Alpha 21264 Microprocessor Data Sheet

Order Number: EC–R4CFA–TE

Revision/Update Information: Revision 1.0, February 1999

Compaq Computer Corporation

Table of Contents

4 Internal Processor Registers

Figures

Tables

Preface

Audience

This data sheet provides a technical overview of the Alpha 21264 microprocessor (referred to as the 21264).

Content

This data sheet contains the following chapters and appendixes:

Chapter 1, Introduction, introduces the 21264 and provides an overview of the Alpha architecture.

Chapter 2, Internal Architecture, describes the major hardware functions and the internal chip architecture.

Chapter 3, Hardware Interface, lists and describes the internal hardware interface signals, and provides mechanical data and packaging information, including signal pin lists.

Chapter 4, Internal Processor Registers, lists and describes the internal processor register set.

Chapter 5, Privileged Architecture Library Code, describes the privileged architecture library code (PALcode).

Chapter 6, Initialization and Configuration, describes the initialization and configuration sequence.

Chapter 7, Electrical Data, provides electrical data and describes signal integrity issues.

Chapter 8, Thermal Management, provides information about thermal management.

Appendix A, Alpha Instruction Set, summarizes the Alpha instruction set.

Documentation Included by Reference

The companion volume to this specification, the *Alpha Architecture Handbook, Version 4*, contains the instruction set architecture. You can access this document from the following website: `ftp.digital.com/pub/Digital/info/semiconductor/literature/dsc-library.html`

Also available is the *Alpha Architecture Reference Manual, Third Edition*, which contains the complete architecture information. That manual is available at bookstores from the Digital Press as EQ-W938E-DP.

Terminology and Conventions

This section defines the abbreviations, terminology, and other conventions used throughout this document.

Abbreviations

- 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
2K pixels	=	2 kilopixels	=	2×2^{10} pixels
4M pixels	=	4 megapixels	=	4×2^{20} pixels

- Register Access

  The abbreviations used to indicate the type of access to register fields and bits have the following definitions:

Abbreviation	Meaning
IGN	Ignore Bits and fields specified are ignored on writes.
MBZ	Must Be Zero Software must never place a nonzero value in bits and fields specified as MBZ. A nonzero read produces an Illegal Operand exception. Also, MBZ fields are reserved for future use.
RAZ	Read As Zero Bits and fields return a zero when read.
RC	Read Clears Bits and fields are cleared when read. Unless otherwise specified, such bits cannot be written.
RES	Reserved Bits and fields are reserved by Compaq and should not be used; however, zeros can be written to reserved fields that cannot be masked.
RO	Read Only The value may be read by software. It is written by hardware. Software write operations are ignored.
RO,*n*	Read Only, and takes the value *n* at power-on reset. The value may be read by software. It is written by hardware. Software write operations are ignored.

Abbreviation	Meaning
RW	Read/Write Bits and fields can be read and written.
RW,*n*	Read/Write, and takes the value *n* at power-on reset. Bits and fields can be read and written.
W1C	Write One to Clear If read operations are allowed to the register, then the value may be read by software. If it is a write-only register, then a read operation by software returns an UNPREDICTABLE result. Software write operations of a 1 cause the bit to be cleared by hardware. Software write operations of a 0 do not modify the state of the bit.
W1S	Write One to Set If read operations are allowed to the register, then the value may be read by software. If it is a write-only register, then a read operation by software returns an UNPREDICTABLE result. Software write operations of a 1 cause the bit to be set by hardware. Software write operations of a 0 do not modify the state of the bit.
WO	Write Only Bits and fields can be written but not read.
WO,*n*	Write Only, and takes the value *n* at power-on reset. Bits and fields can be written but not read.

- Sign extension

  SEXT(x) means *x* is sign-extended to the required size.

Addresses

Unless otherwise noted, all addresses and offsets are hexadecimal.

Aligned and Unaligned

The terms *aligned* and *naturally aligned* are interchangeable and refer to data objects that are powers of two in size. An aligned datum of size 2*n* is stored in memory at a byte address that is a multiple of 2*n*; that is, one that has *n* low-order zeros. For example, an aligned 64-byte stack frame has a memory address that is a multiple of 64.

A datum of size 2*n* is *unaligned* if it is stored in a byte address that is not a multiple of 2*n*.

Bit Notation

Multiple-bit fields can include contiguous and noncontiguous bits contained in square 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 square brackets. For example, [27] specifies bit 27. See also Field Notation.

Caution

Cautions indicate potential damage to equipment or loss of data.

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	2	4	32	Dword
Quadword	4	8	64	2 longword

Do Not Care (X)

A capital X represents any valid value.

External

Unless otherwise stated, external means not contained in the chip.

Field Notation

The names of single-bit and multiple-bit fields can be used rather than the actual bit numbers (see Bit Notation). When the field name is used, it is contained in square brackets ([]). For example, **RegisterName[LowByte]** specifies **RegisterName[7:0]**.

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 square 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 Figures

The gray areas in register figures indicate reserved or unused bits and fields.

Bit ranges that are coupled with the field name specify the bits of the named field that are included in the register. The bit range may, but need not necessarily, correspond to the bit *Extent* in the register. See the explanation above Table 4–1 for more information.

Signal Names

The following examples describe signal-name conventions used in this document.

AlphaSignal[n:n]	Boldface, mixed-case type denotes signal names that are assigned internal and external to the 21264 (that is, the signal traverses a chip interface pin).
AlphaSignal_*x*[n:n]	When a signal has high and low assertion states, a lowercase italic *x* represents the assertion states. For example, **SignalName_*x*[3:0]** represents **SignalName_H[3:0]** and **SignalName_L[3:0]**.

UNDEFINED

Operations specified as UNDEFINED may vary from moment to moment, implementation to implementation, and instruction to instruction within implementations. The operation may vary in effect from nothing to stopping system operation.

UNDEFINED operations may 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.

UNPREDICTABLE

UNPREDICTABLE results or occurrences do not disrupt the basic operation of the processor; it continues to execute instructions in its normal manner. Further:

- Results or occurrences specified as UNPREDICTABLE may 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 may acquire an arbitrary value subject to a few constraints. Such a result may 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 may also produce exceptions.
- An occurrence specified as UNPREDICTABLE may happen or not based on an arbitrary choice function. The choice function is subject to the same constraints as are UNPREDICTABLE results and, in particular, 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 may 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, or

  – 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.

X

Do not care. A capital X represents any valid value.

Revision History

The following table lists the revision history for this document.

Date	Revison	Comments
February 17, 1999	1.0	First release

1 Introduction

This chapter provides a brief introduction to the Alpha architecture, Compaq's RISC (reduced instruction set computing) architecture designed for high performance. The chapter then summarizes the specific features of the Alpha 21264 microprocessor (hereafter called the 21264) that implements the Alpha architecture. Appendix A provides a list of Alpha instructions.

The companion volume to this specification, the *Alpha Architecture Handbook, Version 4*, contains the instruction set architecture. Also available is the *Alpha Architecture Reference Manual, Third Edition*, which contains the complete architecture information.

1.1 The Architecture

The Alpha architecture is a 64-bit load and store RISC architecture designed with particular emphasis on speed, multiple instruction issue, multiple processors, and software migration from many operating systems.

All registers are 64 bits long and all operations are performed between 64-bit registers. All instructions are 32 bits long. Memory operations are either load or store operations. All data manipulation is done between registers.

The Alpha architecture supports the following data types:

- 8-, 16-, 32-, and 64-bit integers
- IEEE 32-bit and 64-bit floating-point formats
- VAX architecture 32-bit and 64-bit floating-point formats

In the Alpha architecture, instructions interact with each other only by one instruction writing to a register or memory location and another instruction reading from that register or memory location. This use of resources makes it easy to build implementations that issue multiple instructions every CPU cycle.

The 21264 uses a set of subroutines, called privileged architecture library code (PALcode), that is specific to a particular Alpha operating system implementation and hardware platform. These subroutines provide operating system primitives for context switching, interrupts, exceptions, and memory management. These subroutines can be invoked by hardware or CALL_PAL instructions. CALL_PAL instructions use the function field of the instruction to vector to a specified subroutine. PALcode is written in standard machine code with some implementation-specific extensions to provide direct access to low-level hardware functions. PALcode supports optimizations for multiple operating systems, flexible memory-management implementations, and multi-instruction atomic sequences.

The Alpha architecture performs byte shifting and masking with normal 64-bit, register-to-register instructions. The 21264 performs single-byte and single-word load and store instructions.

1.1.1 Addressing

The basic addressable unit in the Alpha architecture is the 8-bit byte. The 21264 supports a 48-bit or 43-bit virtual address (selectable under IPR control).

Virtual addresses as seen by the program are translated into physical memory addresses by the memory-management mechanism. The 21264 supports a 44-bit physical address.

1.1.2 Integer Data Types

Alpha architecture supports the four integer data types listed in Table 1–1.

Table 1–1 Integer Data Types

Data Type	Description
Byte	A byte is 8 contiguous bits that start at an addressable byte boundary. A byte is an 8-bit value.
Word	A word is 2 contiguous bytes that start at an arbitrary byte boundary. A word is a 16-bit value.
Longword	A longword is 4 contiguous bytes that start at an arbitrary byte boundary. A longword is a 32-bit value.
Quadword	A quadword is 8 contiguous bytes that start at an arbitrary byte boundary.

Note: Alpha implementations may impose a significant performance penalty when accessing operands that are not naturally aligned. Refer to the *Alpha Architecture Handbook, Version 4,* for details.

1.1.3 Floating-Point Data Types

The 21264 supports the following floating-point data types:

- Longword integer format in floating-point unit
- Quadword integer format in floating-point unit
- IEEE floating-point formats
  - S_floating
  - T_floating
- VAX floating-point formats
  - F_floating
  - G_floating
  - D_floating (limited support)

1.2 21264 Microprocessor Features

The 21264 microprocessor is a superscalar pipelined processor. It is packaged in a 587-pin PGA carrier and has removable application-specific heat sinks. A number of configuration options allow its use in a range of system designs ranging from extremely simple uniprocessor systems with minimum component count to high-performance multiprocessor systems with very high cache and memory bandwidth.

The 21264 can issue four Alpha instructions in a single cycle, thereby minimizing the average cycles per instruction (CPI). A number of low-latency and/or high-throughput features in the instruction issue unit and the onchip components of the memory subsystem further reduce the average CPI.

The 21264 and associated PALcode implements IEEE single-precision and double-precision, VAX F_floating and G_floating data types, and supports longword (32-bit) and quadword (64-bit) integers. Byte (8-bit) and word (16-bit) support is provided by byte-manipulation instructions. Limited hardware support is provided for the VAX D_floating data type.

Other 21264 features include:

- The ability to issue up to four instructions during each CPU clock cycle.
- A peak instruction execution rate of four times the CPU clock frequency.
- An onchip, demand-paged memory-management unit with translation buffer, which, when used with PALcode, can implement a variety of page table structures and translation algorithms. The unit consists of a 128-entry, fully-associative data translation buffer (DTB) and a 128-entry, fully-associative instruction translation buffer (ITB), with each entry able to map a single 8KB page or a group of 8, 64, or 512 8KB pages. The allocation scheme for the ITB and DTB is round-robin. The size of each translation buffer entry's group is specified by hint bits stored in the entry. The DTB and ITB implement 8-bit address space numbers (ASN), MAX_ASN=255.
- Two onchip, high-throughput pipelined floating-point units, capable of executing both VAX and IEEE floating-point data types.
- An onchip, 64KB virtually-addressed instruction cache with 8-bit ASNs (MAX_ASN=255).
- An onchip, virtually-indexed, physically-tagged dual-read-ported, 64KB data cache.
- Supports a 48-bit or 43-bit virtual address (program selectable).
- Supports a 44-bit physical address.
- An onchip I/O write buffer with four 64-byte entries for I/O write transactions.
- An onchip, 8-entry victim data buffer.
- An onchip, 32-entry load queue.
- An onchip, 32-entry store queue.
- An onchip, 8-entry miss address file for cache fill requests and I/O read transactions.
- An onchip, 8-entry probe queue, holding pending system port probe commands.

- An onchip, duplicate tag array used to maintain level 2 cache coherency.
- A 64-bit data bus with onchip parity and error correction code (ECC) support.
- Support for an external second-level (Bcache) cache. The size and some timing parameters of the Bcache are programmable.
- An internal clock generator providing a high-speed clock used by the 21264, and two clocks for use by the CPU module.
- Onchip performance counters to measure and analyze CPU and system performance.
- Chip and module level test support, including an instruction cache test interface to support chip and module level testing.
- A 2.2-V external interface.

Refer to Chapter 7 for 21264 dc and ac electrical characteristics. Refer to the *Alpha Architecture Handbook, Version 4,* Appendix E, for waivers and any other implementation-dependent information.

2 Internal Architecture

This chapter provides both an overview of the 21264 microarchitecture and a system designer's view of the 21264 implementation of the Alpha architecture. The combination of the 21264 microarchitecture and privileged architecture library code (PALcode) defines the chip's implementation of the Alpha architecture. If a certain piece of hardware seems to be "architecturally incomplete," the missing functionality is implemented in PALcode. Chapter 5 provides more information on PALcode.

This chapter describes the major functional hardware units and is not intended to be a detailed hardware description of the chip. It is organized as follows:

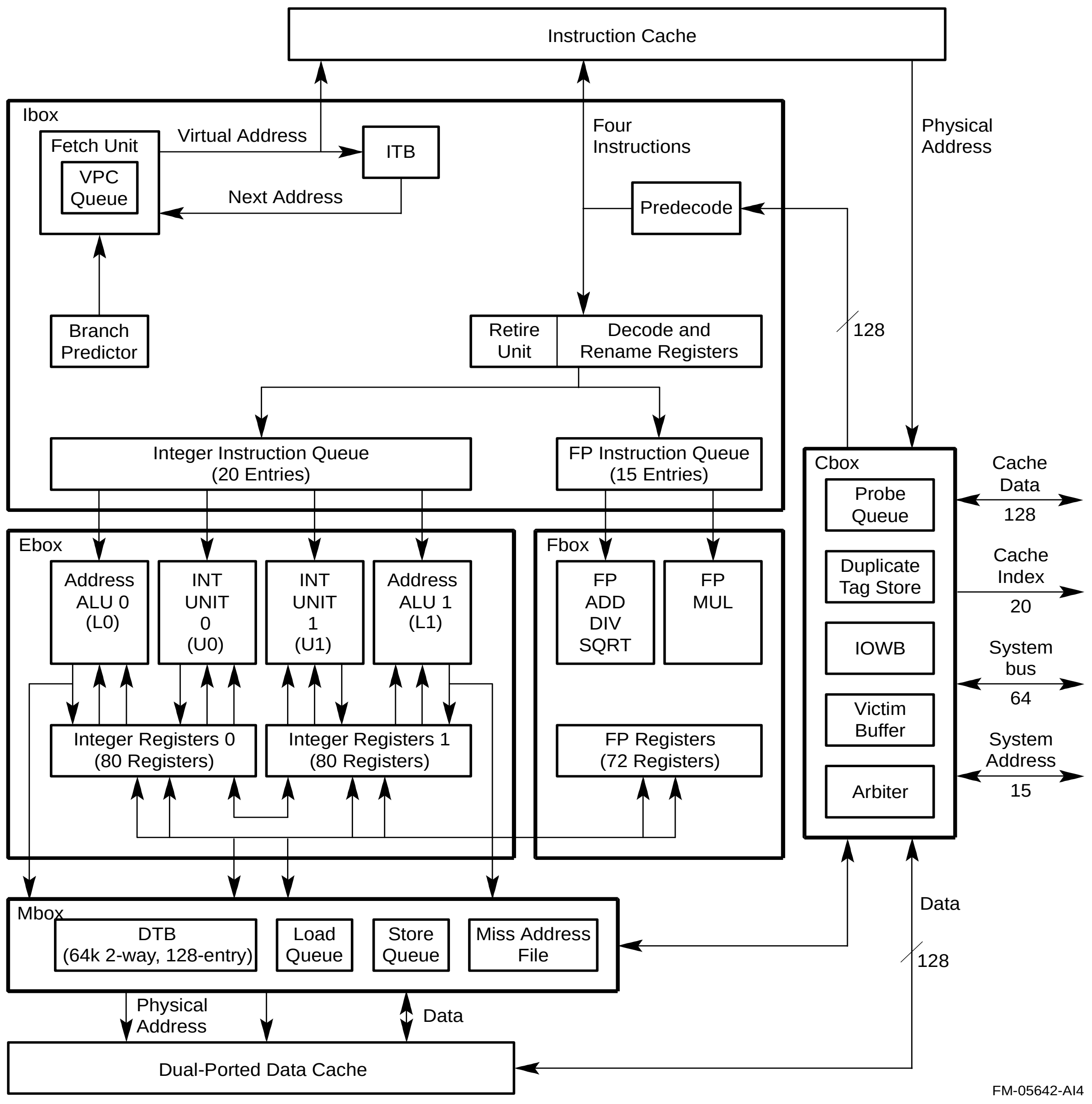
- 21264 microarchitecture
- Pipeline organization
- Floating-point control register
- AMASK and IMPLVER instruction values

2.1 21264 Microarchitecture

The 21264 microprocessor is a high-performance third-generation implementation of the Compaq Alpha architecture. The 21264 consists of the following sections, as shown in Figure 2–1:

- Instruction fetch, issue, and retire unit (Ibox)
- Integer execution unit (Ebox)
- Floating-point execution unit (Fbox)
- Onchip caches (Icache and Dcache)
- Memory reference unit (Mbox)
- External cache and system interface unit (Cbox)
- Pipeline operation sequence

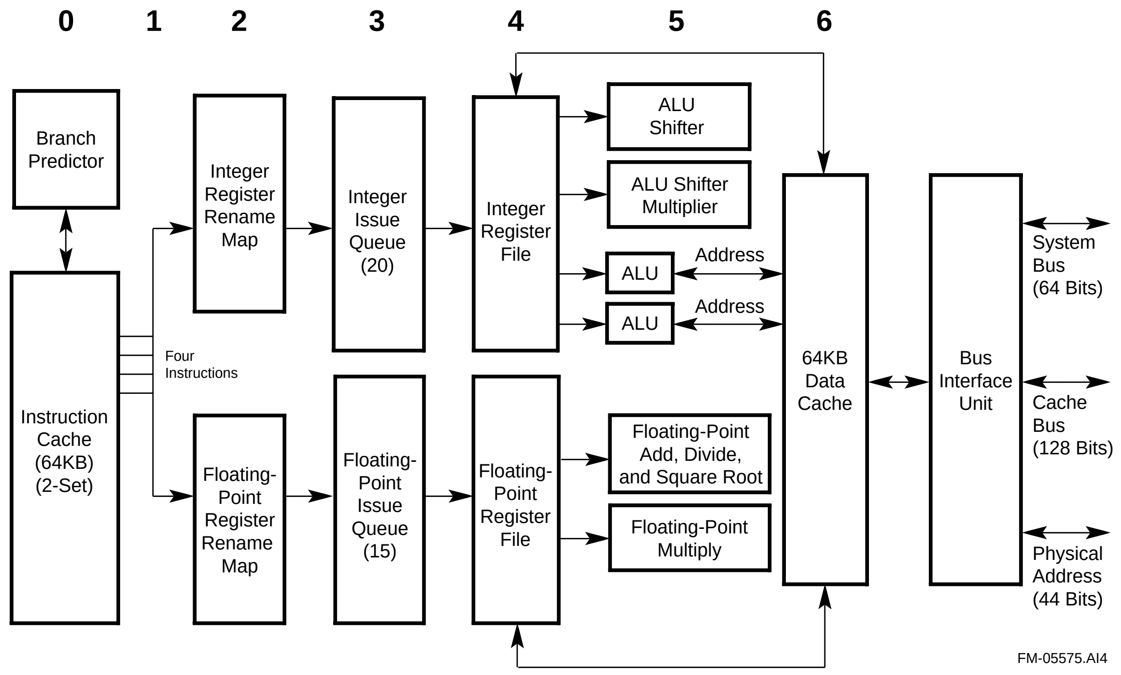
Figure 2–1 21264 Block Diagram



2.2 Pipeline Organization

The 7-stage pipeline provides an optimized environment for executing Alpha instructions. The pipeline stages (0 to 6) are shown in Figure 2–2 and described in the following paragraphs.

Figure 2–2 Pipeline Organization



Stage 0 — Instruction Fetch

The branch predictor uses a branch history algorithm to predict a branch instruction target address.

Up to four aligned instructions are fetched from the Icache, in program order. The branch prediction tables are also accessed in this cycle. The branch predictor uses tables and a branch history algorithm to predict a branch instruction target address for one branch or memory format JSR instruction per cycle. Therefore, the prefetcher is limited to fetching through one branch per cycle. If there is more than one branch within the fetch line, and the branch predictor predicts that the first branch will not be taken, it will predict through subsequent branches at the rate of one per cycle, until it predicts a taken branch or predicts through the last branch in the fetch line.

The Icache array also contains a line prediction field, the contents of which are applied to the Icache in the next cycle. The purpose of the line predictor is to remove the pipeline bubble which would otherwise be created when the branch predictor predicts a branch to be taken. In effect, the line predictor attempts to predict the Icache line which the branch predictor will generate. On fills, the line predictor value at each fetch line is initialized with the index of the next sequential fetch line, and later retrained by the branch predictor if necessary.

Stage1 — Instruction Slot

The Ibox maps four instructions per cycle from the 64KB 2-way set-predict Icache. Instructions are mapped in order, executed dynamically, but are retired in order.

In the slot stage the branch predictor compares the next Icache index that it generates to the index that was generated by the line predictor. If there is a mismatch, the branch predictor wins—the instructions fetched during that cycle are aborted, and the index predicted by the branch predictor is applied to the Icache during the next cycle. Line mispredictions result in one pipeline bubble.

The line predictor takes precedence over the branch predictor during memory format calls or jumps. If the line predictor was trained with a true (as opposed to predicted) memory format call or jump target, then its contents take precedence over the target hint field associated with these instructions. This allows dynamic calls or jumps to be correctly predicted.

The instruction fetcher produces the full VPC address during the fetch stage of the pipeline. The Icache produces the tags for both Icache sets 0 and 1 each time it is accessed. That enables the fetcher to separate set mispredictions from true Icache misses. If the access was caused by a set misprediction, the instruction fetcher aborts the last two fetched slots and refetches the slot in the next cycle. It also retrains the appropriate set prediction bits.

The instruction data is transferred from the Icache to the integer and floating-point register map hardware during this stage. When the integer instruction is fetched from the Icache and slotted into the IQ, the slot logic determines whether the instruction is for the upper or lower subclusters. The slot logic makes the decision based on the resources needed by the (up to four) integer instructions in the fetch block. Although all four instructions need not be issued simultaneously, distributing their resource usage improves instruction loading across the units. For example, if a fetch block contains two instructions that can be placed in either cluster followed by two instructions that must execute in the lower cluster, the slot logic would designate that combination as EELL and slot them as UULL.

Stage 2 — Map

Instructions are sent from the Icache to the integer and floating-point register maps during the slot stage and register renaming is performed during the map stage. Also, each instruction is assigned a unique 8-bit number, called an *inum*, which is used to identify the instruction and its program order with respect to other instructions during the time that it is in flight. Instructions are considered to be in flight between the time they are mapped and the time they are retired.

Mapped instructions and their associated inums are placed in the integer and floating-point queues by the end of the map stage.

Stage 3 — Issue

The 20-entry integer issue queue (IQ) issues instruction at the rate of four per cycle. The 15-entry floating-point issue queue (FQ) issues floating-point operate instructions, conditional branch instructions, and store instructions, at the rate of two per cycle. Normally, instructions are deleted from the IQ or FQ two cycles after they are issued. For example, if an instruction is issued in cycle *n*, it remains in the FQ or IQ in cycle *n*+1 but does not request service, and is deleted in cycle *n*+2.

Stage 4 — Register Read

Instructions issued from the issue queues read their operands from the integer and floating register files and receive bypass data.

Stage 5 — Execute

The Ebox and Fbox pipelines begin execution.

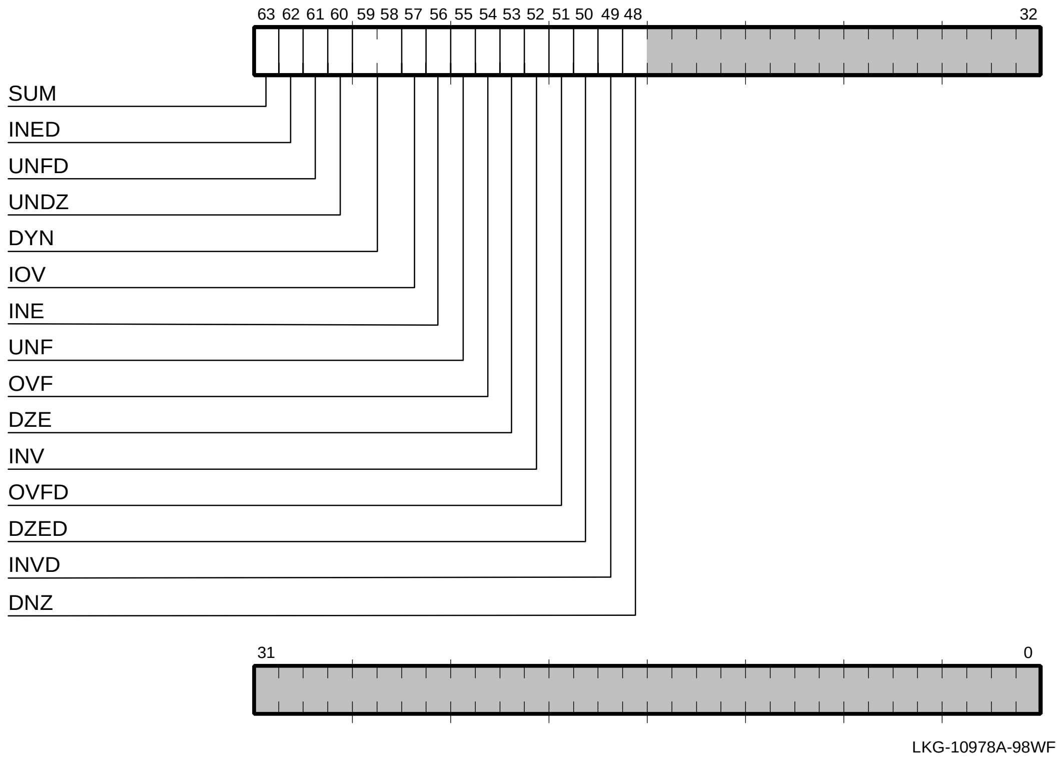
Stage 6 — Dcache Access

Memory reference instructions access the Dcache and data translation buffers. Normally load instructions access the tag and data arrays while store instructions only access the tag arrays. Store data is written to the store queue where it is held until the store instruction is retired. Most integer operate instructions write their register results in this cycle.

2.3 Floating-Point Control Register

The floating-point control register (FPCR) is shown in Figure 2–3.

Figure 2–3 Floating-Point Control Register



The floating-point control register fields are described in Table 2–1.

Table 2–1 Floating-Point Control Register Fields

Name	Extent	Type	Description
SUM	[63]	RW	Summary bit. Records bit-wise OR of FPCR exception bits.
INED	[62]	RW	Inexact Disable. If this bit is set and a floating-point instruction which enables trapping on inexact results generates an inexact value, the result is placed in the destination register and the trap is suppressed.

Table 2–1 Floating-Point Control Register Fields (Continued)

Name	Extent	Type	Description
UNFD	[61]	RW	Underflow Disable. The 21264 hardware cannot generate IEEE compliant denormal results. UNFD is used in conjunction with UNDZ as follows:

UNFD	UNDZ	Result
0	X	Underflow trap.
1	0	Trap to supply a possible denormal result.
1	1	Underflow trap suppressed. Destination is written with a true zero (+0.0).

Name	Extent	Type	Description
UNDZ	[60]	RW	Underflow to zero. When UNDZ is set together with UNFD, underflow traps are disabled and the 21264 places a true zero in the destination register. See UNFD, above.
DYN	[59:58]	RW	Dynamic rounding mode. Indicates the rounding mode to be used by an IEEE floating-point instruction when the instruction specifies dynamic rounding mode:

Bits	Meaning
00	Chopped
01	Minus infinity
10	Normal
11	Plus infinity

Name	Extent	Type	Description
IOV	[57]	RW	Integer overflow. An integer arithmetic operation or a conversion from floating-point to integer overflowed the destination precision.
INE	[56]	RW	Inexact result. A floating-point arithmetic or conversion operation gave a result that differed from the mathematically exact result.
UNF	[55]	RW	Underflow. A floating-point arithmetic or conversion operation gave a result that underflowed the destination exponent.
OVF	[54]	RW	Overflow. A floating-point arithmetic or conversion operation gave a result that overflowed the destination exponent.
DZE	[53]	RW	Divide by zero. An attempt was made to perform a floating-point divide with a divisor of zero.
INV	[52]	RW	Invalid operation. An attempt was made to perform a floating-point arithmetic operation and one or more of its operand values were illegal.
OVFD	[51]	RW	Overflow disable. If this bit is set and a floating-point arithmetic operation generates an overflow condition, then the appropriate IEEE nontrapping result is placed in the destination register and the trap is suppressed.
DZED	[50]	RW	Division by zero disable. If this bit is set and a floating-point divide by zero is detected, the appropriate IEEE nontrapping result is placed in the destination register and the trap is suppressed.
INVD	[49]	RW	Invalid operation disable. If this bit is set and a floating-point operate generates an invalid operation condition and 21264 is capable of producing the correct IEEE nontrapping result, that result is placed in the destination register and the trap is suppressed.

Table 2–1 Floating-Point Control Register Fields (Continued)

Name	Extent	Type	Description
DNZ	[48]	RW	Denormal operands to zero. If this bit is set, treat all Denormal operands as a signed zero value with the same sign as the Denormal operand.
Reserved	[47:0][1]	—	—

[1] Alpha architecture FPCR bit 47 (DNOD) is not implemented by the 21264.

2.4 AMASK and IMPLVER Values

The AMASK and IMPLVER instructions return processor type and supported architecture extensions, respectively.

2.4.1 AMASK

The 21264 returns the AMASK instruction values provided in Table 2–2. The I_CTL register reports the 21264 pass level (see I_CTL[CHIP_ID], Section 4.2.14).

Table 2–2 21264 AMASK Values

21264 Pass Level	AMASK Value Returned
Pass 1	001_{16}
Pass 2	303_{16}

The AMASK bit definitions provided in Table 2–2 are defined in Table 2–3:

Table 2–3 AMASK Bit Assignments

Bit	Meaning
0	Support for the byte/word extension (BWX) The instructions that comprise the BWX extension are LDBU, LDWU, SEXTB, SEXTW, STB, and STW.
1	Support for the square-root and floating-point convert extension (FIX) The instructions that comprise the FIX extension are FTOIS, FTOIT, ITOFF, ITOFS, ITOFT, SQRTF, SQRTG, SQRTS, and SQRTT.
8	Support for the multimedia extension (MVI) The instructions that comprise the MVI extension are MAXSB8, MAXSW4, MAXUB8, MAXUW4, MINSB8, MINSW4, MINUB8, MINUW4, PERR, PKLB, PKWB, UNPKBL, and UNPKBW.
9	Support for precise arithmetic trap reporting in hardware. The trap PC is the same as the instruction PC after the trapping instruction is executed.

2.4.2 IMPLVER

For the 21264, the IMPLVER instruction returns the value 2.

3 Hardware Interface

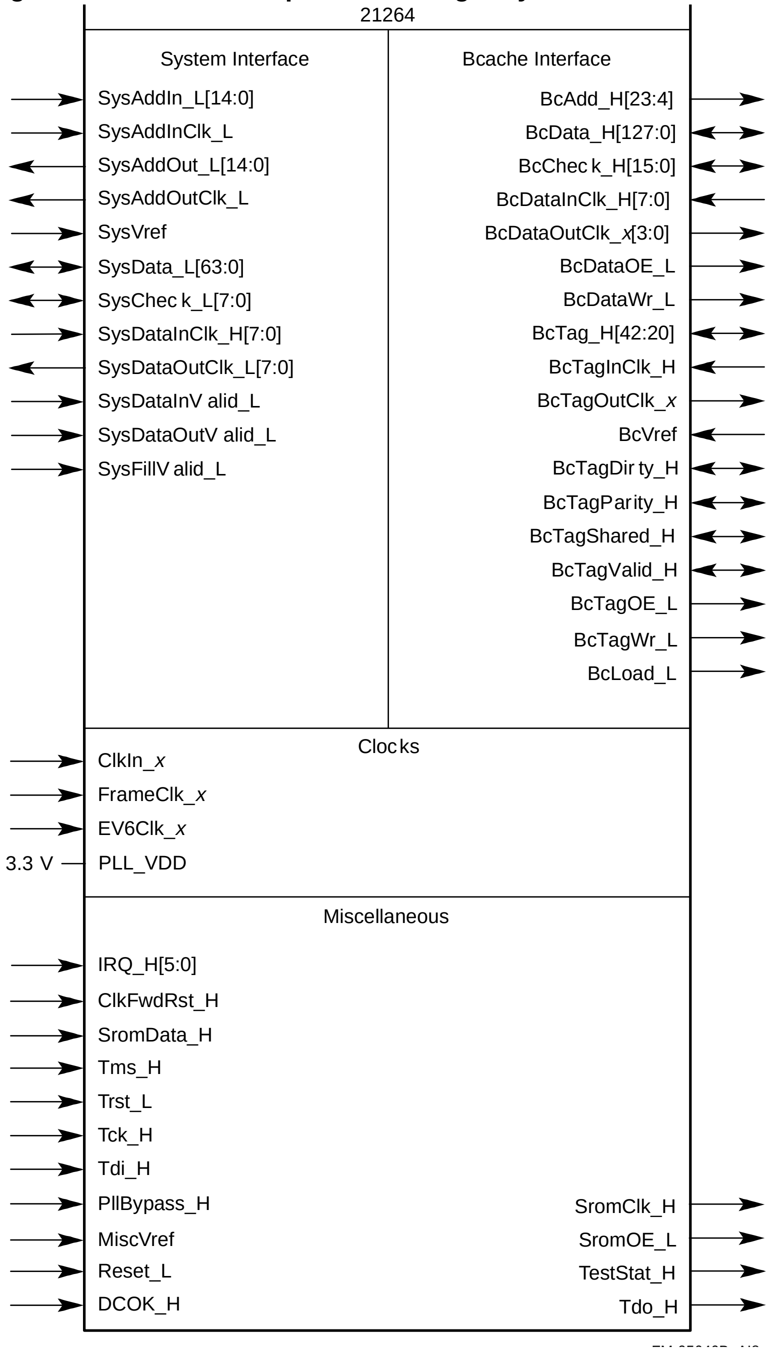
This chapter contains the 21264 microprocessor logic symbol and provides information about signal names, their function, and their location. This chapter also describes the mechanical specifications of the 21264. It is organized as follows:

- The 21264 logic symbol
- The 21264 signal names and functions
- Lists of the signal pins, sorted by name and PGA location
- The specifications for the 21264 mechanical package
- The top and bottom views of the 21264 pinouts

3.1 21264 Microprocessor Logic Symbol

Figure 3–1 show the logic symbol for the 21264 chip.

Figure 3–1 21264 Microprocessor Logic Symbol



3.2 21264 Signal Names and Functions

Table 3–1 defines the 21264 signal types referred to in this section.

Table 3–1 Signal Pin Types Definitions

Signal Type	Definition
Inputs	
I_DC_REF	Input DC reference pin
I_DA	Input differential amplifier receiver
I_DA_CLK	Input clock pin
Outputs	
O_OD	Open drain output driver
O_OD_TP	Open drain driver for test pins
O_PP	Push/pull output driver
O_PP_CLK	Push/pull output clock driver
Bidirectional	
B_DA_OD	Bidirectional differential amplifier receiver with open drain output
B_DA_PP	Bidirectional differential amplifier receiver with push/pull output
Other	
Spare	Reserved to COMPAQ[1]
NoConnect	No connection — Do not connect to these pins for any revision of the 21264. These pins must float.

[1] All Spare connections are Reserved to COMPAQ to maintain compatibility between passes of the chip. Designers should not use these pins.

Table 3–2 lists all signal pins in alphabetic order and provides a full functional description of the pins. Table 3–4 lists the signal pins and their corresponding pin grid array (PGA) locations in alphabetic order for the signal type. Table 3–5 lists the pin grid array locations in alphabetical order.

Table 3–3 lists signals by function and provides an abbreviated description.

Table 3–2 21264 Signal Descriptions

Signal	Type	Count	Description
BcAdd_H[23:4]	O_PP	20	These signals provide the index to the Bcache.
BcCheck_H[15:0]	B_DA_PP	16	ECC check bits for **BcData_H[127:0].**
BcData_H[127:0]	B_DA_PP	128	Bcache data signals.
BcDataInClk_H[7:0]	I_DA	8	Bcache data input clocks. These clocks are used with high speed SDRAMs, such as DDRs, that provide a clock-out with data-output pins to optimize Bcache read bandwidths. The 21264 internally synchronizes the data to its logic with clock forward receive circuits similar to the system interface.

Table 3–2 21264 Signal Descriptions (Continued)

Signal	Type	Count	Description
BcDataOE_L	O_PP	1	Bcache data output enable. The 21264 asserts this signal during Bcache read operations.
BcDataOutClk_H[3:0] **BcDataOutClk_L[3:0]**	O_PP	8	Bcache data output clocks. These free-running clocks are differential copies of the Bcache clock and are derived from the 21264 GCLK. Their period is a multiple of the GCLK and is fixed for all operations. They can be configured so that their rising edge lags **BcAdd_H[23:4]** by 0 to 2 GCLK cycles. The 21264 synchronizes tag output information with these clocks.
BcDataWr_L	O_PP	1	Bcache data write enable. The 21264 asserts this signal when writing data to the Bcache data arrays.
BcLoad_L	O_PP	1	Bcache burst enable.
BcTag_H[42:20]	B_DA_PP	23	Bcache tag bits.
BcTagDirty_H	B_DA_PP	1	Tag dirty state bit. During cache write operations, the 21264 will assert this signal if the Bcache data has been modified.
BcTagInClk_H	I_DA	1	Bcache tag input clock. The 21264 uses this input clock to latch the tag information on Bcache read operations. This clock is used with high-speed SDRAMs, such as DDRs, that provide a clock-out with data-output pins to optimize Bcache read bandwidths. The 21264 internally synchronizes the data to its logic with clock forward receive circuits similar to the system interface.
BcTagOE_L	O_PP	1	Bcache tag output enable. This signal is asserted by the 21264 for Bcache read operations.
BcTagOutClk_H **BcTagOutClk_L**	O_PP	2	Bcache tag output clock. These clocks “echo” the clock-forwarded **BcDataOutClk_*x*[3:0]** clocks.
BcTagParity_H	B_DA_PP	1	Tag parity state bit.
BcTagShared_H	B_DA_PP	1	Tag shared state bit. The 21264 will write a 1 on this signal line if another agent has a copy of the cache line.
BcTagValid_H	B_DA_PP	1	Tag valid state bit. If set, this line indicates that the cache line is valid.
BcTagWr_L	O_PP	1	Tag RAM write enable. The 21264 asserts this signal when writing a tag to the Bcache tag arrays.
BcVref	I_DC_REF	1	Bcache tag reference voltage.
ClkFwdRst_H	I_DA	1	Systems assert this synchronous signal to wake up a powered-down 21264. The **ClkFwdRst_H** signal is clocked into a 21264 register by the captured **FrameClk_*x*** signals. Systems must ensure that the timing of this signal meets 21264 requirements.
ClkIn_H **ClkIn_L**	I_DA_CLK	2	Differential input signals provided by the system.
DCOK_H	I_DA	1	dc voltage OK. Must be deasserted until dc voltage reaches proper operating level. After that, **DCOK_H** is asserted.
EV6Clk_H **EV6Clk_L**	O_PP_CLK	2	Provides an external test point to measure phase alignment of the PLL.

Table 3–2 21264 Signal Descriptions (Continued)

Signal	Type	Count	Description
FrameClk_H **FrameClk_L**	I_DA_CLK	2	A skew-controlled differential 50% duty cycle copy of the system clock. It is used by the 21264 as a reference, or framing, clock.
IRQ_H[5:0]	I_DA	6	These six interrupt signal lines may be asserted by the system. The response of the 21264 is determined by the system software.
MiscVref	I_DC_REF	1	Voltage reference for the miscellaneous pins (see Table 3–3).
PllBypass_H	I_DA	1	When asserted, this signal will cause the two input clocks (**ClkIn_*x***) to be applied to the 21264 internal circuits, instead of the 21264 global clock (GCLK).
PLL_VDD	3.3 V	1	3.3-V dedicated power supply for the 21264 PLL.
Reset_L	I_DA	1	System reset. This signal protects the 21264 from damage during initial power-up. It must be asserted until **DCOK_H** is asserted. After that, it is deasserted and the 21264 begins its reset sequence.
SromClk_H	O_OD_TP	1	Serial ROM clock. Supplies the clock that causes the SROM to advance to the next bit. The cycle time for this clock is 256 times the cycle time of the GCLK (internal 21264 clock).
SromData_H	I_DA	1	Serial ROM data. Input data line from the SROM.
SromOE_L	O_OD_TP	1	Serial ROM enable. Supplies the output enable to the SROM.
SysAddIn_L[14:0]	I_DA	15	Time-multiplexed command/address/ID/Ack from system to the 21264.
SysAddInClk_L	I_DA	1	Single-ended forwarded clock from system for **SysAddIn_L[14:0]** and **SysFillValid_L**.
SysAddOut_L[14:0]	O_OD	15	Time-multiplexed command/address/ID/mask from the 21264 to the system bus.
SysAddOutClk_L	O_OD	1	Single-ended forwarded clock output for **SysAddOut_L[14:0].**
SysCheck_L[7:0]	B_DA_OD	8	Quadword ECC check bits for **SysData_L[63:0].**
SysData_L[63:0]	B_DA_OD	64	Data bus for memory and I/O data.
SysDataInClk_H[7:0]	I_DA	8	Single-ended system-generated clocks for clock forwarded input system data.
SysDataInValid_L	I_DA	1	When asserted, marks a valid data cycle for data transfers to the 21264.
SysDataOutClk_L[7:0]	O_OD	8	Single-ended 21264-generated clocks for clock forwarded output system data.
SysDataOutValid_L	I_DA	1	When asserted, marks a valid data cycle for data transfers from the 21264.
SysFillValid_L	I_DA	1	When asserted, this bit indicates validation for the cache fill delivered in the previous system SysDc command.
SysVref	I_DC_REF	1	System interface reference voltage.

Table 3–2 21264 Signal Descriptions (Continued)

Signal	Type	Count	Description
Tck_H	I_DA	1	IEEE 1149.1 test clock.
Tdi_H	I_DA	1	IEEE 1149.1 test data-in signal.
Tdo_H	O_OD_TP	1	IEEE 1149.1 test data-out signal.
TestStat_H	O_OD_TP	1	Test status pin. System reset drives the test status pin low. The **TestStat_H** pin is forced high at the start of the Icache BiST. If the Icache BiST passes, the pin is deasserted at the end of the BiST operation; otherwise, it remains high. The 21264 generates a timeout reset signal if an instruction is not retired within one billion cycles. The 21264 signals the timeout reset event by outputting a 256 GCLK cycle wide pulse on **TestStat_H**.
Tms_H	I_DA	1	IEEE 1149.1 test mode select signal.
Trst_L	I_DA	1	IEEE 1149.1 test access port (TAP) reset signal.

Table 3–3 21264 Signal Descriptions by Function

Signal	Type	Count	Description
BcVref Domain			
BcAdd_H[23:4]	O_PP	20	Bcache index.
BcCheck_H[15:0]	B_DA_PP	16	ECC check bits for **BcData_H[127:0].**
BcData_H[127:0]	B_DA_PP	128	Bcache data.
BcDataInClk_H[7:0]	I_DA	8	Bcache data input clocks.
BcDataOE_L	O_PP	1	Bcache data output enable.
BcDataOutClk_H[3:0] **BcDataOutClk_L[3:0]**	O_PP	8	Bcache data output clocks.
BcDataWr_L	O_PP	1	Bcache data write enable.
BcLoad_L	O_PP	1	Bcache burst enable.
BcTag_H[42:20]	B_DA_PP	23	Bcache tag bits.
BcTagDirty_H	B_DA_PP	1	Tag dirty state bit.
BcTagInClk_H	I_DA	1	Bcache tag input clock.
BcTagOE_L	O_PP	1	Bcache tag output enable.
BcTagOutClk_H **BcTagOutClk_L**	O_PP	2	Bcache tag output clocks.
BcTagParity_H	B_DA_PP	1	Tag parity state bit.
BcTagShared_H	B_DA_PP	1	Tag shared state bit.
BcTagValid_H	B_DA_PP	1	Tag valid state bit.
BcTagWr_L	O_PP	1	Tag RAM write enable.
BcVref	I_DC_REF	1	Tag data input reference voltage.
SysVref Domain			

Table 3–3 21264 Signal Descriptions by Function (Continued)

Signal	Type	Count	Description
SysAddIn_L[14:0]	I_DA	15	Time-multiplexed SysAddIn, system-to-21264.
SysAddInClk_L	I_DA	1	Single-ended forwarded clock from system for **SysAddIn_L[14:0]** and **SysFillValid_L**.
SysAddOut_L[14:0]	O_OD	15	Time-multiplexed SysAddOut, 21264-to-system.
SysAddOutClk_L	O_OD	1	Single-ended forwarded-clock.
SysCheck_L[7:0]	B_DA_OD	8	Quadword ECC check bits for **SysData_L[63:0].**
SysData_L[63:0]	B_DA_OD	64	Data bus for memory and I/O data.
SysDataInClk_H[7:0]	I_DA	8	Single-ended system-generated clocks for clock forwarded input system data.
SysDataInValid_L	I_DA	1	When asserted, marks a valid data cycle for data transfers to the 21264.
SysDataOutClk_L[7:0]	O_OD	8	Single-ended 21264-generated clocks for clock forwarded output system data.
SysDataOutValid_L	I_DA	1	When asserted, marks a valid data cycle for data transfers from the 21264.
SysFillValid_L	I_DA	1	Validation for fill given in previous SysDC command.
SysVref	I_DC_REF	1	System interface reference voltage.
Clocks and PLL			
ClkIn_H **ClkIn_L**	I_DA_CLK	2	Differential input signals provided by the system.
EV6Clk_H **EV6Clk_L**	O_PP_CLK	2	Provides an external test point to measure phase alignment of the PLL.
FrameClk_H **FrameClk_L**	I_DA_CLK	2	A skew-controlled differential 50% duty cycle copy of the system clock. It is used by the 21264 as a reference, or framing, clock.
PLL_VDD	3.3 V	1	3.3-V dedicated power supply for the 21264 PLL.
MiscVref Domain			
ClkFwdRst_H	I_DA	1	Systems assert this synchronous signal to wake up a powered-down 21264. The **ClkFwdRst_H** signal is clocked into a 21264 register by the captured **FrameClk_*x*** signals.
DCOK_H	I_DA	1	dc voltage OK. Must be deasserted until dc voltage reaches proper operating level. After that, **DCOK_H** is asserted
IRQ_H[5:0]	I_DA	6	These six interrupt signal lines may be asserted by the system.
MiscVref	I_DC_REF	1	Reference voltage for miscellaneous pins.
PllBypass_H	I_DA	1	When asserted, this signal will cause the input clocks (**ClkIn_*x***) to be applied to the 21264 internal circuits, instead of the 21264's global clock (GCLK).

Table 3–3 21264 Signal Descriptions by Function (Continued)

Signal	Type	Count	Description
Reset_L	I_DA	1	System reset. This signal protects the 21264 from damage during initial power-up. It must be asserted until **DCOK_H** is asserted. After that, it is deasserted and the 21264 begins its reset sequence.
SromClk_H	O_OD_TP	1	Serial ROM clock.
SromData_H	I_DA	1	Serial ROM data.
SromOE_L	O_OD_TP	1	Serial ROM enable.
Tck_H	I_DA	1	IEEE 1149.1 test clock.
Tdi_H	I_DA	1	IEEE 1149.1 test data-in signal.
Tdo_H	O_OD_TP	1	IEEE 1149.1 test data-out signal.
TestStat_H	O_OD_TP	1	Test status pin.
Tms_H	I_DA	1	IEEE 1149.1 test mode select signal.
Trst_L	I_DA	1	IEEE 1149.1 test access port (TAP) reset signal.

3.3 Pin Assignments

The 21264 package has 587 pins aligned in a pin grid array (PGA) design. There are 388 functional signal pins, 1 dedicated 3.3-V pin for the PLL, 104 ground **VSS** pins, and 94 **VDD** pins. Table 3–4 lists the signal pins and their corresponding pin grid array (PGA) locations in alphabetical order for the signal type. Table 3–5 lists the pin grid array locations in alphabetical order.

Table 3–4 Pin List Sorted by Signal Name

Signal Name	PGA Location	Signal Name	PGA Location	Signal Name	PGA Location
BcAdd_H_10	B30	**BcAdd_H_11**	D30	**BcAdd_H_12**	C31
BcAdd_H_13	H28	**BcAdd_H_14**	G29	**BcAdd_H_15**	A33
BcAdd_H_16	E31	**BcAdd_H_17**	D32	**BcAdd_H_18**	B34
BcAdd_H_19	A35	**BcAdd_H_20**	B36	**BcAdd_H_21**	H30
BcAdd_H_22	C35	**BcAdd_H_23**	E33	**BcAdd_H_4**	B28
BcAdd_H_5	E27	**BcAdd_H_6**	A29	**BcAdd_H_7**	G27
BcAdd_H_8	C29	**BcAdd_H_9**	F28	**BcCheck_H_0**	F2
BcCheck_H_1	AB4	**BcCheck_H_10**	AW1	**BcCheck_H_11**	BD10
BcCheck_H_12	E45	**BcCheck_H_13**	AC45	**BcCheck_H_14**	AT44
BcCheck_H_15	BB36	**BcCheck_H_2**	AT2	**BcCheck_H_3**	BC11
BcCheck_H_4	M38	**BcCheck_H_5**	AB42	**BcCheck_H_6**	AU43
BcCheck_H_7	BC37	**BcCheck_H_8**	M8	**BcCheck_H_9**	AA3
BcData_H_0	B10	**BcData_H_1**	D10	**BcData_H_10**	L3

Table 3–4 Pin List Sorted by Signal Name (Continued)

Signal Name	PGA Location	Signal Name	PGA Location	Signal Name	PGA Location
BcData_H_100	D42	**BcData_H_101**	D44	**BcData_H_102**	H40
BcData_H_103	H42	**BcData_H_104**	G45	**BcData_H_105**	L43
BcData_H_106	L45	**BcData_H_107**	N45	**BcData_H_108**	T44
BcData_H_109	U45	**BcData_H_11**	M2	**BcData_H_110**	W45
BcData_H_111	AA43	**BcData_H_112**	AC43	**BcData_H_113**	AD44
BcData_H_114	AE41	**BcData_H_115**	AG45	**BcData_H_116**	AK44
BcData_H_117	AL43	**BcData_H_118**	AM42	**BcData_H_119**	AR45
BcData_H_12	T2	**BcData_H_120**	AP40	**BcData_H_121**	BA45
BcData_H_122	AV42	**BcData_H_123**	BB44	**BcData_H_124**	BB42
BcData_H_125	BC41	**BcData_H_126**	BA37	**BcData_H_127**	BD40
BcData_H_13	U1	**BcData_H_14**	V2	**BcData_H_15**	Y4
BcData_H_16	AC1	**BcData_H_17**	AD2	**BcData_H_18**	AE3
BcData_H_19	AG1	**BcData_H_2**	A5	**BcData_H_20**	AK2
BcData_H_21	AL3	**BcData_H_22**	AR1	**BcData_H_23**	AP2
BcData_H_24	AY2	**BcData_H_25**	BB2	**BcData_H_26**	AW5
BcData_H_27	BB4	**BcData_H_28**	BB8	**BcData_H_29**	BE5
BcData_H_3	C5	**BcData_H_30**	BB10	**BcData_H_31**	BE7
BcData_H_32	G33	**BcData_H_33**	C37	**BcData_H_34**	B40
BcData_H_35	C41	**BcData_H_36**	C43	**BcData_H_37**	E43
BcData_H_38	G41	**BcData_H_39**	F44	**BcData_H_4**	C3
BcData_H_40	K44	**BcData_H_41**	N41	**BcData_H_42**	M44
BcData_H_43	P42	**BcData_H_44**	U43	**BcData_H_45**	V44
BcData_H_46	Y42	**BcData_H_47**	AB44	**BcData_H_48**	AD42
BcData_H_49	AE43	**BcData_H_5**	E3	**BcData_H_50**	AF42
BcData_H_51	AJ45	**BcData_H_52**	AK42	**BcData_H_53**	AN45
BcData_H_54	AP44	**BcData_H_55**	AN41	**BcData_H_56**	AW45
BcData_H_57	AU41	**BcData_H_58**	AY44	**BcData_H_59**	BA43
BcData_H_6	H6	**BcData_H_60**	BC43	**BcData_H_61**	BD42
BcData_H_62	BB38	**BcData_H_63**	BE41	**BcData_H_64**	C11
BcData_H_65	A7	**BcData_H_66**	C9	**BcData_H_67**	B6
BcData_H_68	B4	**BcData_H_69**	D4	**BcData_H_7**	E1
BcData_H_70	G5	**BcData_H_71**	D2	**BcData_H_72**	H4
BcData_H_73	G1	**BcData_H_74**	N5	**BcData_H_75**	L1

Table 3–4 Pin List Sorted by Signal Name (Continued)

Signal Name	PGA Location	Signal Name	PGA Location	Signal Name	PGA Location
BcData_H_76	N1	**BcData_H_77**	U3	**BcData_H_78**	W5
BcData_H_79	W1	**BcData_H_8**	J3	**BcData_H_80**	AB2
BcData_H_81	AC3	**BcData_H_82**	AD4	**BcData_H_83**	AF4
BcData_H_84	AJ3	**BcData_H_85**	AK4	**BcData_H_86**	AN1
BcData_H_87	AM4	**BcData_H_88**	AU5	**BcData_H_89**	BA1
BcData_H_9	K2	**BcData_H_90**	BA3	**BcData_H_91**	BC3
BcData_H_92	BD6	**BcData_H_93**	BA9	**BcData_H_94**	BC9
BcData_H_95	AY12	**BcData_H_96**	A39	**BcData_H_97**	D36
BcData_H_98	A41	**BcData_H_99**	B42	**BcDataInClk_H_0**	E7
BcDataInClk_H_1	R3	**BcDataInClk_H_2**	AH2	**BcDataInClk_H_3**	BC5
BcDataInClk_H_4	F38	**BcDataInClk_H_5**	U39	**BcDataInClk_H_6**	AH44
BcDataInClk_H_7	AY40	**BcDataOE_L**	A27	**BcDataOutClk_H_0**	J5
BcDataOutClk_H_1	AU3	**BcDataOutClk_H_2**	J43	**BcDataOutClk_H_3**	AR43
BcDataOutClk_L_0	K4	**BcDataOutClk_L_1**	AV4	**BcDataOutClk_L_2**	K42
BcDataOutClk_L_3	AT42	**BcDataWr_L**	D26	**BcLoad_L**	F26
BcTag_H_20	E13	**BcTag_H_21**	H16	**BcTag_H_22**	A11
BcTag_H_23	B12	**BcTag_H_24**	D14	**BcTag_H_25**	E15
BcTag_H_26	A13	**BcTag_H_27**	G17	**BcTag_H_28**	C15
BcTag_H_29	H18	**BcTag_H_30**	D16	**BcTag_H_31**	B16
BcTag_H_32	C17	**BcTag_H_33**	A17	**BcTag_H_34**	E19
BcTag_H_35	B18	**BcTag_H_36**	A19	**BcTag_H_37**	F20
BcTag_H_38	D20	**BcTag_H_39**	E21	**BcTag_H_40**	C21
BcTag_H_41	D22	**BcTag_H_42**	H22	**BcTagDirty_H**	C23
BcTagInClk_H	G19	**BcTagOE_L**	H24	**BcTagOutClk_H**	C25
BcTagOutClk_L	D24	**BcTagParity_H**	B22	**BcTagShared_H**	G23
BcTagValid_H	B24	**BcTagWr_L**	E25	**BcVref**	F18
ClkFwdRst_H	BE11	**ClkIn_H**	AM8	**ClkIn_L**	AN7
DCOK_H	AY18	**EV6Clk_H**	AM6	**EV6Clk_L**	AL7
FrameClk_H	AV16	**FrameClk_L**	AW15	**IRQ_H_0**	BA15
IRQ_H_1	BE13	**IRQ_H_2**	AW17	**IRQ_H_3**	AV18
IRQ_H_4	BC15	**IRQ_H_5**	BB16	**MiscVref**	AV22
NoConnect	BB14	NoConnect	BD2	**PLL_VDD**	AV8
PllBypass_H	BD12	**Reset_L**	BD16	Spare	E9

Table 3–4 Pin List Sorted by Signal Name (Continued)

Signal Name	PGA Location	Signal Name	PGA Location	Signal Name	PGA Location
Spare	R5	Spare	AG5	Spare	BA7
Spare	D38	Spare	T42	Spare	AG39
Spare	AW41	Spare	F8	Spare	T4
Spare	AJ1	Spare	BD4	Spare	E39
Spare	V38	Spare	AJ43	Spare	BA39
Spare	AT4	Spare	AR3	Spare	BC21
Spare	BE9	**SromClk_H**	AW19	**SromData_H**	BC17
SromOE_L	BE17	**SysAddIn_L_0**	BD30	**SysAddIn_L_10**	BB24
SysAddIn_L_1	BC29	**SysAddIn_L_11**	AV24	**SysAddIn_L_12**	BD24
SysAddIn_L_13	BE23	**SysAddIn_L_14**	AW23	**SysAddIn_L_2**	AY28
SysAddIn_L_3	BE29	**SysAddIn_L_4**	AW27	**SysAddIn_L_5**	BA27
SysAddIn_L_6	BD28	**SysAddIn_L_7**	BE27	**SysAddIn_L_8**	AY26
SysAddIn_L_9	BC25	**SysAddInClk_L**	BB26	**SysAddOut_L_0**	AW33
SysAddOut_L_1	BE39	**SysAddOut_L_10**	BE33	**SysAddOut_L_11**	AW29
SysAddOut_L_12	BC31	**SysAddOut_L_13**	AV28	**SysAddOut_L_14**	BB30
SysAddOut_L_2	BD36	**SysAddOut_L_3**	BC35	**SysAddOut_L_4**	BA33
SysAddOut_L_5	AY32	**SysAddOut_L_6**	BE35	**SysAddOut_L_7**	AV30
SysAddOut_L_8	BB32	**SysAddOut_L_9**	BA31	**SysAddOutClk_L**	BD34
SysCheck_L_0	L7	**SysCheck_L_1**	AA5	**SysCheck_L_2**	AK8
SysCheck_L_3	BA13	**SysCheck_L_4**	L39	**SysCheck_L_5**	AA41
SysCheck_L_6	AM40	**SysCheck_L_7**	AY34	**SysData_L_0**	F14
SysData_L_1	G13	**SysData_L_10**	P6	**SysData_L_11**	T8
SysData_L_12	V8	**SysData_L_13**	V6	**SysData_L_14**	W7
SysData_L_15	Y6	**SysData_L_16**	AB8	**SysData_L_17**	AC7
SysData_L_18	AD8	**SysData_L_19**	AE5	**SysData_L_2**	F12
SysData_L_20	AH6	**SysData_L_21**	AH8	**SysData_L_22**	AJ7
SysData_L_23	AL5	**SysData_L_24**	AP8	**SysData_L_25**	AR7
SysData_L_26	AT8	**SysData_L_27**	AV6	**SysData_L_28**	AV10
SysData_L_29	AW11	**SysData_L_3**	H12	**SysData_L_30**	AV12
SysData_L_31	AW13	**SysData_L_32**	F32	**SysData_L_33**	F34
SysData_L_34	H34	**SysData_L_35**	G35	**SysData_L_36**	F40
SysData_L_37	G39	**SysData_L_38**	K38	**SysData_L_39**	J41
SysData_L_4	H10	**SysData_L_40**	M40	**SysData_L_41**	N39

Table 3–4 Pin List Sorted by Signal Name (Continued)

Signal Name	PGA Location	Signal Name	PGA Location	Signal Name	PGA Location
SysData_L_42	P40	**SysData_L_43**	T38	**SysData_L_44**	V40
SysData_L_45	W41	**SysData_L_46**	W39	**SysData_L_47**	Y40
SysData_L_48	AB38	**SysData_L_49**	AC39	**SysData_L_5**	G7
SysData_L_50	AD38	**SysData_L_51**	AF40	**SysData_L_52**	AH38
SysData_L_53	AJ39	**SysData_L_54**	AL41	**SysData_L_55**	AK38
SysData_L_56	AN39	**SysData_L_57**	AP38	**SysData_L_58**	AR39
SysData_L_59	AT38	**SysData_L_6**	F6	**SysData_L_60**	AY38
SysData_L_61	AV36	**SysData_L_62**	AW35	**SysData_L_63**	AV34
SysData_L_7	K8	**SysData_L_8**	M6	**SysData_L_9**	N7
SysDataInClk_H_0	D8	**SysDataInClk_H_1**	P4	**SysDataInClk_H_2**	AF6
SysDataInClk_H_3	AY6	**SysDataInClk_H_4**	E37	**SysDataInClk_H_5**	R43
SysDataInClk_H_6	AG41	**SysDataInClk_H_7**	AV40	**SysDataInValid_L**	BD22
SysDataOutClk_L_0	G11	**SysDataOutClk_L_1**	U7	**SysDataOutClk_L_2**	AG7
SysDataOutClk_L_3	AY8	**SysDataOutClk_L_4**	H36	**SysDataOutClk_L_5**	R41
SysDataOutClk_L_6	AH40	**SysDataOutClk_L_7**	AW39	**SysDataOutValid_L**	BB22
SysFillValid_L	BC23	**SysVref**	BA25	**Tck_H**	BE19
Tdi_H	BA21	**Tdo_H**	BB20	**TestStat_H**	BA19
Tms_H	BD18	**Trst_L**	AY20	—	—

Table 3–5 Pin List Sorted by PGA Location

PGA Location	Signal Name	PGA Location	Signal Name	PGA Location	Signal Name
A11	**BcTag_H_22**	A13	**BcTag_H_26**	A17	**BcTag_H_33**
A19	**BcTag_H_36**	A27	**BcDataOE_L**	A29	**BcAdd_H_6**
A33	**BcAdd_H_15**	A35	**BcAdd_H_19**	A39	**BcData_H_96**
A41	**BcData_H_98**	A5	**BcData_H_2**	A7	**BcData_H_65**
AA3	**BcCheck_H_9**	AA41	**SysCheck_L_5**	AA43	**BcData_H_111**
AA5	**SysCheck_L_1**	AB2	**BcData_H_80**	AB38	**SysData_L_48**
AB4	**BcCheck_H_1**	AB42	**BcCheck_H_5**	AB44	**BcData_H_47**
AB8	**SysData_L_16**	AC1	**BcData_H_16**	AC3	**BcData_H_81**
AC39	**SysData_L_49**	AC43	**BcData_H_112**	AC45	**BcCheck_H_13**
AC7	**SysData_L_17**	AD2	**BcData_H_17**	AD38	**SysData_L_50**
AD4	**BcData_H_82**	AD42	**BcData_H_48**	AD44	**BcData_H_113**
AD8	**SysData_L_18**	AE3	**BcData_H_18**	AE41	**BcData_H_114**

Table 3–5 Pin List Sorted by PGA Location (Continued)

PGA Location	Signal Name	PGA Location	Signal Name	PGA Location	Signal Name
AE43	**BcData_H_49**	AE5	**SysData_L_19**	AF4	**BcData_H_83**
AF40	**SysData_L_51**	AF42	**BcData_H_50**	AF6	**SysDataInClk_H_2**
AG1	**BcData_H_19**	AG39	Spare	AG41	**SysDataInClk_H_6**
AG45	**BcData_H_115**	AG5	Spare	AG7	**SysDataOutClk_L_2**
AH2	**BcDataInClk_H_2**	AH38	**SysData_L_52**	AH40	**SysDataOutClk_L_6**
AH44	**BcDataInClk_H_6**	AH6	**SysData_L_20**	AH8	**SysData_L_21**
AJ1	Spare	AJ3	**BcData_H_84**	AJ39	**SysData_L_53**
AJ43	Spare	AJ45	**BcData_H_51**	AJ7	**SysData_L_22**
AK2	**BcData_H_20**	AK38	**SysData_L_55**	AK4	**BcData_H_85**
AK42	**BcData_H_52**	AK44	**BcData_H_116**	AK8	**SysCheck_L_2**
AL3	**BcData_H_21**	AL41	**SysData_L_54**	AL43	**BcData_H_117**
AL5	**SysData_L_23**	AL7	**EV6Clk_L**	AM4	**BcData_H_87**
AM40	**SysCheck_L_6**	AM42	**BcData_H_118**	AM6	**EV6Clk_H**
AM8	**ClkIn_H**	AN1	**BcData_H_86**	AN39	**SysData_L_56**
AN41	**BcData_H_55**	AN45	**BcData_H_53**	AN7	**ClkIn_L**
AP2	**BcData_H_23**	AP38	**SysData_L_57**	AP40	**BcData_H_120**
AP44	**BcData_H_54**	AP8	**SysData_L_24**	AR1	**BcData_H_22**
AR3	Spare	AR39	**SysData_L_58**	AR43	**BcDataOutClk_H_3**
AR45	**BcData_H_119**	AR7	**SysData_L_25**	AT2	**BcCheck_H_2**
AT38	**SysData_L_59**	AT4	Spare	AT42	**BcDataOutClk_L_3**
AT44	**BcCheck_H_14**	AT8	**SysData_L_26**	AU3	**BcDataOutClk_H_1**
AU41	**BcData_H_57**	AU43	**BcCheck_H_6**	AU5	**BcData_H_88**
AV10	**SysData_L_28**	AV12	**SysData_L_30**	AV16	**FrameClk_H**
AV18	**IRQ_H_3**	AV22	**MiscVref**	AV24	**SysAddIn_L_11**
AV28	**SysAddOut_L_13**	AV30	**SysAddOut_L_7**	AV34	**SysData_L_63**
AV36	**SysData_L_61**	AV4	**BcDataOutClk_L_1**	AV40	**SysDataInClk_H_7**
AV42	**BcData_H_122**	AV6	**SysData_L_27**	AV8	**PLL_VDD**
AW1	**BcCheck_H_10**	AW11	**SysData_L_29**	AW13	**SysData_L_31**
AW15	**FrameClk_L**	AW17	**IRQ_H_2**	AW19	**SromCLK_H**
AW23	**SysAddIn_L_14**	AW27	**SysAddIn_L_4**	AW29	**SysAddOut_L_11**
AW33	**SysAddOut_L_0**	AW35	**SysData_L_62**	AW39	**SysDataOutClk_L_7**
AW41	Spare	AW45	**BcData_H_56**	AW5	**BcData_H_26**
AY12	**BcData_H_95**	AY18	**DCOK_H**	AY2	**BcData_H_24**

Table 3–5 Pin List Sorted by PGA Location (Continued)

PGA Location	Signal Name	PGA Location	Signal Name	PGA Location	Signal Name
AY20	**Trst_L**	AY26	**SysAddIn_L_8**	AY28	**SysAddIn_L_2**
AY32	**SysAddOut_L_5**	AY34	**SysCheck_L_7**	AY38	**SysData_L_60**
AY40	**BcDataInClk_H_7**	AY44	**BcData_H_58**	AY6	**SysDataInClk_H_3**
AY8	**SysDataOutClk_L_3**	B10	**BcData_H_0**	B12	**BcTag_H_23**
B16	**BcTag_H_31**	B18	**BcTag_H_35**	B22	**BcTagParity_H**
B24	**BcTagValid_H**	B28	**BcAdd_H_4**	B30	**BcAdd_H_10**
B34	**BcAdd_H_18**	B36	**BcAdd_H_20**	B4	**BcData_H_68**
B40	**BcData_H_34**	B42	**BcData_H_99**	B6	**BcData_H_67**
BA1	**BcData_H_89**	BA13	**SysCheck_L_3**	BA15	**IRQ_H_0**
BA19	**TestStat_H**	BA21	**Tdi_H**	BA25	**SysVref**
BA27	**SysAddIn_L_5**	BA3	**BcData_H_90**	BA31	**SysAddOut_L_9**
BA33	**SysAddOut_L_4**	BA37	**BcData_H_126**	BA39	Spare
BA43	**BcData_H_59**	BA45	**BcData_H_121**	BA7	Spare
BA9	**BcData_H_93**	BB10	**BcData_H_30**	BB14	NoConnect
BB16	**IRQ_H_5**	BB2	**BcData_H_25**	BB20	**Tdo_H**
BB22	**SysDataOutValid_L**	BB24	**SysAddIn_L_10**	BB26	**SysAddInClk_L**
BB30	**SysAddOut_L_14**	BB32	**SysAddOut_L_8**	BB36	**BcCheck_H_15**
BB38	**BcData_H_62**	BB4	**BcData_H_27**	BB42	**BcData_H_124**
BB44	**BcData_H_123**	BB8	**BcData_H_28**	BC11	**BcCheck_H_3**
BC15	**IRQ_H_4**	BC17	**SromData_H**	BC21	Spare
BC23	**SysFillValid_L**	BC25	**SysAddIn_L_9**	BC29	**SysAddIn_L_1**
BC3	**BcData_H_91**	BC31	**SysAddOut_L_12**	BC35	**SysAddOut_L_3**
BC37	**BcCheck_H_7**	BC41	**BcData_H_125**	BC43	**BcData_H_60**
BC5	**BcDataInClk_H_3**	BC9	**BcData_H_94**	BD10	**BcCheck_H_11**
BD12	**PllBypass_H**	BD16	**Reset_L**	BD18	**Tms_H**
BD2	NoConnect	BD22	**SysDataInValid_L**	BD24	**SysAddIn_L_12**
BD28	**SysAddIn_L_6**	BD30	**SysAddIn_L_0**	BD34	**SysAddOutClk_L**
BD36	**SysAddOut_L_2**	BD4	Spare	BD40	**BcData_H_127**
BD42	**BcData_H_61**	BD6	**BcData_H_92**	BE11	**ClkFwdRst_H**
BE13	**IRQ_H_1**	BE17	**SromOE_L**	BE19	**Tck_H**
BE23	**SysAddIn_L_13**	BE27	**SysAddIn_L_7**	BE29	**SysAddIn_L_3**
BE33	**SysAddOut_L_10**	BE35	**SysAddOut_L_6**	BE39	**SysAddOut_L_1**
BE41	**BcData_H_63**	BE5	**BcData_H_29**	BE7	**BcData_H_31**

Table 3–5 Pin List Sorted by PGA Location (Continued)

PGA Location	Signal Name	PGA Location	Signal Name	PGA Location	Signal Name
BE9	Spare	C11	**BcData_H_64**	C15	**BcTag_H_28**
C17	**BcTag_H_32**	C21	**BcTag_H_40**	C23	**BcTagDirty_H**
C25	**BcTagOutClk_H**	C29	**BcAdd_H_8**	C3	**BcData_H_4**
C31	**BcAdd_H_12**	C35	**BcAdd_H_22**	C37	**BcData_H_33**
C41	**BcData_H_35**	C43	**BcData_H_36**	C5	**BcData_H_3**
C9	**BcData_H_66**	D10	**BcData_H_1**	D14	**BcTag_H_24**
D16	**BcTag_H_30**	D2	**BcData_H_71**	D20	**BcTag_H_38**
D22	**BcTag_H_41**	D24	**BcTagOutClk_L**	D26	**BcDataWr_L**
D30	**BcAdd_H_11**	D32	**BcAdd_H_17**	D36	**BcData_H_97**
D38	Spare	D4	**BcData_H_69**	D42	**BcData_H_100**
D44	**BcData_H_101**	D8	**SysDataInClk_H_0**	E1	**BcData_H_7**
E13	**BcTag_H_20**	E15	**BcTag_H_25**	E19	**BcTag_H_34**
E21	**BcTag_H_39**	E25	**BcTagWr_L**	E27	**BcAdd_H_5**
E3	**BcData_H_5**	E31	**BcAdd_H_16**	E33	**BcAdd_H_23**
E37	**SysDataInClk_H_4**	E39	Spare	E43	**BcData_H_37**
E45	**BcCheck_H_12**	E7	**BcDataInClk_H_0**	E9	Spare
F12	**SysData_L_2**	F14	**SysData_L_0**	F18	**BcVref**
F2	**BcCheck_H_0**	F20	**BcTag_H_37**	F26	**BcLoad_L**
F28	**BcAdd_H_9**	F32	**SysData_L_32**	F34	**SysData_L_33**
F38	**BcDataInClk_H_4**	F40	**SysData_L_36**	F44	**BcData_H_39**
F6	**SysData_L_6**	F8	Spare	G1	**BcData_H_73**
G11	**SysDataOutClk_L_0**	G13	**SysData_L_1**	G17	**BcTag_H_27**
G19	**BcTagInClk_H**	G23	**BcTagShared_H**	G27	**BcAdd_H_7**
G29	**BcAdd_H_14**	G33	**BcData_H_32**	G35	**SysData_L_35**
G39	**SysData_L_37**	G41	**BcData_H_38**	G45	**BcData_H_104**
G5	**BcData_H_70**	G7	**SysData_L_5**	H10	**SysData_L_4**
H12	**SysData_L_3**	H16	**BcTag_H_21**	H18	**BcTag_H_29**
H22	**BcTag_H_42**	H24	**BcTagOE_L**	H28	**BcAdd_H_13**
H30	**BcAdd_H_21**	H34	**SysData_L_34**	H36	**SysDataOutClk_L_4**
H4	**BcData_H_72**	H40	**BcData_H_102**	H42	**BcData_H_103**
H6	**BcData_H_6**	J3	**BcData_H_8**	J41	**SysData_L_39**
J43	**BcDataOutClk_H_2**	J5	**BcDataOutClk_H_0**	K2	**BcData_H_9**
K38	**SysData_L_38**	K4	**BcDataOutClk_L_0**	K42	**BcDataOutClk_L_2**

Table 3–5 Pin List Sorted by PGA Location (Continued)

PGA Location	Signal Name	PGA Location	Signal Name	PGA Location	Signal Name
K44	**BcData_H_40**	K8	**SysData_L_7**	L1	**BcData_H_75**
L3	**BcData_H_10**	L39	**SysCheck_L_4**	L43	**BcData_H_105**
L45	**BcData_H_106**	L7	**SysCheck_L_0**	M2	**BcData_H_11**
M38	**BcCheck_H_4**	M40	**SysData_L_40**	M44	**BcData_H_42**
M6	**SysData_L_8**	M8	**BcCheck_H_8**	N1	**BcData_H_76**
N39	**SysData_L_41**	N41	**BcData_H_41**	N45	**BcData_H_107**
N5	**BcData_H_74**	N7	**SysData_L_9**	P4	**SysDataInClk_H_1**
P40	**SysData_L_42**	P42	**BcData_H_43**	P6	**SysData_L_10**
R3	**BcDataInClk_H_1**	R41	**SysDataOutClk_L_5**	R43	**SysDataInClk_H_5**
R5	Spare	T2	**BcData_H_12**	T38	**SysData_L_43**
T4	Spare	T42	Spare	T44	**BcData_H_108**
T8	**SysData_L_11**	U1	**BcData_H_13**	U3	**BcData_H_77**
U39	**BcDataInClk_H_5**	U43	**BcData_H_44**	U45	**BcData_H_109**
U7	**SysDataOutClk_L_1**	V2	**BcData_H_14**	V38	Spare
V40	**SysData_L_44**	V44	**BcData_H_45**	V6	**SysData_L_13**
V8	**SysData_L_12**	W1	**BcData_H_79**	W39	**SysData_L_46**
W41	**SysData_L_45**	W45	**BcData_H_110**	W5	**BcData_H_78**
W7	**SysData_L_14**	Y4	**BcData_H_15**	Y40	**SysData_L_47**
Y42	**BcData_H_46**	Y6	**SysData_L_15**	—	—

Table 3–6 lists the 21264 ground and power (**VSS** and **VDD**, respectively) pin list.

Table 3–6 Ground and Power (VSS and VDD) Pin List

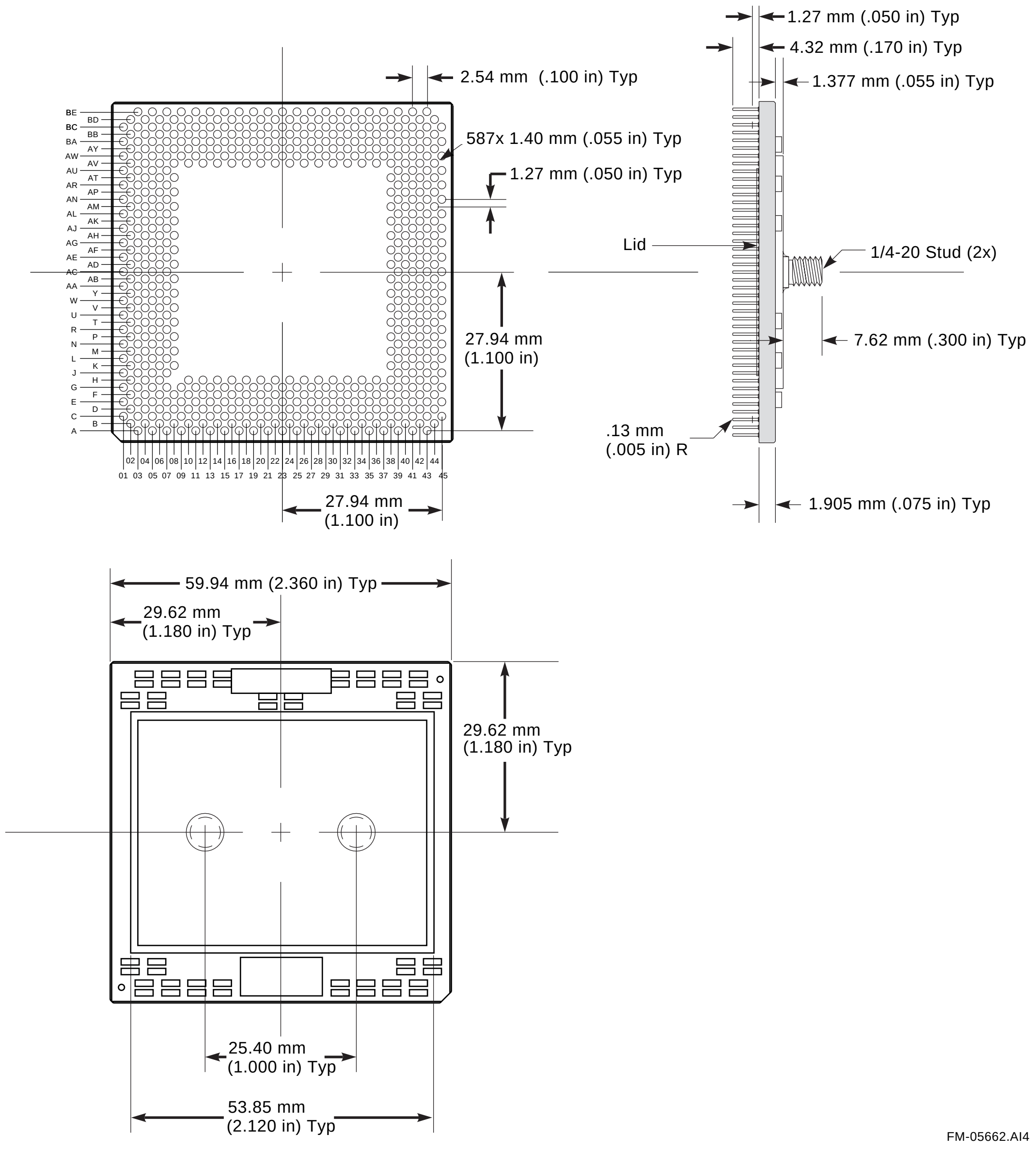
Signal	PGA Location									
VSS	C1	W3	AR5	G9	E17	G25	C33	AA39	BA41	R45
	J1	AG3	BA5	AW9	BA17	AW25	BC33	AE39	A43	AA45
	R1	AN3	C7	C19	BE25	E35	AL39	G43	AE45	
	AA1	AW3	J7	E11	BC19	C27	BA35	AU39	N43	AL45
	AE1	BE3	R7	BA11	A21	BC27	A37	BC39	W43	AU45
	AL1	E5	AA7	C13	G21	E29	G37	E41	AG43	BC45
	AU1	L5	AE7	BC13	AW21	BA29	AW37	L41	AN43	
	BC1	U5	AU7	A15	BE21	A31	BE37	U41	AW43	
	A3	AC5	AW7	G15	E23	G31	C39	AC41	BE43	
	G3	AJ5	BC7	AY14	BA23	AW31	J39	AJ41	C45	
	N3	AN5	A9	BE15	A25	BE31	R39	AR41	J45	
VDD	B2	V4	AP6	D12	B20	H26	BD32	AM38	BB40	Y44
	H2	AH4	AT6	BB12	H20	AV26	D34	AV38	F42	AF44
	P2	AP4	BB6	B14	AV20	BD26	BB34	BD38	M42	AM44
	Y2	AY4	B8	H14	BD20	D28	F36	D40	V42	AV44
	AF2	D6	P8	AV14	F22	BB28	AY36	K40	AH42	BD44
	AM2	K6	Y8	BD14	AY22	F30	B38	T40	AP42	
	AV2	T6	AF8	F16	A23	AY30	H38	AB40	AY42	
	AB6	BD8	AY16	F24	B32	P38	AD40	B44	F4	
	AD6	F10	D18	AY24	H32	Y38	AK40	H44	M4	
	AK6	AY10	BB18	B26	AV32	AF38	AT40	P44		

3.4 Mechanical Specifications

This section shows the 21264 mechanical package dimensions without a heat sink. For heat sink information and dimensions, refer to Chapter 8.

Figure 3–2 shows the package physical dimensions without a heat sink.

Figure 3–2 Package Dimensions



3.5 21264 Packaging

Figure 3–3 shows the 21264 pinout from the top view with pins facing down. Figure 3–4 shows the 21264 pinout from the bottom view with pins facing up.

Figure 3–3 21264 Top View (Pin Down)

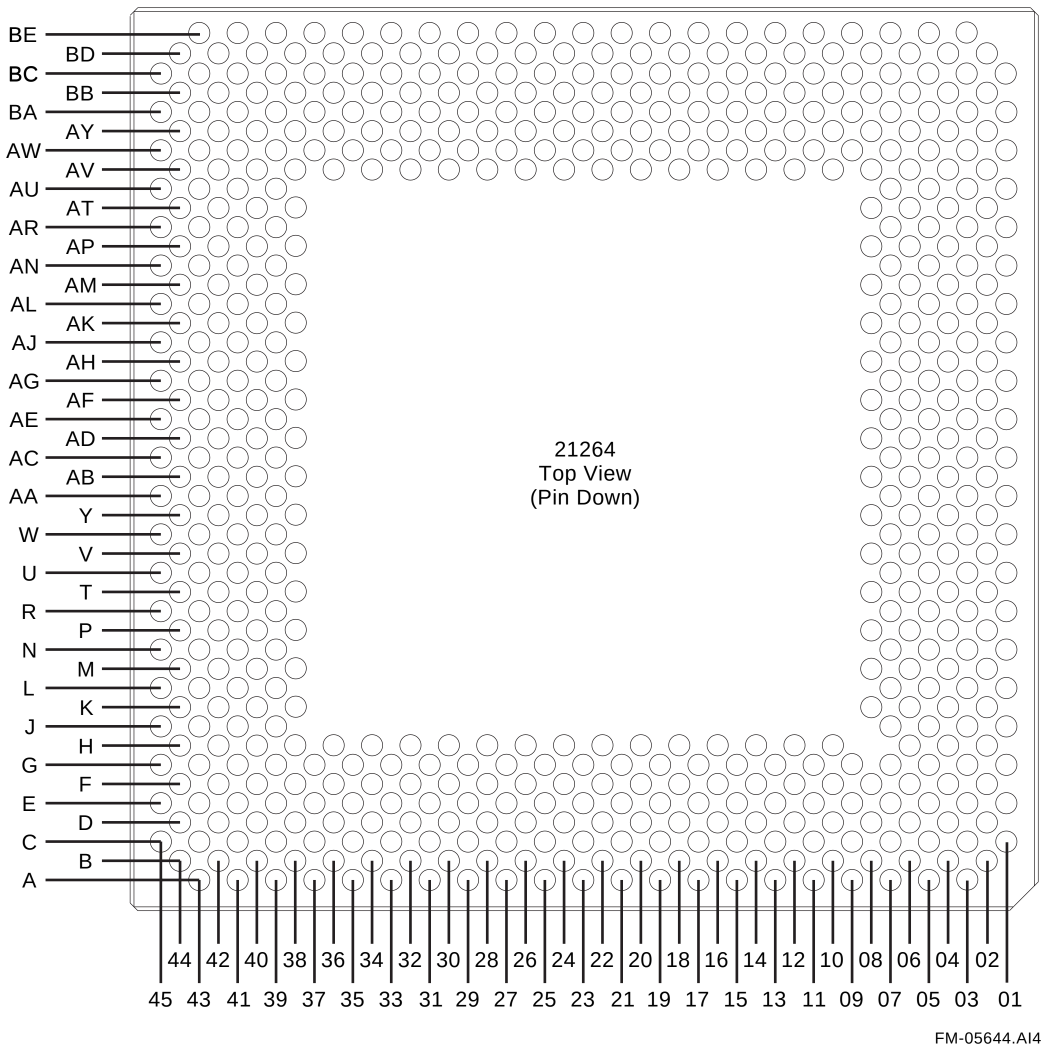
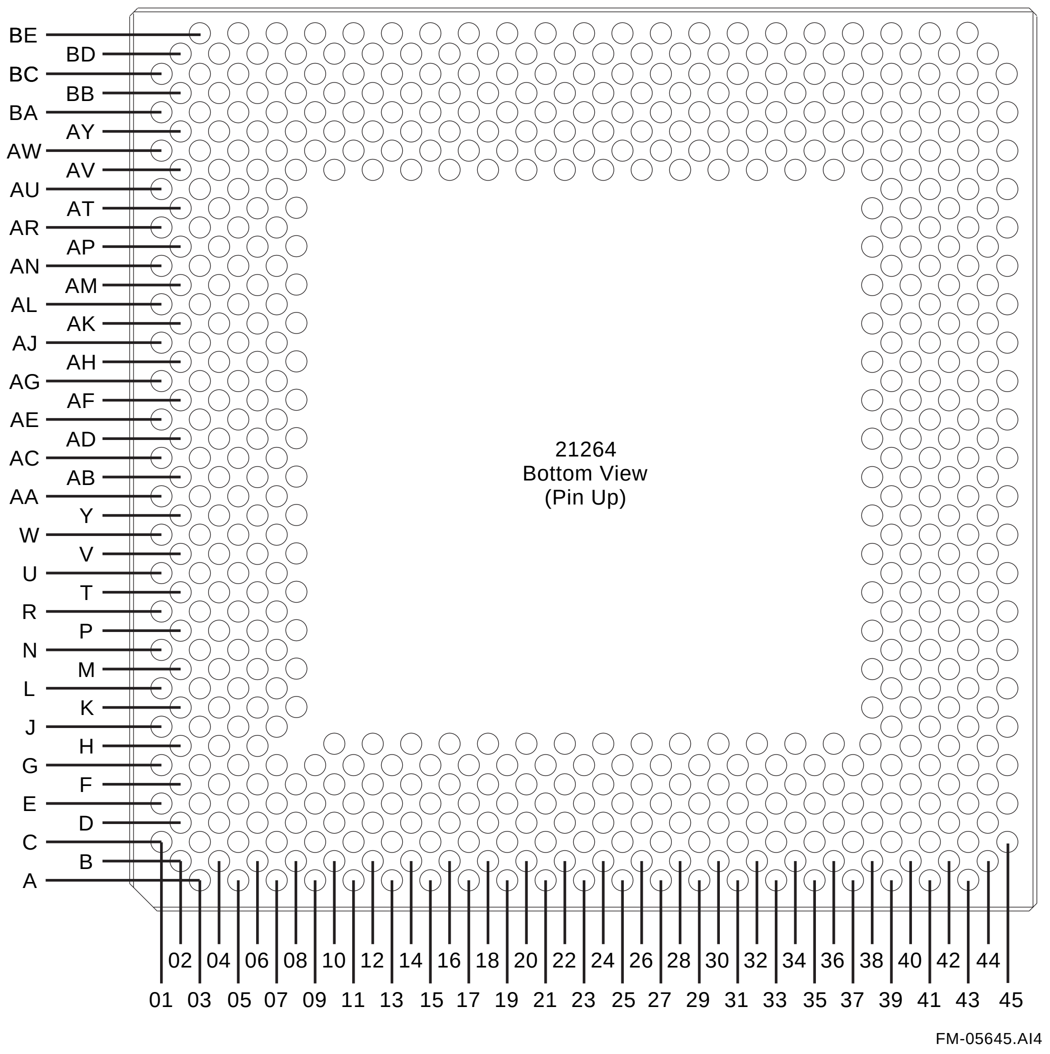


Figure 3–4 21264 Bottom View (Pin Up)



4 Internal Processor Registers

This chapter describes 21264 internal processor registers (IPRs). They are separated into the following circuit logic groups: Ebox, Ibox, Mbox, and Cbox.

The gray areas in register figures indicate reserved fields. Bit ranges that are coupled with the field name specify those bits in that named field that are included in the IPR. For example, in Figure 4–2, the field named COUNTER[31:4] contains bits 31 through 4 of the COUNTER field from Section 4.1.1 . The bit range of COUNTER[31:4] in the IPR is also listed in the column *Extent* in Table 4–2. In many cases, such as this one, the bit ranges correspond. However, the bit range of the named field need not always correspond to the *Extent* in the IPR. For example, in Figure 4–13, the field VA[47:13] resides in IPR IVA_FORM[37:3] under the stated conditions.

The register contents after initialization are listed in Section 6.2.

Table 4–1 lists the 21264 internal processor registers.

Table 4–1 Internal Processor Registers

Register Name	Mnemonic	Index (Binary)	Score-Board Bit	Access	MT/MF Issued from Ebox Pipe	Latency for MFPR (Cycles)
Ebox IPRs						
Cycle counter	CC	1100 0000	5	RW	1L	1
Cycle counter control	CC_CTL	1100 0001	5	W0	1L	—
Virtual address	VA	1100 0010	4, 5, 6, 7	RO	1L	1
Virtual address control	VA_CTL	1100 0100	5	WO	1L	—
Virtual address format	VA_FORM	1100 0011	4, 5, 6, 7	RO	1L	1

Table 4–1 Internal Processor Registers (Continued)

Register Name	Mnemonic	Index (Binary)	Score-Board Bit	Access	MT/MF Issued from Ebox Pipe	Latency for MFPR (Cycles)
Ibox IPRs						
ITB tag array write	ITB_TAG	0000 0000	6	WO	0L	—
ITB PTE array write	ITB_PTE	0000 0001	4, 0	WO	0L	—
ITB invalidate all process (ASM=0)	ITB_IAP	0000 0010	4	WO	0L	—
ITB invalidate all	ITB_IA	0000 0011	4	WO	0L	—
ITB invalidate single	ITB_IS	0000 0100	4, 6	WO	0L	—
Exception address	EXC_ADDR	0000 0110	—	RO	0L	3
Instruction VA format	IVA_FORM	0000 0111	5	RO	0L	3
Current mode	CM	0000 1001	4	RW	0L	3
Interrupt enable	IER	0000 1010	4	RW	0L	3
Interrupt enable and current mode	IER_CM	0000 10xx	4	RW	0L	3
Software interrupt request	SIRR	0000 1100	4	RW	0L	3
Interrupt summary	ISUM	0000 1101	—	RO	—	—
Hardware interrupt clear	HW_INT_CLR	0000 1110	4	WO	0L	—
Exception summary	EXC_SUM	0000 1111	—	RO	0L	3
PAL base address	PAL_BASE	0001 0000	4	RW	0L	3
Ibox control	I_CTL	0001 0001	4	RW	0L	3
Ibox status	I_STAT	0001 0110	4	RW	0L	3
Icache flush	IC_FLUSH	0001 0011	4	W	0L	—
Icache flush ASM	IC_FLUSH_ASM	0001 0010	4	WO	0L	—
Clear virtual-to-physical map	CLR_MAP	0001 0101	4, 5, 6, 7	WO	0L	—
Sleep mode	SLEEP	0001 0111	4, 5, 6, 7	WO	0L	—
Process context register	PCTX	01x*n nnnn*[1]	4	W	0L	3
Process context register	PCTX	01xx xxxx	4	R	0L	3
Performance counter control	PCTR_CTL	0001 0100	4	RW	0L	3
Mbox IPRs						
DTB tag array write 0	DTB_TAG0	0010 0000	2, 6	WO	0L	—
DTB tag array write 1	DTB_TAG1	1010 0000	1, 5	WO	1L	—
DTB PTE array write 0	DTB_PTE0	0010 0001	0, 4	WO	0L	—
DTB PTE array write 1	DTB_PTE1	1010 0001	3, 7	WO	0L	—
DTB alternate processor mode	DTB_ALTMODE	0010 0110	6	WO	1L	—
DTB invalidate all process (ASM = 0)	DTB_IAP	1010 0010	7	WO	1L	—
DTB invalidate all	DTB_IA	1010 0011	7	WO	1L	—

Table 4–1 Internal Processor Registers (Continued)

Register Name	Mnemonic	Index (Binary)	Score-Board Bit	Access	MT/MF Issued from Ebox Pipe	Latency for MFPR (Cycles)
Mbox IPRs (cont.)						
DTB invalidate single (array 0)	DTB_IS0	0010 0100	6	WO	0L	—
DTB invalidate single (array 1)	DTB_IS1	1010 0100	7	WO	1L	—
DTB address space number 0	DTB_ASN0	0010 0101	4	WO	0L	—
DTB address space number 1	DTB_ASN1	1010 0101	7	WO	1L	—
Memory management status	MM_STAT	0010 0111	—	RO	0L	3
Mbox control	M_CTL	0010 1000	6	WO	0L	—
Dcache control	DC_CTL	0010 1001	6	WO	0L	—
Dcache status	DC_STAT	0010 1010	6	RW	0L	3
Cbox IPRs						
Cbox data	C_DATA	0010 1011	6	RW	0L	3
Cbox shift control	C_SHFT	0010 1100	6	WO	0L	Ò

[1]When *n* equals 1, that process context field is selected (FPE, PPCE, ASTRR, ASTER, ASN).

4.1 Ebox IPRs

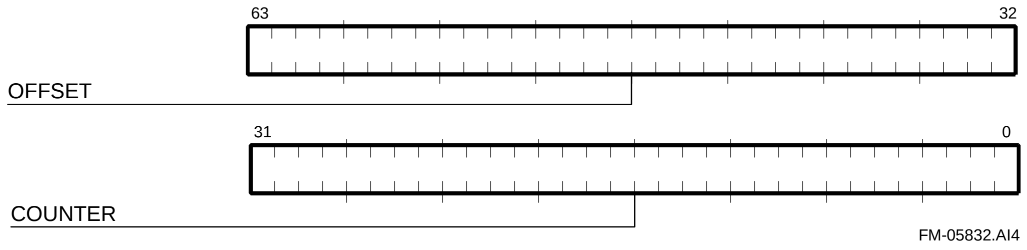
This section describes the internal processor registers that control Ebox functions.

4.1.1 Cycle Counter Register – CC

The cycle counter register (CC) is a read-write register. The lower half of CC is a counter that, when enabled by way of CC_CTL[32], increments once each CPU cycle. The upper half of the register is 32 bits of register storage that may be used as a counter offset as described in the *Alpha Architecture Handbook, Version 4* under Processor Cycle Counter (PCC) Register.

A HW_MTPR instruction to the CC writes the upper half of the register and leaves the lower half unchanged. The RPCC instruction returns the full 64-bit value of the register. Figure 4–1 shows the cycle counter register.

Figure 4–1 Cycle Counter Register



4.1.2 Cycle Counter Control Register – CC_CTL

The cycle counter control register (CC_CTL) is a write-only register through which the lower half of the CC register may be written and its associated counter enabled and disabled. Figure 4–2 shows the cycle counter control register.

Figure 4–2 Cycle Counter Control Register

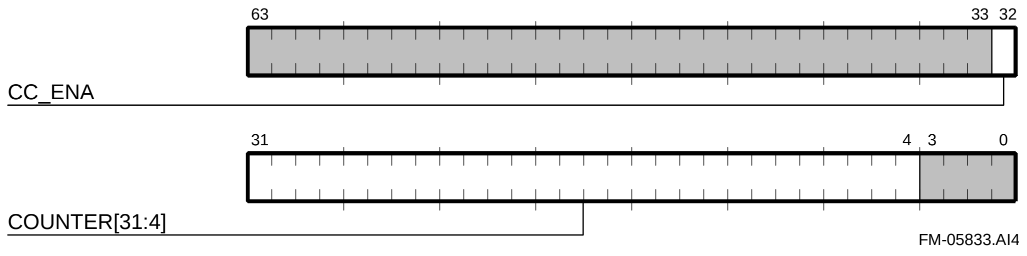


Table 4–2 describes the CC_CTL register fields.

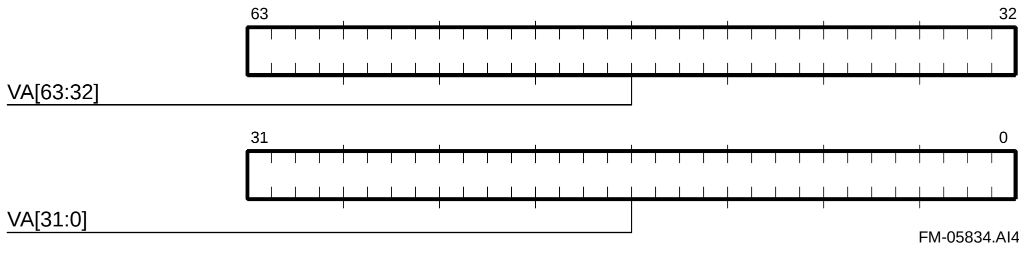
Table 4–2 Cycle Counter Control Register Fields Description

Name	Extent	Type	Description
Reserved	[63:33]	—	—
CC_ENA	[32]	WO	Counter Enable. When set, this bit allows the cycle counter to increment.
COUNTER[31:4]	[31:4]	WO	CC[31:4] may be written by way of this field. Write transactions to CC_CTL result in CC[3:0] being cleared.
Reserved	[3:0]	—	—

4.1.3 Virtual Address Register – VA

The virtual address register (VA) is a read-only register. When a DTB miss or fault occurs, the associated effective virtual address is written into the VA register. VA is not written when a LD_VPTE gets a DTB miss or Dstream fault. Figure 4–3 shows the virtual address register.

Figure 4–3 Virtual Address Register



4.1.4 Virtual Address Control Register – VA_CTL

The virtual address control register (VA_CTL) is a write-only register that controls the way in which the faulting virtual address stored in the VA register is formatted when it is read by way of the VA_FORM register. It also contains control bits that affect the behavior of the memory pipe virtual address sign extension checkers and the behavior of the Ebox extract, insert, and mask instructions. Figure 4–4 shows the virtual address control register.

Figure 4–4 Virtual Address Control Register

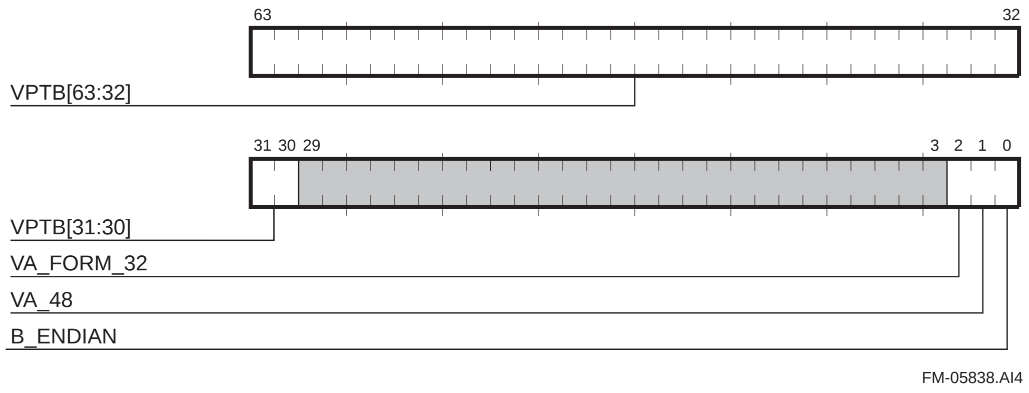


Table 4–3 describes the virtual address control register fields.

Table 4–3 Virtual Address Control Register Fields Description

Name	Extent	Type	Description
VPTB[63:30]	[63:30]	WO	Virtual Page Table Base. See the VA_FORM register section for details.
Reserved	[29:3]	—	—
VA_FORM_32	[2]	WO,0	This bit is used to control address formatting when reading the VA_FORM register. See the section on the VA_FORM register for details.
VA_48	[1]	WO,0	This bit controls the format applied to effective virtual addresses by the VA_FORM register and the memory pipe virtual address sign extension checkers. When VA_48 is clear, the 43-bit virtual address format is used, and when VA_48 is set, the 48-bit virtual address format is used. When VA_48 is set, the sign extension checkers generate an access control violation (ACV) if VA[63:0] ≠ SEXT (VA[47:0]). When VA_48 is clear, the sign extension checkers generate an ACV if VA[63:0] ≠ SEXT(VA[42:0]).
B_ENDIAN	[0]	WO,0	Big Endian Mode. When set, the shift amount (Rbv[2:0]) is inverted for EXTxx, INSxx, and MSKxx instructions. The lower bits of the physical address for Dstream accesses are inverted based upon the length of the reference as follows: Byte: Invert bits [2:0] Word: Invert bits [2:1] Longword: Inverts bit [2]

4.1.5 Virtual Address Format Register – VA_FORM

The virtual address format register (VA_FORM) is a read-only register. It contains the virtual page table entry address derived from the faulting virtual address stored in the VA register. It also contains the virtual page table base and associated control bits stored in the VA_CTL register.

Figure 4–5 shows VA_FORM when VA_CTL(VA_48) equals 0 and VA_CTL(VA_FORM_32) equals 0.

Figure 4–5 Virtual Address Format Register (VA_48 = 0, VA_FORM_32 = 0)

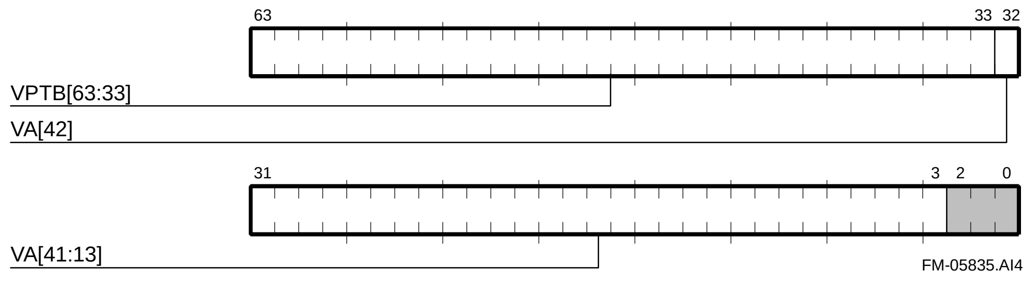


Figure 4–6 shows VA_FORM when VA_CTL(VA_48) equals 1 and VA_CTL(VA_FORM_32) equals 0.

Figure 4–6 Virtual Address Format Register (VA_48 = 1, VA_FORM_32 = 0)

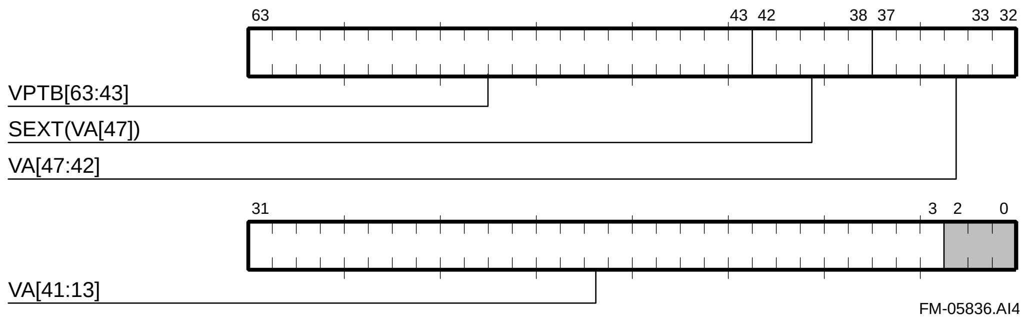
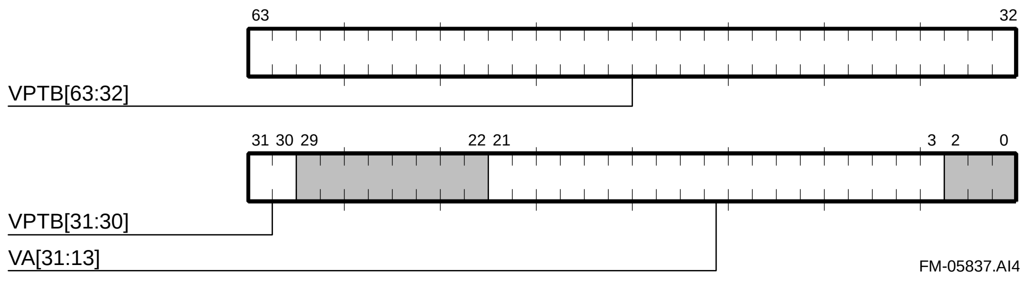


Figure 4–7 shows VA_FORM when VA_CTL(VA_48) equals 0 and VA_CTL(VA_FORM_32) equals 1.

Figure 4–7 Virtual Address Format Register (VA_48 = 0, VA_FORM_32 = 1)



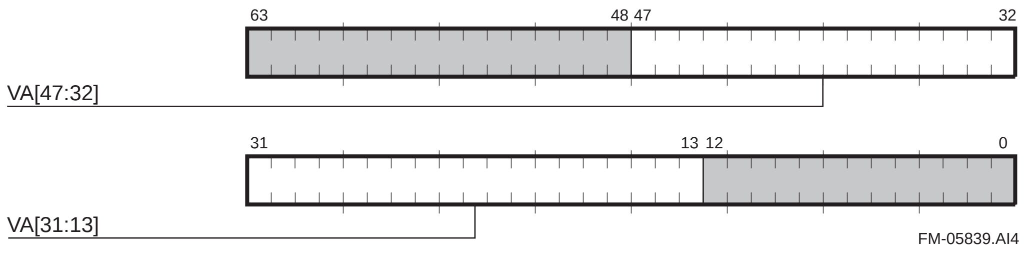
4.2 Ibox IPRs

This section describes the internal processor registers that control Ibox functions.

4.2.1 ITB Tag Array Write Register – ITB_TAG

The ITB tag array write register (ITB_TAG) is a write-only register. The ITB tag array is written by way of this register. A write transaction to ITB_TAG writes a register outside the ITB array. When a write to the ITB_PTE register is retired, the contents of both the ITB_TAG and ITB_PTE registers are written into the ITB entry. The specific ITB entry that is written is determined by a round-robin algorithm; the algorithm writes to entry number 0 as the first entry after the 21264 is reset. Figure 4–8 shows the ITB tag array write register.

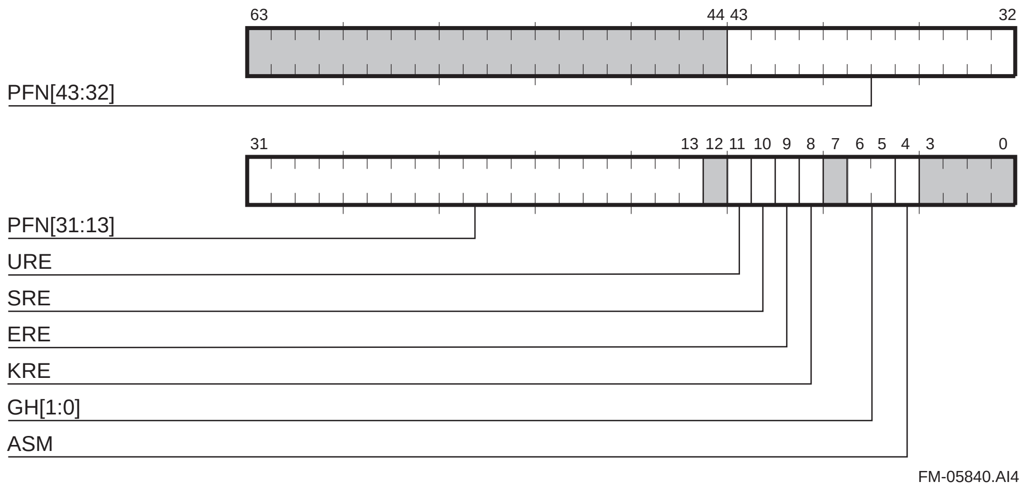
Figure 4–8 ITB Tag Array Write Register



4.2.2 ITB PTE Array Write Register – ITB_PTE

The ITB PTE array write register (ITB_PTE) is a write-only register through which the ITB PTE array is written. A round-robin allocation algorithm is used. A write to the ITB_PTE array, when retired, results in both the ITB_TAG and ITB_PTE arrays being written. The specific entry that is written is chosen by the round-robin algorithm described above. Figure 4–9 shows the ITB PTE array write register.

Figure 4–9 ITB PTE Array Write Register



4.2.3 ITB Invalidate All Process (ASM=0) Register – ITB_IAP

The ITB invalidate all process register (ITB_IAP) is a pseudo register that, when written to, invalidates all ITB entries whose ASM bit is clear. An explicit write to IC_FLUSH_ASM is required to flush the Icache of blocks with ASM equal to zero.

4.2.4 ITB Invalidate All Register – ITB_IA

The ITB invalidate all register (ITB_IA) is a pseudo register that, when written to, invalidates all ITB entries and resets the allocation pointer to its initial state. An explicit write to IC_FLUSH is required to flush the Icache.

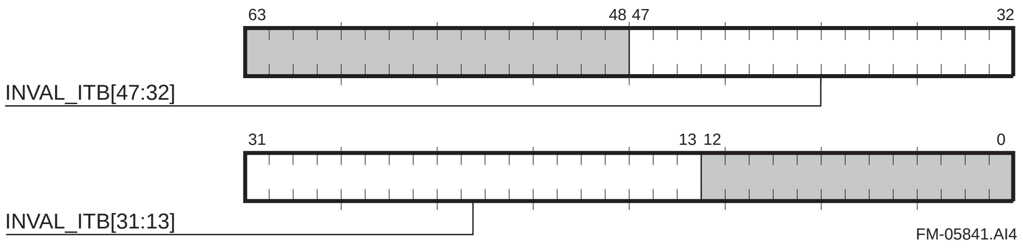
4.2.5 ITB Invalidate Single Register – ITB_IS

The ITB invalidate single register (ITB_IS) is a write-only register. Writing a virtual page number to this register invalidates any ITB entry that meets one of the following criteria:

- The ITB entry's virtual page number matches ITB_IS[47:13] (or fewer bits if granularity hint bits are set in the ITB entry) and its ASN field matches the address space number supplied in PCTX[46:39].
- The ITB entry's virtual page number matches ITB_IS[47:13] and its ASM bit is set.

Figure 4–10 shows the ITB invalidate single register.

Figure 4–10 ITB Invalidate Single Register



Note: Because the Icache is virtually indexed and tagged, it is normally not necessary to flush the Icache when paging. Therefore, a write to ITB_IS will not flush the Icache.

4.2.6 Exception Address Register – EXC_ADDR

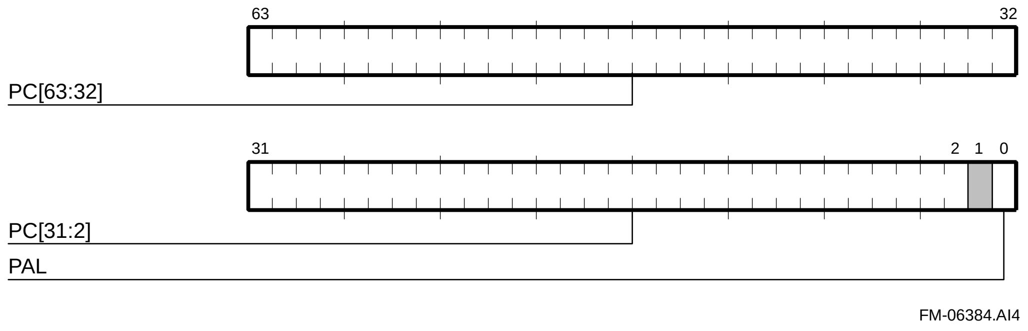
The exception address register (EXC_ADDR) is a read-only register that is updated by hardware when it encounters an exception or interrupt.

EXC_ADDR[0] is set if the associated exception occurred in PAL mode. The exception actions are listed here:

- If the exception was a fault or a synchronous trap, EXC_ADDR contains the PC of the instruction that triggered the fault or trap.
- If the exception was an interrupt, EXC_ADDR contains the PC of the next instruction that would have executed if the interrupt had not occurred.

Figure 4–11 shows the exception address register.

Figure 4–11 Exception Address Register



4.2.7 Instruction Virtual Address Format Register — IVA_FORM

The instruction virtual address format register (IVA_FORM) is a read-only register. It contains the virtual PTE address derived from the faulting virtual address stored in the EXC_ADDR register, and from the virtual page table base, VA_48 and VA_FORM_32 bits, stored in the I_CTL register.

Figure 4–12 shows IVA_FORM when VA_48 equals 0 and VA_FORM_32 equals 0.

Figure 4–12 Instruction Virtual Address Format Register (VA_48 = 0, VA_FORM_32 = 0)

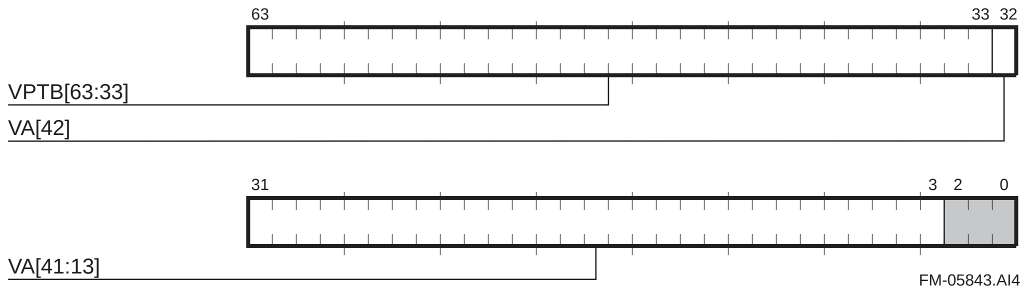


Figure 4–13 shows IVA_FORM when VA_48 equals 1 and VA_FORM_32 equals 0.

Figure 4–13 Instruction Virtual Address Format Register (VA_48 = 1, VA_FORM_32 = 0)

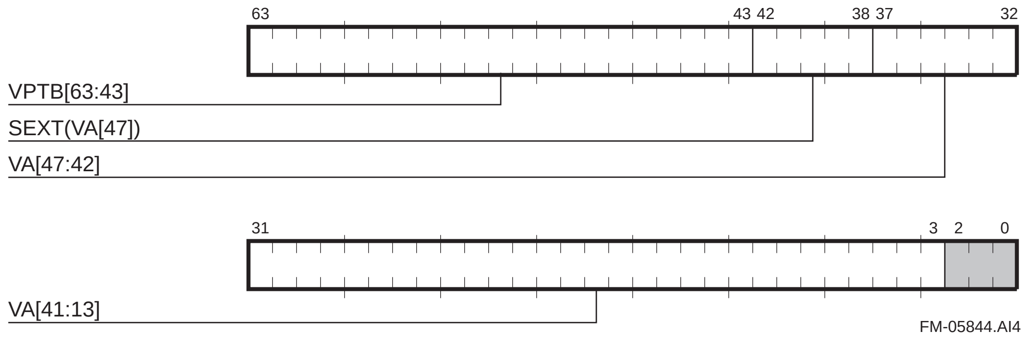
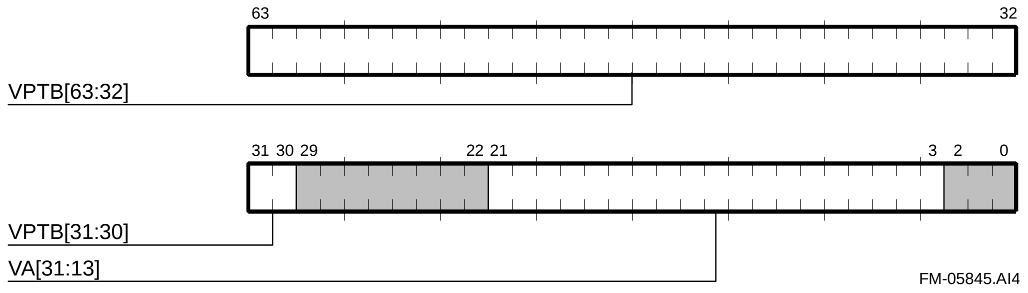


Figure 4–14 shows IVA_FORM when VA_48 equals 0 and VA_FORM_32 equals 1.

Figure 4–14 Instruction Virtual Address Format Register (VA_48 = 0, VA_FORM_32 = 1)



4.2.8 Interrupt Enable and Current Processor Mode Register – IER_CM

The interrupt enable and current processor mode register (IER_CM) contains the interrupt enable and current processor mode bit fields. These bit fields can be written either individually or together with a single HW_MTPR instruction. When bits [7:2] of the IPR index field of a HW_MTPR instruction contain the value 000010_2, this register is selected. Bits [1:0] of the IPR index indicate which bit fields are to be written: bit[1] corresponds to the IER field and bit[0] corresponds to the processor mode field. A HW_MFPR instruction to this register returns the values in both fields. Figure 4–15 shows the interrupt enable and current processor mode register.

Figure 4–15 Interrupt Enable and Current Processor Mode Register

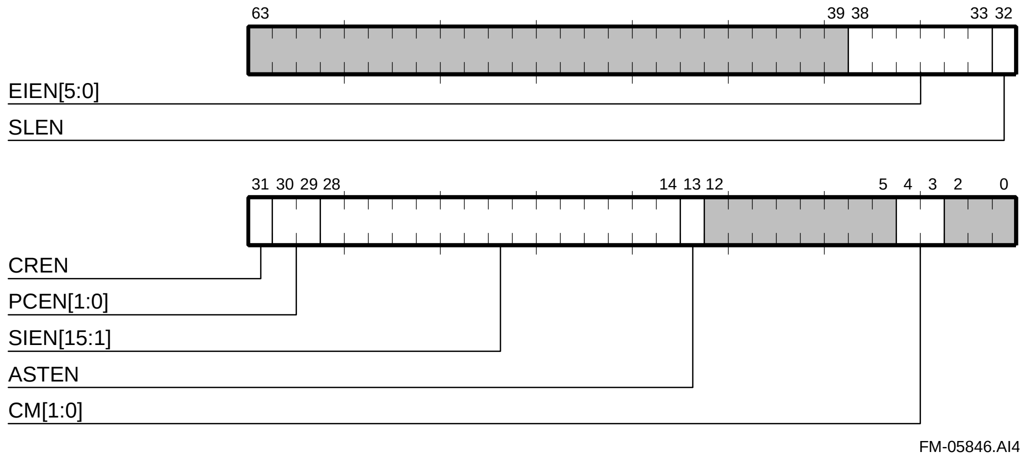


Table 4–4 describes the interrupt enable and current processor mode register fields.

Table 4–4 IER_CM Register Fields Description

Name	Extent	Type	Description
Reserved	[63:39]	—	—
EIEN[5:0]	[38:33]	RW	External Interrupt Enable
SLEN	[32]	RW	Serial Line Interrupt Enable
CREN	[31]	RW	Corrected Read Error Interrupt Enable
PCEN[1:0]	[30:29]	RW	Performance Counter Interrupt Enables
SIEN[15:1]	[28:14]	RW	Software Interrupt Enables

Table 4–4 IER_CM Register Fields Description (Continued)

Name	Extent	Type	Description
ASTEN	[13]	RW	AST Interrupt Enable When set, enables those AST interrupt requests that are also enabled by the value in ASTER.
Reserved	[12:5]	—	—
CM[1:0]	[4:3]	RW	Current Mode 00 Kernel 01 Executive 10 Supervisor 11 User
Reserved	[2:0]	—	—

4.2.9 Software Interrupt Request Register – SIRR

The software interrupt request register (SIRR) is a read-write register containing bits to request software interrupts. To generate a particular software interrupt, its corresponding bits in SIRR and IER[SIER] must both be set. Figure 4–16 shows the software interrupt request register.

Figure 4–16 Software Interrupt Request Register

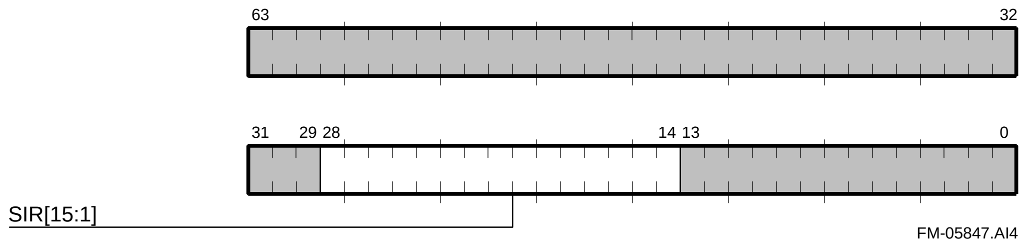


Table 4–5 describes the software interrupt request register fields.

Table 4–5 Software Interrupt Request Register Fields Description

Name	Extent	Type	Description
Reserved	[63:29]	—	—
SIR[15:1]	[28:14]	RW	Software Interrupt Requests
Reserved	[13:0]	—	—

4.2.10 Interrupt Summary Register – ISUM

The interrupt summary register (ISUM) is a read-only register that records all pending hardware, software, and AST interrupt requests that have their corresponding enable bit set.

If a new interrupt (hardware, serial line, crd, or performance counters) occurs simultaneously with an ISUM read, the ISUM read returns zeros. That condition is normally assumed to be a passive release condition. The interrupt is signaled again when the PALcode returns to native mode. The effects of this condition can be minimized by reading ISUM twice and ORing the results.

Figure 4–17 shows the interrupt summary register.

Figure 4–17 Interrupt Summary Register

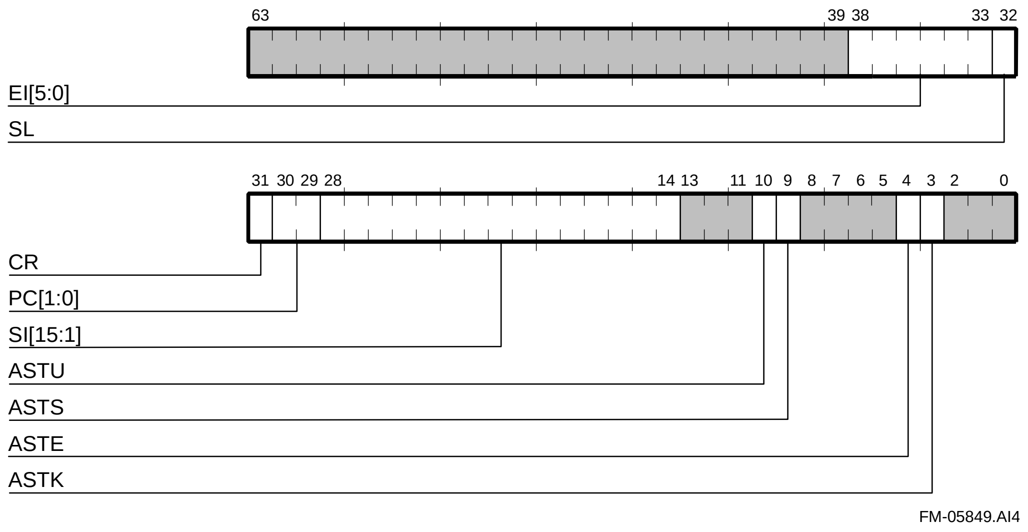


Table 4–6 describes the interrupt summary register fields.

Table 4–6 Interrupt Summary Register Fields Description

Name	Extent	Type	Description
Reserved	[63:39]	—	—
EI[5:0]	[38:33]	RO	External Interrupts
SL	[32]	RO	Serial Line Interrupt
CR	[31]	RO	Corrected Read Error Interrupts
PC[1:0]	[30:29]	RO	Performance Counter Interrupts PC0 when PC[0] is set. PC1 when PC[1] is set.
SI[15:1]	[28:14]	RO	Software Interrupts
Reserved	[13:11]	—	—
ASTU, ASTS	[10],[9]	RO	AST Interrupts For each processor mode, the bit is set if an associated AST interrupt is pending. This includes the mode's ASTER and ASTRR bits and whether the processor mode value held in the IER_CM register is greater than or equal to the value for the mode.

Table 4–6 Interrupt Summary Register Fields Description (Continued)

Name	Extent	Type	Description
Reserved	[8:5]	—	—
ASTE, ASTK	[4],[3]	RO	AST Interrupts For each processor mode, the bit is set if an associated AST interrupt is pending. This includes the mode’s ASTER and ASTRR bits and whether the processor mode value held in the IER_CM register is greater than or equal to the value for the mode.
Reserved	[2:0]	—	—

4.2.11 Hardware Interrupt Clear Register – HW_INT_CLR

The hardware interrupt clear register (HW_INT_CLR) is a write-only register used to clear edge-sensitive interrupt requests. Figure 4–18 shows the hardware interrupt clear register.

Figure 4–18 Hardware Interrupt Clear Register

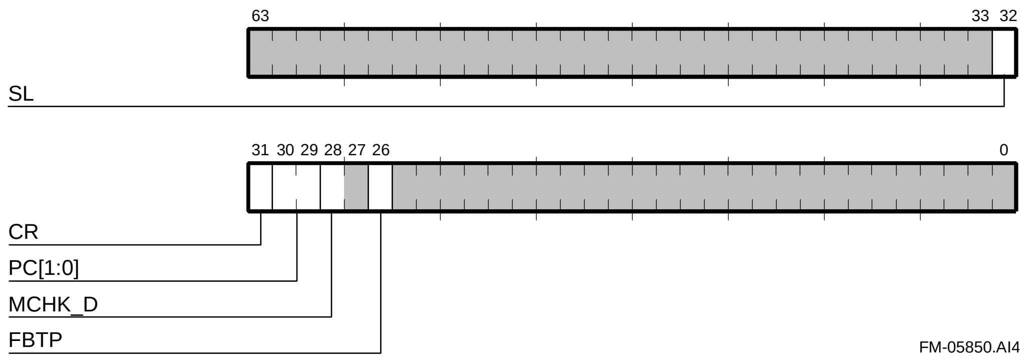


Table 4–7 describes the hardware interrupt clear register fields.

Table 4–7 Hardware Interrupt Clear Register Fields Description

Name	Extent	Type	Description
Reserved	[63:33]	—	—
SL	[32]	W1C	Clears serial line interrupt request
CR	[31]	W1C	Clears corrected read error interrupt request
PC[1:0]	[30:29]	W1C	Clears performance counter interrupt requests
MCHK_D	[28]	W1C	Clears Dstream machine check interrupt request
Reserved	[27]	—	—
FBTP	[26]	W1S	Forces the next Bcache hit that fills the Icache to generate bad Icache fill parity
Reserved	[25:0]	—	—

4.2.12 Exception Summary Register – EXC_SUM

The exception summary register (EXC_SUM) is a read-only register that contains information about instructions that have triggered traps. The register is updated at trap delivery time. Its contents are valid only if it is read (by way of a HW_MFPR) in the first fetch block of the exception handler. There are three types of traps for which this register captures related information:

- Arithmetic traps: The instruction generated an exceptional condition that should be reported to the operating system, and/or the FPCR status bit associated with this condition is clear and should be set by PALcode. Additionally, the REG field contains the register number of the destination specifier for the instruction that triggered the trap.
- Istream ACV: The BAD_IVA bit of this register indicates whether the offending Istream virtual address is latched into the EXC_ADDR register or the VA register.
- Dstream exceptions: The REG field contains the register number of either the source specifier (for stores) or the destination specifier (for loads) of the instruction that triggered the trap.

Figure 4–19 shows the exception summary register.

Figure 4–19 Exception Summary Register

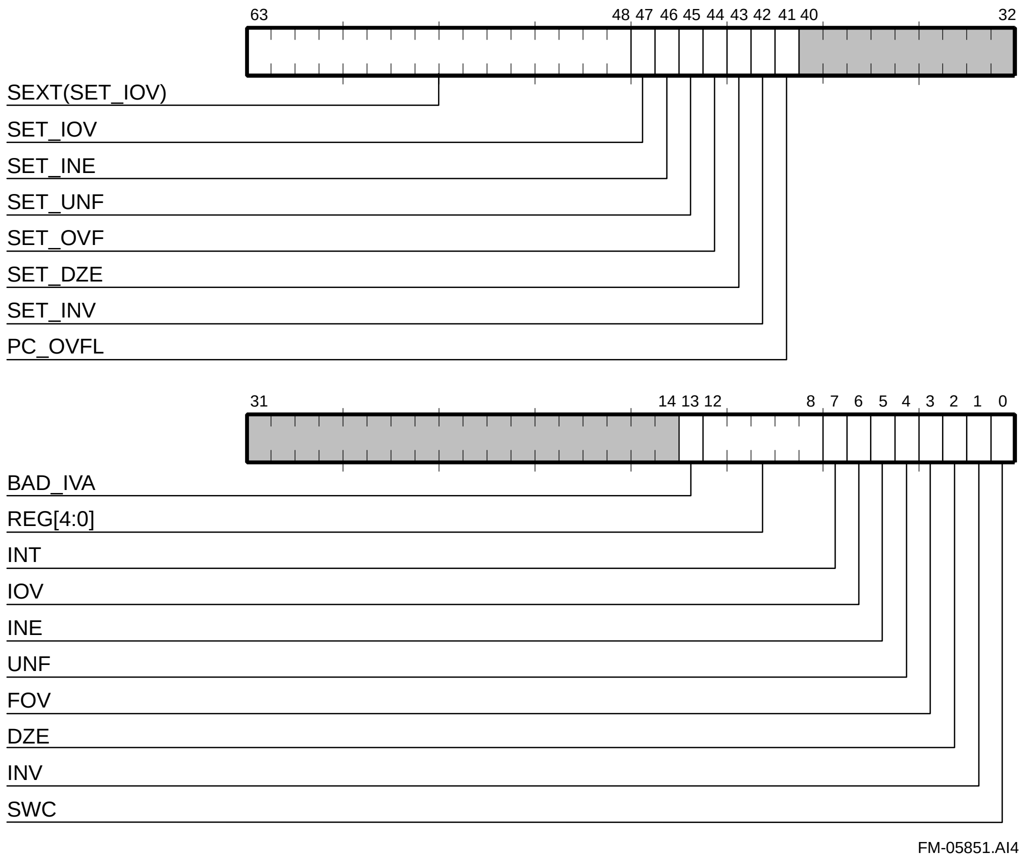


Table 4–8 describes the exception summary register fields.

Table 4–8 Exception Summary Register Fields Description

Name	Extent	Type	Description
SEXT(SET_IOV)	[63:48]	RO, 0	Sign-extended value of SET_IOV because it is bit 47.
SET_IOV	[47]	RO	PALcode should set FPCR[IOV].
SET_INE	[46]	RO	PALcode should set FPCR[INE].
SET_UNF	[45]	RO	PALcode should set FPCR[UNF].
SET_OVF	[44]	RO	PALcode should set FPCR[OVF].
SET_DZE	[43]	RO	PALcode should set FPCR[DZE].
SET_INV	[42]	RO	PALcode should set FPCR[INV].
PC_OVFL	[41]	RO	Indicates that EXC_ADDR was improperly sign extended for 48-bit mode over/underflow IACV.
Reserved	[40:14]	RO, 0	Reserved for COMPAQ.
BAD_IVA	[13]	RO	Bad Istream VA. This bit should be used by the IACV PALcode routine to determine whether the offending I-stream virtual address is latched in the EXC_ADDR register or the VA register. If BAD_IVA is clear, then EXC_ADDR contains the address, if BAD_IVA is set then VA contains the address.
REG[4:0]	[12:8]	RO	Destination register of load or operate instruction that triggered the trap OR source register of store that triggered the trap. These bits may contain the Rc field of an operate instruction or the Ra field of a load or store instruction. The value is UNPREDICTABLE if the trap was triggered by an ITB miss, interrupt, OPCDEC, or other non load/st/operate.
INT	[7]	RO	Set to indicate Ebox integer overflow trap, clear to indicate Fbox trap condition.
IOV	[6]	RO	Indicates Fbox convert-to-integer overflow or Ebox integer overflow trap.
INE	[5]	RO	Indicates floating-point inexact error trap.
UNF	[4]	RO	Indicates floating-point underflow trap.
FOV	[3]	RO	Indicates floating-point overflow trap.
DZE	[2]	RO	Indicates divide by zero trap.
INV	[1]	RO	Indicates invalid operation trap.
SWC	[0]	RO	Indicates software completion possible. This bit is set if the instruction that triggered the trap contained the /S modifier.

4.2.13 PAL Base Register – PAL_BASE

The PAL base register (PAL_BASE) is a read-write register that contains the base physical address for PALcode. Its contents are cleared by chip reset but are not cleared after waking up from sleep mode or from fault reset. Figure 4–20 shows the PAL base register.

Figure 4–20 PAL Base Register

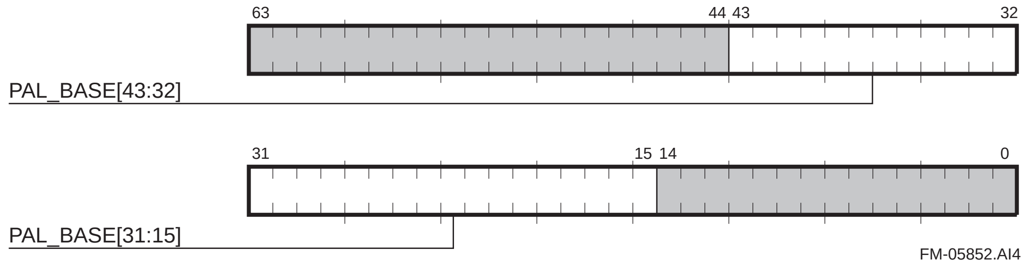


Table 4–9 describes the PAL base register fields.

Table 4–9 PAL Base Register Fields Description

Name	Extent	Type	Description
Reserved	[63:44]	RO, 0	Reserved for COMPAQ.
PAL_BASE[43:15]	[43:15]	RW	Base physical address for PALcode.
Reserved	[14:0]	RO, 0	Reserved for COMPAQ.

4.2.14 Ibox Control Register – I_CTL

The Ibox control register (I_CTL) is a read-write register that controls various Ibox functions. Its contents are cleared by chip reset. Figure 4–21 shows the Ibox control register.

Figure 4–21 Ibox Control Register

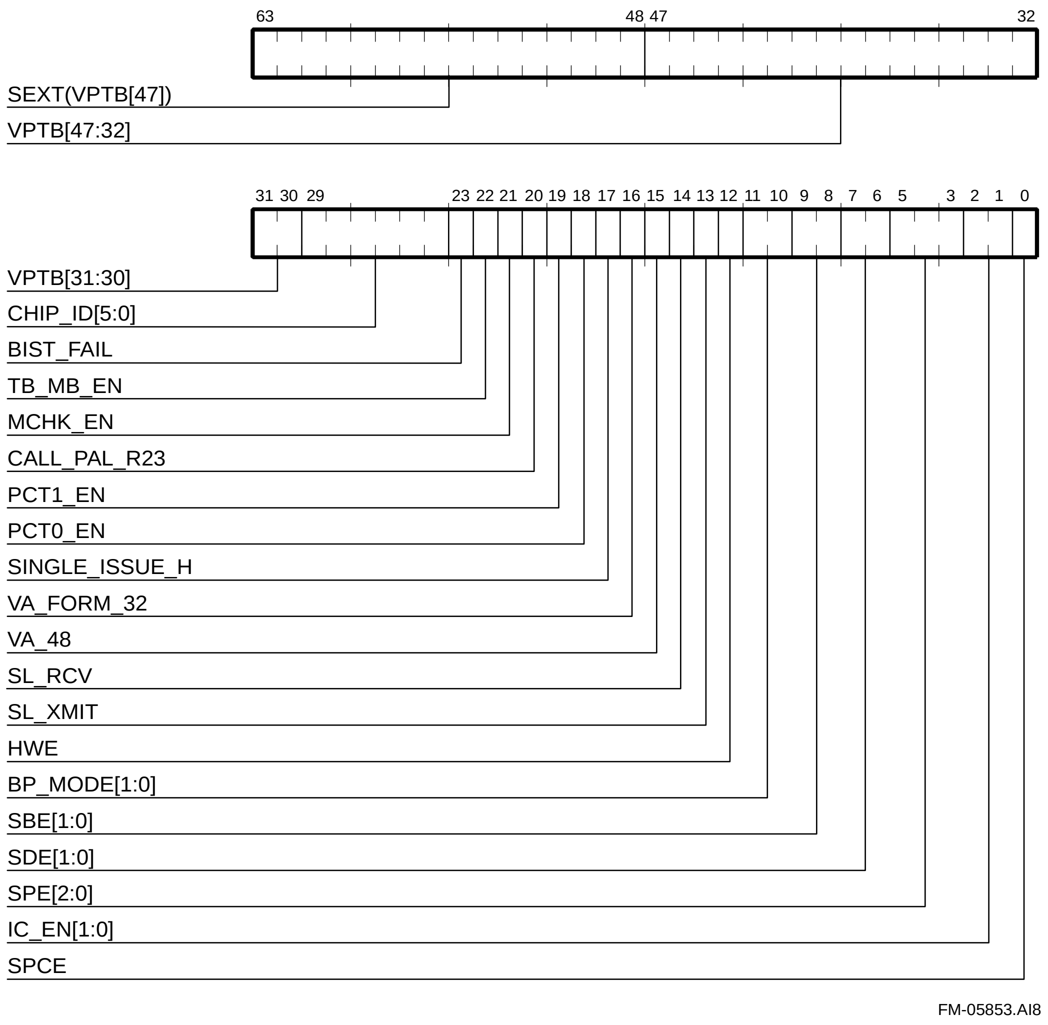


Table 4–10 describes the Ibox control register fields.

Table 4–10 Ibox Control Register Fields Description

Name	Extent	Type	Description
SEXT(VPTB[47])	[63:48]	RW,0	Sign extended VPTB[47].
VPTB[47:30]	[47:30]	RW,0	Virtual Page Table Base. See Section 4.1.5 for details.
CHIP_ID[5:0]	[29:24]	RO	This is a read-only field that supplies the revision ID number for the 21264 part. 21264 pass 1 ID is 000000_2. 21264 pass 2 ID is 000001_2. 21264 pass 2.2 ID is 000010_2. 21264 pass 2.3 ID is 000011_2.
BIST_FAIL	[23]	RO,0	Indicates the status of BIST (set = pass, clear = fail).

Table 4–10 Ibox Control Register Fields Description (Continued)

Name	Extent	Type	Description
TB_MB_EN	[22]	RW,0	When set, the hardware ensures that the virtual-mode loads in DTB and ITB fill flows that access the page table and the subsequent virtual mode load or store that is being retried are ‘ordered’ relative to another processor’s stores. This must be set for multiprocessor systems in which no MB instruction is present in the TB fill flow, unless there are other mechanisms present that ensure coherency.
MCHK_EN	[21]	RW,0	Machine check enable — set to enable machine checks.
CALL_PAL_R23	[20]	RW,0	CALL_PAL linkage register. If this bit is one, the CALL_PAL linkage register is R23; when zero, it is R27. Coordinate setting this bit with SDE[1:0] to ensure that the shadow register is used as the linkage register.
PCT1_EN	[19]	RW,0	Enable performance counter #1. If this bit is one, the performance counter will count if either the system (SPCE) or process (PPCE) performance counter enable is asserted.
PCT0_EN	[18]	RW,0	Enable performance counter #0. If this bit is one, the performance counter will count if EITHER the system (SPCE) or process (PPCE) performance counter enable is set.
SINGLE_ISSUE_H	[17]	RW,0	When set, this bit forces instructions to issue only from the bottom-most entries of the IQ and FQ.
VA_FORM_32	[16]	RW,0	This bit controls address formatting on a read of the IVA_FORM register.
VA_48	[15]	RW,0	This bit controls the format applied to effective virtual addresses by the IVA_FORM register and the Ibox virtual address sign extension checkers. When VA_48 is clear, 43-bit virtual address format is used, and when VA_48 is set, 48-bit virtual address format is used. The effect of this bit on the IVA_FORM register is identical to the effect of VA_CTL[VA_48] on the VA_FORM register. See Section 4.1.5. When VA_48 is set, the sign extension checkers generate an ACV if va[63:0] ≠ SEXT(va[47:0]). When VA_48 is clear, the sign extension checkers generate an ACV if va[63:0] ≠ SEXT(va[42:0]). This bit also affects DTB_DOUBLE Traps. If set, the DTB double miss traps vector to the DTB_DOUBLE_4 entry point. DTB_DOUBLE PALcode flow selection is not affected by VA_CTL[VA_48].
SL_RCV	[14]	RO	When in native mode, any transition on SL_RCV, driven from the **SromData_H** pin, results in a trap to the PALcode interrupt handler. When in PALmode, all interrupts are blocked. The interrupt routine then begins sampling SL_RCV under a software timing loop to input as much data as needed, using the chosen serial line protocol.
SL_XMIT	[13]	WO	When set, drives a value on **SromClk_H**.

Table 4–10 Ibox Control Register Fields Description (Continued)

Name	Extent	Type	Description
HWE	[12]	RW,0	If set, allow PALRES intructions to be executed in kernel mode. Note that modification of the ITB while in kernel mode/native mode may cause UNPREDICTABLE behavior.
BP_MODE[1:0]	[11:10]	RW,0	Branch Prediction Mode Selection. BP_MODE[1], if set, forces all branches to be predicted to fall through. If clear, the dynamic branch predictor is chosen. BP_MODE[0]. If set, the dynamic branch predictor chooses local history prediction. If clear, the dynamic branch predictor chooses local or global prediction based on the state of the chooser.
SBE[1:0]	[9:8]	RW,0	Stream Buffer Enable. The value in this bit field specifies the number of Istream buffer prefetches (besides the demand-fill) that are launched after an Icache miss. If the value is zero, only demand requests are launched.
SDE[1:0]	[7:6]	RW,0	PALshadow Register Enable. Enables access to the PALshadow registers. If SDE[1] is set, R4-R7 and R20-R23 are used as PALshadow registers. SDE[0] does not affect 21264 operation.
SPE[2:0]	[5:3]	RW,0	Super Page Mode Enable. Identical to the SPE bits in the Mbox M_CTL SPE[2:0]. See Section 4.3.9.
IC_EN[1:0]	[2:1]	RW,3	Icache Set Enable. At least one set must be enabled. The entire cache may be enabled by setting both bits. Zero, one, or two Icache sets can be enabled. This bit does not clear the Icache, but only disables fills to the affected set.
SPCE	[0]	RW,0	System Performance Counting Enable. Enables performance counting for the entire system if individual counters (PCTR0 or PCTR1) are enabled by setting PCT0_EN or PCT1_EN, respectively. Performance counting for individual processes can be enabled by setting PCTX[PPCE].

4.2.15 Ibox Status Register – I_STAT

The Ibox status register (I_STAT) is a read/write-1-to-clear register that contains Ibox status information.

Figure 4–22 shows the Ibox status register.

Figure 4–22 Ibox Status Register

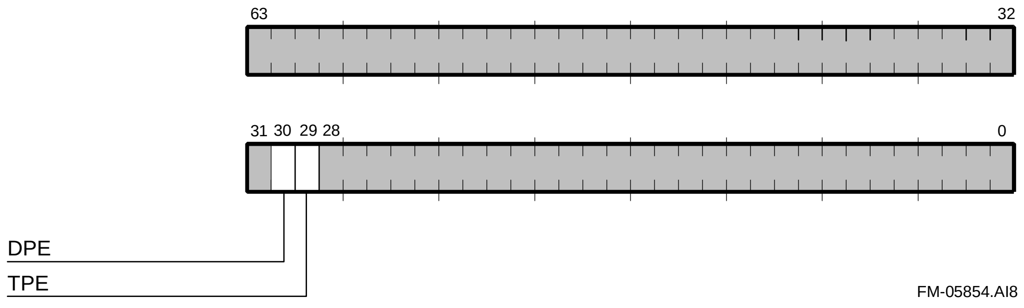


Table 4–11 describes the Ibox status register fields.

Table 4–11 Ibox Status Register Fields Description

Name	Extent	Type	Description
Reserved	[63:31]	RO	Reserved for COMPAQ.
DPE	[30]	W1C	Icache data parity error When set, this bit indicates that the Icache encountered a data parity error on instruction fetch.
TPE	[29]	W1C	Icache tag parity error When set, this bit indicates that the Icache encountered a tag parity error on instruction fetch.
Reserved	[28:0]	RO	Reserved for COMPAQ.

4.2.16 Icache Flush Register – IC_FLUSH

The Icache flush register (IC_FLUSH) is a pseudo register. Writing to this register invalidates all Icache blocks. The cache is flushed when the next HW_RET/STALL instruction is retired.

4.2.17 Icache Flush ASM Register – IC_FLUSH_ASM

The Icache flush ASM register (IC_FLUSH_ASM) is a pseudo register. Writing to this register invalidates all Icache blocks with their ASM bit clear.

4.2.18 Clear Virtual-to-Physical Map Register – CLR_MAP

The clear virtual-to-physical map register (CLR_MAP) is a pseudo register that, when written, results in the clearing of the current map of virtual to physical registers. This register must only be written after there are no register-borne dependencies present and there are no unretired instructions. See an example in the PALcode restrictions.

4.2.19 Sleep Mode Register – SLEEP

The sleep mode register (SLEEP) is a pseudo register that, when written, results in the PLL speed being reduced and the chip entering a low-power mode. This register must only be written after a sequence of code has been run which saves all necessary state to DRAM, flushes the caches, and unmasks certain interrupts so the chip can be woken up.

4.2.20 Process Context Register – PCTX

The process context register (PCTX) contains information associated with the context of a process. Any combination of the bit fields within this register may be written with a single HW_MTPR instruction. When bits [7:6] of the IPR index field of a HW_MTPR instruction contain the value 01_2, this register is selected. Bits [4:0] of the IPR index indicate which bit fields are to be written. Table 4–12 lists the correspondence between IPR index bits and register fields.

Table 4–12 IPR Index Bits and Register Fields

IPR Index Bit	Register Field
0	ASN
1	ASTER
2	ASTRR
3	PPCE
4	FPE

A HW_MFPR from this register returns the values in all of its component bit fields.

Figure 4–23 shows the process context register.

Figure 4–23 Process Context Register

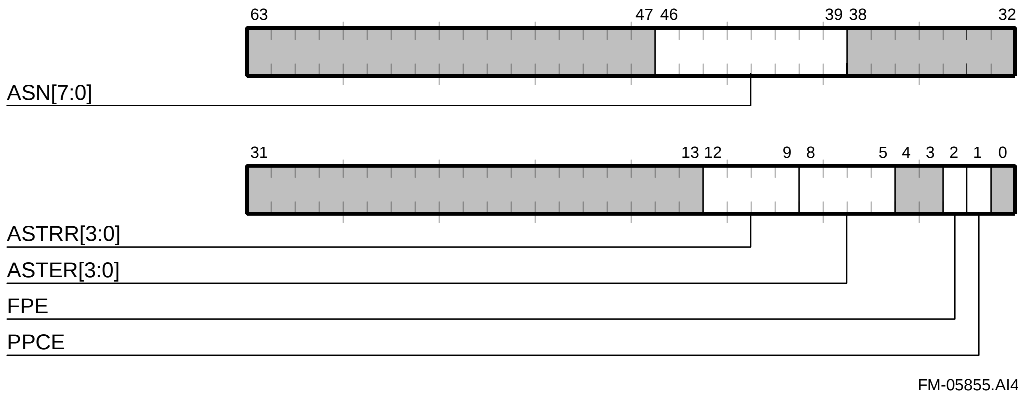


Table 4–13 describes the process context register fields.

Table 4–13 Process Context Register Fields Description

Name	Extent	Type	Description
Reserved	[63:47]	—	—
ASN[7:0]	[46:39]	RW	Address space number.
Reserved	[38:13]	—	—

Table 4–13 Process Context Register Fields Description (Continued)

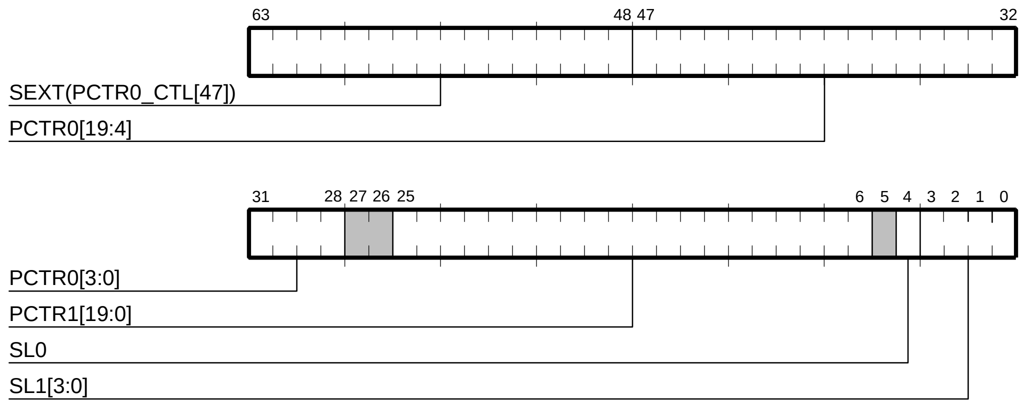
Name	Extent	Type	Description
ASTRR[3:0]	[12:9]	RW	AST request register—used to request AST interrupts in each of the four processor modes. To generate a particular AST interrupt, its corresponding bits in ASTRR and ASTER must be set, along with the ASTE bit in IER. Further, the value of the current mode bits in the PS register must be equal to or higher than the value of the mode associated with the AST request. The bit order with this field is: User Mode 12 Supervior Mode 11 Executive Mode 10 Kernel Mode 9
ASTER[3:0]	[8:5]	RW	AST enable register—used to individually enable each of the four AST interrupt requests. The bit order with this field is: User Mode 8 Supervisor Mode 7 Executive Mode 6 Kernel Mode 5
Reserved	[4:3]	—	—
FPE	[2]	RW,1	Floating-point enable—if clear, floating-point instructions generate FEN exceptions. This bit is set by hardware on reset.
PPCE	[1]	RW	Process performance counting enable. Enables performance counting for an individual process with counters PCTR0 or PCTR1, which are enabled by setting PCT0_EN or PCT1_EN, respectively. Performance counting for the entire system can be enabled by setting I_CTL[SPCE].
Reserved	[0]	—	—

4.2.21 Performance Counter Control Register – PCTR_CTL

The performance counter control register (PCTR_CTL) is a read-write register that controls the function of the performance counters.

Figure 4–24 shows the performance counter control register.

Figure 4–24 Performance Counter Control Register



FM-05856.AI8

Table 4–14 describes the performance counter control register fields.

Table 4–14 Performance Counter Control Register Fields Description

Name	Extent	Type	Description
SEXT(PCTR0_CTL[47])	[63:48]	RO	When read, this field is sign extended from PCTR_CTL[47]. Writes to this field are ignored.
PCTR0[19:0]	[47:28]	—	Performance counter 0. Mode is determined by PCTR_CTL[SL0] and operation is described in Table 4–15.
Reserved	[27:26]	RO	Reads to this field return zero. Writes to this field are ignored.
PCTR1[19:0]	[25:6]	—	Performance counter 1. PCTR1 must be enabled by I_CTL[PCT1_EN] and either I_CTL[SPCE] or PCTX[PPCE]. On overflow, an interrupt is triggered at ISUM[PC1] if enabled by IER_CM[PCEN1]. When enabled, PCTR1 is incremented at each cycle by the selected input.
Reserved	[5]	RO	Reads to this field return zero. Writes to this field are ignored.
SL0	[4]	—	SL0 input select 0. Selects counter PCTR0. 0: Cycles 1: Retired instructions See Table 4–15 for more information.

Table 4–14 Performance Counter Control Register Fields Description (Continued)

Name	Extent	Type	Description	
SL1[3:0]	[3:0]	—	SL1 input select 1. Selects counter PCTR1.	
			Bit Value	**Meaning**
			0000	Counter 1 counts cycles.
			0001	Counter 1 counts retired conditional branches.
			0010	Counter 1 counts retired branch mispredicts.
			0011	Counter 1 counts retired DTB single misses * 2.
			0100	Counter 1 counts retired DTB double double misses.
			0101	Counter 1 counts retired ITB misses.
			0110	Counter 1 counts retired unaligned traps.
			0111	Counter 1 counts replay traps.

Table 4–15 Performance Counter Control Register Input Select Field SL0

SL0[0] Bit Value	Meaning
0	Counts cycles.
1	Counts retired instructions. PCTR0 is incremented by up to 8 retired instructions per cycle when enabled via I_CTL[PCT0_EN] and either I_CTL[SPCE] or PCTX[PPCE]. On overflow, an interrupt is triggered at ISUM[PC0] if enabled via IER_CM[PCEN0]. The 21264 can retire up to 11 instructions per cycle, which exceeds PCTR0's maximum increment of 8 per cycle. However, in aggregate counting mode, no retires go uncounted because the 21264 cannot sustain 11 retires/cycle, and the 21264 corrects PCTR0 in subsequent cycles. A squashed instruction does not count as a retire.

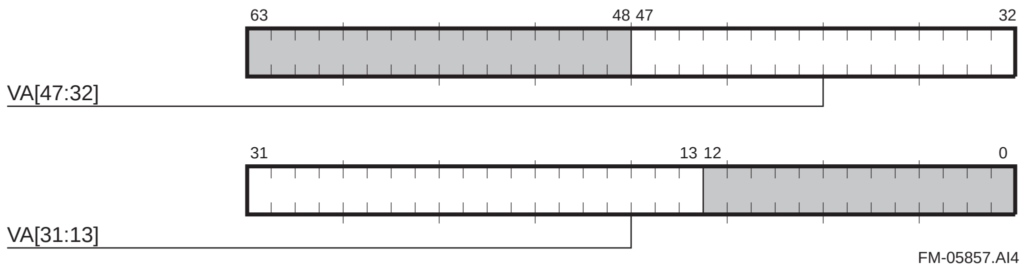
4.3 Mbox IPRs

This section describes the internal processor registers that control Mbox functions.

4.3.1 DTB Tag Array Write Registers 0 and 1 – DTB_TAG0, DTB_TAG1

The DTB tag array write registers 0 and 1 (DTB_TAG0 and DTB_TAG1) are write-only registers through which the two memory pipe DTB tag arrays are written. Write transactions to DTB_TAG0 and DTB_TAG1 writes data to registers outside the DTB arrays. When write transactions to the corresponding DTB_PTE registers are retired, the contents of both the DTB_TAG and DTB_PTE registers are written into their respective DTB arrays, at locations determined by the round-robin allocation algorithm. Figure 4–25 shows the DTB tag array write registers 0 and 1.

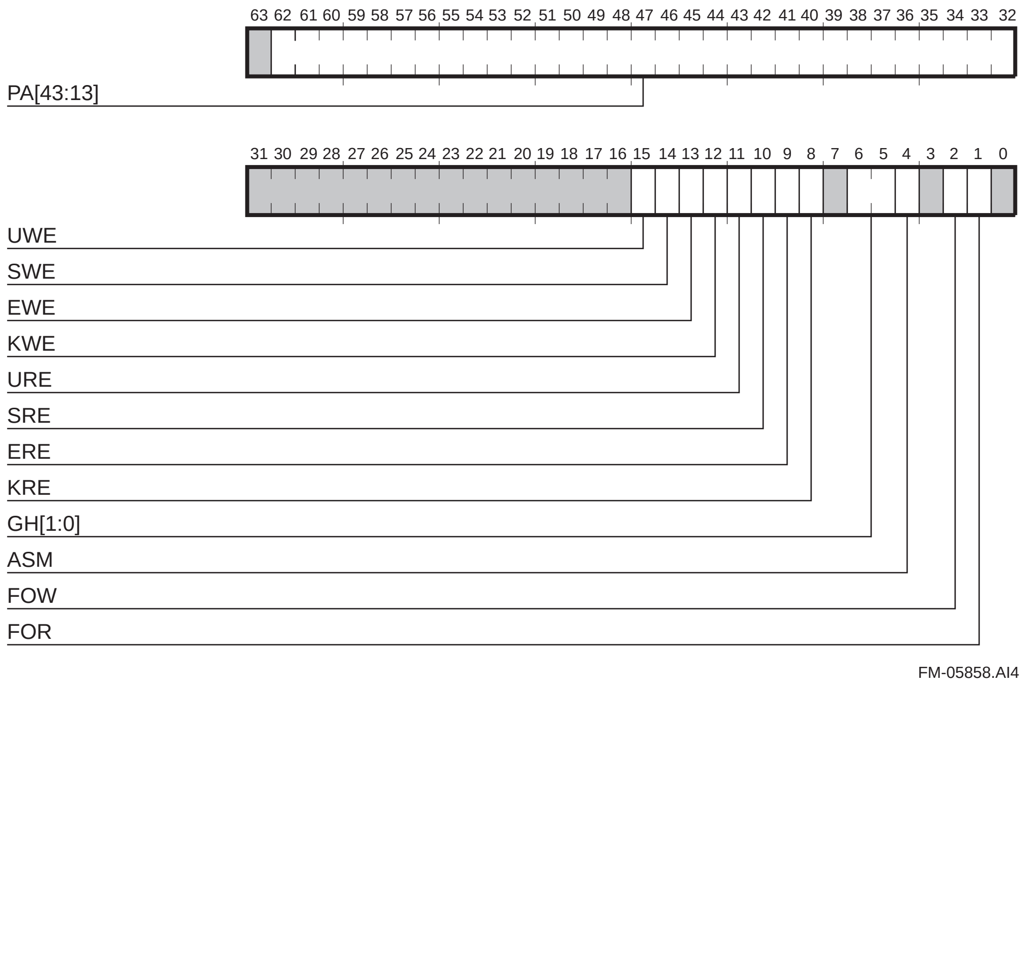
Figure 4–25 DTB Tag Array Write Registers 0 and 1



4.3.2 DTB PTE Array Write Registers 0 and 1 – DTB_PTE0, DTB_PTE1

The DTB PTE array write registers 0 and 1 (DTB_PTE0 and DTB_PTE1) are registers though which the DTB PTE arrays are written. The entries to be written are chosen by a round-robin allocation scheme. Write transactions to the DTB_PTE registers, when retired, result in both the DTB_TAG and DTB_PTE arrays being written. Figure 4–26 shows the DTB PTE array write registers 0 and 1.

Figure 4–26 DTB PTE Array Write Registers 0 and 1



4.3.3 DTB Alternate Processor Mode Register – DTB_ALTMODE

The DTB alternate processor mode register (DTB_ALTMODE) is a write-only register whose contents specify the alternate processor mode used by some HW_LD and HW_ST instructions. Figure 4–27 shows the DTB alternate processor mode register.

Figure 4–27 DTB Alternate Processor Mode Register

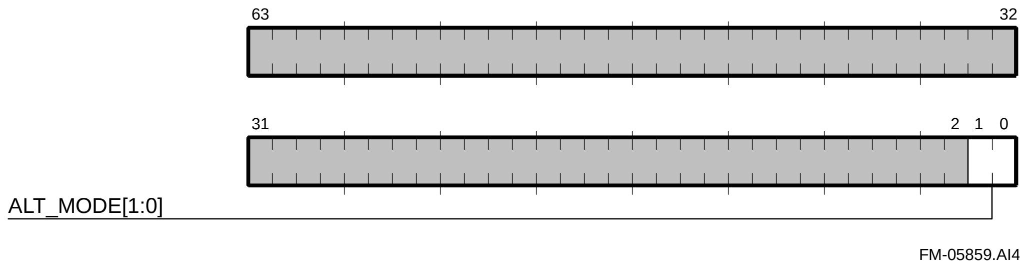


Table 4–16 describes the DTB_ALTMODE register fields.

Table 4–16 DTB Alternate Processor Mode Register Fields Description

Name	Extent	Type	Description
Reserved	[63:2]	—	—
ALT_MODE[1:0]	[1:0]	RW	Alt_Mode: ALT_MODE[1:0] Mode 00 Kernel 01 Executive 10 Supervisor 11 User

4.3.4 Dstream TB Invalidate All Process (ASM=0) Register – DTB_IAP

The Dstream translation buffer invalidate all (ASM=0) process register (DTB_IAP) is a write-only pseudo register. Write transactions to this register invalidate all DTB entries in which the address space match (ASM) bit is clear.

4.3.5 Dstream TB Invalidate All Register – DTB_IA

The Dstream translation buffer invalidate all register (DTB_IA) is a write-only pseudo register. Write transactions to this register invalidate all DTB entries and reset the DTB not-last-used pointer to its initial state.

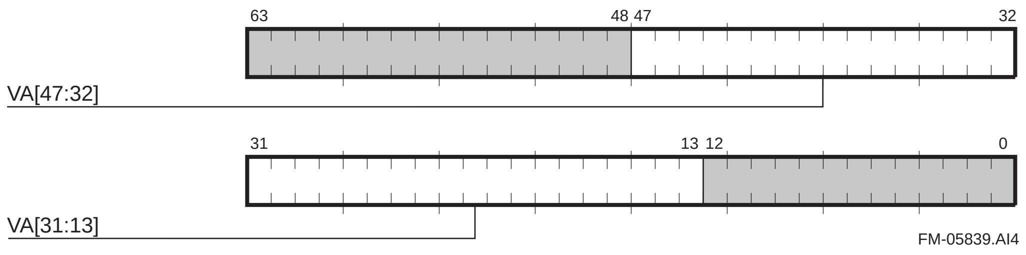
4.3.6 Dstream TB Invalidate Single Registers 0 and 1 – DTB_IS0,1

The Dstream translation buffer invalidate single registers (DTB_IS0 and DTB_IS1) are write-only pseudo registers through which software may invalidate a single entry in the DTB arrays. Writing a virtual page number to one of these registers invalidates any DTB entry in the corresponding memory pipeline which meets one of the following criteria:

- The DTB entry's virtual page number matches DTB_IS[47:13] and its ASN field matches DTB_ASN[63:56].
- The DTB entry's virtual page number matches DTB_IS[47:13] and its ASM bit is set.

Figure 4–28 shows the Dstream translation buffer invalidate single registers.

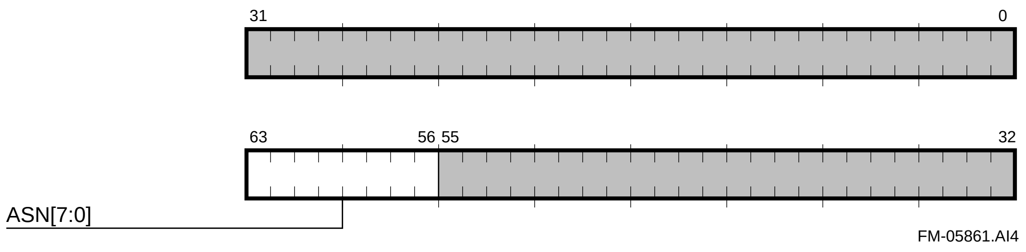
Figure 4–28 Dstream Translation Buffer Invalidate Single Registers



4.3.7 Dstream TB Address Space Number Registers 0 and 1 – DTB_ASN0,1

The Dstream translation buffer address space number registers (DTB_ASN0 and DTB_ASN1) are write-only registers that should be written with the address space number (ASN) of the current process. Figure 4–29 shows the Dstream translation buffer address space number registers 0 and 1.

Figure 4–29 Dstream Translation Buffer Address Space Number Registers 0 and 1



4.3.8 Memory Management Status Register – MM_STAT

The memory management status register (MM_STAT) is a read-only register. When a Dstream TB miss or fault occurs, information about the error is latched in MM_STAT. MM_STAT is not updated when a LD_VPTE gets a DTB miss instruction. Figure 4–30 shows the memory management status register.

Figure 4–30 Memory Management Status Register

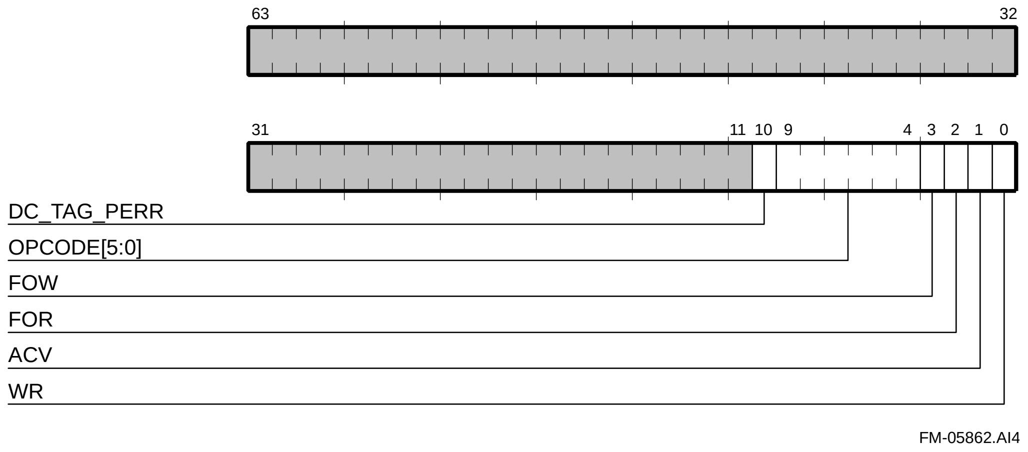


Table 4–17 describes the memory management status register fields.

Table 4–17 Memory Management Status Register Fields Description

Name	Extent	Type	Description
Reserved	[63:11]	—	—
DC_TAG_PERR	[10]	RO	This bit is set when a Dcache tag parity error occurred during the initial tag probe of a load or store instruction. The error created a synchronous fault to the D_FAULT PALcode entry point and is correctable. The virtual address associated with the error is available in the VA register.
OPCODE[5:0]	[9:4]	RO	Opcode of the instruction that caused the error. HW_LD is displayed as 3 and HW_ST is displayed as 7.
FOW	[3]	RO	This bit is set when a fault-on-write error occurs during a write transaction and PTE[FOW] was set.
FOR	[2]	RO	This bit is set when a fault-on-read error occurs during a read transaction and PTE[FOR] was set.
ACV	[1]	RO	This bit is set when an access violation occurs during a transaction. Access violations include a bad virtual address.
WR	[0]	RO	This bit is set when an error occurs during a write transaction.

Note: The Ra field of the instruction that triggered the error can be obtained from the Ibox EXC_SUM register.

4.3.9 Mbox Control Register – M_CTL

The Mbox control register (M_CTL) is a write-only register. Its contents are cleared by chip reset. Figure 4–31 shows the Mbox control register.

Figure 4–31 Mbox Control Register

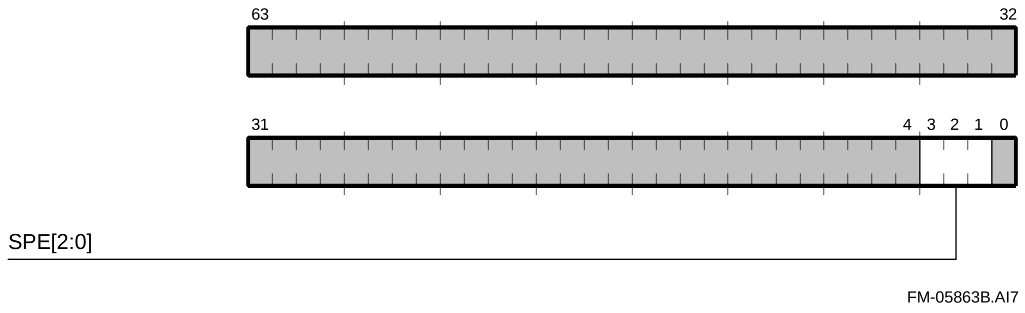


Table 4–18 describes the Mbox control register fields.

Table 4–18 Mbox Control Register Fields Description

Name	Extent	Type	Description
Reserved	[63:4]	—	—
SPE[2:0]	[3:1]	WO,0	Super page mode enables. SPE[2], when set, enables super page mapping when VA[47:46] = 2. In this mode, VA[43:13] are mapped directly to PA[43:13] and VA[45:44] are ignored. SPE[1], when set, enables super page mapping when VA[47:41] = $7E_{16}$. In this mode, VA[40:13] are mapped directly to PA[40:13] and PA[43:41] are copies of PA[40] (sign extension). SPE[0], when set, enables super page mapping when VA[47:30] = $3FFFE_{16}$. In this mode, VA[29:13] are mapped directly to PA[29:13] and PA[43:30] are cleared.
Reserved	[0]	—	—

Note: Super page accesses are only allowed in kernel mode. Non-kernel mode references to super pages result in access violations.

4.3.10 Dcache Control Register – DC_CTL

The Dcache control register (DC_CTL) is a write-only register that controls Dcache activity. The contents of DC_CTL are initialized by chip reset as indicated. Figure 4–32 shows the Dcache control register.

Figure 4–32 Dcache Control Register

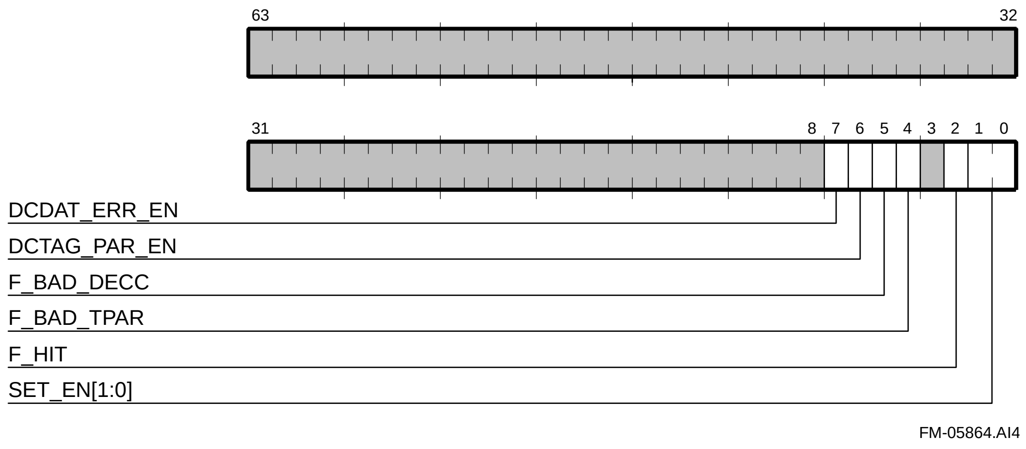


Table 4–19 describes the Dcache control register fields.

Table 4–19 Dcache Control Register Fields Description

Name	Extent	Type	Description
Reserved	[63:8]	—	—
DCDAT_ERR_EN	[7]	WO,0	Dcache data ECC and parity error enable.
DCTAG_PAR_EN	[6]	WO,0	Dcache tag parity enable.

Table 4–19 Dcache Control Register Fields Description (Continued)

Name	Extent	Type	Description
F_BAD_DECC	[5]	WO,0	Force Bad Data ECC. When set, ECC data is *not* written into the cache along with the block that is loaded by a fill or store. Writing data that is different from that already in the block will cause bad ECC to be present. Since the old ECC value will remain, the ECC will be *bad*.
F_BAD_TPAR	[4]	WO,0	Force Bad Tag Parity. When set, this bit causes bad tag parity to be put into the Dcache tag array during Dcache fill operations.
Reserved	[3]	—	—
F_HIT	[2]	WO,0	Force Hit. When set, this bit causes all memory space load and store instructions to hit in the Dcache, independent of the Dcache tag address compare. F_HIT does not force the status of the block to register as DIRTY (the tag status bits are still consulted), so stores may still generate offchip activity. In this mode, only one of the two sets may be enabled, and tag parity checking must be disabled (set DCTAG_PER_EN to zero).
SET_EN[1:0]	[1:0]	WO,3	Dcache Set Enable. At least one set must be enabled.

4.3.11 Dcache Status Register – DC_STAT

The Dcache status register (DC_STAT) is a read-write register. If a Dcache tag parity error or data ECC error occurs, information about the error is latched in this register. Figure 4–33 shows the Dcache status register.

Figure 4–33 Dcache Status Register

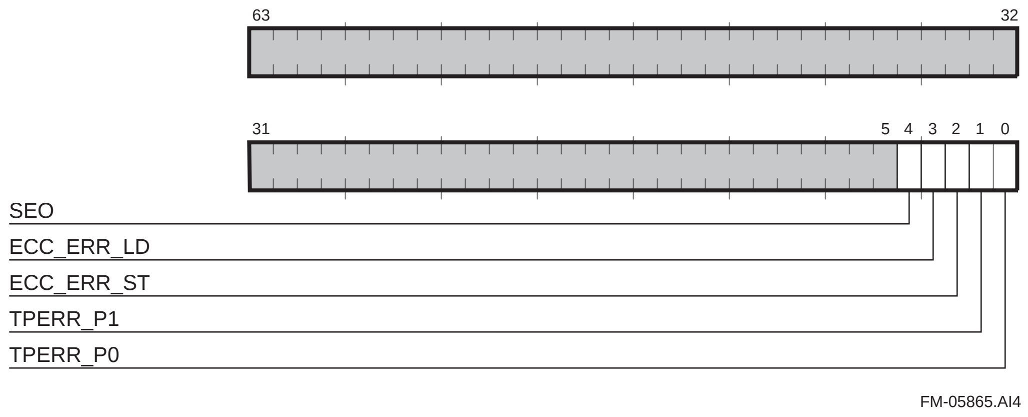


Table 4–20 describes the Dcache status register fields.

Table 4–20 Dcache Status Register Fields Description

Name	Extent	Type	Description
Reserved	[63:5]	—	—
SEO	[4]	W1C	Second error occured. When set, this bit indicates that a second Dcache store ECC error occurred within 6 cycles of the previous Dcache store ECC error.

Table 4–20 Dcache Status Register Fields Description

Name	Extent	Type	Description
ECC_ERR_LD	[3]	W1C	ECC error on load. When set, this bit indicates that a single-bit ECC error occurred while processing a load from the Dcache or any fill.
ECC_ERR_ST	[2]	W1C	ECC error on store. When set, this bit indicates that an ECC error occurred while processing a store.
TPERR_P1	[1]	W1C	Tag parity error — pipe 1. When set, this bit indicates that a Dcache tag probe from pipe 1 resulted in a tag parity error. The error is uncorrectable and results in a machine check.
TPERR_P0	[0]	W1C	Tag parity error — pipe 0. When set, this bit indicates that a Dcache tag probe from pipe 0 resulted in a tag parity error. The error is uncorrectable and results in a machine check.

4.4 Cbox CSRs and IPRs

This section describes the Cbox CSRs and IPRs.

The Cbox configuration registers are split into three shift register chains:

- The hardware allocates 367 bits for the WRITE_ONCE chain, of which the 21264 uses 303 bits. During hardware reset (after BiST), 367 bits are always shifted into the WRITE_ONCE chain from the SROM, MSB first, so that any unused bits are shifted out the end of the WRITE_ONCE chain.
- A 36-bit WRITE_MANY chain that is loaded using MTPR instructions to the Cbox data register. Six bits of information are shifted into the WRITE_MANY chain during each write transaction to the Cbox data register.
- A 60-bit Cbox ERROR_REG chain that is read by using MFFR instructions from the Cbox data register in combination with MTPR instructions to the Cbox shift register. Each write transaction to the Cbox shift register destructively shifts six bits of information out of the Cbox error register.

4.4.1 Cbox Data Register – C_DATA

Figure 4–34 shows the Cbox data register.

Figure 4–34 Cbox Data Register

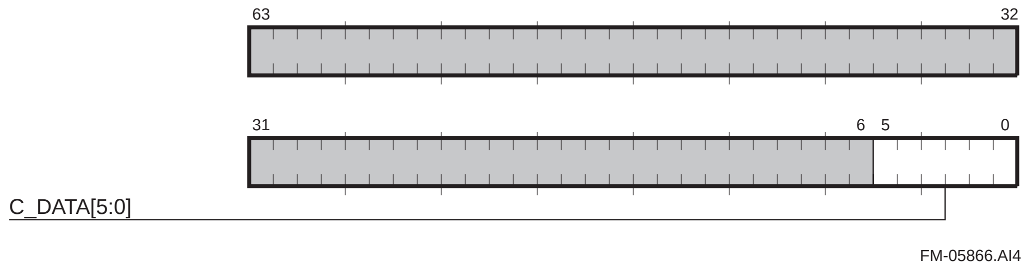


Table 4–21 describes the Cbox data register fields.

Table 4–21 Cbox Data Register Fields Description

Name	Extent	Type	Description
Reserved	[63:6]	—	—
C_DATA[5:0]	[5:0]	RW	Cbox data register. A HW_MTPR instruction to this register causes six bits of data to be placed into a serial shift register. When the HW_MTPR instruction is retired, the data is shifted into the Cbox. After the Cbox shift register has been accessed, performing a HW_MFPR instruction to this register will return six bits of data.

4.4.2 Cbox Shift Register – C_SHFT

Figure 4–35 shows the Cbox shift register.

Figure 4–35 Cbox Shift Register

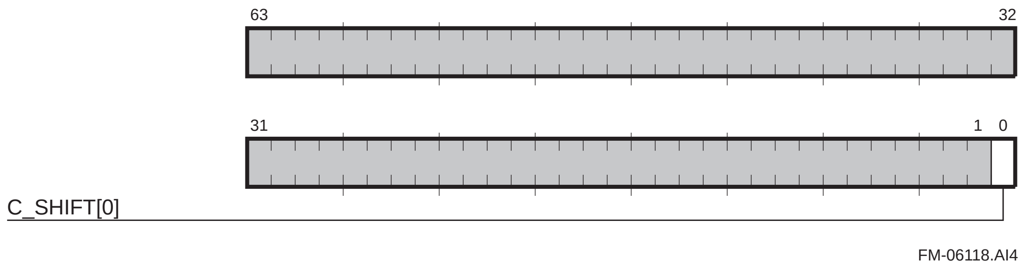


Table 4–22 describes the Cbox shift register fields.

Table 4–22 Cbox Shift Register Fields Description

Name	Extent	Type	Description
Reserved	[63:1]	—	—
C_SHIFT[0]	[0]	W1	Writing a 1 to this register bit causes six bits of Cbox IPR data to shift into the Cbox data register. Software can then use a HW_MFPR read operation to the Cbox data register to read the six bits of data.

4.4.3 Cbox WRITE_ONCE Chain Description

The WRITE_ONCE chain order is contained in Table 4–23. In the table:

- Many CSRs are duplicated for ease of hardware implementation. These CSRs are indicated in italics. They must be written with values that are identical to the values written to the original CSRs.
- Only a brief description of each CSR is given.
- The order of multibit vectors is [MSB:LSB], so the LSB is first bit in the Cbox chain.

Table 4–23 describes the Cbox WRITE_ONCE chain order from LSB to MSB.

Table 4–23 Cbox WRITE_ONCE Chain Order

Cbox WRITE_ONCE Chain	Description
32_BYTE_IO[0]	Enable 32_BYTE I/O mode.
SKEWED_FILL_MODE[0]	Asserted when Bcache is at 1.5X ratio.
SKEWED_FILL_MODE[0]	*Duplicate of prior bit.*
DCVIC_THRESHOLD[7:0]	Threshold of the number of Dcache victims that will accumulate before streamed write transactions to the Bcache are initiated. The Cbox can accumulate up to six victims for streamed Dcache processing. This register is programmed with the decoded value of the threshold count.
BC_CLEAN_VICTIM[0]	Enable clean victims to the system interface.
SYS_BUS_SIZE[1:0]	Size of SysAddOut and SysAddOut buses.
SYS_BUS_FORMAT[0]	Indicates system bus format.
SYS_CLK_RATIO[4:1]	Speed of system bus. Code Multiplier 0001 1.5X 0010 2.0X 0100 2.5X 1000 3.0X
DUP_TAG_ENABLE[0]	Enable duplicate tag mode in the 21264.
PRB_TAG_ONLY[0]	Enable probe-tag only mode in the 21264.
FAST_MODE_DISABLE[0]	When asserted, disables fast data movement mode.
BC_RDVICTIM[0]	Enables RdVictim mode on the pins.
BC_CLEAN_VICTIM[0]	*Duplicate CSR.*
RDVIC_ACK_INHIBIT	Enable inhibition of incrementing acknowledge counter for RdVic commands.
SYSBUS_MB_ENABLE	Enable MB commands offchip.
SYSBUS_ACK_LIMIT[0:4]	Sysbus acknowledge limit CSR.
SYSBUS_VIC_LIMIT[0:2]	Limit for victims.
BC_CLEAN_VICTIM[0]	*Duplicate CSR.*
BC_WR_WR_BUBBLE[0]	Write to write GCLK bubble.
BC_RD_WR_BUBBLES[0:5]	Read to write GCLK bubbles for the Bcache interface.
BC_RD_RD_BUBBLE[0]	Read to read GCLK bubble for banked Bcaches.
BC_SJ_BANK_ENABLE	Enable bank mode for Bcache.
BC_WR_RD_BUBBLES[0:3]	Write to read GCLK bubbles.
DUP_TAG_ENABLE	*Duplicate CSR.*
SKEWED_FILL_MODE	*Duplicate CSR.*
BC_RDVICTIM	*Duplicate CSR.*

Table 4–23 Cbox WRITE_ONCE Chain Order (Continued)

Cbox WRITE_ONCE Chain	Description
SKEWED_FILL_MODE	*Duplicate CSR.*
BC_RDVICTIM	*Duplicate CSR.*
BC_CLEAN_VICTIM	*Duplicate CSR.*
DUP_TAG_MODE	*Duplicate CSR.*
SKEWED_FILL_MODE	*Duplicate CSR.*
ENABLE_PROBE_CHECK	Enable error checking during probe processing.
SPEC_READ_ENABLE[0]	Enable speculative references to the system port.
SKEWED_FILL_MODE	*Duplicate CSR.*
SKEWED_FILL_MODE	*Duplicate CSR.*
MBOX_BC_PRB_STALL	Must be asserted when BC_RATIO = 4.0X, 5.0X, 6.0X, 7.0X, or 8.0X.
BC_LAT_DATA_PATTERN[0:31]	Bcache data latency pattern.
BC_LAT_TAG_PATTERN[0:23]	Bcache tag latency pattern.
BC_RDVICTIM	*Duplicate CSR.*
ENABLE_STC_COMMAND[0]	Enable STx_C instructions to the pins.
BC_LATE_WRITE_NUM[0:2]	Number of Bcache clocks to delay the data for Bcache write commands.
BC_CPU_LATE_WRITE_NUM[0:1]	Number of GCLK cycles to delay the Bcache clock/data from index.
BC_BURST_MODE_ENABLE[0]	Burst mode enable signal.
BC_PENTIUM_MODE[0]	Enable Pentium mode RAM behavior.
SKEWED_FILL_MODE	*Duplicate CSR.*
BC_FRM_CLK[0]	Force all Bcache transactions to start on rising edges of the A phase of a GCLK.
BC_CLK_DELAY[0:1]	Delay of Bcache clock for 0,0,1,2 GCLK phases.
BC_DDMR_ENABLE[0]	Enables the rising edge of the Bcache forwarded clock (always enabled).
BC_DDMF_ENABLE[0]	Enable the falling edge of the Bcache forwarded clock (always enabled).
BC_LATE_WRITE_UPPER[0]	Asserted when (BC_LATE_WRITE_NUM > 3) or ((BC_LATE_WRITE_NUM = 3) and (BC_CPU_LATE_WRITE_NUM > 1)).
BC_TAG_DDM_FALL_EN[0]	Enables the update of the 21264 Bcache tag outputs based on the falling edge of the forwarded clock.
BC_TAG_DDM_RISE_EN[0]	Enables the update of the 21264 Bcache tag outputs based on the rising edge of the forwarded clock.
BC_CLKFWD_ENABLE[0]	Enable clock forwarding on the Bcache interface.
BC_RCV_MUX_CNT_PRESET[0:1]	Initial value for the Bcache clock forwarding unload pointer FIFO.

Table 4–23 Cbox WRITE_ONCE Chain Order (Continued)

Cbox WRITE_ONCE Chain	Description
BC_LATE_WRITE_UPPER[0]	*Duplicate CSR.*
SYS_DDM_FALL_EN[0]	Enables the update of the 21264 system outputs based on the falling edge of the system forwarded clock.
SYS_DDM_RISE_EN[0]	Enables the update of the 21264 system outputs based on the rising edge of the system forwarded clock.
SYS_CLKFWD_ENABLE[0]	Enables clock forwarding on the system interface.
SYS_RCV_MUX_CNT_PRESET[0:1]	Initial value for the system clock forwarding unload pointer FIFO.
SYS_CLK_DELAY[0:1]	Delay of 0 to 2 phases between the forwarded clock out and address/data.
SYS_DDMR_ENABLE[0]	Enables the rising edge of the system forwarded clock (always enabled).
SYS_DDMF_ENABLE[0]	Enables the falling edge of the system forwarded clock (always enabled).
BC_DDM_FALL_EN[0]	Enables update of data/address on the rising edge of the system forwarded clock.
BC_DDM_RISE_EN[0]	Enables the update of data/address on the falling edge of the system forwarded clock.
BC_CLKFWD_ENABLE	*Duplicate CSR.*
BC_RCV_MUX_CNT_PRESET[0:1]	*Duplicate CSR.*
BC_CLK_DELAY[0:1]	*Duplicate CSR.*
BC_DDMR_ENABLE	*Duplicate CSR.*
BC_DDMF_ENABLE	*Duplicate CSR.*
SYS_DDM_FALL_EN	*Duplicate CSR.*
SYS_DDM_RISE_EN	*Duplicate CSR.*
SYS_CLKFWD_ENABLE	*Duplicate CSR.*
SYS_RCV_MUX_CNT_PRESET[0:1]	*Duplicate CSR.*
SYS_CLK_DELAY[0:1]	*Duplicate CSR.*
SYS_DDMR_ENABLE	*Duplicate CSR.*
SYS_DDMF_ENABLE	*Duplicate CSR.*
BC_DDM_FALL_EN	*Duplicate CSR.*
BC_DDM_RISE_EN	*Duplicate CSR.*
BC_CLKFWD_ENABLE	*Duplicate CSR.*
BC_RCV_MUX_CNT_PRESET[0:1]	*Duplicate CSR.*
SYS_DDM_FALL_EN	*Duplicate CSR.*
SYS_DDM_RISE_EN	*Duplicate CSR.*
SYS_CLKFWD_ENABLE	*Duplicate CSR.*
SYS_RCV_MUX_CNT_PRESET[0:1]	*Duplicate CSR.*

Table 4–23 Cbox WRITE_ONCE Chain Order (Continued)

Cbox WRITE_ONCE Chain	Description
SYS_CLK_DELAY[0:1]	*Duplicate CSR.*
SYS_DDMR_ENABLE	*Duplicate CSR.*
SYS_DDMF_ENABLE	*Duplicate CSR.*
BC_DDM_FALL_EN	*Duplicate CSR.*
BC_DDM_RISE_EN	*Duplicate CSR.*
BC_CLKFWD_ENABLE	*Duplicate CSR.*
BC_RCV_MUX_CNT_PRESET[0:1]	*Duplicate CSR.*
BC_CLK_DELAY[0:1]	*Duplicate CSR.*
BC_DDMR_ENABLE	*Duplicate CSR.*
BC_DDMF_ENABLE	*Duplicate CSR.*
SYS_DDM_FALL_EN	*Duplicate CSR.*
SYS_DDM_RISE_EN	*Duplicate CSR.*
SYS_CLKFWD_ENABLE	*Duplicate CSR.*
SYS_RCV_MUX_CNT_PRESET[0:1]	*Duplicate CSR.*
SYS_CLK_DELAY[1:0]	*Duplicate CSR.*
SYS_DDMR_ENABLE	*Duplicate CSR.*
SYS_DDMF_ENABLE	*Duplicate CSR.*
BC_DDM_FALL_EN	*Duplicate CSR.*
BC_DDM_RISE_EN	*Duplicate CSR.*
BC_CLKFWD_ENABLE	*Duplicate CSR.*
BC_RCV_MUX_CNT_PRESET[1:0]	*Duplicate CSR.*
SYS_CLK_DELAY[0:1]	*Duplicate CSR.*
SYS_DDMR_ENABLE	*Duplicate CSR.*
SYS_DDMF_ENABLE	*Duplicate CSR.*
SYS_DDM_FALL_EN	*Duplicate CSR.*
SYS_DDM_RISE_EN	*Duplicate CSR.*
SYS_CLKFWD_ENABLE	*Duplicate CSR.*
SYS_RCV_MUX_CNT_PRESET[0:1]	*Duplicate CSR.*
CFR_GCLK_DELAY[0:3]	Number of GCLK cycles to delay internal ClkFwdRst.
CFR_EV6CLK_DELAY[0:2]	Number of EV6Clk_*x* cycles to delay internal ClkFwdRst.
CFR_FRMCLK_DELAY[0:1]	Number of FrameClk_*x* cycles to delay internal ClkFwdRst.
BC_LATE_WRITE_NUM[0:2]	*Duplicate CSR.*
BC_CPU_LATE_WRITE_NUM[1:0]	*Duplicate CSR.*
JITTER_CMD[0]	Add one GCLK cycle to the SYSDC write path.

Table 4–23 Cbox WRITE_ONCE Chain Order (Continued)

Cbox WRITE_ONCE Chain	Description
FAST_MODE_DISABLE[0]	*Duplicate CSR.*
SYSDC_DELAY[3:0]	Number of GCLK cycles to delay SysDc fill commands before action by the Cbox.
DATA_VALID_DLY[1:0]	Number of Bcache clock cycles to delay signal SysDataInValid before sample by the Cbox.
BC_DDM_FALL_EN	*Duplicate CSR.*
BC_DDM_RISE_EN	*Duplicate CSR.*
BC_CPU_CLK_DELAY[0:1]	Delay of Bcache clock for 0, 1, 2, 3 GCLK cycles.
BC_FDBK_EN[0:7]	CSR to program the Bcache forwarded clock shift register feed-back points.
BC_CLK_LD_VECTOR[0:15]	CSR to program the Bcache forwarded clock shift register load values.
BC_BPHASE_LD_VECTOR[0:3]	CSR to program the Bcache forwarded clock b-phase enables.
SYS_DDM_FALL_EN	*Duplicate CSR.*
SYS_DDM_RISE_EN	*Duplicate CSR.*
SYS_CPU_CLK_DELAY[0:1]	Delay of 0..3 GCLK cycles between the forwarded clock out and address/data.
SYS_FDBK_EN[0:7]	CSR to program the system forwarded clock shift register feed-back points.
SYS_CLK_LD_VECTOR[0:15]	CSR to program the system forwarded clock shift register load val-ues.
SYS_BPHASE_LD_VECTOR[0:3]	CSR to program the system forwarded clock b-phase enables.
SYS_FRAME_LD_VECTOR[0:4]	CSR to program the ratio between frame clock and system for-warded clock.

4.4.4 Cbox WRITE_MANY Chain Description

The WRITE_MANY chain order is contained in Table 4–24. Note the following:

- Many CSRs are duplicated for ease of hardware implementation. These CSR names are indicated in italics and have two leading asterisks.
- Only a brief description of each CSR is given.
- The order of multibit vectors is [MSB:LSB], so the LSB is first bit in the Cbox chain.

Table 4–24 describes the Cbox WRITE_MANY chain order from LSB to MSB.

Table 4–24 Cbox WRITE_MANY Chain Order

Cbox WRITE_MANY Chain	Description
BC_ENABLE[0]	Enable the Bcache
INIT_MODE[0]	Enable initialize mode
BC_SIZE[3:0]	Bcache size
BC_ENABLE	*Duplicate CSR*
BC_ENABLE	*Duplicate CSR*
BC_SIZE[0:3]	*Duplicate CSR*
BC_ENABLE[1]	*Duplicate CSR*
BC_ENABLE[1]	*Duplicate CSR*
BC_ENABLE[1]	*Duplicate CSR*
INVAL_TO_DIRTY_ENABLE[1]	WH64 acknowledges
ENABLE_EVICT	Enable issue evict
BC_ENABLE	*Duplicate CSR*
INVAL_TO_DIRTY_ENABLE[0]	WH64 acknowledges
BC_ENABLE	*Duplicate CSR*
BC_ENABLE	*Duplicate CSR*
BC_ENABLE	*Duplicate CSR*
SET_DIRTY_ENABLE[0]	SetDirty acknowledge programming
INVAL_TO_DIRTY_ENABLE[0]	*Duplicate CSR*
SET_DIRTY_ENABLE[2:1]	SetDirty acknowledge programming
BC_BANK_ENABLE[0]	Enable bank mode for Bcache
BC_SIZE[0:3]	*Duplicate CSR*
INIT_MODE	*Duplicate CSR*
BC_WRT_STS[0:3]	Write status for Bcache in initialize-mode (Valid, Dirty, Shared, Parity)

[1] MBZ during initialization mode.

Figure 4–36 shows an example of PALcode used to write to the WRITE_MANY chain.

Figure 4–36 WRITE_MANY Chain Write Transaction Example

```
;
; Initialize the Bcache configuration in the Cbox
;
;       BC_ENABLE = 1
;       INIT_MODE = 0
;       BC_SIZE = 0xF
;       INVALID_TO_DIRTY_ENABLE = 3
;         ENABLE_EVICT = 1
;         SET_DIRTY_ENABLE = 6
;         BC_BANK_ENABLE = 1
;         BC_WRT_STS = 0
;
; The value for the write_many chain is based on Table 4–24.
;
; The value is sampled from MSB, 6 bits at a time, as it is written
; to EV6__DATA. Therefore, before the value can be shifted in, it must be
; inverted on a by 6 basis. The code then writes out 6 bits at a time,
; shifting right by 6 after each write.
;
; So the following transformation is done on the write_many value:
;
;         [35:30]||[29:24]||[23:18]||[17:12]||[11:06]||[05:00] =>
;         [05:00]||[11:06]||[17:12]||[23:18]||[29:24]||[35:30]
;
;         WRITE_MANY chain = 0x07FBFFFFD
;         value to be shifted in  = 0xF7FFEFFC1
;
; Before the chain can be written, I_CTL[SBE] must be disabled,
; and the code must be forced into the Icache.
;
          ALIGN_CACHE_BLOCK <^x47FF041F>; align with nops

          mb                                          ; wait for MEM-OP’s to complete
          lda     r0, ^x0086(r31)                     ; load I_CTL.....
          hw_mtpr    r0, EV6__I_CTL                   ; .....SDE=2, IC_EN=3, SBE=0
          br      r0, .                               ; create dest address

          addq    r0, #17, r0                         ; finish computing dest address
          hw_mtpr r31, EV6__IC_FLUSH                  ; flush the Icache
          bne     r31, .                              ; separate retires
          hw_jmp_stall (r0)                           ; force flush

          ALIGN_CACHE_BLOCK <^x47FF041F>              ; align with nops

bc_config:
          mb                                          ; pull this block in Icache
          lda         r1, ^xFFC1(r31)                 ; data[15:00] = 0xFFC1
          ldah        r0, ^x7FFE(r31)                 ; data[31:16] = 0x7FFE
          zap         r1, #^x0c, r1                   ; clear out bits [31:16]

          bis         r1, r0, r1                      ; or in bits [31:16]
          addq        r31, #6, r0                     ; shift in 6 x 6 bits
bc_config_shift_in:
          hw_mtpr     r1, EV6__DATA                   ; shift in 6 bits
          subq        r0, #1, r0                      ; decrement R0

          beq         r0, bc_config_done              ; done if R0 is zero
```

```
        srl      r1, #6, r1                  ; align next 6 bits
        br       r31, bc_config_shift_in     ; continue shifting
bc_config_done:
        hw_mtpr  r31, <EV6__MM_STAT ! 64>    ; wait until last shift

        beq      r31, bc_config_end          ; predicts fall thru
        br       r31, .-4                    ; predict infinite loop
        bis      r31, r31, r31               ; nop
        bis      r31, r31, r31               ; nop

bc_config_end:
```

4.4.5 Cbox Read Register (IPR) Description

The Cbox read register is read 6 bits at a time. Table 4–25 shows the ordering from LSB to MSB.

Table 4–25 Cbox Read IPR Fields Description

Name	Description
C_SYNDROME_1[7:0]	Syndrome for upper QW in OW of victim that was scrubbed.
C_SYNDROME_0[7:0]	Syndrome for lower QW in OW of victim that was scrubbed.

Table 4–25 Cbox Read IPR Fields Description (Continued)

Name	Description	
C_STAT[4:0]	**Bits**	**Error Status**
	0 0 0 0 0	Either no error, or error on a speculative load, or a Bcache victim read due to a Dcache/Bcache miss
	0 0 0 0 1	BC_PERR (Bcache tag parity error)
	0 0 0 1 0	DC_PERR (duplicate tag parity error)
	0 0 0 1 1	DSTREAM_MEM_ERR
	0 0 1 0 0	DSTREAM_BC_ERR
	0 0 1 0 1	DSTREAM_DC_ERR
	0 0 1 1 X	PROBE_BC_ERR
	0 1 0 0 0	Reserved
	0 1 0 0 1	Reserved
	0 1 0 1 0	Reserved
	0 1 0 1 1	ISTREAM_MEM_ERR
	0 1 1 0 0	ISTREAM_BC_ERR
	0 1 1 0 1	Reserved
	1 XXXX	DOUBLE_BIT_ERROR
C_STS[3:0]	If C_STAT equals *xxx*_MEM_ERR or *xxx*_BC_ERR, then C_STS contains the status of the block as follows; otherwise, the value of C_STS is X:	
	Bit Value	**Status of Block**
	7:4	Reserved
	3	Parity
	2	Valid
	1	Dirty
	0	Shared
C_ADDR[6:42]	Address of last reported ECC or parity error. If C_STAT value is DSTREAM_DC_ERR, only bits 6:19 are valid.	

5 Privileged Architecture Library Code

This chapter describes the 21264 privileged architecture library code (PALcode). The chapter is organized as follows:

- PALcode description
- PALmode environment
- Required PALcode function codes
- Opcodes reserved for PALcode
- Internal processor register access mechanisms
- PALshadow registers
- PALcode emulation of FPCR
- PALcode entry points

5.1 PALcode Description

PALcode is macrocode that provides an architecturally-defined, operating-system-specific programming interface that is common across all Alpha microprocessors. The actual implementation of PALcode differs for each operating system. PALcode runs with privileges enabled, instruction stream (Istream) mapping disabled, and interrupts disabled. PALcode has privilege to use five special opcodes that allow functions such as physical data stream (Dstream) references and internal processor register (IPR) manipulation.

PALcode can be invoked by the following events:

- Reset
- System hardware exceptions (MCHK, ARITH)
- Memory-management exceptions
- Interrupts
- CALL_PAL instructions

PALcode has characteristics that make it appear to be a combination of microcode, ROM BIOS, and system service routines, though the analogy to any of these other items is not exact. PALcode exists for several major reasons:

- There are some necessary support functions that are too complex to implement directly in a processor chip's hardware, but that cannot be handled by a normal operating system software routine. Routines to fill the translation buffer (TB), acknowledge interrupts, and dispatch exceptions are some examples. In some architectures, these functions are handled by microcode, but the Alpha architecture is careful not to mandate the use of microcode so as to allow reasonable chip implementations.

- There are functions that must run atomically, yet involve long sequences of instructions that may need complete access to all of the underlying computer hardware. An example of this is the sequence that returns from an exception or interrupt.
- There are some instructions that are necessary for backward compatibility or ease of programming; however, these are not used often enough to dedicate them to hardware, or are so complex that they would jeopardize the overall performance of the computer. For example, an instruction that does a VAX style interlocked memory access might be familiar to someone used to programming on a CISC machine, but is not included in the Alpha architecture. Another example is the emulation of an instruction that has no direct hardware support in a particular chip implementation.

In each of these cases, PALcode routines are used to provide the function. The routines are nothing more than programs invoked at specified times, and read in as Istream code in the same way that all other Alpha code is read. Once invoked, however, PALcode runs in a special mode called PALmode.

5.2 PALmode Environment

PALcode runs in a special environment called PALmode, defined as follows:

- Istream memory mapping is disabled. Because the PALcode is used to implement translation buffer fill routines, Istream mapping clearly cannot be enabled. Dstream mapping is still enabled.
- The program has privileged access to all of the computer hardware. Most of the functions handled by PALcode are privileged and need control of the lowest levels of the system.
- Interrupts are disabled. If a long sequence of instructions need to be executed atomically, interrupts cannot be allowed.

An important aspect of PALcode is that it uses normal Alpha instructions for most of its operations; that is, the same instruction set that nonprivileged Alpha programmers use. There are a few extra instructions that are only available in PALmode, and will cause a dispatch to the OPCDEC PALcode entry point if attempted while not in PALmode. The Alpha architecture allows some flexibility in what these special PALmode instructions do. In the 21264, the special PALmode-only instructions perform the following functions:

- Read or write internal processor registers (HW_MFPR, HW_MTPR)
- Perform memory load or store operations without invoking the normal memory-management routines (HW_LD, HW_ST)
- Return from an exception or interrupt (HW_RET)

When executing in PALmode, there are certain restrictions for using the privileged instructions because PALmode gives the programmer complete access to many of the internal details of the 21264. Refer to Section 5.4 for information on these special PALmode instructions.

Caution: It is possible to cause unintended side effects by writing what appears to be perfectly acceptable PALcode. As such, PALcode is not something that many users will want to change.

5.3 Required PALcode Function Codes

Table 5–1 lists opcodes required for all Alpha implementations. The notation used is oo.ffff, where oo is the hexadecimal 6-bit opcode and ffff is the hexadecimal 26-bit function code.

Table 5–1 Required PALcode Function Codes

Mnemonic	Type	Function Code
DRAINA	Privileged	00.0002
HALT	Privileged	00.0000
IMB	Unprivileged	00.0086

5.4 Opcodes Reserved for PALcode

Table 5–2 lists the opcodes reserved by the Alpha architecture for implementation-specific use. These opcodes are privileged and are only available in PALmode.

Table 5–2 Opcodes Reserved for PALcode

Mnemonic	Opcode	Architecture Mnemonic	Function
HW_LD	1B	PAL1B	Dstream load instruction
HW_ST	1F	PAL1F	Dstream store instruction
HW_RET	1E	PAL1E	Return from PALcode routine
HW_MFPR	19	PAL19	Copies the value of an IPR into an integer GPR
HW_MTPR	1D	PAL1D	Writes the value of an integer GPR into an IPR

These instructions generally produce an OPCDEC exception if executed while the processor is not in PALmode. If I_CTL[HWE] is set, these instructions can also be executed in kernel mode. Software that uses these instructions must adhere to the PALcode restrictions listed in this section.

5.4.1 HW_LD Instruction

PALcode uses the HW_LD instruction to access memory outside the realm of normal Alpha memory management and to perform special Dstream load transactions. Data alignment traps are disabled for the HW_LD instruction.

Figure 5–1 shows the HW_LD instruction format.

Figure 5–1 HW_LD Instruction Format

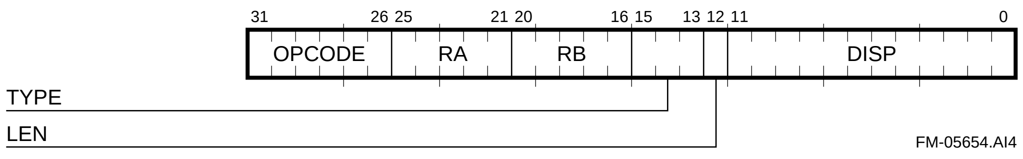


Table 5–3 describes the HW_LD instruction fields.

Table 5–3 HW_LD Instruction Fields Descriptions

Extent	Mnemonic	Value	Description
[31:26]	OPCODE	$1B_{16}$	The opcode value.
[25:21]	RA	—	Destination register number.
[20:16]	RB	—	Base register for memory address.
[15:13]	TYPE	000_2	Physical — The effective address for the HW_LD instruction is physical.
		001_2	Physical/Lock — The effective address for the HW_LD instruction is physical. It is the load lock version of the HW_LD instruction.
		010_2	Virtual/VPTE — Flags a virtual PTE fetch (LD_VPTE). Used by trap logic to distinguish a single TB miss from a double TB miss. Kernel mode access checks are performed.
		100_2	Virtual — The effective address for the HW_LD instruction is virtual.
		101_2	Virtual/WrChk — The effective address for the HW_LD instruction is virtual. Access checks for fault-on-read (FOR), fault-on-write (FOW), read and write protection.
		110_2	Virtual/Alt — The effective address for the HW_LD instruction is virtual. Access checks use DTB_ALT_MODE IPR.
		111_2	Virtual/WrChk/Alt — The effective address for the HW_LD instruction is virtual. Access checks for FOR, FOW, read and write protection. Access checks use DTB_ ALT_MODE IPR.
[12]	LEN	0	Access length is longword.
		1	Access length is quadword.
[11:0]	DISP	—	Holds a 12-bit signed byte displacement.

5.4.2 HW_ST Instruction

PALcode uses the HW_ST instruction to access memory outside the realm of normal Alpha memory management and to do special forms of Dstream store instructions. Data alignment traps are inhibited for HW_ST instructions. Figure 5–2 shows the HW_ST instruction format.

Figure 5–2 HW_ST Instruction Format

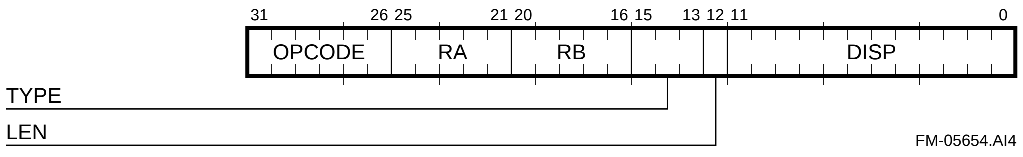


Table 5–4 describes the HW_ST instruction fields.

Table 5–4 HW_ST Instruction Fields Descriptions

Extent	Mnemonic	Value	Description
[31:26]	OPCODE	$1F_{16}$	The opcode value.
[25:21]	RA	—	Write data register number.
[20:16]	RB	—	Base register for memory address.
[15:13]	TYPE	000_2	Physical — The effective address for the HW_ST instruction is physical.
		001_2	Physical/Cond — The effective address for the HW_ST instruction is physical. Store conditional version of the HW_ST instruction. The lock flag is returned in RA. Refer to PALcode restrictions for correct use of this function.
		010_2	Virtual — The effective address for the HW_ST instruction is virtual.
		110_2	Virtual/Alt — The effective address for the HW_ST instruction is virtual. Access checks use DTB_ ALT_MODE IPR.
		All others	Unused.
[12]	LEN	0	Access length is longword.
		1	Access length is quadword.
[11:0]	DISP	—	Holds a 12-bit signed byte displacement.

5.4.3 HW_RET Instruction

The HW_RET instruction is used to return instruction flow to a specified PC. The RB field of the HW_RET instruction specifies an integer GPR, which holds the new value of the PC. Bit [0] of this register provides the new value of PALmode after the HW_RET instruction is executed. Bits [15:14] of the instruction determine the stack action.

Normally the HW_RET instruction succeeds a CALL_PAL instruction, or a trap handler that pushed its PC onto the prediction stack. In this mode, the HINT should be set to ‘10’ to pop the PC and generate a predicted target address for the HW_RET instruction.

In some conditions, the HW_RET instruction is used in the middle of a PALcode flow to cause a group of instructions to retire. In these cases, if the HW_RET instruction does not have a corresponding instruction that pushed a PC onto the stack, the HINT field should be set to ‘00’ to keep the stack from being modified.

In the rare circumstance that the HW_RET instruction might be used like a JSR or JSR_COROUTINE, the stack can be managed by setting the HINT bits accordingly.

Figure 5–3 shows the HW_RET instruction format.

Figure 5–3 HW_RET Instruction Format

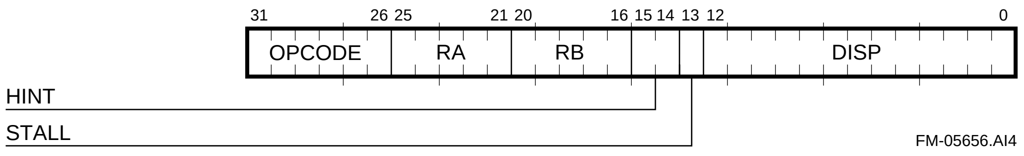


Table 5–5 describes the HW_RET instruction fields.

Table 5–5 HW_RET Instruction Fields Descriptions

Extent	Mnemonic	Value	Description
[31:26]	OPCODE	$1E_{16}$	The opcode value.
[25:21]	RA	—	Register number. It should be R31.
[20:16]	RB	—	Target PC of the HW_RET instruction. Bit [0] of the register's contents determines the new value of PALmode.
[15:14]	HINT	00	HW_JMP — The PC is not pushed onto the prediction stack. The predicted target is PC + (4*DISP[12:0]).
		01	HW_JSR — The PC is pushed onto the prediction stack. The predicted target is PC + (4*DISP[12:0]).
		10	HW_RET — The prediction is popped off the stack and used as the target.
		11	HW_COROUTINE — The prediction is popped off the stack and used as the target. The PC is pushed onto the stack.
[13]	STALL	—	If set, the fetcher is stalled until the HW_RET instruction is retired or aborted. The 21264 will: • Force a mispredict • Kill instructions that were fetched beyond the HW_RET instruction • Refetch the target of the HW_RET instruction • Stall until the HW_RET instruction is retired or aborted If instructions beyond the HW_RET have been issued out of order, they will be killed and refetched.
[12:0]	DISP	—	Holds a 13-bit signed longword displacement

5.4.4 HW_MFPR and HW_MTPR Instructions

The HW_MFPR and HW_MTPR instructions are used to access internal processor registers. The HW_MFPR instruction reads the value from the specified IPR into the integer register specified by the RA field of the instruction. The HW_MTPR instruction writes the value from the integer GPR, specified by the RB field of the instruction, into the specified IPR. Figure 5–4 shows the HW_MFPR and HW_MTPR instructions format.

Figure 5–4 HW_MFPR and HW_MTPR Instructions Format

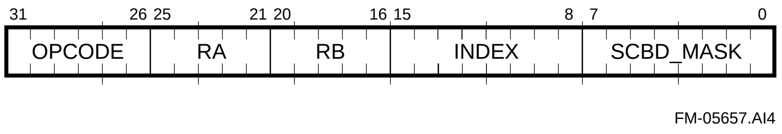


Table 5–6 describes the HW_MFPR and HW_MTPR instructions fields.

Table 5–6 HW_MFPR and HW_MTPR Instructions Fields Descriptions

Extent	Mnemonic	Value	Description
[31:26]	OPCODE	19_{16}	The opcode value for the HW_MFPR instruction.
		$1D_{16}$	The opcode value for the HW_MTPR instruction.
[25:21]	RA	—	Destination register for the HW_MFPR instruction. It should be R31 for the HW_MTPR instruction.
[20:16]	RB	—	Source register for the HW_MTPR instruction. It should be R31 for the HW_MFPR instruction.
[15:8]	INDEX	—	IPR index.
[7:0]	SCBD_ MASK	—	Specifies which IPR scoreboard bits in the IQ are to be applied to this instruction. If a mask bit is set, it indicates that the corresponding IPR scoreboard bit should be applied to this instruction.

5.5 Internal Processor Register Access Mechanisms

This section describes the hardware and software access mechanisms that are used for the 21264 IPRs.

Because the Ibox reorders and executes instructions speculatively, extra hardware is required to provide software with the correct view of the architecturally-defined state. The Alpha architecture defines two classes of state: general-purpose registers and memory. Register renaming is used to provide architecturally-correct register file behavior. The Ibox and Mbox each have dedicated hardware that provides correct memory behavior to the programmer. Because the internal processor registers are implementation-specific, and their state is not defined by the Alpha architecture, access mechanisms for these registers may be defined that impose restrictions and limitations on the software that uses them.

For every IPR, each instruction type can be classified by how it affects and is affected by the value held by that IPR.

- Explicit readers are HW_MFPR instructions that explicitly read the value of the IPR.
- Implicit readers are instructions whose behavior is affected by the value of the IPR. For example, each load instruction is an implicit reader of the DTB.
- Explicit writers are HW_MTPR instructions that explicitly write a value into the IPR.
- Implicit writers are instructions that may write a value into the IPR as a side effect of execution. For example, a load instruction that generates an access violation is an implicit writer of the VA, MM_STAT, and EXC_ADDR IPRs. In the 21264, only instructions that generate an exception will act as implicit IPR writers.

Only certain IPRs, such as those with write-one-to-clear bits, are both implicitly and explicitly written. The read-write semantics of these IPRs is controlled by software.

5.5.1 IPR Scoreboard Bits

In previous Alpha implementations, IPR registers were not scoreboarded in hardware. Software was required to schedule HW_MTPR and HW_MFPR instructions for each machine's pipeline organization in order to ensure correct behavior. This software scheduling task is more difficult in the 21264 because the Ibox performs dynamic scheduling. Hence, eight extra scoreboard bits are used within the IQ to help maintain correct IPR access order. The HW_MTPR and HW_MFPR instruction formats contain an 8-bit field that is used as an IPR scoreboard bit mask to specify which of the eight IPR scoreboard bits are to be applied to the instruction.

If any of the unmasked scoreboard bits are set when an instruction is about to enter the IQ, then the instruction, and those behind it, are stalled outside the IQ until all the unmasked scoreboard bits are clear and the queue does not contain any implicit or explicit readers that were dependent on those bits when they entered the queue. When all the unmasked scoreboard bits are clear, and the queue does not contain any of those readers, the instruction enters the IQ and the unmasked scoreboard bits are set.

HW_MFPR instructions are stalled in the IQ until all their unmasked IPR scoreboard bits are clear.

When scoreboard bits [3:0] and [7:4] are set, their effect on other instructions is different, and they are cleared in a different manner.

If any of scoreboard bits [3:0] are set when a load or store instruction enters the IQ, that load or store instruction will not be issued from the IQ until those scoreboard bits are clear.

Scoreboard bits [3:0] are cleared when the HW_MTPR instructions that set them are issued (or are aborted). Bits [7:4] are cleared when the HW_MTPR instructions that set them are retired (or are aborted).

Bits [3:0] are used for the DTB_TAG and DTB_PTE register pairs within the DTB fill flows. These bits can be used to order writes to the DTB for load and store instructions. See Section 4.3.1.

Bit [0] is used in both DTB and ITB fill flows to trigger, in hardware, a light-weight memory barrier (TB-MB) to be inserted between a LD_VPTE and the corresponding virtual-mode load instruction that missed in the TB.

5.5.2 Hardware Structure of Explicitly Written IPRs

IPRs that are written by software are physically implemented as two registers. When the HW_MTPR instruction that writes the IPR executes, it writes its value to the *first* register. When the HW_MTPR instruction is retired, the contents of the *first* register are written into the *second* register. Instructions that either implicitly or explicitly read the value of the IPR access the *second* register. Read-after-write and write-after-write dependencies are managed using the IPR scoreboard bits. To avoid write-after-read conflicts, the *second* register is not written until the writer is retired. The writer will not be retired until the previous reader is retired, and the reader is retired after it has read its value from the *second* register.

Some groups of IPRs are built using a single shared *first* register. To prevent write-after-write conflicts, IPRs that share a *first* register also share scoreboard bits.

5.5.3 Hardware Structure of Implicitly Written IPRs

Implicitly written IPRs are physically built using only a single level of register, however the IPR has two hardware states associated with it:

1. Default State—The contents of the register may be written when an instruction generates an exception. If an exception occurs, write a new value into the IPR and go to state 2.
2. Locked State—The contents of the register may only be overwritten by an excepting instruction that is older than the instruction associated with the contents of the IPR. If such an exception occurs, overwrite the value of the IPR. When the triggering instruction, or instruction that is older than the triggering instruction, is killed by the Ibox, go to state 1.

5.5.4 IPR Access Ordering

IPR access mechanisms must allow values to be passed through each IPR from a producer to its intended consumers.

Table 5–7 lists all of the paired instruction orderings between instructions of the four IPR access types. It specifies whether access order must be maintained, and if so, the mechanisms used to ensure correct ordering.

Table 5–7 Paired Instruction Fetch Order

Second Instruction	**First Instruction**			
	Implicit Reader	Implicit Writer	Explicit Reader	Explicit Writer
Implicit Reader	Read transactions can be reordered.	No IPRs in this class.	Read transactions can be reordered.	A variety of of mechanisms are used to ensure order: scoreboard bits to stall issue of reader; HW_RET_STALL to stall reader; double write plus buffer blocks to force retire and allow for propogation delay.
Implicit Writer	No IPRs in this class.	The hardware structure of implicitly written IPRs handles this case.	IPR-specific PALcode restrictions are required for this case. An interlock mechanism must be placed between the explicit reader and the implicit writer (a read transaction).	No IPRs in this class.
Explicit Reader	Read transactions can be reordered.	If the reader is in the PALcode routine invoked by the exception associated with the writer, then ordering is guaranteed.	Read transactions can be reordered.	Scoreboard bits stall issue of reader until writer is retired.

Table 5–7 Paired Instruction Fetch Order (Continued)

Second Instruction	First Instruction			
Explicit Writer	Reader reads second register. Writer cannot write second register until it is retired.	Write-one-to-clear bits, or performance counter special case. For example, performance counter increments are typically not scoreboarded against read transactions.	Reader reads second register. Writer cannot write second register until it is retired.	Scoreboard bits stall second writer in map stage until first writer is retired.

For convenience of implemenation, there is no IPR scoreboard bit checking within the same fetch block (octaword-aligned octaword).

- Within one fetch block, there can be only one explicit writer (HW_MTPR) to an IPR in a particular scoreboard group.
- Within one fetch block, an explicit writer (HW_MTPR) to an IPR in a particular scoreboard group cannot be followed by an explicit reader (HW_MFPR) to an IPR in that same scoreboard group.
- Within one fetch block, an explicit writer (HW_MTPR) to an IPR in a particular scoreboard group cannot be followed by an implicit reader to an IPR in that scoreboard group. This case covers writes to DTB_PTE or DTB_TAG followed by a LD, ST, or any memory operation, including HW_RETs without the 'stall' bit set.

5.5.5 Correct Ordering of Explicit Writers Followed by Implicit Readers

Across fetch blocks, the correct ordering of the explicit write of the DTB_PTE or DTB_TAG followed by an implicit reader (memory operation) is guaranteed using the IPR scoreboard bits.

However, there are cases where correct ordering of explicit writers followed by implicit readers cannot be guaranteed using the IPR scoreboard mechanism. If the instruction that implicitly reads the IPR does so before the issue stage of the pipeline, the scoreboard mechanism is not sufficient.

For example, modification of the ITB affects instructions before the issue state of the pipeline. In this case, PALcode must contain a HW_RET instruction, with its stall bit set, before any instruction that implicitly reads the IPR(s) in question. This prevents instructions that are newer than the HW_RET instruction from being successfully fetched, issued, and retired until after the HW_RET instruction is retired (or aborted).

There are also cases when the HW_RET with the STALL bit mechanism is not sufficient. There may be additional propagation delay past the retirement of the HW_RET instruction. In these cases, instead of using a HW_RET, a suggested method of ensuring the ordering is coding a group of 5 fetch blocks, where the first contains the HW_MTPR to the IPR, the second contains a HW_MTPR to the same IPR or one in the same scoreboard group, and where the following 3 fetch blocks each contain at least one non-NOP instruction.

5.5.6 Correct Ordering of Explicit Readers Followed by Implicit Writers

Certain IPRs that are updated as a result of faulting memory operations require PALcode assistance to maintain ordering against newer instructions. Consider the following code sequence:

```
HW_MFPR IPR_MM_STAT
LDQ rx,(ry)
```

It is typically the case that these instructions would issue in-order:

- The MFPR is data-ready and both instructions use a lower subcluster. However, the HW_MFPRs (and HW_MTPRs) respond to certain resource-busy indications and do not issue when the MBOX informs the IBOX that a certain set of resources (store bubbles) are busy.
- The LDs respond to a different set of resource-busy indications (load-bubbles) and could issue around the HW_MFPR in the presence of the former. PALcode assistance is required to enforce the issue order.

One totally reliable method is to insert an MB (memory barrier) instruction before the first load that occurs after the HW_MFPR MM_STAT. Another method would be to force a register dependency between the HW_MFPR and the LD.

5.6 PALshadow Registers

The 21264 contains eight extra virtual integer registers, called shadow registers, which are available to PALcode for use as scratch space and storage for commonly used values. These registers are made available under the control of the SDE[1] field of the I_CTL IPR. These shadow registers overlay R4 through R7 and R20 through R23, when the CPU is in PALmode and SDE[1] is set.

PALcode generally runs with shadow mode enabled. Any PALcode that supports CALL_PAL instructions must run in that mode because the hardware writes a PALshadow register with the return address of CALL_PAL instructions.

PALcode may occasionally be required to toggle shadow mode to obtain access to the overlayed registers.

5.7 PALcode Emulation of the FPCR

The FPCR register contains status and control bits. They are accessed by way of the MT_FPCR and MF_FPCR instructions. The register is physically implemented like an explicitly written IPR. It may be written with a value from the floating-point register file by way of the MT_FPCR instruction. Architecturally-compliant FPCR behavior requires PALcode assistance. The FPCR register must operate as listed here:

1. Correct operation of the status bits, which must be set when a floating-point instruction encounters an exceptional condition, independent of whether a trap for the condition is enabled.
2. Correct values must be returned when the FPCR is read by way of a MF_FPCR instruction.
3. Correct actions must occur when the FPCR is written by way of a MT_FPCR instruction.

5.7.1 Status Flags

The FPCR status bits in the 21264 are set with PALcode assistance. Floating-point exceptions, for which the associated FPCR status bit is clear or for which the associated trap is enabled, result in a hardware trap to the ARITH PALcode routine. The EXC_SUM register contains information to allow this routine to update the FPCR appropriately, and to decide whether to report the exception to the operating system.

5.7.2 MF_FPCR

The MF_FPCR is issued from the floating-point queue and executed by the Fbox. No PALcode assistance is required.

5.7.3 MT_FPCR

The MT_FPCR instruction is issued from the floating-point queue. This instruction is implemented as an explicit IPR write operation. The value is written into the *first* latch, and when the instruction is retired, the value is written into the *second* latch. There is no IPR scoreboarding mechanism in the floating-point queue, so PALcode assistance is required to ensure that subsequent readers of the FPCR get the updated value.

After writing the *first* latch, the MT_FPCR instruction invokes a synchronous trap to the MT_FPCR PALcode entry point. The PALcode can return using a HW_RET instruction with its STALL bit set. This sequence ensures that the MT_FPCR instruction will be correctly ordered for subsequent readers of the FPCR.

5.8 PALcode Entry Points

PALcode is invoked at specific entry points, of which there are two classes: CALL_PAL and exceptions.

5.8.1 CALL_PAL Entry Points

CALL_PAL entry points are used whenever the Ibox encounters a CALL_PAL instruction in the Istream. To speed the processing of CALL_PAL instructions, CALL_PAL instructions do not invoke pipeline aborts but are processed as normal jumps to the offset from the contents of the PAL_BASE register, which is specified by the CALL_PAL instruction's function field.

The Ibox fetches a CALL_PAL instruction, bubbles one cycle, and then fetches the instructions at the CALL_PAL entry point. For convenience of implementation, returns from CALL_PAL are aided by a linkage register (much like JSRs). PALshadow register R23 is used as the linkage register. The Ibox loads the PC of the instruction after the CALL_PAL instruction, into the linkage register. Bit [0] of the linkage register is set if the CALL_PAL instruction was executed while the processor was in PALmode.

The Ibox pushes the value of the return PC onto the return prediction stack. CALL_PAL instructions start at the following offsets:

- Privileged CALL_PAL instructions start at offset 2000_{16}.
- Nonprivileged CALL_PAL instructions start at offset 3000_{16}.

Each CALL_PAL instruction includes a function field that is used to calculate the PC of its associated PALcode entry point. The PALcode OPCDEC exception flow will be invoked if the CALL_PAL function field satisfies any of the following requirements:

- Is in the range of 40_{16} to $7F_{16}$ inclusive
- Is greater than BF_{16}
- Is between 00_{16} and $3F_{16}$ inclusive, and IER_CM[CM] is not equal to the kernel mode value 0

If none of the conditions above are met, the PALcode entry point PC is as follows:

- PC[63:15] = PAL_BASE[63:15]
- PC[14] = 0
- PC[13] = 1
- PC[12] = CALL_PAL function field [7]
- PC[11:6] = CALL_PAL function field [5:0]
- PC[5:1] = 0
- PC[0] = 1 (PALmode)

5.8.2 PALcode Exception Entry Points

When hardware encounters an exception, Ibox execution jumps to a PALcode entry point at a PC determined by the type of exception. The return PC of the instruction that triggered the exception is placed in the EXC_ADDR register and onto the return prediction stack.

Table 5–8 shows the PALcode exception entry locations and their offset from the PAL_BASE IPR. The entry points are listed in decreasing order of priority.

Table 5–8 PALcode Exception Entry Locations

Entry Name	Type	Offset$_{16}$	Description
DTBM_DOUBLE_3	Fault	100	Dstream TB miss on virtual page table entry fetch. Use three-level flow.
DTBM_DOUBLE_4	Fault	180	Dstream TB miss on virtual page table entry fetch. Use four-level flow.
FEN	Fault	200	Floating point disabled.
UNALIGN	Fault	280	Unaligned Dstream reference.
DTBM_SINGLE	Fault	300	Dstream TB miss.
DFAULT	Fault	380	Dstream fault or virtual address sign check error.
OPCDEC	Fault	400	Illegal opcode or function field: • Opcode 1, 2, 3, 4, 5, 6 or 7 • Opcode 19_{16}, $1B_{16}$, $1D_{16}$, $1E_{16}$ or $1F_{16}$, not PALmode or not I_CTL[HWE] • Extended precision IEEE format • Unimplemented function field of opcodes 14_{16} or $1C_{16}$
IACV	Fault	480	Istream access violation or virtual address sign check error

Table 5–8 PALcode Exception Entry Locations (Continued)

Entry Name	Type	Offset$_{16}$	Description
MCHK	Interrupt	500	Machine check.
ITB_MISS	Fault	580	Istream TB miss.
ARITH	Synch. Trap	600	Arithmetic exception or update to FPCR.
INTERRUPT	Interrupt	680	Interrupts: hardware, software, and AST.
MT_FPCR	Synch. Trap	700	Invoked when a MT_FPCR instruction is issued.
RESET/WAKEUP	Interrupt	780	Chip reset or wake-up from sleep mode.

6

Initialization and Configuration

This chapter provides information on 21264-specific microprocessor system initialization and configuration. It is organized as follows:

- Power-up reset flow
- Internal processor register (IPR) reset state

Initialization is controlled by the reset state machine, which is responsible for four major operations. Table 6–1 describes the four major operations.

Table 6–1 21264 Reset State Machine Major Operations

Operation	Function
Ramp up	Sequence the PLL input and output dividers (X_{div} and Z_{div}) to gradually raise the internal GCLK frequency and generate time intervals for the PLL to re-establish lock.
BiST/SROM	Receive a synchronous transfer on the **ClkFwdRst_H** pin in order to start built-in self-test and SROM load at a predictable GCLK cycle.
Clock forward interface	Receive a synchronous transfer on the **ClkFwdRst_H** pin in order to initialize the clock forwarding interface.
Ramp down	Sequence the PLL input and output dividers (X_{div} and Z_{div}) to gradually lower the internal GCLK frequency during sleep mode.

6.1 Power-Up Reset Flow and the RESET_L and DCOK_H Pins

The 21264 reset sequence is triggered using the two input signals: **Reset_L** and **DCOK_H** in a sequence that is described in Section 6.1.1. After **Reset_L** is deasserted, the following sequence of operations takes place:

1. The clock forwarding and system clock ratio configuration information is loaded onto the 21264. See Section 6.1.2.
2. The internal PLL is ramped up to operating frequency.
3. The internal arrays built-in self-test (BiST) is run, followed by Icache initialization using an external serial ROM (SROM) interface.

   The 21264 systems, unlike the Alpha 21064 and 21164 microprocessor systems, are required to have an SROM. The SROM provides the only means to configure the system port, and the SROM pins can be used as a software-controlled UART.

The Icache must contain PALcode that starts at location 0x780. This code is used to configure the 21264 IPRs as necessary before causing any offchip read or write commands. This allows the 21264 to be configured to match the external system implementation.

4. After configuring the 21264, control can be transferred to code anywhere in memory, including the noncacheable regions. The Icache can be flushed by a write operation to the ITB invalidate-all register after control is transferred. This transfer of control should be to addresses not loaded in the Icache by the SROM interface or the Icache may provide unexpected instructions.
5. Typically, any state required by the PALcode is initialized and then the console is started (switching out of PALmode and into native mode). The console code initializes and configures the system and boots an operating system from an I/O device such as a disk or the network.

Figure 6–1 shows the sequence of events at power-up, or cold reset. In Figure 6–1, note the following symbols for constraints and information:

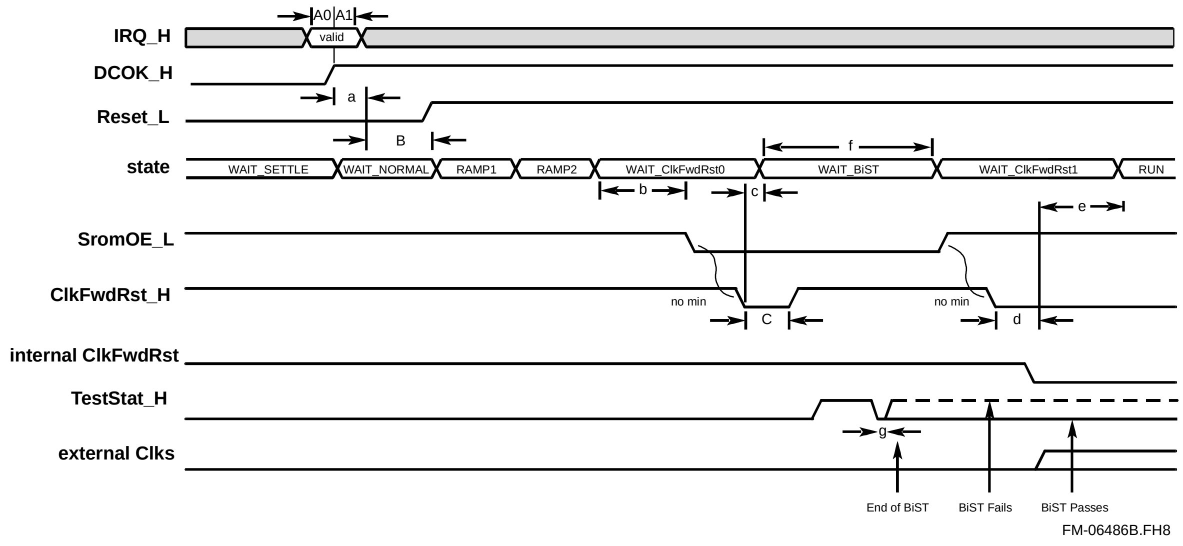
Constraints:

A	Setup (A0) and hold (A1) for IRQ’s to be latched by DCOK (2 ns for each).
B	Enough time for **Reset_L** to propogate through 5 stages of RESET synchronizer (clocked by the internal framing clock, which is driven by **EV6Clk_*x***). Worst case is 5x8x8 = 320 GCLK cycles, because Y_{div} values above 8 are out of range.
C	Min = 1 FrameClk cycle.

Information:

a	8 GCLK cycles from DCOK assertion to first “real” **EV6Clk_*x*** cycle.
b	Approximately 264 GCLK cycles for external framing clock to be sampled and captured.
c	1 **FrameClk_*x*** cycle.
d	3 **FrameClk_*x*** cycles.
e	Approximately 264 GCLK cycles to prevent first command from appearing too early.
f	Approximately 700,000 GCLK cycles for BiST + approximately 100,000 GCLK cycles fixed time + approximately 50,000 GCLK cycles per line of Icache for SROM load.
g	16 GCLK cycles.

Figure 6–1 Power-Up Timing Sequence



6.1.1 Power Sequencing and Reset State for Signal Pins

Power sequencing and avoiding potential failure mechanisms is described in Section 7.3.

The reset state for the signal pins is listed in Table 6–2.

Table 6–2 Signal Pin Reset State

Signal	Reset State	Signal	Reset State
Bcache			
BcAdd_H[23:4]	Tristated		
BcCheck_H[15:0]	Tristated	**BcTagInClk_H**	NA (input)
BcData_H[127:0]	Tristated	**BcTagOE_L**	Tristated
BcDataInClk_H[7:0]	NA (input)	**BcTagOutClk_*x***	Tristated
BcDataOE_L	Tristated	**BcTagParity_H**	Tristated
BcDataOutClk_*x*[3:0]	Tristated	**BcTagShared_H**	Tristated
BcDataWr_L	Tristated	**BcTagValid_H**	Tristated
BcLoad_L	Tristated	**BcTagWr_L**	Tristated
BcTag_H[42:20]	Tristated	**BcVref**	NA (I_DC_REF)
BcTagDirty_H	Tristated		

Table 6–2 Signal Pin Reset State (Continued)

Signal	Reset State	Signal	Reset State
System Interface			
IRQ_H[5:0]	NA (input)	**SysDataInClk_H[7:0]**	NA (input)
SysAddIn_L[14:0]	NA (input)	**SysDataInValid_L**	NA (input)
SysAddInClk_L	NA (input)	**SysDataOutClk_L[7:0]**	Tristated
SysAddOut_L[14:0]	Initially, during power-up reset, state is not defined. If not during power-up, preserves previous state. Then, after the clock forward reset period (as the external clocks start), signal driven to NZNOP until the reset state machine enters RUN, when it is driven to NOP.	**SysDataOutValid_L**	NA (input)
SysAddOutClk_L	Tristated	**SysFillValid_L**	NA (input)
SysCheck_L[7:0]	Tristated	**SysVref**	NA (I_DC_REF)
SysData_L[63:0]	Tristated		
Clocks			
ClkFwdRst_H	NA (input)	**FrameClk_*x***	NA (input)
ClkIn_H **ClkIn_L**	NA (input)	**PLL_VDD**	NA (I_DC_REF)
EV6Clk_H **EV6Clk_L**	NA (input)		
Miscellaneous			
DCOK_H	Must be deasserted until dc voltage reaches proper operating level.	**Tck_H**	NA (input)
PllBypass_H	NA (input)	**Tdi_H**	NA (input)
Reset_L	NA (input)	**Tdo_H**	Unspecified
SromClk_H	Tristated	**TestStat_H**	Tristated
SromData_H	NA (input)	**Tms_H**	NA (input)
SromOE_L	Tristated	**Trst_L**	NA (input)

In addition, as power is being ramped, **Reset_L** must be asserted — this allows the 21264 to reset internal state. Once the target voltage levels are attained, systems should assert **DCOK_H**. This indicates to the 21264 that internal logic functions can be evaluated correctly and that the power-up sequence should be continued. Prior to **DCOK_H** being asserted, the logic internal to the 21264 is being reset and the internal clock network is running (either clocked by the PLL VCO, which is at a nominal speed, or by **ClkIn_H**, if the PLL is bypassed).

The reset state machine is in state WAIT_SETTLE.

6.1.2 Clock Forwarding and System Clock Ratio Configuration

When **DCOK_H** is asserted, the 21264 samples several pins and latches in some initialization state, including the value of the PLL Y_{div} divisor, which specifies the ratio of the system clock to the internal clock, and enables the charge pump on the phase-locked loop.

Table 6–3 summarizes the pins and the suggested/required initialization state. Most of this information is supplied by placing (switch-selectable or hardwired) weak pull-ups or pull-downs on the **IRQ_H** pins. The **IRQ_H** pins are sampled on the rising edge of **DCOK_H**, during which time the 21264 is in reset and is not generating any system activity. During normal operation, the **IRQ_H** pins supply interrupt requests to the 21264.

It is possible to disable the 21264 PLL and source GCLK directly from **ClkIn_*x***. This mode is selected via **PllBypass_H**. The 21264 still produces a divided-down clock on **EV6Clk_*x***; this output clock, which tracks GCLK, can be used in a feedback loop to generate a locked input clock via an external PLL. The input clock can be locked against a slower speed system reference clock.

Table 6–3 Pin Signal Names and Initialization State

Signal Name	Sample Time	Function	Value
PllBypass_H	Continuous input	Select **ClkIn_*x*** onto GCLK instead of internal PLL.	0 Bypass[1] 1 Use PLL
ClkFwdRst_H	Sampling method according to **IRQ_H[4]**	—	—
Reset_L	Continuous input	—	—
IRQ_H[5]	Rising edge of **DCOK_H**	Select 1:1 FrameClk mode. Internal FrameClk can be generated two ways: 1 By sampling **FrameClk_H**. Used if **FrameClk_H** is slower than **ClkIn_H**. 2 As a direct copy of **EV6Clk_H**. Used if **FrameClk_H** is the same frequency as **ClkIn_H** or is DC.	0 Sample with **FrameClk_H** 1 Use a copy of **EV6Clk_H**

Table 6–3 Pin Signal Names and Initialization State

Signal Name	Sample Time	Function	Value	
IRQ_H[4]	Rising edge of **DCOK_H**	Select method of sampling **ClkFwdRst_H** to produce internal ClkFwdRst — either with external or internal copy of **FrameClk_*x***.	0 Sample with External **FrameClk_*x*** 1 Sample with Internal Frameclk	
IRQ_H[3:0]	Rising edge of **DCOK_H**	Select Y_{div} divisor value. This is the divide-down factor between GCLK and **EV6Clk_x**. When the PLL is in use and the 21264 is ramped-up to full speed, the VCO adjusts in order to phase-align (and rate-match) **EV6Clk_x** to **ClkIn_*x***. When the PLL is not in use, and **ClkIn_*x*** is bypassed onto GCLK, **EV6Clk_x** is slower than **ClkIn_*x*** by the divisor Y_{div}.	**IRQ_H[3:0]**	**Divisor**
			0011	3
			0100	4
			0101	5
			0110	6
			0111	7
			0000	8
			1000	9
			1001	10
			1010	11
			1011	12
			1100	13
			1101	14
			1110	15
			1111	16
DCOK_H	Continuous input	When deasserted, initializes the internal 21264 reset state machine and keeps the PLL internal oscillator running at a nominal speed. Assertion, which implies power to the 21264 is good, causes configuration information to be sampled.	—	

[1] The maximum permissible instantaneous change in **ClkIn_*x*** frequency is 333MHz (to prevent current spikes).

6.1.3 PLL Ramp Up

After the configuration is loaded through the **IRQ_H** pins, the next phase in the power up flow is the internal PLL ramp up sequence. Ramping up of the PLL is required to guarantee that the dynamic change in frequency will not cause the supply on the 21264 to fall due to the supply loop inductance. Clock control circuitry steps GCLK from power-up/reset clocking to 1/16th operating frequency, to ½ operating frequency, and finally normal operating frequency.

After the assertion of **DCOK_H**, the 21264 waits for the deassertion of **Reset_L** from the system while the PLL attempts to achieve a lock. The PLL internal ramp dividers are set to divide down the input clock by 16 and the PLL attempts to achieve lock against an effective input frequency of **ClkIn_*x***/16. Once lock is achieved, the actual internal frequency (GCLK) is **ClkIn_*x****(Y_{div} divisor value)/16. There should be a minimum delay of 100 ms between the assertion of **DCOK_H** and the deassertion of **Reset_L** to allow for this locking The reset state machine is in the WAIT_NOMINAL state.

After the deassertion of **Reset_L**, the reset state machine goes into the RAMP1 state. The 21264 ramps the internal frequency, by changing the effective input frequency of the PLL to **ClkIn_*x***/2 for a sufficient lock interval (about 20 μs). The state machine then goes into the RAMP2 state, changing the effective input frequency to ClkIn/1 for an additional lock interval (about 20 µs). The lock periods are generated by the internal duration counter, which is driven by GCLK. The counter counts 4108 GCLK cycles during the **ClkIn_*x***/2 lock interval. Note that GCLK is produced by the output of the PLL, which is locking to an input clock which is 1/2 of the operating frequency — therefore, the 4108 cycle interval constitutes a 12-20 µs interval when the operating frequency is 400–666 MHz. Then, the counter counts 8205 GCLK cycles during the **ClkIn_*x***/1 lock interval.

6.1.4 BiST and SROM Load and the TestStat_H Pin

The 21264 uses the deassertion of **ClkFwdRst_H** (which must be deasserted for a minimum of one **FrameClk_H** cycle and then reasserted) to begin built-in self-test (BiST). The reset state machine goes into the WAIT_BiST state. The power-up BiST lasts approximately 700,000 cycles. The result of the self-test is made available on the **TestStat_H** pin. The pin is forced low by the system reset. It is then forced high during BiST.

As BiST completes, the **TestStat_H** pin is held low for 16 GCLK cycles. Then, if BiST succeeds, the pin remains low. Otherwise, it is asserted. After successfully completing BiST, the 21264 then performs the SROM load sequence. After the SROM load sequence is finished, the 21264 deasserts **SromOE_L**.

6.1.5 Clock Forward Reset and System Interface Initialization

After the deassertion of **SromOE_L**, the reset state machine enters the WAIT_ClkFwdRst1 state, where the 21264 waits for the system to deassert **ClkFwdReset_H**. The 21264 samples the deasserting edge of **ClkFwdReset_H** to take synchronous actions. It uses this synchronous event to reset the clock forwarding interface, start the outgoing clocks, and deassert internal reset. The chip then waits 264 cycles before issuing commands. The reset state machine is then in RUN and the 21264 begins fetching code at address 0x780.

Table 6–4 lists signals relevant to the power-up flow, provides a short description of each, and any relevant constraints.

Table 6–4 Power-Up Flow Signals and Their Constraints

Signal Name	Description	Constraint
ClkIn_*x*	Differential clocks that are inputs to PLL or are bypassed onto GCLK directly	Clocks must be running before **DCOK_H** is asserted.
PLL_VDD	VDD supply to PLL	**PLL_VDD** must lead **VDD.**
VDD	VDD supply to the 21264 chip logic (except PLL)	—
DCOK_H	Logic signal to the 21264 that the VDD supply is good	—

Table 6–4 Power-Up Flow Signals and Their Constraints (Continued)

Signal Name	Description	Constraint
Reset_L	RESET pin asserted by SYSTEM to the 21264	**Reset_L** must be asserted prior to **DCOK_H** and must remain asserted for at least 100 ms after **DCOK_H** is asserted. This allows for PLL settling time. Deassertion of **Reset_L** causes the 21264 to ramp divisors to their final value and begin BiST.
ClkFwdRst_H Deassertion #1	Signal asserted by SYSTEM to synchronously commence built-in self-test and SROM load	**ClkFwdRst_H** must be deasserted after PLL has achieved its lock in its final divisor value (about 20 µs). The deassertion causes built-in self-test to begin on an internal clock cycle that corresponds to one framing clock cycle after **ClkFwdRst_H** is deasserted. **ClkFwdRst_H** can be asserted after one frame clock cycle. See Figure 6–1.
ClkFwdRst_H Deassertion #2	Signal asserted by SYSTEM to initialize and reset clock forwarding interfaces	**ClkFwdRst_H** must be deasserted when the Cbox has loaded configuration information. This occurs as the first part of the serial ROM load, after BiST is run. Once **ClkFwdRst_H** is deasserted, the interface is initialized and can receive probe requests from the 21264.

6.2 Internal Processor Register Power-Up Reset State

Many IPR bits are not initialized by reset. They are located in error-reporting registers and other IPR states. They must be initialized by initialization PALcode. Table 6–5 lists the state of all internal processor registers (IPRs) immediately following power-up reset. The table also specifies which registers need to be initialized by power-up PALcode.

Table 6–5 Internal Processor Registers at Power-Up Reset State

Mnemonic	Register Name	Reset State	Comments
Ibox IPRs			
ITB_TAG	ITB tag array write	X	—
ITB_PTE	ITB PTE array write	X	—
ITB_IAP	ITB invalidate-all (ASM=0)	X	—
ITB_IA	ITB invalidate all	X	Must be written to in PALcode.
ITB_IS	ITB invalidate single	X	—
EXC_ADDR	Exception address	X	—
IVA_FORM	Instruction VA format	X	—
IER_CM	Interrupt enable current mode	X	Must be written to in PALcode.
SIRR	Software interrupt request	X	—
ISUM	Interrupt summary	X	—

Table 6–5 Internal Processor Registers at Power-Up Reset State (Continued)

Mnemonic	Register Name	Reset State	Comments
HW_INT_CLR	Hardware interrupt clear	X	Must be cleared in PALcode.
EXC_SUM	Exception summary	X	—
PAL_BASE	PAL base address	Cleared	—
I_CTL	Ibox control	IC_EN = 3	All other bits are cleared on reset.
I_STAT	Ibox status	X	Must be cleared in PALcode.
IC_FLUSH	Icache flush	X	—
CLR_MAP	Clear virtual-to-physical map	X	—
SLEEP	Sleep mode	X	—
PCTX	Ibox process context	PCTX[FPE] is set. All other bits are cleared.	
PCTR_CTL	Performance counter control	X	Must be cleared in PALcode.
Ebox IPRs			
CC	Cycle counter	X	Must be cleared in PALcode.
CC_CTL	Cycle counter control	X	Must be cleared in PALcode.
VA	Virtual address	X	—
VA_FORM	Virtual address format	X	—
VA_CTL	Virtual address control	X	Must be cleared in PALcode.
Mbox IPRs			
DTB_TAG0	DTB tag array write 0	Cleared	—
DTB_TAG1	DTB tag array write 1	Cleared	—
DTB_PTE0	DTB PTE array write 0	Cleared	—
DTB_PTE1	DTB PTE array write 1	Cleared	—
DTB_ALTMODE	DTB alternate processor mode	X	PALcode must initialize.
DTB_IAP	DTB invalidate all process ASM = 0	X	—
DTB_IA	DTB invalidate all process	X	Must be written to in PALcode.
DTB_IS0	DTB invalidate single (array 0)	X	—
DTB_IS1	DTB invalidate single (array 1)	X	—
DTB_ASN0	DTB address space number 0	Cleared	—
DTB_ASN1	DTB address space number 1	Cleared	—
MM_STAT	Memory management status	X	—
M_CTL	Mbox control	Cleared	—
DC_CTL	Dcache control	DC_CTL[7:2] are cleared at reset. DC_CTL[1:0] are set at power up.	
DC_STAT	Dcache status	X	Must be cleared in PALcode.

Table 6–5 Internal Processor Registers at Power-Up Reset State (Continued)

Mnemonic	Register Name	Reset State	Comments
Cbox IPRs			
C_DATA	Cbox data	X	Must be read in PALcode.
C_SHFT	Cbox shift control	X	—

7 Electrical Data

This chapter describes the electrical characteristics of the 21264 and its interface pins. The chapter contains both ac and dc electrical characteristics and power supply considerations, and is organized as follows:

- Electrical characteristics
- DC characteristics
- Power supply sequencing
- AC characteristics

7.1 Electrical Characteristics

Table 7–1 lists the maximum electrical ratings for the 21264.

Table 7–1 Maximum Electrical Ratings

Characteristics	Ratings	
Storage temperature	–55° C to +125° C (–67° F to 257° F)	
Junction Temperature	0° C to 100° C (32° F to 212° F)	
Maximum dc voltage on signal pins	**VDD** + 400 mV	
Minimum dc voltage on signal pins	**VSS** – 400 mV	
Maximum power @ indicated VDD for the following frequencies:	**Frequency**	**Peak Power**
	466 MHz	82.0 W @ 2.30 V VDD
	500 MHz	91.0 W @ 2.30 V VDD
	550 MHz	100.0 W @ 2.30 V VDD
	575 MHz	107.5 W @ 2.35 V VDD
	600 MHz	109.0 W @ 2.30 V VDD

Notes: Stresses above those listed under the given maximum electrical ratings may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to these limits for extended periods of time may affect device reliability

Power data is preliminary and based on measurements from a limited set of material.

7.2 DC Characteristics

This section contains the dc characteristics for the 21264. The 21264 pins can be divided into 10 distinct electrical signal types. The mapping between these signal types and the package pins is shown in Chapter 3. Table 7–2 shows the signal types.

Table 7–2 Signal Types

Signal Type	Description
I_DC_POWER	Supply voltage pins (**VDD/PLL_VDD**)
I_DC_REF	Input dc reference pin
I_DA	Input differential amplifier receiver
I_DA_CLK	Input differential amplifier clock receiver
O_OD	Open-drain output driver
O_OD_TP	Open-drain driver for test pins
O_PP	Push-pull output driver
O_PP_CLK	Push-pull output clock driver
B_DA_OD	Bidirectional differential amplifier receiver — open-drain
B_DA_PP	Bidirectional differential amplifier receiver — push-pull

DC Switching Characteristics for Each Signal Type

Tables 7–3 through 7–12 show the dc switching characteristics of each signal type.

Notes for Tables 7–3 to 7–12

The following notes apply to Tables 7–3 to 7–12.

1. The differential voltage, Vdiff, is the absolute difference between the differential input pins.
2. Delta V_{BIAS} is defined as the open-circuit differential voltage on the appropriate differential pairs. Test condition for these inputs are to let the input network self bias and measure the open circuit voltage. The test load must be ≥ 1M ohm. In normal operation, these inputs are coupled with a 680-pF capacitor.
3. Functional operation of the 21264 with less than all **VDD** and **VSS** pins connected is not implied.
4. Please see the special supply decoupling and noise requirements for the **PLL_VDD** outlined in the *21264 PLL Specification.*
5. The test load is a 50-ohm resistor to VDD/2. The resistor can be connected to the 21264 pin by a 50-ohm transmission line of any length.
6. DC test conditions set the minimum swing required. These dc limits set the trip point precision.
7. Input pin capacitance values include 2.0 pF added for package capacitance.

Note: Current out of a 21264 pin is represented by a – symbol while a + symbol indicates current flowing into a 21264 pin.

Table 7–3 VDD (I_DC_POWER)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
VDD	Processor core supply voltage	—	2.1 V	2.3 V
Power (sleep)	Processor power required (sleep)	@ VDD = 2.3 V Note 3	—	19 W[1]
PLL_VDD	PLL supply voltage (Note 4)	—	3.135 V	3.465 Vc
PLL_IDD	PLL supply current (running)	Freq = 600 MHz	—	25 mA

[1] Power measured at 37.5 MHz while running the “Ebox aliveness test.”

Table 7–4 Input DC Reference Pin (I_DC_REF)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
VREF	DC input reference voltage	—	600 mV	VDD – 650 mV
$\lvert I_I \rvert$	Input current	VSS ≤ V ≤ VDD	—	150 µA

Table 7–5 Input Differential Amplifier Receiver (I_DA)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{IL}	Low-level input voltage	Note 6	—	VREF – 200 mV
V_{IH}	High-level input voltage	—	VREF + 200 mV	—
$\lvert I_I \rvert$	Input current	VSS ≤ V ≤ VDD	—	150 µA
C_{IN}	Input-pin capacitance	Freq =10 MHz	—	5.7 pF Note 7

Table 7–6 Input Differential Amplifier Clock Receiver (I_DA_CLK)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{diff}	Differential input voltage	—	200 mv Note 1	—
$\lvert \Delta V_{BIAS} \rvert$	Open-circuit differential	I ≤ ± 1 µA Note 2	—	50 mV
$\lvert I_I \rvert$	Input current	VSS ≤ V ≤ VDD	—	150 µA
C_{IN}	Input-pin capacitance	Freq =10 MHz	—	5.0 pF Note 7

Table 7–7 Open-Drain Output Driver (O_OD)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{OL}	Low-level output voltage	I_{OL} = 70 mA	—	400 mV
$\|I_{OZ}\|$	High impedance output current	0 < V < VDD	—	150 µA
C_{OD}	Open-drain pin capacitance	Freq = 10 MHz	—	5.7 pF Note 7

Table 7–8 Bidirectional, Differential Amplifier Receiver, Open-Drain Output Driver (B_DA_OD)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{IL}	Low-level input voltage	Note 6	—	VREF –200 mv
V_{IH}	High-level input voltage	—	VREF + 200 mV	—
V_{OL}	Low-level output voltage	I_{OL} = 70 mA	—	400 mV
$\|I_I\|$	Input current	VSS ≤ V ≤ VDD	—	150 µA[1]
C_{IN}	Input-pin capacitance	Freq =10 MHz	—	5.7 pF Note 7

[1] Measurement taken with output driver disabled.

Table 7–9 Open-Drain Driver for Test Pins (O_OD_TP)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{OL}	Low-level output voltage	I_{OL} = 15 mA	—	400 mV
$\|I_{OZ}\|$	High-impedance output current	0 < V < VDD	—	150 µA
C_{OD}_TP	Pin capacitance	Freq = 10 MHz	—	5.2 pF Note 7

Table 7–10 Bidirectional, Differential Amplifier Receiver, Push-Pull Output Driver (B_DA_PP)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{IL}	Low-level input voltage	—	—	VREF – 200 mV
V_{IH}	High-level input voltage	—	VREF + 200 mV	—
V_{OL}	Low-level output voltage	I_{OL} = 6 mA	—	400 mV
V_{OH}	High-level output voltage	I_{OH} = –6 mA	VDD – 400 mV	—
$\|I_I\|$	Input current	VSS ≤ V ≤ VDD	—	150 µA[1]
C_{IN}	Input-pin capacitance	Freq =10 MHz	—	6.0 pF Note 7

[1] Measurement taken with output driver disabled.

Table 7–11 Push-Pull Output Driver (O_PP)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{OL}	Low-level output voltage	I_{OL} = 40 mA	—	500 mV
V_{OH}	High-level output voltage	I_{OL} = –40 mA	VDD – 500 mV	—
$\| I_{OZ} \|$	High-impedance output current	0 < V < VDD	—	150 μA
C_{OD}	Open-drain pin capacitance	Freq = 10 MHz	—	6.0 pF Note 7

Table 7–12 Push-Pull Output Clock Driver (O_PP_CLK)

Parameter Symbol	Description	Test Conditions	Minimum	Maximum
V_{OL}	Low-level output voltage	Note 5	—	VDD/2 – 325 mV
V_{OH}	High-level output voltage	Note 5	VDD/2 + 325 mV	—
$\| I_{OZ} \|$	High-impedance output current	0 < V < VDD	—	40 mA[1]

[1] Measured value includes current from onchip termination structures.

7.3 Power Supply Sequencing and Avoiding Potential Failure Mechanisms

Before the power-on sequencing can occur, systems should ensure that **DCOK_H** is deasserted and **Reset_L** is asserted. Then, systems ramp power to the 21264 **PLL_VDD** @ 3.3 V and the 21264 power planes (**VDD** @ 2.2 V, not to exceed 2.3 V under any circumstances), with **PLL_VDD** leading **VDD**. Systems should supply differential clocks to the 21264 on **ClkIn_H** and **ClkIn_L**. The clocks should be running as power is supplied.

When enabling the power supply inputs in a system, three failure mechanisms must be avoided:

1. Bidirectional signal buses must not conflict during power-up. A conflict on these buses can generate high current conditions, which can compromise the reliability of the associated chips.
2. Similarly, input receivers should not see intermediate voltage levels that can also generate high current conditions, which can compromise the reliability of the receiving chip.
3. Finally, no CMOS chip should see an input voltage that is higher than its internal VDD. In such a condition, a reasonable level of charge can be injected into the bulk of the die. This condition can expose the chip to a positive-feedback latchup condition.

The 21264 addresses those three failure mechanisms by disabling all of its outputs and bidirectional pins (with three exceptions) until the assertion of **DCOK_H**. The three exceptions are **Tdo_H**, **EV6Clk_L**, and **EV6Clk_H**. **Tdo_H** is used only in the tester environment and does not need to be disabled. **EV6Clk_L** and **EV6Clk_H** are outputs that are both generated and consumed by the 21264; thus, **VDD** tracks for both the producer and consumer.

On the push-pull interfaces:

- Disabling all output drivers leaves the output signal at the DC bias point of the termination network.
- Disabling the bidirectional drivers leaves the other consumers of the bus as the bus master.

On the open-drain interfaces:

- Disabling all output drivers leaves the output signal at the voltage of the open-drain pull-up.
- Disabling all bidirectional drivers leaves the other consumers of the bus as the bus master.

To avoid failure mechanism number two, systems must sequence and control external signal flow in such a way as to avoid zero differential into the 21264 input receivers (I_DA, I_DA_CLK, B_DA_OD, B_DA_PP, and B_DA_PP). Finally, to avoid failure mechanism number three, systems must sequence input and bidirectional pins (I_DA, I_DA_CLK, B_DA_OD, B_DA_PP, and I_DC_REF) such that the 21264 does not see a voltage above its VDD.

In addition, as power is being ramped, **Reset_L** must be asserted — this allows the 21264 to reset internal state. Once the target voltage levels are attained, systems should assert **DCOK_H**. This indicates to the 21264 that internal logic functions can be evaluated correctly and that the power-up sequence should be continued. Prior to **DCOK_H** being asserted, the logic internal to the 21264 is being reset and the internal clock network is running (either clocked by the VCO, which is at a nominal speed, or by **ClkIn_H**, if the PLL is bypassed).

The reset state machine is in state WAIT_SETTLE.

7.4 AC Characteristics

Abbreviations:

The following abbreviations apply to Table 7–13:

- TSU = Setup time
- Duty cycle = Minimum clock duty cycle
- TDH = Hold time
- Slew rate = referenced to signal edge

AC Test Conditions:

The following conditions apply to the measurements that are listed in Table 7–13:

- VDD is in the range between 2.1 V and 2.3 V.
- SysVref is VDD/2 Volts.
- BcVref is 0.75 Volts.

- The input voltage swing is Vref ± 0.40 Volts.
- All output skew data is based on simulation into a 50-ohm transmission line that is terminated with 50 ohms to VDD/2 for Bcache timing, and with 50 ohms to VDD for all other timing.

  Timings are measured at the pins as follows:

  – For open-drain outputs, timing is measured to $(V_{ol} + V_{term})/2$. Where V_{term} is the off-chip termination voltage for system signals.
  – For non-open-drain outputs, timing is measured to $(V_{ol} + V_{oh})/2$.
  – For all inputs other than type I_DA_CLK, timing is measured to the point where the input signal crosses VREF.
  – For type I_DA_CLK inputs, timing is measured when the voltage on the complementary inputs is equal.

Table 7–13 AC Specifications

Signal Name	Type	Reference Signal	TSU[1]	TDH[2]	TSkew	Duty Cycle	TSlew
SysAddIn_L[14:0	I_DA	**SysAddInClk_L**	400 ps	400 ps	NA	NA	1.0 V/ns
SysFillValid_L	I_DA	**SysAddInClk_L**	400 ps	400 ps	NA	NA	1.0 V/ns
SysDataInValid_L	I_DA	**SysAddInClk_L**	400 ps	400 ps	NA	NA	1.0 V/ns
SysDataOutValid_L	I_DA	**SysAddInClk_L**	400 ps	400 ps	NA	NA	1.0 V/ns
SysAddInClk_L	I_DA	NA	NA	NA	NA	45–55%	1.0 V/ns
SysAddOut_L[14:0]	O_OD	**SysAddOutClk_L**	NA	NA	± 300 ps[3]	NA	NA
SysAddOutClk_L	O_OD	**EV6Clk_*x***	NA	NA	± 400 ps	45-55%	NA
SysData_L[63:0]	B_DA_OD	**SysDataInClk_H[7:0]**	400 ps	400 ps	NA	NA	1.0 V/ns
		SysDataOutClk_L[7:0][4]	NA	NA	± 300 ps[3]	NA	NA
SysCheck_L[7:0]	B_DA_OD	**SysDataInClk_H[7:0]**	400 ps	400 ps	NA	NA	1.0 V/ns
		SysDataOutClk_L[7:0][4]	NA	NA	± 300 ps[3]	NA	NA
SysDataInClk_H[7:0]	I_DA	NA	NA	NA	NA	45-55%	1.0 V/ns
SysDataOutClk_L[7:0]	O_OD	**EV6Clk_*x***	NA	NA	± 400 ps	45-55%	NA
BcAdd_H[23:4]	O_PP	**BcTagOutClk_*x***	NA	NA	± 300 ps[5,6]	NA	—
BcDataOE_L	O_PP	**BcDataOutClk_*x*[3:0]**[7]				45-55%	—
BcLoad_L	O_PP					38-63%[8]	—
BcDataWr_L	O_PP					40-60%[9]	—
BcData_H[127:0]	B_DA_PP	**BcDataOutClk_*x*[3:0]**[10]	**NA**	**NA**	± 300 ps[6]	45-55%	1.0 V/ns
						38-63%[8]	NA
						40-60%[9]	NA
		BcDataInClk_H[7:0]	400 ps	400 ps	NA	NA	NA
BcDataInClk_H[7:0]	I_DA	NA	NA	NA	NA	45-55%	
BcDataOutClk_H[3:0]	O_PP	**EV6Clk_*x***	NA	NA	± 400 ps		
BcDataOutClk_L[3:0]	O_PP	**EV6Clk_*x***	NA	NA	± 400 ps		
BcTag_H[42:20]	B_DA_PP	**BcTagInClk_H**	400 ps	400 ps	NA	NA	1.0 V/ns
BcTagValid_H	B_DA_PP	**BcTagOutClk_*x***	NA	NA	± 300 ps[6]	45-55%	NA

Table 7–13 AC Specifications (Continued)

Signal Name	Type	Reference Signal	TSU[1]	TDH[2]	TSkew	Duty Cycle	TSlew
BcTagDirty_H	B_DA_PP					38-63%[8]	NA
BcTagShared_H	B_DA_PP					40-60%[9]	NA
BcTagParity_H	B_DA_PP						
BcTagOE_L	O_PP						
BcTagWr_L	O_PP						
BcTagInClk_H	I_DA	NA	NA	NA	NA	45-55%	
BcTagOutClk_*x*	O_PP	**EV6Clk_*x***	NA	NA	± 400 ps		
IRQ_H[5:0]	I_DA	**DCOK_H**	10 ns[11]	10 ns[11]	NA	NA	100 mV/ns
Reset_L[12]	I_DA		NA	NA	NA	NA	100 mV/ns
DCOK_H[13]	I_DA		NA	NA	NA	NA	100 mV/ns
PllBypass_H[14]	I_DA		NA	NA	NA	NA	100 mV/ns
ClkIn_*x*[15]	I_DA_CLK		NA	NA	NA	40–60%[16]	1.0 V/ns
FrameClk_*x*[17]	I_DA_CLK	**ClkIn_*x***	400 ps	400 ps	NA	NA	1.0 V/ns
EV6Clk_*x*[18]	O_PP_CLK	**ClkIn_*x***	**NA**	**NA**	±1.0 ns	YDiv±5%	NA
EV6Clk_x[19]		Cycle Compression Specification:			See Note 19		
ClkFwdRst_H	**I_DA**	**FrameClk_*x***	400 ps	400 ps	NA	NA	1.0 V/ns
SromData_H	I_DA	**SromClk_H**	2.0 ns	2.0 ns	NA		100 mV/ns
SromOE_L	O_OD	**EV6Clk_*x***	NA	NA	± 2.0 ns		
SromClk_H[20]	O_OD	**EV6Clk_*x***	**NA**	**NA**	± 7.0 ns		
Tms_H	I_DA	**Tck_H**	2.0 ns	2.0 ns	NA	NA	100 mV/ns
Trst_L[21]	I_DA	**Tck_H**	**NA**	**NA**	**NA**	**NA**	100 mV/ns
Tdi_H	I_DA	**Tck_H**	2.0 ns	2.0 ns	NA	NA	100 mV/ns
Tdo_H	O_OD	**Tck_H**	NA	NA	± 7.0 ns	NA	NA
Tck_H	I_DA	IEEE 1149.1 Port Freq. = 5.0 MHz Max.	NA	NA	NA	45-55%	100 mV/ns
TestStat_H	O_OD	**EV6Clk_*x***	NA	NA	± 4.0 ns	NA	NA

1 The TSU specified for all clock-forwarded signal groups is with respect to the associated clock.

2 The TDH specified for all clock-forwarded signal groups is with respect to the associated clock.

3 The TSkew value applies only when the SYS_CLK_DELAY[0:1] entry in the Cbox WRITE_ONCE chain (Table 4–23) is set to zero phases of delay between forwarded clock out and address/data.

4 The TSkew specified for **SysData_L** signals is only with respect to the associated clock.

5 These signals should be referenced to **BcTagOutClk_*x*** when measuring TSkew, provided that **BcTagOutClkl_*x*** and **BcDataOutClk_*x*** have no programmed offset.

6 The TSkew value applies only when the BC_CLK_DELAY[0:1] entry in the Cbox WRITE_ONCE chain (Table 4–23) is set to zero phases of delay for Bcache clock.

7 The TSkew specified for **BcAdd_H** signals is only with respect to the associated clock.

8 The duty cycle for 2.5X single data mode 2 GCLK phases high and 3 GCLK phases low.

[9] The duty cycle for 3.5X single data mode 3 GCLK phases high and 4 GCLK phases low.

[10] The TSkew specified for **BcData_H** signals is only with respect to the associated clock pair.

[11] **IRQ_H[5:0]** must have their TSU and TDH times referenced to **DCOK_H** during power-up to ensure the correct Y divider and resulting **EV6Clk_*x*** duty cycle. When the 21264 is executing instructions **IRQ_H[5:0]** act as normal asynchronous pins to handle interrupts.

[12] **Reset_L** is an asynchronous pin. It may be asserted asynchronously.

[13] **DCOK_H** is an asynchronous pin. Note the minimum slew rate on the assertion edge.

[14] **PllBypass_H** may not switch when **ClkIn_*x*** is running. This pin must either be deasserted during power-up or the 21264 core power pin (**VDD** pins) indicating the 21264's internal PLL will be used. Note that it is illegal to use **PllBypass_H** asserted during power-up unless a **ClkIn_*x*** is present.

[15] **ClkIn_*x*** has specific input jitter requirements to ensure optimum performance of the internal 21264 PLL.

[16] In PLL bypass mode, duty cycle deviation from 50%–50% directly degrades device operating frequency.

[17] The TSU and TDH of **FrameClk_*x*** are referenced to the deasserting edge of **ClkIn_*x***.

[18] This signal is a feedback to the internal PLL and may be monitored for overall 21264 jitter. It can also be used as a feedback signal to an external PLL when in PLL bypass mode. Proper termination of **EV6Clk_*x*** is imperative.

[19] The cycle or phase cannot be more than 5% shorter than the nominal. Do not confuse this measurement with duty cycle. Refer to Appendix F in the *21264 PLL Specifications* for the system measurement procedure.

[20] The period for **SromClk_H** is 256 GCLK cycles.

[21] When **Trst_L** is deasserted, **Tms_H** must not change state. **Trst_L** is asserted asynchronously but may be deasserted synchronously.

8
Thermal Management

This chapter describes the 21264 thermal management and thermal design considerations, and is organized as follows:

- Operating temperature
- Heat sink specifications
- Thermal design considerations

8.1 Operating Temperature

The 21264 is specified to operate when the temperature at the center of the heat sink (T_c) is as shown in Table 8–1. Temperature T_c should be measured at the center of the heat sink, between the two package studs. The GRAFOIL pad is the interface material between the package and the heat sink.

Table 8–1 Operating Temperature at Heat Sink Center (T_c)

T_c	Frequency
76.9° C	466 MHz
75.1° C	500 MHz
72.7° C	550 MHz
71.5° C	575 MHz
70.3° C	600 MHz

Note: Compaq recommends using the heat sink because it greatly improves the ambient temperature requirement.

Table 8–2 lists the values for the center of heat-sink-to-ambient ($\theta_c a$) for the 21264 587-pin PGA. Tables 8–3 through 8–7 show the allowable T_a (without exceeding T_c) at various airflows.

Table 8–2 q_ca at Various Airflows for 21264

Airflow (linear ft/min)	100	200	400	800	1000
$\theta_c a$ with heat sink type 1 (°C/W)	2.0	1.2	0.65	0.40	0.37
$\theta_c a$ with heat sink type 2 (°C/W)	1.4	0.78	0.45	0.33	0.31
$\theta_c a$ with heat sink type 3[1] (°C/W)			**— 0.38 —**		

[1] Heat sink type 3 has a 80 mm × 80 mm × 15 mm fan attached.

Table 8–3 Maximum T_a for 21264 @ 466 MHz with Various Airflows

Airflow (linear ft/min)	100	200	400	800	1000
Maximum T_a with heat sink type 1 (°C)	—	—	26.9	46.1	48.4
Maximum T_a with heat sink type 2 (°C)	—	—	42.3	51.5	53.1
Maximum T_a with heat sink type 3[1] (°C)			**— 47.7 —**		

[1] Heat sink type 3 has a 80 mm × 80 mm × 15 mm fan attached.

Table 8–4 Maximum T_a for 21264 @ 500 MHz with Various Airflows

Airflow (linear ft/min)	100	200	400	800	1000
Maximum T_a with heat sink type 1 (°C)	—	—	21.2	41.9	44.4
Maximum T_a with heat sink type 2 (°C)	—	—	37.8	47.7	49.4
Maximum T_a with heat sink type 3[1] (°C)			**— 43.6 —**		

[1] Heat sink type 3 has a 80 mm × 80 mm × 15 mm fan attached.

Table 8–5 Maximum T_a for 21264 @ 550 MHz with Various Airflows

Airflow (linear ft/min)	100	200	400	800	1000
Maximum T_a with heat sink type 1 (°C)	—	—	—	36.3	39.1
Maximum T_a with heat sink type 2 (°C)	—	—	31.8	42.7	44.5
Maximum T_a with heat sink type 3[1] (°C)			**— 38.2 —**		

[1] Heat sink type 3 has a 80 mm × 80 mm × 15 mm fan attached.

Table 8–6 Maximum T_a for 21264 @ 575 MHz with Various Airflows

Airflow (linear ft/min)	100	200	400	800	1000
Maximum T_a with heat sink type 1 (°C)	—	—	—	33.5	36.4
Maximum T_a with heat sink type 2 (°C)	—	—	28.8	40.2	42.1
Maximum T_a with heat sink type 3[1] (°C)			**— 35.4 —**		

[1] Heat sink type 3 has a 80 mm × 80 mm × 15 mm fan attached.

Table 8–7 Maximum T_a for 21264 @ 600 MHz with Various Airflows

Airflow (linear ft/min)	100	200	400	800	1000
Maximum T_a with heat sink type 1 (°C)	—	—	—	30.7	33.7
Maximum T_a with heat sink type 2 (°C)	—	—	25.8	37.7	39.6
Maximum T_a with heat sink type 3[1] (°C)				**— 32.7 —**	

[1] Heat sink type 3 has a 80 mm × 80 mm × 15 mm fan attached.

8.2 Heat Sink Specifications

Three heat sink types are specified. The mounting holes for all three are in line with the cooling fins.

Figure 8–1 shows the heat sink type 1, along with its approximate dimensions.

Figure 8–1 Type 1 Heat Sink

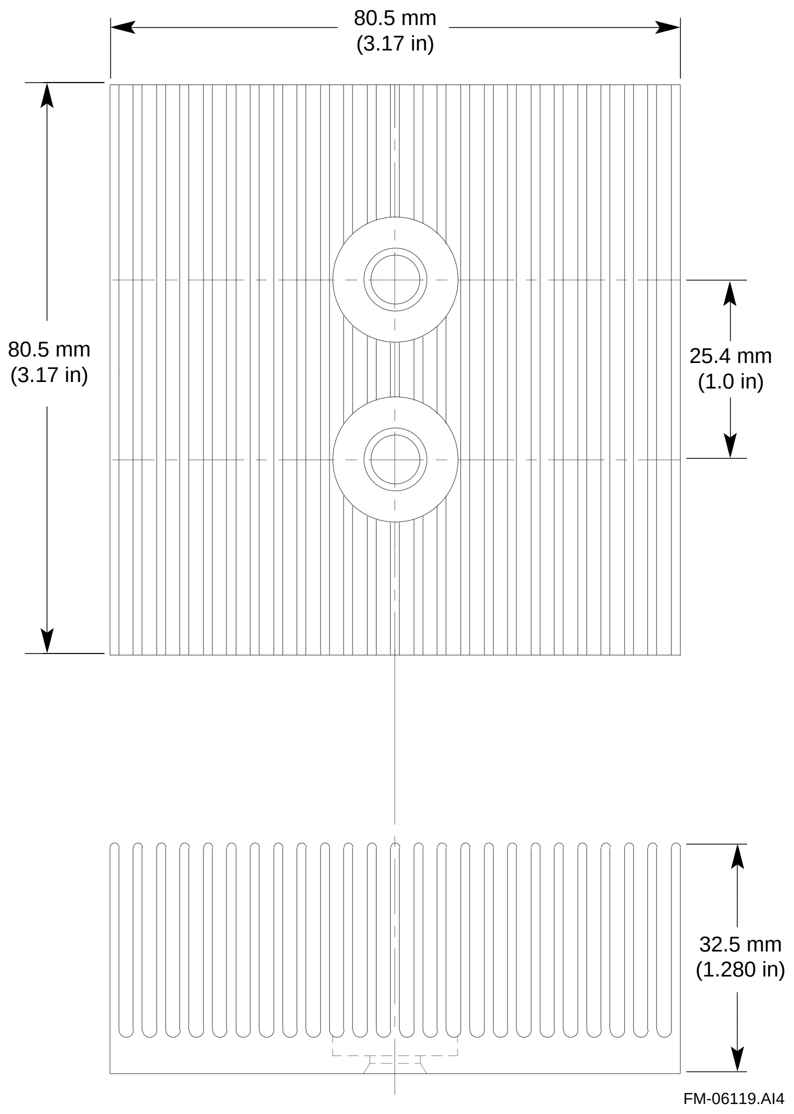


Figure 8–2 shows the heat sink type 2, along with its approximate dimensions.

Figure 8–2 Type 2 Heat Sink

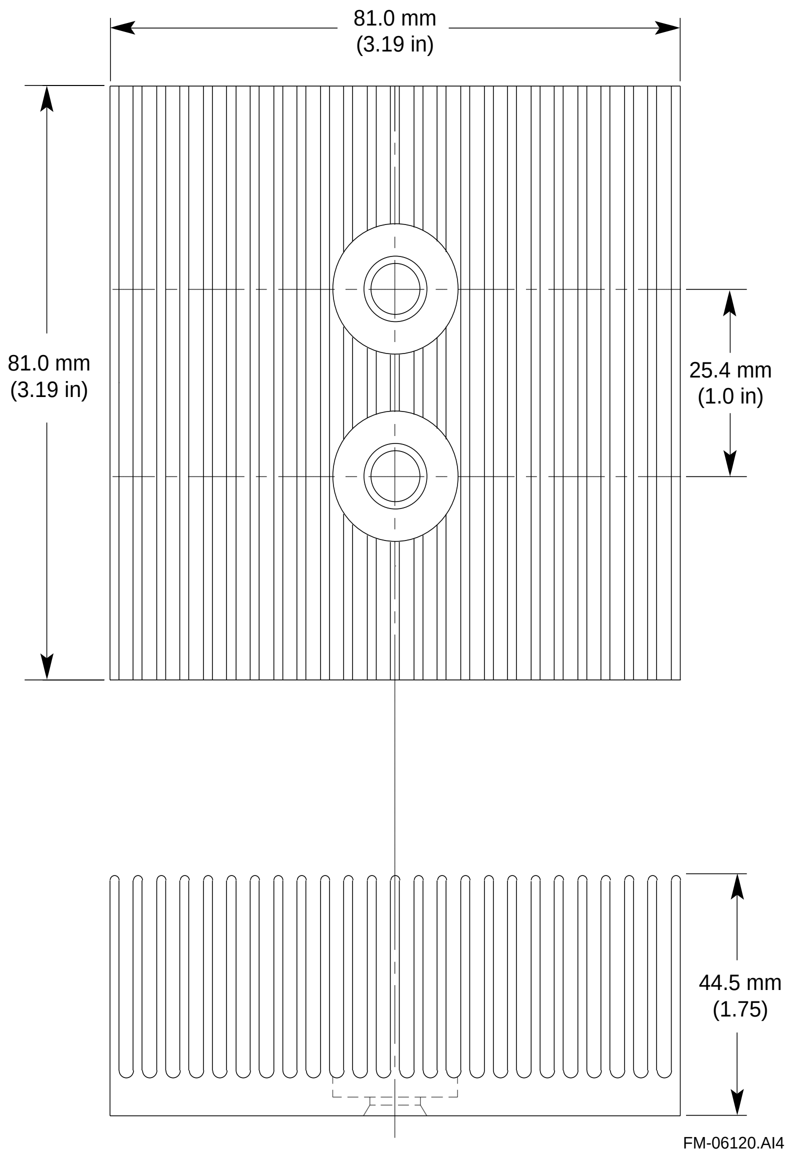
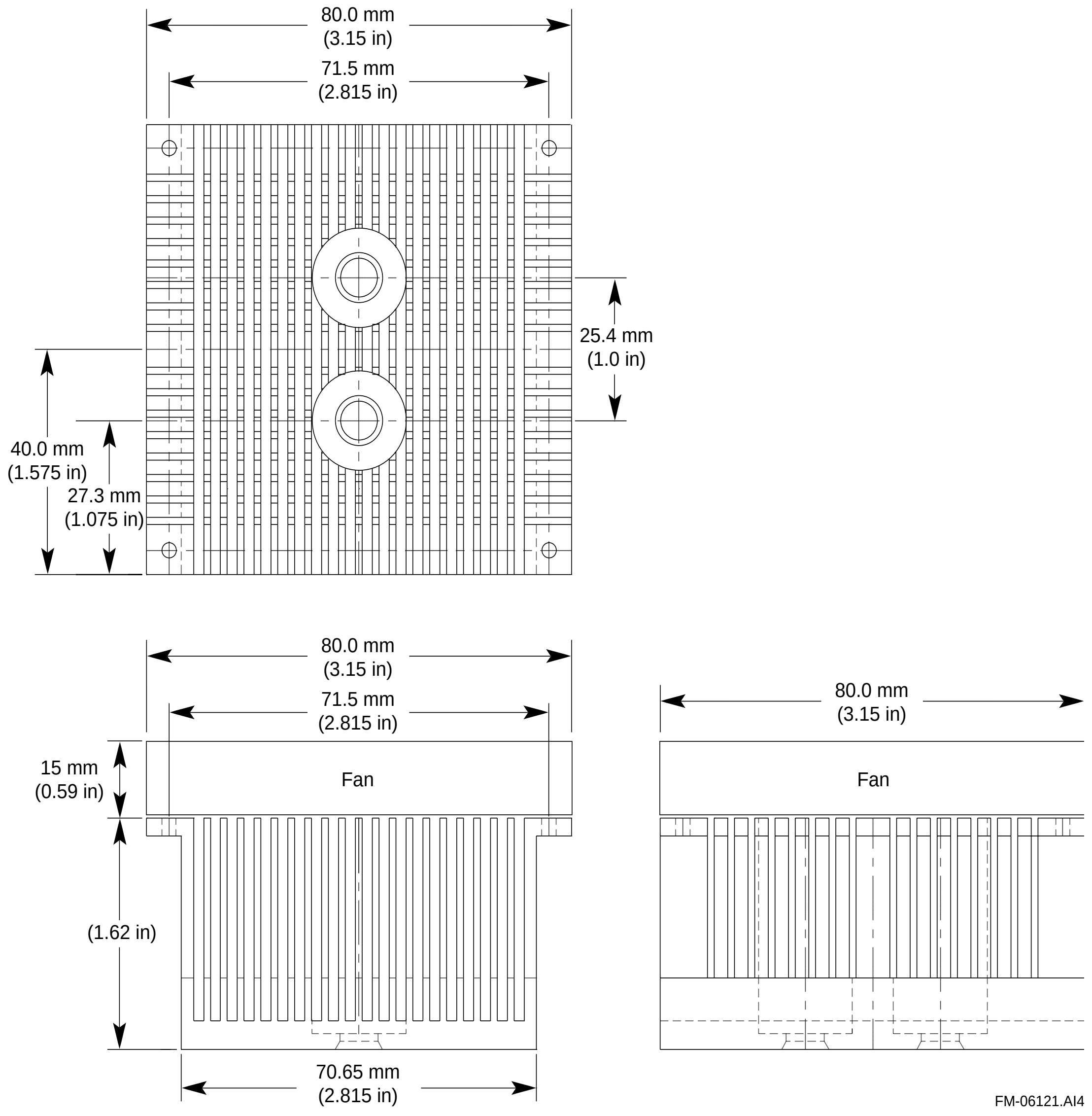


Figure 8–3 shows heat sink type 3, along with its approximate dimensions.

The cooling fins of heat sink type 3 are cross-cut. Also, an 80 mm × 80 mm × 15 mm fan is attached to heat sink type 3.

Figure 8–3 Type 3 Heat Sink



8.3 Thermal Design Considerations

Follow these guidelines for printed circuit board (PCB) component placement:

- Orient the 21264 on the PCB with the heat sink fins aligned with the airflow direction.
- Avoid preheating ambient air. Place the 21264 on the PCB so that inlet air is not preheated by any other PCB components.
- Do not place other high power devices in the vicinity of the 21264.

Do not restrict the airflow across the 21264 heat sink. Placement of other devices must allow for maximum system airflow in order to maximize the performance of the heat sink.

A
Alpha Instruction Set

This appendix provides a summary of the Alpha instruction set and describes the 21264 IEEE floating-point conformance. It is organized as follows:

- Alpha instruction summary
- Reserved opcodes
- IEEE floating-point instructions
- VAX floating-point instructions
- Independent floating-point instructions
- Opcode summary
- Required PALcode function codes
- IEEE floating-point conformance

A.1 Alpha Instruction Summary

This section contains a summary of all Alpha architecture instructions. All values are in hexadecimal radix. Table A–1 describes the contents of the Format and Opcode columns that are in Table A–2.

Table A–1 Instruction Format and Opcode Notation

Instruction Format	Format Symbol	Opcode Notation	Meaning
Branch	Bra	oo	*oo* is the 6-bit opcode field.
Floating-point	F-P	oo.fff	*oo* is the 6-bit opcode field . *fff* is the 11-bit function code field.
Memory	Mem	oo	*oo* is the 6-bit opcode field.
Memory/function code	Mfc	oo.ffff	*oo* is the 6-bit opcode field. *ffff* is the 16-bit function code in the displacement field.
Memory/ branch	Mbr	oo.h	*oo* is the 6-bit opcode field. *h* is the high-order 2 bits of the displacement field.

Table A–1 Instruction Format and Opcode Notation

Instruction Format	Format Symbol	Opcode Notation	Meaning
Operate	Opr	oo.ff	*oo* is the 6-bit opcode field. *ff* is the 7-bit function code field.
PALcode	Pcd	oo	*oo* is the 6-bit opcode field; the particular PALcode instruction is specified in the 26-bit function code field.

Qualifiers for operate instructions are shown in Table A–2. Qualifiers for IEEE and VAX floating-point instructions are shown in Tables A–5 and A–6, respectively.

Table A–2 Architecture Instructions

Mnemonic	Format	Opcode	Description
ADDF	F-P	15.080	Add F_floating
ADDG	F-P	15.0A0	Add G_floating
ADDL	Opr	10.00	Add longword
ADDL/V	—	10.40	—
ADDQ	Opr	10.20	Add quadword
ADDQ/V	—	10.60	—
ADDS	F-P	16.080	Add S_floating
ADDT	F-P	16.0A0	Add T_floating
AMASK	Opr	11.61	Architecture mask
AND	Opr	11.00	Logical product
BEQ	Bra	39	Branch if = zero
BGE	Bra	3E	Branch if ≥ zero
BGT	Bra	3F	Branch if > zero
BIC	Opr	11.08	Bit clear
BIS	Opr	11.20	Logical sum
BLBC	Bra	38	Branch if low bit clear
BLBS	Bra	3C	Branch if low bit set
BLE	Bra	3B	Branch if ≤ zero
BLT	Bra	3A	Branch if < zero
BNE	Bra	3D	Branch if ≠ zero
BR	Bra	30	Unconditional branch
BSR	Mbr	34	Branch to subroutine

Table A–2 Architecture Instructions (Continued)

Mnemonic	Format	Opcode	Description
CALL_PAL	Pcd	00	Trap to PALcode
CMOVEQ	Opr	11.24	CMOVE if = zero
CMOVGE	Opr	11.46	CMOVE if ≥ zero
CMOVGT	Opr	11.66	CMOVE if > zero
CMOVLBC	Opr	11.16	CMOVE if low bit clear
CMOVLBS	Opr	11.14	CMOVE if low bit set
CMOVLE	Opr	11.64	CMOVE if ≤ zero
CMOVLT	Opr	11.44	CMOVE if < zero
CMOVNE	Opr	11.26	CMOVE if ≠ zero
CMPBGE	Opr	10.0F	Compare byte
CMPEQ	Opr	10.2D	Compare signed quadword equal
CMPGEQ	F-P	15.0A5	Compare G_floating equal
CMPGLE	F-P	15.0A7	Compare G_floating less than or equal
CMPGLT	F-P	15.0A6	Compare G_floating less than
CMPLE	Opr	10.6D	Compare signed quadword less than or equal
CMPLT	Opr	10.4D	Compare signed quadword less than
CMPTEQ	F-P	16.0A5	Compare T_floating equal
CMPTLE	F-P	16.0A7	Compare T_floating less than or equal
CMPTLT	F-P	16.0A6	Compare T_floating less than
CMPTUN	F-P	16.0A4	Compare T_floating unordered
CMPULE	Opr	10.3D	Compare unsigned quadword less than or equal
CMPULT	Opr	10.1D	Compare unsigned quadword less than
CPYS	F-P	17.020	Copy sign
CPYSE	F-P	17.022	Copy sign and exponent
CPYSN	F-P	17.021	Copy sign negate
CVTDG	F-P	15.09E	Convert D_floating to G_floating
CVTGD	F-P	15.0AD	Convert G_floating to D_floating
CVTGF	F-P	15.0AC	Convert G_floating to F_floating
CVTGQ	F-P	15.0AF	Convert G_floating to quadword
CVTLQ	F-P	17.010	Convert longword to quadword
CVTQF	F-P	15.0BC	Convert quadword to F_floating

Table A–2 Architecture Instructions (Continued)

Mnemonic	Format	Opcode	Description
CVTQG	F-P	15.0BE	Convert quadword to G_floating
CVTQL	F-P	17.030	Convert quadword to longword
CVTQS	F-P	16.0BC	Convert quadword to S_floating
CVTQT	F-P	16.0BE	Convert quadword to T_floating
CVTST	F-P	16.2AC	Convert S_floating to T_floating
CVTTQ	F-P	16.0AF	Convert T_floating to quadword
CVTTS	F-P	16.0AC	Convert T_floating to S_floating
DIVF	F-P	15.083	Divide F_floating
DIVG	F-P	15.0A3	Divide G_floating
DIVS	F-P	16.083	Divide S_floating
DIVT	F-P	16.0A3	Divide T_floating
ECB	Mfc	18.E800	Evict cache block
EQV	Opr	11.48	Logical equivalence
EXCB	Mfc	18.0400	Exception barrier
EXTBL	Opr	12.06	Extract byte low
EXTLH	Opr	12.6A	Extract longword high
EXTLL	Opr	12.26	Extract longword low
EXTQH	Opr	12.7A	Extract quadword high
EXTQL	Opr	12.36	Extract quadword low
EXTWH	Opr	12.5A	Extract word high
EXTWL	Opr	12.16	Extract word low
FBEQ	Bra	31	Floating branch if = zero
FBGE	Bra	36	Floating branch if ≥ zero
FBGT	Bra	37	Floating branch if > zero
FBLE	Bra	33	Floating branch if ≤ zero
FBLT	Bra	32	Floating branch if < zero
FBNE	Bra	35	Floating branch if ≠ zero
FCMOVEQ	F-P	17.02A	FCMOVE if = zero
FCMOVGE	F-P	17.02D	FCMOVE if ≥ zero
FCMOVGT	F-P	17.02F	FCMOVE if > zero
FCMOVLE	F-P	17.02E	FCMOVE if ≤ zero
FCMOVLT	F-P	17.02C	FCMOVE if < zero

Table A–2 Architecture Instructions (Continued)

Mnemonic	Format	Opcode	Description
FCMOVNE	F-P	17.02B	FCMOVE if ≠ zero
FETCH	Mfc	18.8000	Prefetch data
FETCH_M	Mfc	18.A000	Prefetch data, modify intent
FTOIS	F-P	1C.78	Floating to integer move, S_floating
FTOIT	F-P	1C.70	Floating to integer move, T_floating
IMPLVER	Opr	11.6C	Implementation version
INSBL	Opr	12.0B	Insert byte low
INSLH	Opr	12.67	Insert longword high
INSLL	Opr	12.2B	Insert longword low
INSQH	Opr	12.77	Insert quadword high
INSQL	Opr	12.3B	Insert quadword low
INSWH	Opr	12.57	Insert word high
INSWL	Opr	12.1B	Insert word low
ITOFF	F-P	14.014	Integer to floating move, F_floating
ITOFS	F-P	14.004	Integer to floating move, S_floating
ITOFT	F-P	14.024	Integer to floating move, T_floating
JMP	Mbr	1A.0	Jump
JSR	Mbr	1A.1	Jump to subroutine
JSR_COROUTINE	Mbr	1A.3	Jump to subroutine return
LDA	Mem	08	Load address
LDAH	Mem	09	Load address high
LDBU	Mem	0A	Load zero-extended byte
LDF	Mem	20	Load F_floating
LDG	Mem	21	Load G_floating
LDL	Mem	28	Load sign-extended longword
LDL_L	Mem	2A	Load sign-extended longword locked
LDQ	Mem	29	Load quadword
LDQ_L	Mem	2B	Load quadword locked
LDQ_U	Mem	0B	Load unaligned quadword
LDS	Mem	22	Load S_floating
LDT	Mem	23	Load T_floating
LDWU	Mem	0C	Load zero-extended word

Table A–2 Architecture Instructions (Continued)

Mnemonic	Format	Opcode	Description
MAXSB8	Opr	1C.3E	Vector signed byte maximum
MAXSW4	Opr	1C.3F	Vector signed word maximum
MAXUB8	Opr	1C.3C	Vector unsigned byte maximum
MAXUW4	Opr	1C.3D	Vector unsigned word maximum
MB	Mfc	18.4000	Memory barrier
MF_FPCR	F-P	17.025	Move from FPCR
MINSB8	Opr	1C.38	Vector signed byte minimum
MINSW4	Opr	1C.39	Vector signed word minimum
MINUB8	Opr	1C.3A	Vector unsigned byte minimum
MINUW4	Opr	1C.3B	Vector unsigned word minimum
MSKBL	Opr	12.02	Mask byte low
MSKLH	Opr	12.62	Mask longword high
MSKLL	Opr	12.22	Mask longword low
MSKQH	Opr	12.72	Mask quadword high
MSKQL	Opr	12.32	Mask quadword low
MSKWH	Opr	12.52	Mask word high
MSKWL	Opr	12.12	Mask word low
MT_FPCR	F-P	17.024	Move to FPCR
MULF	F-P	15.082	Multiply F_floating
MULG	F-P	15.0A2	Multiply G_floating
MULL	Opr	13.00	Multiply longword
MULL/V		13.40	
MULQ	Opr	13.20	Multiply quadword
MULQ/V		13.60	
MULS	F-P	16.082	Multiply S_floating
MULT	F-P	16.0A2	Multiply T_floating
ORNOT	Opr	11.28	Logical sum with complement
PERR	Opr	1C.31	Pixel error
PKLB	Opr	1C.37	Pack longwords to bytes
PKWB	Opr	1C.36	Pack words to bytes
RC	Mfc	18.E000	Read and clear
RET	Mbr	1A.2	Return from subroutine

Table A–2 Architecture Instructions (Continued)

Mnemonic	Format	Opcode	Description
RPCC	Mfc	18.C000	Read process cycle counter
RS	Mfc	18.F000	Read and set
S4ADDL	Opr	10.02	Scaled add longword by 4
S4ADDQ	Opr	10.22	Scaled add quadword by 4
S4SUBL	Opr	10.0B	Scaled subtract longword by 4
S4SUBQ	Opr	10.2B	Scaled subtract quadword by 4
S8ADDL	Opr	10.12	Scaled add longword by 8
S8ADDQ	Opr	10.32	Scaled add quadword by 8
S8SUBL	Opr	10.1B	Scaled subtract longword by 8
S8SUBQ	Opr	10.3B	Scaled subtract quadword by 8
SEXTB	Opr	1C.00	Sign extend byte
SEXTW	Opr	1C.01	Sign extend word
SLL	Opr	12.39	Shift left logical
SQRTF	F-P	14.08A	Square root F_floating
SQRTG	F-P	14.0AA	Square root G_floating
SQRTS	F-P	14.08B	Square root S_floating
SQRTT	F-P	14.0AB	Square root T_floating
SRA	Opr	12.3C	Shift right arithmetic
SRL	Opr	12.34	Shift right logical
STB	Mem	0E	Store byte
STF	Mem	24	Store F_floating
STG	Mem	25	Store G_floating
STL	Mem	2C	Store longword
STL_C	Mem	2E	Store longword conditional
STQ	Mem	2D	Store quadword
STQ_C	Mem	2F	Store quadword conditional
STQ_U	Mem	0F	Store unaligned quadword
STS	Mem	26	Store S_floating
STT	Mem	27	Store T_floating
STW	Mem	0D	Store word
SUBF	F-P	15.081	Subtract F_floating
SUBG	F-P	15.0A1	Subtract G_floating

Table A–2 Architecture Instructions (Continued)

Mnemonic	Format	Opcode	Description
SUBL	Opr	10.09	Subtract longword
SUBL/V		10.49	
SUBQ	Opr	10.29	Subtract quadword
SUBQ/V		10.69	
SUBS	F-P	16.081	Subtract S_floating
SUBT	F-P	16.0A1	Subtract T_floating
TRAPB	Mfc	18.0000	Trap barrier
UMULH	Opr	13.30	Unsigned multiply quadword high
UNPKBL	Opr	1C.35	Unpack bytes to longwords
UNPKBW	Opr	1C.34	Unpack bytes to words
WH64	Mfc	18.F800	Write hint — 64 bytes
WMB	Mfc	18.4400	Write memory barrier
XOR	Opr	11.40	Logical difference
ZAP	Opr	12.30	Zero bytes
ZAPNOT	Opr	12.31	Zero bytes not

A.2 Reserved Opcodes

This section describes the opcodes that are reserved in the Alpha architecture. They can be reserved for Compaq or for PALcode.

A.2.1 Opcodes Reserved for Compaq

Table A–3 lists opcodes reserved for Compaq.

Table A–3 Opcodes Reserved for Compaq

Mnemonic	Opcode	Mnemonic	Opcode
OPC01	01	OPC05	05
OPC02	02	OPC06	06
OPC03	03	OPC07	07
OPC04	04	—	—

A.2.2 Opcodes Reserved for PALcode

Table A–4 lists the 21264-specific instructions. See Chapter 2 for more information.

Table A–4 Opcodes Reserved for PALcode

21264 Mnemonic	Opcode	Architecture Mnemonic	Function
HW_LD	1B	PAL1B	Performs Dstream load instructions.
HW_ST	1F	PAL1F	Performs Dstream store instructions.
HW_REI	1E	PAL1E	Returns instruction flow to the program counter (PC) pointed to by EXC_ADDR internal processor register (IPR).
HW_MFPR	19	PAL19	Accesses the Ibox, Mbox, and Dcache IPRs.
HW_MTPR	1D	PAL1D	Accesses the Ibox, Mbox, and Dcache IPRs.

A.3 IEEE Floating-Point Instructions

Table A–5 lists the hexadecimal value of the 11-bit function code field for the IEEE floating-point instructions, with and without qualifiers. The opcode for these instructions is 16_{16}.

Table A–5 IEEE Floating-Point Instruction Function Codes

Mnemonic	None	/C	/M	/D	/U	/UC	/UM	/UD
ADDS	080	000	040	0C0	180	100	140	1C0
ADDT	0A0	020	060	0E0	1A0	120	160	1E0
CMPTEQ	0A5	—	—	—	—	—	—	—
CMPTLT	0A6	—	—	—	—	—	—	—
CMPTLE	0A7	—	—	—	—	—	—	—
CMPTUN	0A4	—	—	—	—	—	—	—
CVTQS	0BC	03C	07C	0FC	—	—	—	—
CVTQT	0BE	03E	07E	0FE	—	—	—	—
CVTST	See below	—	—	—	—	—	—	—
CVTTQ	See below	—	—	—	—	—	—	—
CVTTS	0AC	02C	06C	0EC	1AC	12C	16C	1EC
DIVS	083	003	043	0C3	183	103	143	1C3
DIVT	0A3	023	063	0E3	1A3	123	163	1E3
MULS	082	002	042	0C2	182	102	142	1C2

Table A–5 IEEE Floating-Point Instruction Function Codes (Continued)

MULT	0A2	022	062	0E2	1A2	122	162	1E2
SQRTS	08B	00B	04B	0CB	18B	10B	14B	1CB
SQRTT	0AB	02B	06B	0EB	1AB	12B	16B	1EB
SUBS	081	001	041	0C1	181	101	141	1C1
SUBT	0A1	021	061	0E1	1A1	121	161	1E1

Mnemonic	/SU	/SUC	/SUM	/SUD	/SUI	/SUIC	/SUIM	/SUID
ADDS	580	500	540	5C0	780	700	740	7C0
ADDT	5A0	520	560	5E0	7A0	720	760	7E0
CMPTEQ	5A5							
CMPTLT	5A6							
CMPTLE	5A7							
CMPTUN	5A4							
CVTQS					7BC	73C	77C	7FC
CVTQT					7BE	73E	77E	7FE
CVTTS	5AC	52C	56C	5EC	7AC	72C	76C	7EC
DIVS	583	503	543	5C3	783	703	743	7C3
DIVT	5A3	523	563	5E3	7A3	723	763	7E3
MULS	582	502	542	5C2	782	702	742	7C2
MULT	5A2	522	562	5E2	7A2	722	762	7E2
SQRTS	58B	50B	54B	5CB	78B	70B	74B	7CB
SQRTT	5AB	52B	56B	5EB	7AB	72B	76B	7EB
SUBS	581	501	541	5C1	781	701	741	7C1
SUBT	5A1	521	561	5E1	7A1	721	761	7E1

Mnemonic	None	/S
CVTST	2AC	6AC

Mnemonic	None	/C	/V	/VC	/SV	/SVC	/SVI	/SVIC
CVTTQ	0AF	02F	1AF	12F	5AF	52F	7AF	72F

Mnemonic	D	/VD	/SVD	/SVID	/M	/VM	/SVM	/SVIM
CVTTQ	0EF	1EF	5EF	7EF	06F	16F	56F	76F

Programming Note:

In order to use CMPTxx with software completion trap handling, it is necessary to specify the /SU IEEE trap mode, even though an underflow trap is not possible. In order to use CVTQS or CVTQT with software completion trap handling, it is necessary to specify the /SUI IEEE trap mode, even though an underflow trap is not possible.

A.4 VAX Floating-Point Instructions

Table A–6 lists the hexadecimal value of the 11-bit function code field for the VAX floating-point instructions. The opcode for these instructions is 15_{16}.

Table A–6 VAX Floating-Point Instruction Function Codes

Mnemonic	None	/C	/U	/UC	/S	/SC	/SU	/SUC
ADDF	080	000	180	100	480	400	580	500
ADDG	0A0	020	1A0	120	4A0	420	5A0	520
CMPGEQ	0A5				4A5			
CMPGLE	0A7				4A7			
CMPGLT	0A6				4A6			
CVTDG	09E	01E	19E	11E	49E	41E	59E	51E
CVTGD	0AD	02D	1AD	12D	4AD	42D	5AD	52D
CVTGF	0AC	02C	1AC	12C	4AC	42C	5AC	52C
CVTGQ	See below							
CVTQF	0BC	03C						
CVTQG	0BE	03E						
DIVF	083	003	183	103	483	403	583	503
DIVG	0A3	023	1A3	123	4A3	423	5A3	523
MULF	082	002	182	102	482	402	582	502
MULG	0A2	022	1A2	122	4A2	422	5A2	522
SQRTF	08A	00A	18A	10A	48A	40A	58A	50A
SQRTG	0AA	02A	1AA	12A	4AA	42A	5AA	52A
SUBF	081	001	181	101	481	401	581	501
SUBG	0A1	021	1A1	121	4A1	421	5A1	521
Mnemonic	**None**	**/C**	**/V**	**/VC**	**/S**	**/SC**	**/SV**	**/SVC**
CVTGQ	0AF	02F	1AF	12F	4AF	42F	5AF	52F

A.5 Independent Floating-Point Instructions

Table A–7 lists the hexadecimal value of the 11-bit function code field for the floating-point instructions that are not directly tied to IEEE or VAX floating point. The opcode for the following instructions is 17_{16}.

Table A–7 Independent Floating-Point Instruction Function Codes

Mnemonic	None	/V	/SV
CPYS	020	—	—
CPYSE	022	—	—
CPYSN	021	—	—
CVTLQ	010	—	—
CVTQL	030	130	530
FCMOVEQ	02A	—	—
FCMOVGE	02D	—	—
FCMOVGT	02F	—	—
FCMOVLE	02E	—	—
FCMOVLT	02C	—	—
MF_FPCR	025	—	—
MT_FPCR	024	—	—

A.6 Opcode Summary

Table A–8 lists all Alpha opcodes from 00 (CALL_PAL) through 3F (BGT). In the table, the column headings that appear over the instructions have a granularity of 8_{16}. The rows beneath the Offset column supply the individual hexadecimal number to resolve that granularity.

If an instruction column has a 0 in the right (low) hexadecimal digit, replace that 0 with the number to the left of the backslash (\) in the Offset column on the instruction’s row. If an instruction column has an 8 in the right (low) hexadecimal digit, replace that 8 with the number to the right of the backslash in the Offset column.

For example, the third row (2/A) under the 10_{16} column contains the symbol INTS*, representing the all-integer shift instructions. The opcode for those instructions would then be 12_{16} because the 0 in 10 is replaced by the 2 in the Offset column. Likewise, the third row under the 18_{16} column contains the symbol JSR*, representing all jump instructions. The opcode for those instructions is 1A because the 8 in the heading is replaced by the number to the right of the backslash in the Offset column. The instruction format is listed under the instruction symbol..

Table A–8 Opcode Summary

Offset	00	08	10	18	20	28	30	38
0/8	PAL* (pal)	LDA (mem)	INTA* (op)	MISC* (mem)	LDF (mem)	LDL (mem)	BR (br)	BLBC (br)
1/9	Res	LDAH (mem)	INTL* (op)	\PAL\	LDG (mem)	LDQ (mem)	FBEQ (br)	BEQ (br)
2/A	LDBU	Res	INTS* (op)	JSR* (mem)	LDS (mem)	LDL_L (mem)	FBLT (br)	BLT (br)

Table A–8 Opcode Summary (Continued)

Offset	00	08	10	18	20	28	30	38
3/B	Res	LDQ_U (mem)	INTM* (op)	\PAL\	LDT (mem)	LDQ_L (mem)	FBLE (br)	BLE (br)
4/C	LDWU	Res	ITFP*	FPTI*	STF (mem)	STL (mem)	BSR (br)	BLBS (br)
5/D	Res	STW	FLTV* (op)	\PAL\	STG (mem)	STQ (mem)	FBNE (br)	BNE (br)
6/E	Res	STB	FLTI* (op)	\PAL\	STS (mem)	STL_C (mem)	FBGE (br)	BGE (br)
7/F	Res	STQ_U (mem)	FLTL* (op)	\PAL\	STT (mem)	STQ_C (mem)	FBGT (br)	BGT (br)

Table A–9 explains the symbols used in Table A–8.

Table A–9 Key to Opcode Summary Used in Table A–8

Symbol	Meaning
FLTI*	IEEE floating-point instruction opcodes
FLTL*	Floating-point Operate instruction opcodes
FLTV*	VAX floating-point instruction opcodes
FPTI*	Floating-point to integer register move opcodes
INTA*	Integer arithmetic instruction opcodes
INTL*	Integer logical instruction opcodes
INTM*	Integer multiply instruction opcodes
INTS*	Integer shift instruction opcodes
ITFP*	Integer to floating-point register move opcodes
JSR*	Jump instruction opcodes
MISC*	Miscellaneous instruction opcodes
PAL*	PALcode instruction (CALL_PAL) opcodes
\PAL\	Reserved for PALcode
Res	Reserved for Compaq

A.7 Required PALcode Function Codes

Table A–10 lists opcodes required for all Alpha implementations. The notation used is *oo.ffff*, where *oo* is the hexadecimal 6-bit opcode and *ffff* is the hexadecimal 26-bit function code.

Table A–10 Required PALcode Function Codes

Mnemonic	Type	Function Code
DRAINA	Privileged	00.0002
HALT	Privileged	00.0000
IMB	Unprivileged	00.0086

A.8 IEEE Floating-Point Conformance

The 21264 supports the IEEE floating-point operations defined in the *Alpha System Reference Manual, Revision* 7 and therefore also from the *Alpha Architecture Handbook, Version 4*. Support for a complete implementation of the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Standard 754 1985) is provided by a combination of hardware and software. The 21264 provides several hardware features to facilitate complete support of the IEEE standard.

The 21264 provides the following hardware features to facilitate complete support of the IEEE standard:

- The 21264 implements precise exception handling in hardware, as denoted by the AMASK instruction returning bit 9 set. TRAPB instructions are treated as NOPs and are not issued.
- The 21264 accepts both Signaling and Quiet NaNs as input operands and propagates them as specified by the Alpha architecture. In addition, the 21264 delivers a canonical Quiet NaN when an operation is required to produce a NaN value and none of its inputs are NaNs. Encodings for Signaling NaN and Quiet NaN are defined by the *Alpha Architecture Handbook, Version 4*.
- The 21264 accepts infinity operands and implements infinity arithmetic as defined by the IEEE standard and the *Alpha Architecture Handbook, Version 4*.
- The 21264 implements SQRT for single (SQRTS) and double (SQRTT) precision in hardware.

Note: In addition, the 21264 also implements the VAX SQRTF and SQRTG instructions.

- The 21264 implements the FPCR[DNZ] bit. When FPCR[DNZ] is set, denormal input operand traps can be avoided for arithmetic operations that include the /S qualifier. When FPCR[DNZ] is clear, denormal input operands for arithmetic operations produce an unmaskable denormal trap. CPYSE/CPYSN, FCMOVxx, and MF_FPCR/MT_FPCR are not arithmetic operations, and pass denormal values without initiating arithmetic traps.
- The 21264 implements the following disable bits in the floating-point control register (FPCR):
  - Underflow disable (UNFD)
  - Overflow disable (OVFD)
  - Inexact result disable (INED)
  - Division by zero disable (DZED)
  - Invalid operation disable (INVD)

  If one of these bits is set, and an instruction with the /S qualifier set generates the associated exception, the 21264 produces the IEEE nontrapping result and suppresses the trap. These nontrapping responses include correctly signed infinity, largest finite number, and Quiet NaNs as specified by the IEEE standard.

The 21264 will not produce a Denormal result for the underflow exception. Instead, a true zero (+0) is written to the destination register. In the 21264 the FPCR underflow to zero (UNDZ) bit must be set if underflow disable (UNFD) bit is set. If desired, trapping on underflow can be enabled by the instruction and the FPCR, and software may compute the Denormal value as defined in the IEEE Standard.

The 21264 records floating-point exception information in two places:

- The FPCR status bits record the occurence of all exceptions that are detected, whether or not the corresponding trap is enabled. The status bits are cleared only through an explicit clear command (MT_FPCR); hence, the exception information they record is a summary of all exceptions that have occurred since the last time they were cleared.
- If an exception is detected and the corresponding trap is enabled by the instruction, and is not disabled by the FPCR control bits, the 21264 will record the condition in the EXC_SUM register and initiate an arithmetic trap.

The following items apply to Table A–11:

- The 21264 traps on a Denormal input operand for all arithmetic operations unless FPCR[DNZ] = 1.
- Input operand traps take precedence over arithmetic result traps.
- The following abbreviations are used:

  Inf: Infinity

  QNaN: Quiet NaN

  SNaN: Signalling NaN

  CQNaN: Canonical Quiet NaN

  For IEEE instructions with /S, Table A–11 lists all exceptional input and output conditions recognized by the 21264, along with the result and exception generated for each condition.

Table A–11 Exceptional Input and Output Conditions

Alpha Instructions	21264 Hardware Supplied Result	Exception
ADDx SUBx INPUT		
Inf operand	±Inf	(none)
QNaN operand	QNaN	(none)
SNaN operand	QNaN	Invalid Op
Effective subtract of two Inf operands	CQNaN	Invalid Op
ADDx SUBx OUTPUT		
Exponent overflow	±Inf or ±MAX	Overflow
Exponent underflow	+0	Underflow
Inexact result	Result	Inexact
MULx INPUT		

Table A–11 Exceptional Input and Output Conditions (Continued)

Alpha Instructions	21264 Hardware Supplied Result	Exception
Inf operand	±Inf	(none)
QNaN operand	QNaN	(none)
SNaN operand	QNaN	Invalid Op
0 * Inf	CQNaN	Invalid Op
MULx OUTPUT (same as ADDx)		
DIVx INPUT		
QNaN operand	QNaN	(none)
SNaN operand	QNaN	Invalid Op
0/0 or Inf/Inf	CQNaN	Invalid Op
A/0 (A not 0)	±Inf	Div Zero
A/Inf	±0	(none)
Inf/A	±Inf	(none)
DIVx OUTPUT (same as ADDx)		
SQRTx INPUT		
+Inf operand	+Inf	(none)
QNaN operand	QNaN	(none)
SNaN operand	QNaN	Invalid Op
-A (A not 0)	CQNaN	Invalid Op
-0	-0	(none)
SQRTx OUTPUT		
Inexact result	root	Inexact
CMPTEQ CMPTUN INPUT		
Inf operand	True or False	(none)
QNaN operand	False for EQ, True for UN	(none)
SNaN operand	False for EQ,True for UN	Invalid Op
CMPTLT CMPTLE INPUT		
Inf operand	True or False	(none)
QNaN operand	False	Invalid Op
SNaN operand	False	Invalid Op
CVTfi INPUT		
Inf operand	0	Invalid Op
QNaN operand	0	Invalid Op

Table A–11 Exceptional Input and Output Conditions (Continued)

Alpha Instructions	21264 Hardware Supplied Result	Exception
SNaN operand	0	Invalid Op
CVTfi OUTPUT		
Inexact result	Result	Inexact
Integer overflow	Truncated result	Invalid Op
CVTif OUTPUT		
Inexact result	Result	Inexact
CVTff INPUT		
Inf operand	±Inf	(none)
QNaN operand	QNaN	(none)
SNaN operand	QNaN	Invalid Op
CVTff OUTPUT (same as ADDx)		
FBEQ FBNE FBLT FBLE FBGT FBGE LDS LDT STS STT CPYS CPYSN FCMOVx		

See Section 2.3 for information about the floating-point control register (FPCR).

B
Products and Documentation

To view current product update and errata revision information, visit the Alpha OEM World Wide Web Internet site. You can also visit this website for information about other Alpha microprocessors or help with deciding which documentation best meets your needs:

http://www.digital.com/alphaoem/

- For documentation needs, click on **Technical Information**.
- For product update information or information about other Alpha microprocessors, click on **Microprocessor Products**.
- For contact information and sales offices, click on **Customer Support.**

Alpha Products

Microprocessor	Order Number
Alpha 21264 500-MHz microprocessor	21264-A1

Alpha Documentation

The following table lists some of the Alpha documentation that is available at the Alpha OEM website, listed above:

Title	Order Number
Alpha Architecture Reference Manual[1]	EY–W938E–DP
Alpha Architecture Handbook	EC–QD2KC–TE
Alpha 21264 Microprocessor Product Brief	EC–R2YTB–TE
AlphaPC 264DP Product Brief	EC-RBD0A-TE

[1] To purchase the *Alpha Architecture Reference Manual,* contact your local technical bookstore or call Butterworth-Heinemann (Digital Press) at 1-800-366-2665.

Index