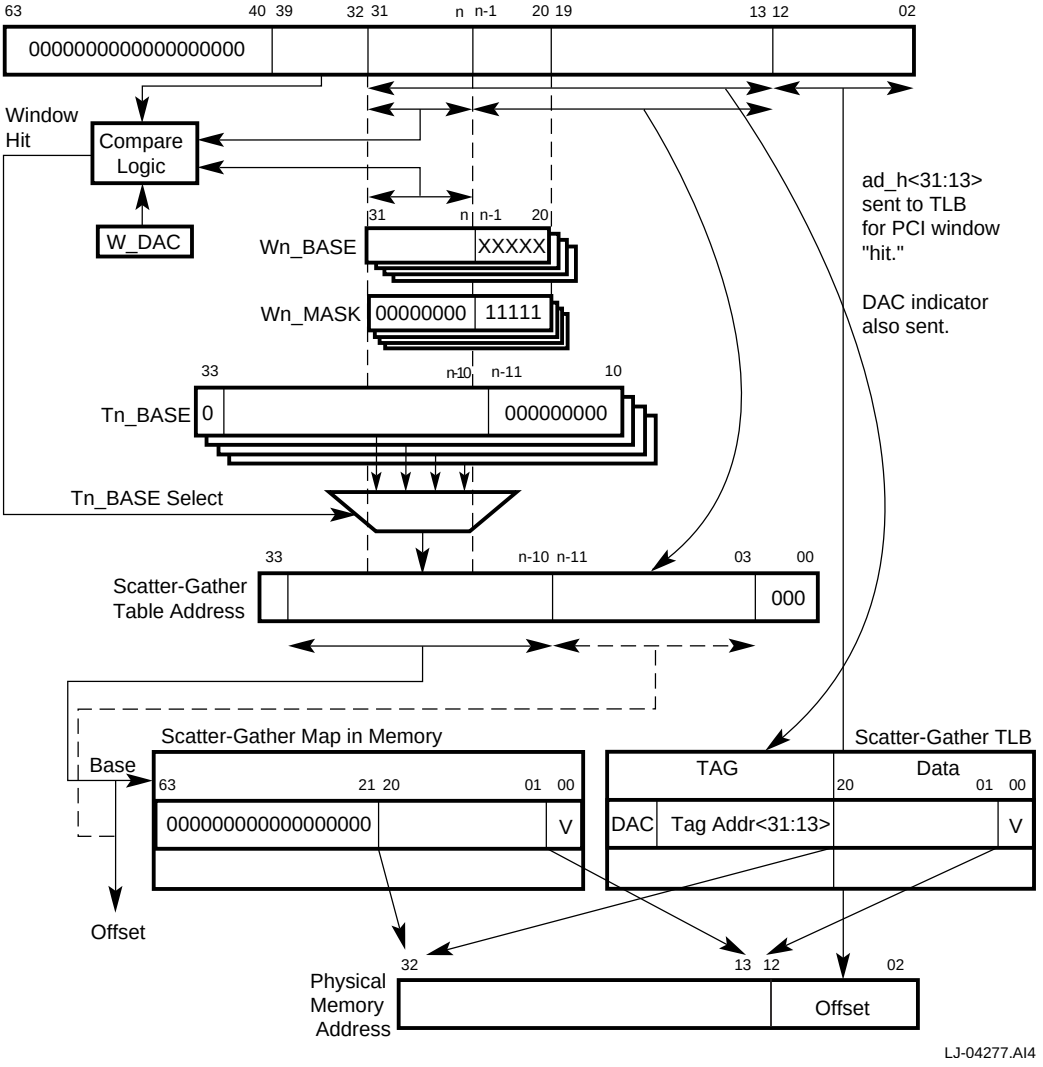
1. The window compare logic determines if the PCI address has hit in one of the four windows, and the PCI_BASE<SG> bit determines if the scatter-gather path should be taken. If window 3 has DAC-mode enabled, and the PCI cycle is a DAC cycle, then a further comparison is made between the high-order PCI bits and the PCI DAC BASE register.
2. PCI address **ad<31:13>** is sent to the TLB associative tag together with the DAC hit indication. If **ad<31:13>** and the DAC bits match in the TLB, then the corresponding 8KB 21164 page address is read out of the TLB. If this entry is valid, then a TLB hit has occurred and this page address is concatenated with **ad<12:2>** to form the physical memory address. If the data entry is invalid, or if the TAG compare failed, then a TLB miss occurs.

A.15.2 Scatter-Gather TLB Miss Process

The process for a scatter-gather TLB miss is as follows:

1. The relevant bits of the PCI address (as determined by the window mask register) are concatenated with the relevant translated base register bits to form the address used to access the scatter-gather map entry (PTE) from a table located in main memory.
2. Bits <20:1> of the map entry (PTE from memory) are used to generate the physical page address, which is appended to the page offset to generate the physical memory address. The TLB is also updated at this point, using a round-robin algorithm, with the four PTE entries that correspond to the 32KB PCI page address that first missed the TLB. The tag portion of the TLB is loaded with this PCI page address, and the DAC bit is set if this PCI cycle is a DAC cycle.
3. If the requested PTE is marked invalid (bit 0 is clear), then a TLB invalid entry exception is taken.

Figure A–21 Scatter-Gather Map Translation



A.16 Suggested Use of a PCI Window

Figure A–22 shows the PCI window assignment after power is turned on (configured by firmware), and Table A–15 lists the details. PCI window 0 was chosen for the 8MB to 16MB EISA region because this window incorporates the **mem_cs_l** logic. PCI window 3 was not used as it incorporates the DAC cycle logic. PCI window 1 was chosen arbitrarily for the 1GB, direct-mapped region, and PCI window 2 is not assigned.

Figure A–22 Default PCI Window Allocation

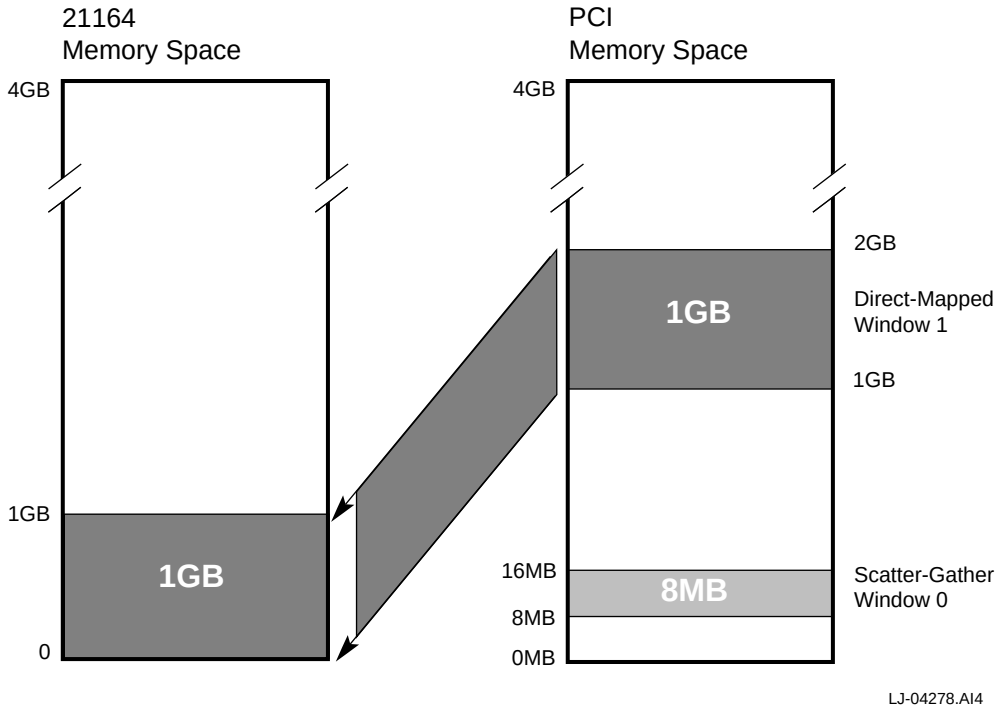


Table A–15 lists the PCI window power-up configuration characteristics.

Table A–15 PCI Window Power-Up Configuration

PCI Window	Assignment	Size	Comments
0	Scatter-gather	8MB	Not used by firmware; **mem_cs_l** disabled
1	Direct-mapped	1GB	Mapped to 0GB to 1GB of main memory
2	Disabled	—	—
3	Disabled	—	—

A.16.1 Peripheral Component Architecture Compatibility Addressing and Holes

The peripheral component architecture allows certain (E)ISA devices to respond to hardwired memory addresses. An example is a VGA graphics device that has its frame buffer located in memory address region A0000–BFFFF. Such devices "pepper" memory space with holes, which are collectively known as peripheral component compatibility holes.

The PCI-EISA bridge decodes PCI addresses and generates a signal, **mem_cs_l**, which takes into account the various PC compatibility holes.

A.16.2 Memory Chip Select Signal mem_cs_l

The PCI-EISA bridge can be made using the following two chips:

- Intel 82374EB EISA System Component (ESC)
- Intel 82375EB PCI-EISA Bridge (PCEB)

The PCI-EISA bridge provides address decode logic with considerable attributes (such as read only, write only, VGA frame buffer, memory holes, and BIOS shadowing) to help manage the EISA memory map and peripheral component compatibility holes.

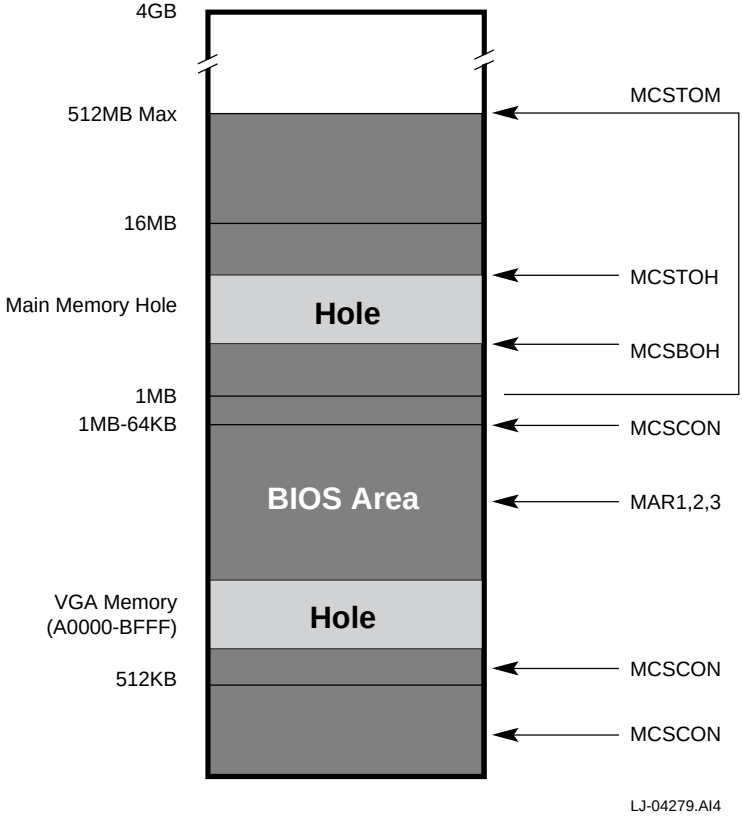
This is known as main memory decoding in the PCI-EISA chip, and results in the generation of the memory chip select (**mem_cs_l**) signal. One exception is the VGA memory hole region that never asserts **mem_cs_l**. If enabled, the 21174 uses this signal with the W0_BASE register.

In Figure A–23, the two main holes are shown lightly shaded, while the **mem_cs_l** range is darkly shaded.

This **mem_cs_l** range in Figure A–23 is subdivided into several portions (such as the BIOS areas) that are individually enabled/disabled using CSRs as listed here:

- The MCSTOM (top of memory) register has a 2MB granularity and can be programmed to select the regions from lMB up to 512MB.
- The MCSTOH (top of hole) and MCSBOH (bottom of hole) registers define a memory hole region where **mem_cs_l** is not selected. The granularity of the hole is 64KB.
- The MARl,2,3 registers enable various BIOS regions.
- The MCSCON (control) register enables the **mem_cs_l** decode logic, and in addition selects a number of regions (0KB to 512KB).
- The VGA memory hole region never asserts **mem_cs_l**.

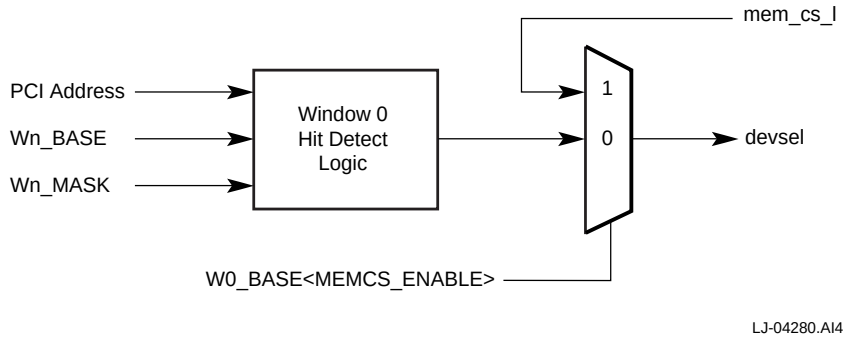
Figure A–23 mem_cs_l Decode Area



Note: For more detail, please refer to the *Intel 82378 System I/O Manual*.

As shown in Figure A–24, PCI window 0 in the 21174 can be enabled to accept the **mem_cs_l** signal as the PCI memory decode signal. With this path enabled, the PCI window hit logic simply uses the **mem_cs_l** signal. For example, if **mem_cs_l** is asserted, then a PCI window 0 hit occurs and the **devsel** signal is asserted on the PCI.

Figure A–24 mem_cs_l Logic



Consequently, the window address area must be large enough to encompass the **mem_cs_l** region programmed into the PCI-EISA bridge. The remaining window attributes are still applicable and/or required:

- The Wx_BASE_SG bit in the W0_BASE register determines if scatter-gather or direct-mapping is applicable.
- The W0_ MASK register size information must match the **mem_cs_l** size for the scatter-gather and direct-mapping algorithms to correctly use the translated base register.
- The **mem_cs_l** enable bit, W0_BASE<MEMCS_ENABLE>, takes precedence over W0_BASE<W_EN>.

B
Supporting Products

This appendix lists sources for components and accessories that are not included with the AlphaPC 164LX. For the latest information, visit the Alpha website at URL: **http://www.alpha.digital.com**. Click on **Motherboard Products**. The hardware compatibility list (HCL) and the qualified-memory vendor list are also available at this location.

B.1 Memory

The following list of dual inline memory modules (DIMMs) is updated frequently. For the most recent additions, please visit the Alpha website at URL: **http://www.digital.com/semiconductor/alpha/alpha-memlist.htm**.

Advantage Memory Corporation

P.O. Box 30
Danvers, MA 01923
attn: Paul Johnson
Phone: 1-800-839-5041
Fax: 978-762-4998

ATP Electronics, Inc.

750 North Mary Avenue
Sunnyvale, CA 94086
attn: Regina Kao
Phone: 408-732-3936
Fax: 408-732-5055

Dataram

P.O. Box 7528
Princeton, NJ 08543
Phone: 1-800-DATARAM (328-2726), ext. 239

Kingston Technology Company

17600 Newhope Street
Fountain Valley, CA 92708
Phone: 1-800-845-2545

Micron Semiconductor Products, Inc.

8000 South Federal Way
Mail Stop 607
Boise, ID 83706
Phone: 208-368-3900
Fax: 208-368-5018

NEC Electronics, Inc.

The Meadows Building, 4th Floor
161 Worcester Road
Framingham, MA 01701
Phone: 508-935-2000
Fax: 508-935-2233

Samsung America, Inc.

3655 N. First Street
San Jose, CA 95134
Phone: 1-800-423-7364

Viking Components

30200 Avenida de la Banderas
Rancho Santa Margarita, CA 92688
attn: Jeff Jones, OEM Development Executive
Phone: 1-800-338-2361, ext. 423
Fax: 714-643-7250

B.2 Thermal Products

Components included in this heat-sink and fan solution are heat sink, GRAFOIL pad, two hex nuts, heat-sink clips, 60-mm fan, and four screws. These are available from:

United Machine and Tool Design

River Road
Fremont, NH 03044
Phone: 603-642-5040
Fax: 603-642-5819
PN 70-32810-02

B.3 Power Supply

An ATX form-factor power supply, suitable for use with the AlphaPC 164LX (+3.3 V, +5 V, –5 V, +12 V, –12 V), is available from:

Quantum Power Labs, Inc.

1410 Gail Borden Place C-4
El Paso, TX 79935
Phone: 915-599-2688
Fax: 915-599-2699
PN AP2-5300FRV (300 W)

Antec, Inc.

2859 Bayview Drive
Fremont, CA 94538
Phone: 510-770-1200, ext. 312
PN PP-253V (250 W)

B.4 Enclosure

An enclosure, suitable for housing the AlphaPC 164LX and its power supply, is available from:

Axxion Group Corporation

7801 Trade Center Avenue
El Paso, TX 79912
Phone: 915-877-5288
PN DL17

C

Support, Products, and Documentation

If you need technical support, a *DIGITAL Semiconductor Product Catalog*, or help deciding which documentation best meets your needs, visit the DIGITAL Semiconductor World Wide Web Internet site:

http://www.digital.com/semiconductor

You can also contact the DIGITAL Semiconductor Information Line or the DIGITAL Semiconductor Customer Technology Center for support.

For documentation and general information:	
DIGITAL Semiconductor Information Line	
United States and Canada:	1-800-332-2717
Outside North America:	1-510-490-4753
Electronic mail address:	semiconductor@digital.com

For technical support:	
DIGITAL Semiconductor Customer Technology Center	
Phone (U.S. and international):	1-978-568-7474
Fax:	1-978-568-6698
Electronic mail address:	ctc@hlo.mts.dec.com

DIGITAL Semiconductor Products

To order the AlphaPC 164LX motherboard, contact your local distributor. The following tables list some of the semiconductor products available from DIGITAL Semiconductor.

Note: The following products and order numbers might have been revised. For the latest versions, contact your local distributor.

Chips	Order Number
DIGITAL Semiconductor 21164 Alpha microprocessor (466 MHz)	21164-IB
DIGITAL Semiconductor 21164 Alpha microprocessor (533 MHz)	21164-P8
DIGITAL Semiconductor 21164 Alpha microprocessor (600 MHz)	21164-MB

Motherboard kits include the motherboard and motherboard user's manual.

Motherboard Kits	Order Number
DIGITAL Semiconductor AlphaPC 164LX Motherboard Kit for Windows NT	21A04-C0
DIGITAL Semiconductor AlphaPC 164LX Motherboard Kit for DIGITAL UNIX	21A04-C1

Design kits include full documentation and schematics. They do not include related hardware.

Design Kits	Order Number
AlphaPC 164LX Motherboard Software Developer's Kit (SDK) and Firmware Update	QR-21A04-12

DIGITAL Semiconductor Documentation

The following table lists some of the available DIGITAL Semiconductor documentation.

Title	Order Number
Alpha AXP Architecture Reference Manual[1]	EY–T132E–DP
Alpha Architecture Handbook[2]	EC–QD2KB–TE
DIGITAL Semiconductor 21164 Alpha Microprocessor Hardware Reference Manual	EC-QP99B-TE
DIGITAL Semiconductor 21164 Alpha Microprocessor Data Sheet	EC-QP98B-TE
DIGITAL Semiconductor 21174 Core Logic Chip Technical Reference Manual	EC-R12GB-TE

[1]To purchase the *Alpha AXP Architecture Reference Manual*, contact your local distributor or call Butterworth-Heinemann (Digital Press) at 1-800-366-2665.
[2]This handbook provides information subsequent to the *Alpha AXP Architecture Reference Manual*.

Third–Party Documentation

You can order the following third-party documentation directly from the vendor.

Title	Vendor
PCI Local Bus Specification, Revision 2.1 PCI Multimedia Design Guide, Revision 1.0 PCI System Design Guide PCI-to-PCI Bridge Architecture Specification, Revision 1.0 PCI BIOS Specification, Revision 2.1	PCI Special Interest Group U.S. 1–800–433–5177 International 1–503–797–4207 Fax 1–503–234–6762

Index

Numerics

21164 microprocessor. *See* Microprocessor.
21174 Core logic chip. *See* Core logic chip.
21174-CA. *See* Core logic chip.
37C935. *See* Combination controller.
82378ZB. *See* SIO.

A

Abbreviations, x
Address space, A–1 to A–47
    memory remapping, A–6
    PCI, A–6
    system, A–1, A–7
Airflow requirements, 3–2
ATX hole specification, 3–3
ATX I/O shield requirements, 3–4

B

Bcache
    interface, 4–2
    subsystem, 1–4
Bit notation, xi
Block diagram, 1–2

C

CAS, 4–5
Clocks, 1–4
    14.3-MHz reference, 4–8
    time-of-year, 4–8
Combination controller, 1–4, 4–7
Communication ports, 4–7
Component list, 2–3
Components and features, 1–1
Configuration
    memory, 5–1
Configuration jumpers, 2–4
Connectors, 2–3
    pinouts, 2–6 to 2–15
Conventions
    numbering, xii
Core logic chip, 4–3 to 4–5
CPU. *See* Microprocessor.
Current
    dc ampere requirements, 3–1

D

Data field size, xi
Data units, xii
DC power requirements, 3–1

Debug monitor
system support, 1–5
Design support, 1–6
DIGITAL UNIX
SDK support, 1–6
SRM console firmware, 1–5
Dimensions
motherboard, 3–2
Direct mapping, 4–4
Diskette controller, 4–7
DMA conversion, 4–4
Documentation
ordering, C–3

E

Environmental requirements, 3–2
Extents and ranges, xii

F

Fan sensor, 3–1
FDC37C935. *See* Combination controller.
Flash ROM
AlphaBIOS firmware, 1–5
Ubus memory device, 4–9

H

Heat sink/fan, 5–5

I

I/O shield dimensions, 3–4
Interface
main memory, 4–4
Interrupts, 4–10
system assignment, 4–12
INTnn, xi
ISA
bus, 4–5
devices, 4–7
expansion slots, 4–7
interface, 1–4

J

Jumper configurations, 2–4
Jumper descriptions, 2–3
Jumpers
Bcache size, 2–5
boot option, 2–5
flash ROM update, 2–6
password bypass, 2–5

K

Keyboard
controller, 4–8

M

Mapping
direct (DMA), 4–4
scatter-gather, 4–4
Memory
access rules and operation, A–18
alternate mode, 4–5
configurations, 5–1
DIMM sizes, 5–1
main, 4–4
PCI dense space, A–15
PCI sparse space, A–17
remapping, A–6
SDRAM DIMM pinouts, 2–9
subsystem, 1–3
Microprocessor
heat sink/fan, 5–5
speeds, 1–1
upgrading, 5–2 to 5–6

Motherboard
ATX hole specification, 3–3
ATX I/O shield, 3–4
component descriptions, 2–3
configuration jumpers, 2–4
dimensions, 3–2
layout, 2–2
Mouse
controller, 4–8

N

Numbering convention, xii

O

Operating systems
software support, 1–5
Ordering products and documentation, C–2

P

Packaging
21174 chip, 4–3
Parallel port, 4–7
PCI
21174 role, 4–6
bus, 4–5
bus hierarchy, A–30
bus speed, 4–5
configuration space, A–26
dense memory space, A–15
device implementation, 4–5
expansion slots, 4–7
interface, 1–4
memory remapping, A–6
sparse memory space, A–17
Pinouts
connectors, 2–6 to 2–15
Power
distribution, 4–19
monitor, 4–17
requirements, 3–1
Power supply
dc ampere requirements, 3–1
wattage requirements, 3–1
Processor. *See* Microprocessor.
PTE, 4–4

R

Ranges and extents, xii
RAS, 4–5
Reset, 4–17
RO
definition, xi
RW
definition, xi

S

Scatter-gather
mapping, 4–4
SDK, 1–5 to 1–6
Serial ports, 4–7
Serial ROM. *See* SROM.
Shield dimensions
I/O, 3–4
SIO, 4–6, 4–10
Software support, 1–5
SRM Console, 1–5
SROM
description, 1–5
functions, 4–18
Support
technical, C–1
System
address space, A–1 to A–47
components and features, 1–1
software support, 1–5

T

Time-of-year clock, 4–8

TLB, 4–4

U

UARTs, 4–7
Ubus, 4–7, 4–9
UNDEFINED
    definition, xiii
UNIX. *See* DIGITAL UNIX.
UNPREDICTABLE
    definition, xiii
Upgrading
    memory, 5–2
    microprocessor, 5–2
Utility bus. *See* Ubus.

W

Wave pipelining, 4–2
Windows NT
    AlphaBIOS firmware, 1–5
    SDK support, 1–6
WO
    definition, xi