PB22H-KB System Module

Hardware Reference Information

Order Number: EK-A0638-TD.001

14 July, 1993

This manual gives hardware reference information for the PB22H-KB system module.

Revision Information: Final draft.

Digital Equipment Corporation
Maynard, Massachusetts

Sign-off Draft, 14 July, 1993

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

2 Backup Cache

3 Lock Logic

4 System Module Memory

5 System Registers

6 Exceptions and Interrupts

7 Direct Memory Access

8 Local Buses

9 Error Handling

10 Power-Up Initialization

Part II DECchip 21064 CPU Overview

11 Alpha AXP Architecture

12 I-Box Internal Processor Registers

13 A-Box Internal Processor Registers

14 PAL Temporary Registers

15 CPU Cycle Types, Transactions, and Initialization

Part III Intel 82357 Integrated System peripheral Chip Functions

16 DMA Controller

17 Interrupt Controller

18 Nonmaskable Interrupt Ports

19 Interval Timer

Part IV VLSI Technology VL82C106 Combination Chip Functions

20 Serial Communications Ports

21 Line Printer Port

22 Real-Time Clock

23 Keyboard Controller

24 Chip Select Registers

Part V Appendixes

A System I/O Map

B Connector Pin Specifications

Glossary

Index

Examples

Figures

Tables

Preface

Purpose

This manual describes the chip sets and registers of the PB22H-KB system module. Use this manual with the *Alpha™ AXP™ Architecture Reference Manual* as a hardware reference to the PB22H-KB system module.

Audience

This manual is for design engineers and systems programmers who develop systems that use the PB22H-KB system module.

Structure of This Manual

This manual is divided into 5 parts, a glossary, and an index.

- Part I gives an overview of the PB22H-KB system module.
- Part II describes the DECchip™ 21064 CPU data types, registers and functions.
- Part II describes the registers and functions of the Intel 82357 integrated system peripheral (ISP) chip.
- Part IV describes the functions and registers of the VLSI Technology VL82C106 combination chip.
- Part V describes technical and other information about the PB22H-KB system module.
- The glossary defines the terms used in this manual.

Conventions

The following conventions are used in this manual:

Convention	Description
<x:y>	Represents a bit field, or an extent of a set of lines or signals ranging from x through y. For example, R0 <7:4> indicates bits 4 through 7 in the general purpose register R0.
x..y	Represents a range of bits from x to y.
n, n_{16}, n_2	Numbers are decimal unless otherwise marked with a subscript number. If there is ambiguity, the radix is explicitly stated.
2.0123.FFFF	Nine digit numbers typically represent 34-bit hexadecimal addresses and are grouped in four-digit clusters, separated by periods.
Note	A note contains information that might be of special importance to the user.
Caution	A caution contains information that the user needs to know to avoid damaging the software or hardware.
n	Boldface small n indicates a variable.
{}	Represents a console command element.
[ ]	Represents a console command element that is optional.
...	Horizontal ellipsis points represent a list command element.
SIGNAL(H)	Signal names are shown in small capitals and follow the conventions in the ANSI /IEEE Standard 991-1986 publication entitled *IEEE Standard for Logic Circuit Diagrams*.

Table 1 lists the conventions for naming bits.

Table 1 Bit Name Conventions

Bit Name	Description
0	Denotes a bit that is ignored on write operations and is read as 0.
1	Denotes a bit that is ignored on write operations and is read as 1.
R/W	Read/write. A bit or field that may be read or written by software.
RO	Read-only. A read-only bit that can be read by software. It is written by hardware. Software writes are ignored.
WO	Write-only. A write-only bit that can be written by software. It is used by hardware. Reads by software return unpredictable results.
W	A write bit that can be written by software. It is used by hardware. Reads by software return a 0.
WC	Write-one-to-clear. Software writes of a 1 cause this bit to be cleared by hardware. Software writes of a 0 do not modify the state of the bit.
W0C	Write-zero-to-clear. Software writes of a 0 cause this bit to be cleared by hardware. Software writes of a 1 do not modify the state of the bit.
WA	Write-anything. Software writes of any value to the register cause the bit to be cleared by hardware.
RC	Read-to-clear. The value is written by hardware and remains unchanged until read by software, at which point, hardware may write a new value into the field.
IGN	Ignored. These bits or fields are ignored when written.
RAZ	Read-as-zero. These bits or fields return a 0 when read.

(continued on next page)

Table 1 (Cont.) Bit Name Conventions

Bit Name	Description
MBZ	Must-be-zero. These bits or fields must never be written by software with a non-zero value. A reserved operand exception occurs if a non-zero value in an MBZ field is encountered by the processor.
SBZ	Should-be-zero. These bits or fields should be filled by software with a zero value. These fields may be used at a future time. Nonzero values in SBZ fields produce unpredictable results.
RES	Reserved. These bits or fields are reserved for future expansion.
X	A don't care bit. The value of don't care bits is ignored.

Related Documents

The following documents contain related information:

- *Alpha AXP Architecture Handbook*, EC-H1689-10
- *Alpha AXP Architecture Reference Manual*, EK-VAXAR-RM
- *DECchip 21064 RISC Microprocessor User Guide*, EK-21064-UG
- *Intel Peripheral Components*
- *VLSI Technology VL16C450 Asynchronous Communications Element Data Sheet*
- *VLSI Technology VL82C106 Combination Chip Data Sheet*

Part I

System Module Overview

Part I provides an overview of the PB22H-KB system module, its components and functions.

This part includes the following chapters.

- Chapter 1, System Module Overview
- Chapter 2, Backup Cache
- Chapter 3, Lock Logic
- Chapter 4, System Module Memory
- Chapter 5, System Registers
- Chapter 6, Exceptions and Interrupts
- Chapter 7, Direct Memory Access
- Chapter 8, Local Buses
- Chapter 9, Error Handling
- Chapter 10, Power-Up Initialization

1

System Module Overview

This chapter contains a technical description of the system module. It contains the following sections:

- Overview
- DECchip 21064 CPU
- Intel 82350DT EISA Chip Set
- VLSI Technology VL82C106 Combination Chip

Overview

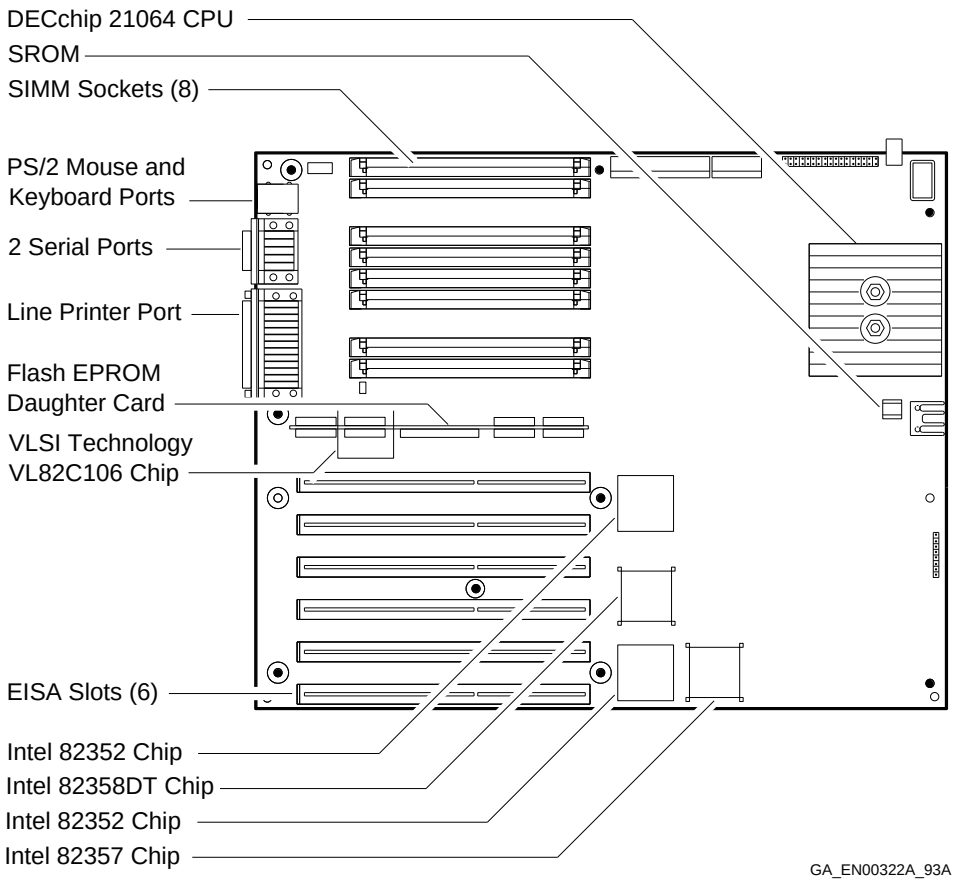
The PB22H-KB system module contains the following components:

- A DECchip 21064 64-bit RISC microprocessor
- 512K-byte write-back backup cache
- Lock logic
- Main memory ranging from 16M bytes to 128M bytes using industry standard single in-line memory modules (SIMMs)
- An extended industry standard architecture (EISA) bus interface using a subset of the Intel 82350DT EISA chip set
- DMA logic that supports full burst mode
- Interrupt logic
- Local buses (H_BUS and L_BUS)
- Six full size EISA bus option slots, each with full bus master capability
- Battery-backed memory
- Firmware in Flash EPROM (FEPROM)
- A real-time clock (RTC)
- An interval timer
- A PS/2™ compatible keyboard port
- A PS/2 compatible mouse port
- Two serial ports
- A line (parallel) printer port
- LED diagnostics display

System Module Layout

Figure 1–1 shows how the components are arranged on the PB22H-KB system module.

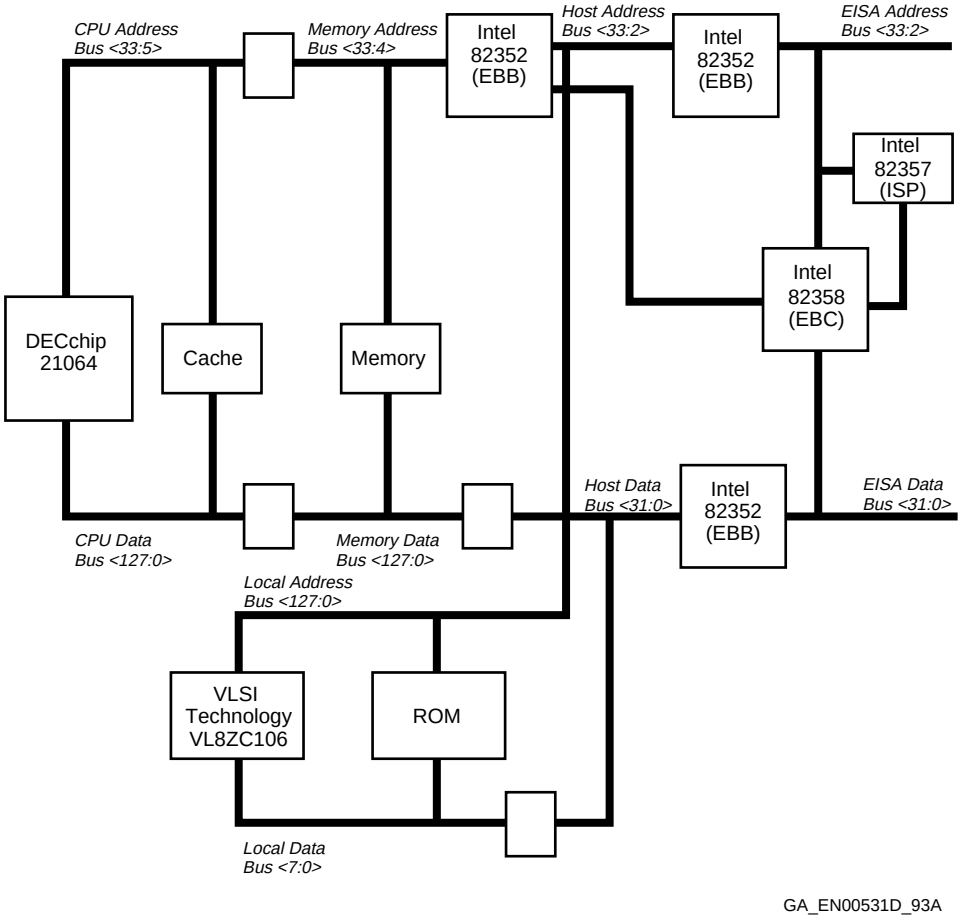
Figure 1–1 Component Layout



Block Diagram

Figure 1–2 shows a block diagram of the system module.

Figure 1–2 PB22H-KB System Module Block Diagram



DECchip 21064 CPU

Summary

The PB22H-KB system module uses the DECchip 21064 64-bit RISC microprocessor as the central processing unit (CPU). The DECchip 21064 CPU is a superscalar, superpipelined implementation of the 64-bit Alpha AXP architecture.

DECchip 21064 CPU Features

The DECchip 21064 CPU includes the following features:

- Supports the following Alpha AXP architecture instruction and data types:
  - Byte
  - Word
  - Longword
  - Quadword
  - Digital floating point data types (F_floating, D_floating, and G_floating)
  - IEEE floating point data types (S_floating and T_ floating)
- Contains a demand-page memory management unit that with properly written privileged architecture library code (PAL code), which is stored in firmware Flash EPROMs, fully implements the Alpha AXP memory management architecture. The translation buffer can be used with alternative PAL code to implement a different page table structure.
- Contains an on-chip, 8-entry, instruction-stream translation buffer for mapping 8K-byte physical pages and a 4-entry instruction stream translation buffer for mapping groups of up to 512 contiguous 8K-byte pages. It also contains a 32-entry D-stream translation buffer for mapping 8K-byte physical pages, and a 4-entry data stream translation buffer for mapping aligned groups of 512 contiguous 8K-byte pages.
- Implements two dynamic branch prediction algorithms using a 2K bytes x 1-bit branch history table. An internal control register bit selects one of these two algorithms.

- Contains an integer execution unit that supports scaled add instructions that improve the performance of address calculations for longword-length and quadword-length array elements.
- Uses a 6.6 nanoseconds (ns) cycle time at the DECchip 21064 CPU's nominal frequency. The DECchip 21064 CPU cycles at 6.6 ns and divides the 6.6 ns clock by 6 to generate a 39.6 ns system clock, which is distributed throughout the machine. The EISA bus interface divides the system clock by 3 to generate an 8.4 megahertz (MHz) clock for the EISA peripherals.
- Provides low average cycles per instruction (CPI). The DECchip 21064 CPU can issue two Alpha AXP instructions in a single cycle, minimizing the average CPI. A branch history table minimizes the branch latency, further reducing the average CPI.
- Contains a fully pipelined floating-point execution unit capable of executing both Digital and IEEE floating-point instructions. The floating-point unit can accept a new instruction every cycle, except for divide instructions. The operate-to-operate latency for all instructions other than divide is six CPU cycles. The latencies for single and double precision divide instructions are 17 and 59 cycles, respectively.
- Contains an on-chip, 8K-byte direct-mapped, write-through, physical data cache with a block size of 32 bytes.
- Contains an on-chip, 8K-byte direct-mapped, read-only, physical instruction cache with a block size of 32 bytes, which is managed as a virtual cache.
- A single-entry stream buffer to prefetch 32-byte instruction cache blocks.
- An on-chip, 4-entry (32 bytes per entry) write buffer with byte merging capability.

DECchip 21064 CPU Execution Units

The DECchip 21064 CPU consists of the following three independent execution units:

- Integer execution unit (E-box)
- Floating point unit (F-box)
- The address generation, memory management, write buffer, and bus interface unit (A-box)

Each execution unit can accept at most one instruction per cycle. However, if code is properly scheduled, this CPU can issue two instructions to two independent units in a single cycle. A fourth box, the I-box, is the central control unit. The I-box issues instructions, maintains the pipeline, and performs all of the program counter calculations.

More Information

See Part II for more information on the DECchip 21064 registers and functions.

Intel 82350DT EISA Chip Set

The PB22H-KB system module contains the following subset of the Intel 82350DT EISA chip:

- One 82358 EISA bus controller (EBC) chip
- One 82357 integrated system peripheral (ISP) chip
- Two 82352 EISA bus buffer (EBB) chips

The following sections briefly describe each chip.

Intel 82358 EISA Bus Controller

The EBC is the central component of the EISA system. The EBC performs the translations between host DECchip 21064 CPU cycles, ISA cycles, and EISA cycles. Masters on any of the three buses communicate with the other buses through the EBC. The EBC controls all necessary timing alignments and translations for the different buses to communicate.

The EBC resides between the fast host (DECchip 21064 CPU) bus and the approximately 8 MHz ISA and EISA buses. It monitors cycles initiated on all buses. When the DECchip 21064 CPU places an address on the bus, which is in EISA space, the 82358 chip decodes it and places the resulting address on the EISA bus, if directed by the local memory or I/O decode. For more information, see the *Intel Peripheral Components* manual.

Intel 82357 Integrated System Peripheral

The ISP is a multifunction support peripheral that is designed to work with the 82358 EISA bus controller to provide most of the system functions necessary in EISA applications. The 82357 consists of the following:

- A high-performance 7-channel programmable DMA controller
- An arbitration scheme that enables efficient bus sharing among multiple EISA masters and DMA devices
- A 15-level programmable interrupt controller
- NMI logic for NMI control and generation
- Refresh address generation and control (the address generated is ignored)
- Five counter-timers that provide a system timer interrupt for bus timeouts

- DRAM refresh requests
- Other system timing operations (not all are used by the system software)

The ISP is accessed in EISA I/O space at the usual (PC/AT) addresses. For more information, see the *Intel Peripheral Components* manual.

Intel 82352 EISA Bus Buffer

Two EBBs are used on the PB22H-KB system module: one to integrate the data swap logic, and the other as an address buffer. This chip integrates approximately 17 components, lowering the system module chip count and cost.

More Information

See Part III for more information on the Intel 82357 integrated system peripheral chip registers and functions.

VLSI Technology VL82C106 Combination Chip

A VLSI Technology VL82C106 ISA combination chip provides a number of low-speed I/O devices. Although the VL82C106 chip contains fully programmable address decoders, it reverts to hard-wired addresses on reset. Default addresses are used for the following:

- Serial line A (COM1) at $3F8_{16}$
- Serial line B (COM2) at $2F8_{16}$
- Line printer port (LPT) at $3BC_{16}$
- Keyboard and mouse at 060_{16}
- Real-time clock (RTC) and RAM at 170_{16} (RTCMAP = GND)

Real-Time Clock

The VL82C106 chip contains an RTC that is program compatible with the Motorola MC14818A RTC. A 4.5 Volt (V) battery pack keeps the RTC running when the system power is turned off. The VL82C106 chip contains a total of 66 bytes of battery-backed RAM, which can be used to store configuration information. The battery for the RTC and BBRAM is mounted externally to the system module, and connects to the system module through the standard PC/AT battery connector (4 pin 100 millimetre (mm) connector with one pin missing). This is a standard PC/AT 4.5V lithium or alkaline battery.

Serial Lines

The VL82C106 chip contains two serial line interfaces. The serial lines have the following features:

- Full modem control
- Program selectable line format
- Program selectable data rate

Speeds range from 50 to 38.4K baud, and are double buffered. The external line drivers and receivers comply with EIA standard RS-232C. The serial lines are ESD protected, EMI filtered, and terminate in PC/AT standard serial port connectors (male DB9s). The interrupt request lines from the serial ports are ORed together and brought into a dedicated interrupt pin on the DECchip 21064 CPU. PAL code makes the serial lines interrupt in a normal way (once per character).

Line Printer Port

The VL82C106 chip contains an interface for a standard IBM® PC line printer. The printer port can be used as either a printer port, or a general purpose bidirectional I/O port. The printer port is EMI filtered, ESD protected, and terminates in an ISA standard printer connector (a female DB25). The data wires and the STB wire have 2200pF capacitors on them, in the traditional style. The interrupt request line from the printer port is brought into the IRQ1 interrupt on the 82357 chip. If running in PS/2 mode (which is the only reasonable mode), the printer interface generates its interrupt by clocking a flip-flop with the printer's ACK signal.

Parallel I/O

The VL82C106 chip contains a number of parallel I/O ports, which are read from and written to indirectly through the keyboard and mouse interface. The parallel I/O ports are used to read switch closures, and an output port is used to drive the LED located on the front panel. The bits called P10 to P17 of the parallel I/O port are inputs (P17 indicates port 1, bit 7). The P17 (KKSW) input, normally used for the keyboard lock switch, is held high to avoid disabling other VL82C106 chip functions when the keyboard lock is in the locked position. The keyboard lock switch is brought in through the P16 (KCM) input. This input is low when the key is pointing at the locked padlock on the front panel. The other inputs are unused and are tied low.

The LED on the front panel is driven from the A20 output (1 = LED on).

Keyboard and Mouse Ports

The VL82C106 chip contains an interface for a standard IBM PS/2® compatible keyboard and for a standard PS/2 serial mouse. The keyboard wires are EMI filtered, ESD protected, and are brought back to a standard IBM PS/2 keyboard connector (a female 6 pin mini-DIN). The mouse clock and data wires are ESD protected and brought back to a standard IBM PS/2 mouse connector (a female 6 pin mini DIN). The 5-volt power supply brought to the keyboard connector is short-circuit protected by a PTC device, and is EMI filtered. The interrupt request lines from the keyboard and mouse are ORed together and brought into a dedicated interrupt pin on the DECchip 21064 CPU. PAL code makes the keyboard and mouse interrupt in a normal way (once per character).

Periodic Interrupt Source

The VL82C106 chip contains a source of periodic interrupts that has a programmable rate The 976.562 μs (1024 Hz)) meets the Alpha AXP architectural requirement. The periodic interrupt is brought into a dedicated interrupt pin on the DECchip 21064 CPU so that PAL can always take the interrupt, as required by the Alpha AXP architecture. The VL82C106 RTC register A rate select bits RS<3:0> must be set to 6 to generate this interrupt period.

More Information

See Part IV for more information on the VLSI Technology VL82C106 combination chip registers and functions.

2

Backup Cache

Introduction

This section describes the PB22H-KB system module backup cache.

In This Chapter

This chapter contains the following sections.

- Backup Cache
- Control Store
- Tag Store
- Data Store
- Cacheable and Noncacheable Memory Locations
- Backup Cache Control
- Backup Cache Address Translation

Backup Cache

The system backup cache is a 512K-byte, direct mapped, write-back cache organized into 32-byte blocks with parity protection. Write-back mode denotes that both reads and writes are normally serviced from the backup cache without external logic intervention. This implies that the backup cache contains the only valid copy of a data block after it has been modified. The DECchip 21064 CPU manipulates the state of the DIRTY bit to signify that the block has been written to since it was initially read from memory.

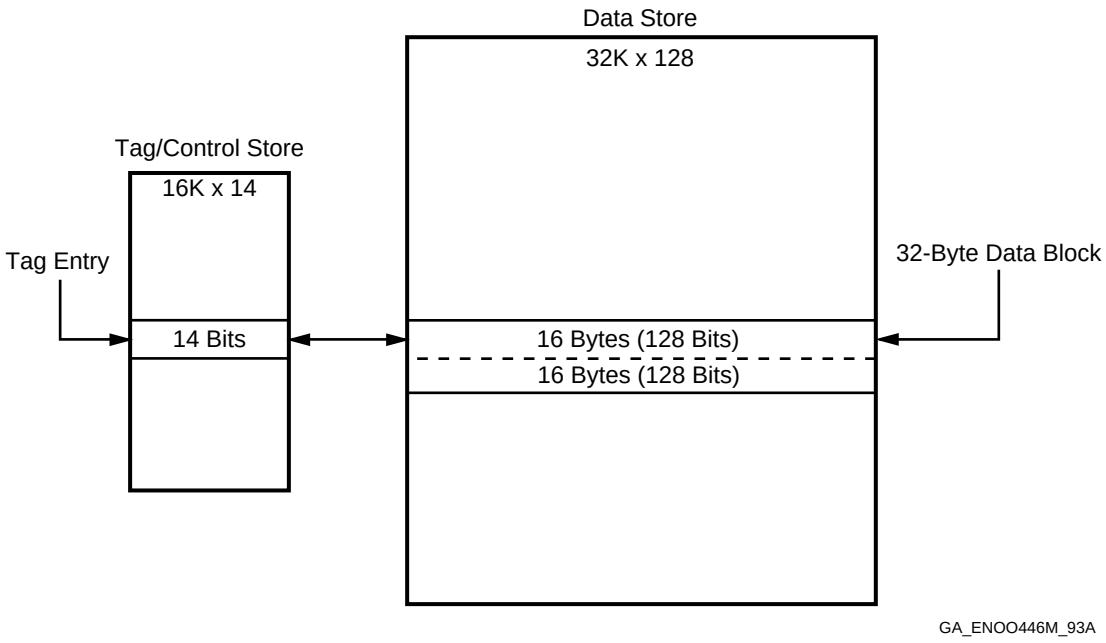
Each backup cache entry consists of the following three stores:

- The control store, which is parity protected, contains the binary flags that indicate whether a cache block entry is either valid or dirty.
- The tag store, which is parity protected, contains the high order address bits of the data currently stored in a cache block entry.
- The data store, which is protected by longword parity, and contains the 32 bytes of cached data.

Backup Cache organization

Figure 2–1 shows how the backup cache is organized.

Figure 2–1 Backup Cache Organization



Control Store

The backup cache control store is 16K x 4 in size and is implemented using one 16K x 4 static RAM (SRAM). The control store contains the binary flags that indicate the status of the cache block. Table 2–1 defines these flags.

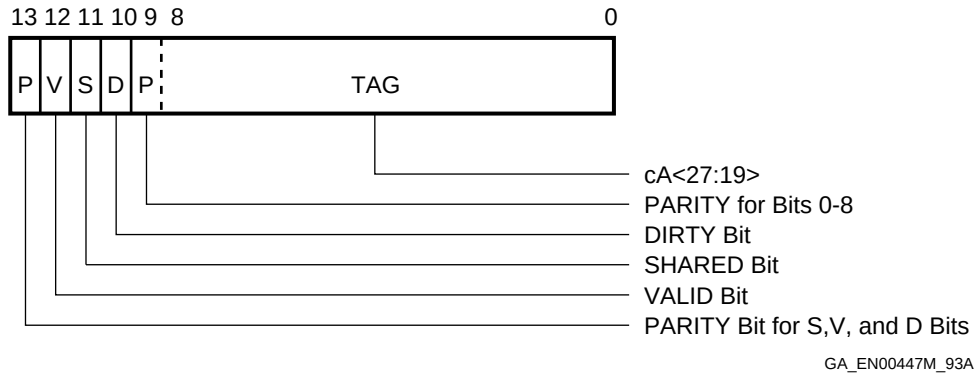
Table 2–1 Control Store Flags

Flag	Description
VALID	When set, this flag indicates that the data found in the other bits of the control store, the tag store, and the data store contain valid information. The VALID bit is set only by the backup cache controller when a cache block is filled with new data.
DIRTY	When set, the DIRTY and VALID bits indicate the data store contains an updated copy of a main memory location. This cache data must be written back to main memory when the cache location is victimized. There is only one copy of any given memory location marked dirty. The DIRTY bit is set by the processor performing a fast backup cache write-hit cycle, or by the backup cache controller assisting the processor to perform a STxC cycle.
SHARED	The PB22H-KB system module does not use the SHARED bit. It is always 0.
PARITY	This flag contains even parity over the contents of the control store. Parity is checked by the processor during every backup cache probe cycle, and by the backup cache controller during every probe cycle initiated by the system bus.

Tag Store

The tag store is 16K x 10 in size and is implemented by using three 16K x 4 SRAMs. The tag store contains the high order address bits (CA<27:19>) of the memory location that currently resides in the cache entry. There is a single parity bit that provides even parity over the tag store. Figure 2–2 shows the tag and control portions of a cache entry.

Figure 2–2 Backup Cache Entry Tag and Control Bits



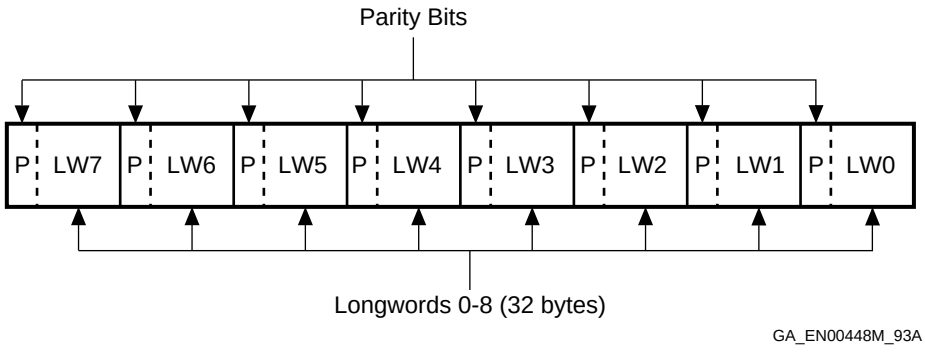
Data Store

The data store is 512K bytes in size and is implemented by using 17 ns 32K x 9 SRAMs. The data store contains the data of the memory location that is cached. Every data store portion of a cache entry contains 32 bytes (8 longwords). Each longword is protected by parity.

Error checking occurs when the processor hits the cache on a read or on a DMA.

Physically, the cache is only 128 bits wide (4 longwords), so a cache block consists of 2 consecutive addresses aligned on a 32-byte block boundary. Figure 2–3 shows a cache data block.

Figure 2–3 Backup Cache Data Block



Cacheable and Noncacheable Memory Locations

Cacheable Memory Locations

Only *memory-like* locations are cached. Memory-like locations are defined as locations with address bits CA<33:32> equal to 0 (quadrant 0). These locations are placed in the backup cache when it is enabled as a side effect of the processor issuing a READ_BLOCK transaction. They are also placed in the primary cache as the read data is acknowledged with OK.

Noncacheable Memory Locations

Nonmemory-like locations are not cached. Nonmemory-like locations are defined as those locations with address bit 33 equal to 1. These locations are not placed in the backup cache, and the read data is acknowledged with OK_NCACHE or OK_NCACHE_NCHCK. Writes to noncacheable space are restricted to aligned quadword accesses only. The quadword write data is presented to the system bus in the proper quadword associated with the address of the access. The data presented in the other quadwords of the cache block (during that system bus cycle) is undefined. However, the correct system bus parity is driven. No read merge occurs. All I/O locations are noncacheable locations.

Backup Cache Control

When the backup cache is enabled, the DECchip 21064 CPU probes it for each memory access, except for lock-related cycles.

Review Question

Is the following paragraph valid?

When the initial tag probe by the DECchip 21064 CPU finds that the entry is valid and unshared, the backup cache is under the control of the DECchip 21064 CPU.

When a backup cache probe results in a miss, or when a lock-associated command is invoked, the DECchip 21064 CPU initiates an external cycle. During the external cycle, the backup cache is under the control of the system logic. Depending on the cycle type, this logic either returns the data to the DECchip 21064 CPU, or accepts the data from the DECchip 21064 CPU and acknowledges the cycle to give control back to the DECchip 21064 CPU. If the cycle necessitates a backup cache fill, the system logic loads the data into the data store, the upper address bits <27:19> with good parity into the tag store, and the proper VALID and DIRTY bits into the control store.

During DMA, the DECchip 21064 CPU is forced off the backup cache SRAMs (using the HOLDREQ and HOLDACK mechanism) and the backup cache is controlled by the I/O system. The backup cache supplies the data to the I/O system on the DMA reads that are hits, and the backup cache accepts data from the I/O system on the DMA writes that are hits, without changing the state of the DIRTY bit.

Note

The data is simultaneously written to memory with the merged data from the cache.

The behavior of the processor relative to the backup cache is controlled or monitored by the processor's BIU_STAT, BIU_ADDR, FILL_ADDR, BIU_CTL, and BC_TAG internal processor registers (see Chapter 12).

Backup Cache Address Translation

During cache probes, the physical address must be translated to determine if the contents of the referenced location are in the backup cache. The cache index field, bits <18:5> of the physical address, is used to select one of the 16K entries in the backup cache. The cache tag field, bits <27:19> of the physical address, is then compared to the tag block of the selected entry. This implies the following:

- That the CPU TAGADR<33:28> inputs are held low
- That the parity logic that generates the tag address parity assumes that the CA<33:28> bits are all 0s
- That the tag comparator ignores the CA<33:28> bits

You must ensure that any address with CA<33> = 0 also has CA<32:28> = 0.

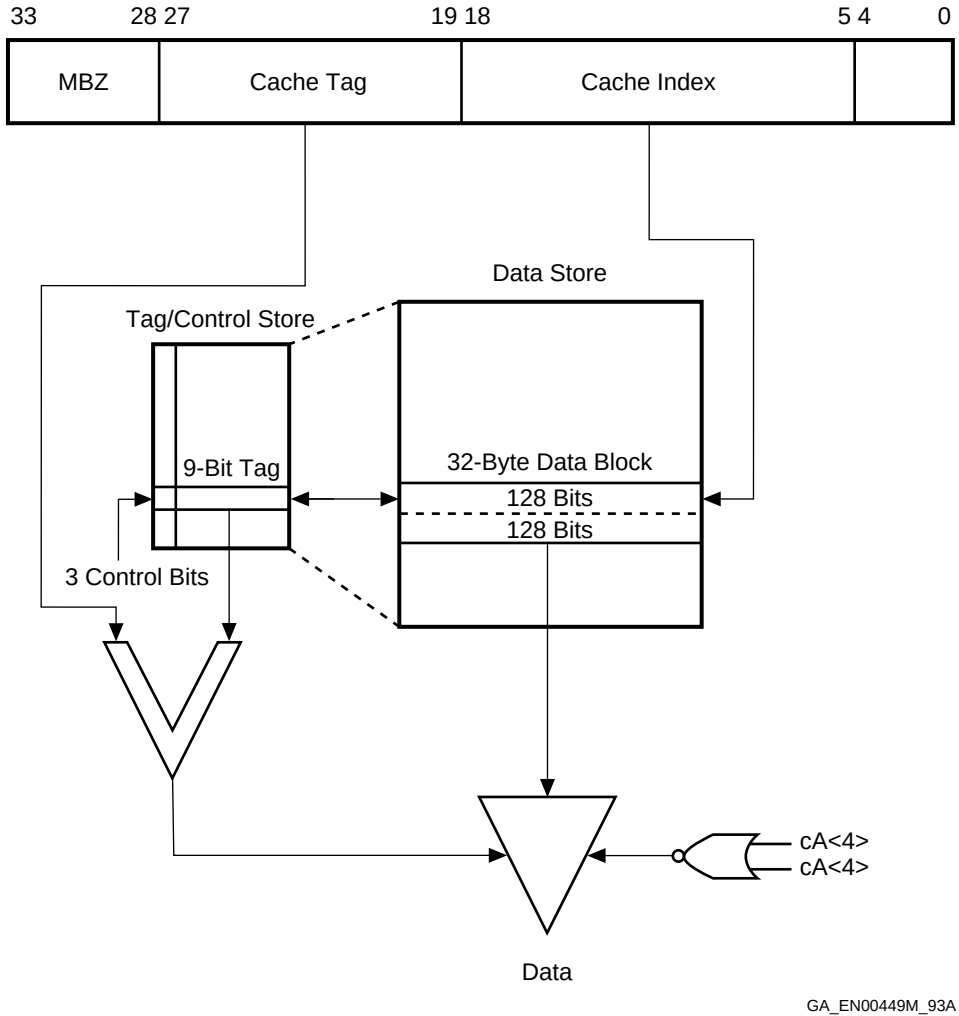
If the backup cache probe results in a valid match, the cycle finishes without performing a main memory access.

Signals CA<4> or AA<4>, depending on whether the backup cache is being controlled by the DECchip 21064 CPU or the system logic, are used to control which data half of the 32-byte data block is being written to or read from.

Address Translation

Figure 2–4 shows the cache address translation.

Figure 2–4 Backup Cache Address Translation



3
Lock Logic

Introduction This section describes the PB22H-KB system module lock logic.

In This Chapter This chapter contains the following section:

- Lock Logic

Lock Logic

The lock logic consists of a lock flag. There is no lock address register or lock address comparator. The lock flag is affected as follows:

- Cleared by reset
- Set by load-locked instructions
- Tested and cleared by store conditional instructions
- Cleared by all DMA writes (hits or misses)

An STxC instruction fails if there is a lock pending on the EISA bus.

Example

Assume a critical section is marked by a Boolean variable called *lock_loc*, with 0 indicating the section is not owned, and 1 indicating the section is owned. Example 3–1 shows an example of Intel code for updating data in the section.

To perform the same function using Alpha AXP code is a simple translation of the Intel code. Because a longword is the smallest instruction you can issue in the Alpha AXP architecture, the lock must be put in a longword. Example 3–2 shows the Alpha AXP translation.

The lock has to be released with an STLC instruction and a loop for the store to replay if an XCHG occurs (whose write always happens). In this case, the XCHG reads and writes a 1 (the lock is untouched) and the STLC replays, eventually releasing the lock.

Note

These sequences do not deal with any starvation issues.

Intel Code Example

Example 3–1 shows an example of Intel code for updating data in the section.

Example 3–1 Intel Lock Logic Code Fragment

```
10:     MOV     AX, 1
        LOCK XCHG       AX, LOCK_LOC
        LNE     10
; ... perform necessary operations on data
; ... protected by the lock
        MOV     LOCK_LOC, 0
```

Alpha AXP Code Example

Example 3–2 shows the Alpha AXP translation.

Example 3–2 Alpha AXP Lock Logic Code Fragment

```
10:     BIS     R31, 1, R0
        LDLL    R1, LOCK_LOC
        STLC    R0, LOCK_LOC
        BEQ     R0, 10      ; if 0, stlc failed, replay
        BNE     R1, 10      ; if 1, section owned, replay
; ... perform necessary operations on data
; ... protected by the lock
11:     BIS     R31, 0, R0
        STLC    R0, LOCK_LOC
        BEQ     R0, 11      ; if 0, stlc failed, replay
```

4

System Module Memory

Introduction

This chapter describes the PB22H-KB system module memory.

In This Chapter

This chapter contains the following sections:

- Memory Configurations
- Memory Address Generation
- Refreshing Memory

Memory Configurations

Memory Sizes

The PB22H-KB System Module supports two sizes of memory option: 16M bytes (4 1Mx36 SIMMs) and 64M bytes (4 4Mx36 SIMMs). Using combinations of these two memory options, the system supports between 16M bytes and 128M bytes of memory.

Configuring Memory

The PB22H-KB system module supports two banks of 128-bit wide, longword parity protected memory. Each bank contains four SIMM connectors. Table 4–1 shows how to install both types of memory modules in bank 0 and bank 1 to achieve the supported memory capacities.

Table 4–1 SIMM Socket Configurations

Total Memory (bytes)	Bank 0 Sockets				Bank 1 Sockets			
16M	1Mx36	1Mx36	1Mx36	1Mx36				
32M	1Mx36	1Mx36	1Mx36	1Mx36	1Mx36	1Mx36	1Mx36	1Mx36
64M	4Mx36	4Mx36	4Mx36	4Mx36				
80M	4Mx36	4Mx36	4Mx36	4Mx36	1Mx36	1Mx36	1Mx36	1Mx36
80M	1Mx36	1Mx36	1Mx36	1Mx36	4Mx36	4Mx36	4Mx36	4Mx36
128M	4Mx36	4Mx36	4Mx36	4Mx36	4Mx36	4Mx36	4Mx36	4Mx36

Configuration Rules

Follow these rules when installing memory modules in these banks:

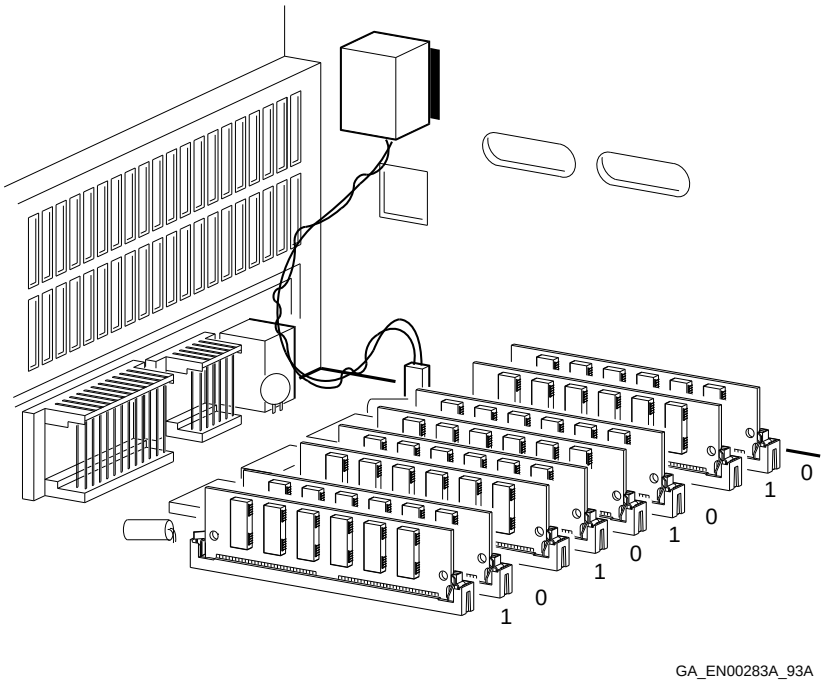
- Use only memory modules that are 36 bits wide and rated at a speed of 70 ns.
- Bank 0 must contain a memory option (four modules).
- A memory option consists of four memory modules. When you install a memory option in a memory bank, you must install a memory module in all of the connectors in that bank.

- Do not install different types of memory modules in the same bank.

SIMM Socket Locations

Figure 4–1 shows the locations of the SIMM sockets on the system module and identifies the SIMM connectors associated with each bank.

Figure 4–1 SIMM Socket Locations



Memory Address Generation

The DECchip 21064 CPU addresses with CA<33:32> = 00 are memory addresses. Address bits CA<32:28> are ignored by the memory system (although they must be zero for the cache to work properly). The CA<13:4> bits are used as memory column addresses, and CA<23:14> are used as memory row addresses. The CA<25:24> bits and the memory configuration field of the SYSCTL register (bits <7:4>) are used as bank selects (see Chapter 5 for information about the SYSCTL register and how bits <7:4> are used to indicate the installed memory configuration).

Table 4–2 Memory Address Generation

Total Memory	Bank 0	Bank 1	SYSCTL<7:4>	CA<27:24>	Target Bank
16M bytes	16M bytes		0000	0000	Bank 0
32M bytes	16M bytes	16M bytes	0000	0000	Bank 0
				0001	Bank 1
64M bytes	64M bytes		0010	00xx	Bank 0
80M bytes	64M bytes	16M bytes	0010	00xx	Bank 0
				0100	Bank 1
80M bytes	16M bytes	64M bytes	1000	0100	Bank 0
				00xx	Bank 1
128M bytes	64M bytes	64M bytes	1010	00xx	Bank 0
				01xx	Bank 1

Caution

Do not use address combinations that are not shown in Table 4–2, because they can cause memory wrapping or other problems.

Refreshing Memory

Main memory is always refreshed if the refresh timer in the 82357 chip is turned on. The refresh address output by the 82357 chip is ignored, and main memory is refreshed using a CAS-before-RAS refresh cycle. The DRAMs used in the memory system require 8 RAS cycles before proper device operation is achieved (they also require a 100 microsecond [μs] delay after power-up, but this is guaranteed by the reset network). This can be achieved either by enabling refresh and waiting or by reading memory eight times.

5
System Registers

Introduction

This chapter describes the registers that are unique to the PB22H-KB system module.

In This Chapter

This chapter contains the following sections:

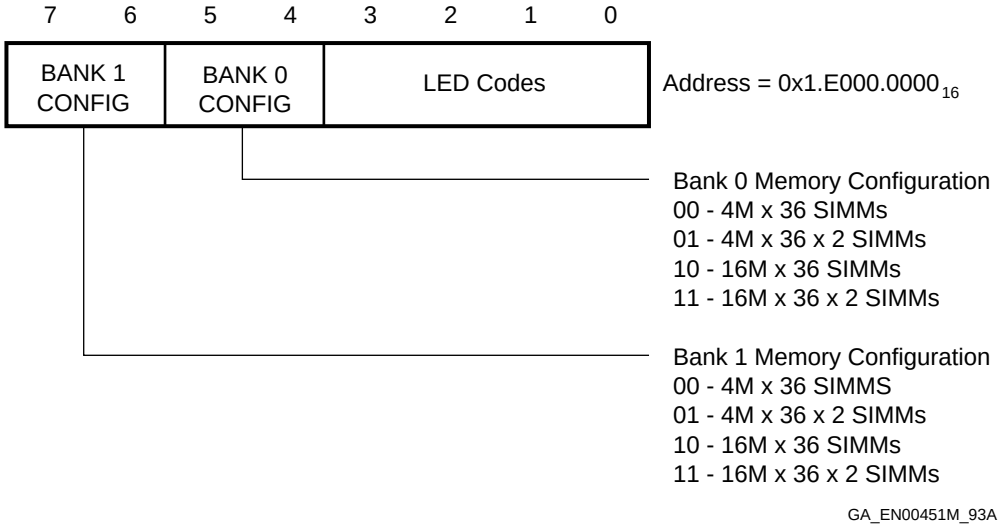
- System Control Register
- Host Address Extension Register

System Control Register

Description

The system control register (SYSCTL) is an 8-bit register that contains memory configuration information and the LED display code bits. Figure 5–1 shows the format of the SYSCTL register.

Figure 5–1 System Control Register



Memory Configuration Bits

Bits 4-7 of the SYSCTL register indicate the memory configuration of the system. These bits are set by the firmware in SROM, which examines memory and sets SYSCTL<7:4> accordingly. You must disable error checking for this process. The SYSCTL<7:4> bits are set to 00 at power-up. The firmware performs writes and reads to determine if memory is present at the first locations (0x0, $0x0.0400.0000_{16}$) of each bank. Then, for each bank, the firmware determines if the bank contains 16M bytes or 64M bytes of memory.

For example, a 64M-byte bank can be detected (with SYSCTL<7:4> still set to 00) because only a 64M-byte bank has memory at base plus 16M-byte. Examining $0x0.0100.0000_{16}$ (for bank 0) and $0x0.0500.0000_{16}$ (for bank 1) confirms whether a 64M-byte bank is present.

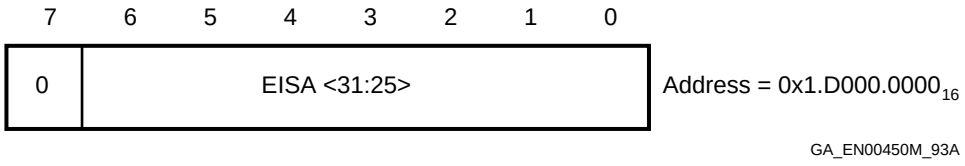
LED Display Code Bits

Bits 0-3 of the SYSCTL register are the LED display code bits (1 = on, 0 = off). See the system *Hardware Service Information* manual for an explanation of the the LED code meanings.

Host Address Extension Register

The host address extension register (HAE) is an 8-bit read/write register that contains the upper bits of addresses destined for the EISA bus. The HAE is unpredictable after system reset. Figure 5–2 shows the HAE register format.

Figure 5–2 Host Address Extension Register Format



For compatibility with future systems, your software must use only the lower segment of EISA address space (EA<31:25> = 0). You can encapsulate EISA references to provide support for future systems with a different numbers of address bits.

6

Exceptions and Interrupts

Introduction

This section describes the PB22H-KB system module exception and interrupt handling.

In This Chapter

This chapter contains the following sections:

- Exceptions and Interrupts
- General Exceptions
- Machine Check Exceptions
- Exception Handling
- PAL Priority Level
- PAL Code Entry 0020
- PAL Code Errors
- Interrupts
- Interrupt Handling
- PAL Priority Levels
- PAL Code Entry 00E0
- Hardware Interrupt Levels

Exceptions and Interrupts

When an interrupt or exception occurs, the DECchip 21064 CPU does the following:

- Drains the pipeline
- Loads the program counter into the EXC_ADDR internal processor register
- Dispatches to one of the PAL code exception routines
- If multiple exceptions occur, the DECchip 21064 CPU dispatches to the highest priority PAL code entry point

See the *Alpha AXP System Reference Manual* for more information.

General Exceptions

General exceptions are caused by badly written software that causes arithmetic traps or attempts illegal opcode execution, or by normal system operation, for example, a translation buffer miss. The list of general exceptions is shown in Table 6–1.

Table 6–1 General Exception Isolation Matrix

PAL Entry	Cause	Cause Isolation
External signal RESET(L) asserted.	RESET	NR†
ARITH	Arithmetic exception (divide by 0, and so on).	EXC_SUM IPR
DTB_MISS	Data translation buffer miss.	NR†
UNALIGN	D-stream unaligned reference.	NR†
DTB_FAULT	Remaining D-stream memory management errors.	NR†
ITB_MISS	Instruction translation buffer miss.	NR†
ITB_ACV	Instruction stream access violation.	NR†
CALLPAL	CALLPAL instruction executed.	Entry based on EXC_ ADDR <7..0>
OPDEC	Attempted execution of a reserved or privileged opcode.	NR†, EXC_ADDR points to instruction
FEN	Floating point operation attempted with floating point unit disabled, under or overflows, inexact errors, divide by 0, or invalid opcodes.	IPR EXC_SUM

†Isolation is not required (NR), because the PAL code entry identifies the cause.

Machine Check Exceptions

Machine check exceptions are special exceptions that are caused by errors in the hardware system. They all dispatch to the general MCHK PAL entry. The list of causes of machine check exceptions is shown in Table 6–2.

Table 6–2 Machine Check Isolation Matrix

Cause	Cause Isolation
BIU detects a backup cache tag store parity error.	BIU_STAT IPR
BIU detects a backup cache tag control store parity error.	BIU_STAT IPR
BIU detects a backup cache data store parity error.	BIU_STAT IPR
System external transaction terminated with HARD_ERROR.	BIU_STAT IPR

Note

When a system error occurs, all caches in the system must be examined to make sure that a location has not been purposely marked bad.

Exception Handling

Most exceptions have unique entries through which control flows. This allows easy identification, and fast dispatch to the appropriate system code. The exceptions are as follows:

- Reset
- Arithmetic
- DTB miss
- Unaligned reference
- Data access fault
- ITB miss
- ITB access violation
- Reserved opcode fault
- Floating point operation

These exceptions can occur relatively frequently as part of normal system operation.

The class of exceptions that occur as a result of hardware system errors are called machine checks. These exceptions result when an uncorrectable system error is detected during the processing of a data request.

Generally, exceptions are handled as follows by the PAL code:

- The PAL code determines the cause of the exception.
- If possible, it corrects the problem and returns the system to normal operation.
- If a problem is not correctable, or error logging is required, control is passed through the system control block (SCB) to the appropriate exception handler.

PAL Priority Level

Table 6–3 shows the prioritized list of the exceptions that can occur on the system. This list goes from the highest to the lowest priority. The interrupt PAL entry $00E0_{16}$ is included for completeness. For more information about PAL entry $00E0_{16}$ see the section entitled Interrupts in this chapter.

Note

Some of the information contained in Table 6–3 may not apply an all operating systems.

Table 6–3 Exception Priorities

Priority	Name	Description	PAL Offset (n_{16})	SCB Offset (n_{16})	IPL (n_{16})
1	RESET	Power-up or machine reset	0000	NA[1]	NA
2	MCHK	Machine check	0020	0660	31
3	ARITH	Arithmetic exception	0060	TBD[2]	TBD
4	INTERRUPT	Interrupt has occurred	TBD	TBD	TBD
5	DTB_MISS (PAL)	DTB miss has occurred in PAL mode	09E0	NA	NA
6	DTB_MISS (Native)	DTB miss has occurred in native mode	08E0	NA	NA
7	UNALIGN	Unaligned data	11E0	0300-03F0	X[3]

[1] Not applicable.

[2] To be done.

[3] *What does this mean??*

(continued on next page)

Table 6–3 (Cont.) Exception Priorities

Priority	Name	Description	PAL Offset (n_{16})	SCB Offset (n_{16})	IPL (n_{16})
8	DTB_FAULT	Remaining D-stream memory management errors	01E0	0080-00C0	X
9	ITB_MISS	ITB miss has occurred	03E0	NA	NA
10	ITB_ACV	I-stream access violation	07E0	0080-00C0	X
11	DPE	I-stream cache data parity error	0FE0	TBD	TBD
12	TPE	I-stream cache tag parity error	0BE0	TBD	TBD
13	CALL_PAL	256 locations based on instructions (7:0)	2000	TBD	TBD
14			2040	TBD	TBD
15			2060 - 3EF0	TBD	TBD
16	OPCDEC	Reserved opcode fault	13E0	0420	X
17	FEN	Floating point operation attempted with FPU disabled	17E0	0010	X

PAL Code Entry 0020_{16}

PAL Code Entry 0020_{16} Characteristics

Exceptions occur directly, except during disconnected write operations that occur as a result of masked write operations causing two consecutive system bus transactions. If an error is detected between the completion of the first and second system bus transactions, because of an unrelated intervening system bus transaction, a machine check occurs.

The PAL code found at the PAL entry 0020_{16} must sift through the system error information and determine the severity of the error. In some cases, the PAL code at this entry may correct the error and allow the machine to continue execution without any higher-level software intervention.

PAL Code Entry 0020_{16} Parse Tree

Because of the nature of backup cache errors, the PAL code executed on entry is restricted to read-only, as shown in Figure 6–1. If a backup cache error does not occur, the restrictions are lifted.

When a backup cache error occurs, the state of the backup cache is effectively frozen. However, memory system coherence is still maintained. This is done by suspending cache allocation when an error is detected. Backup cache probing by the processor must also be disabled by the PAL code by writing a 0 to the BC_ENA bit of the BIU_CTL register.

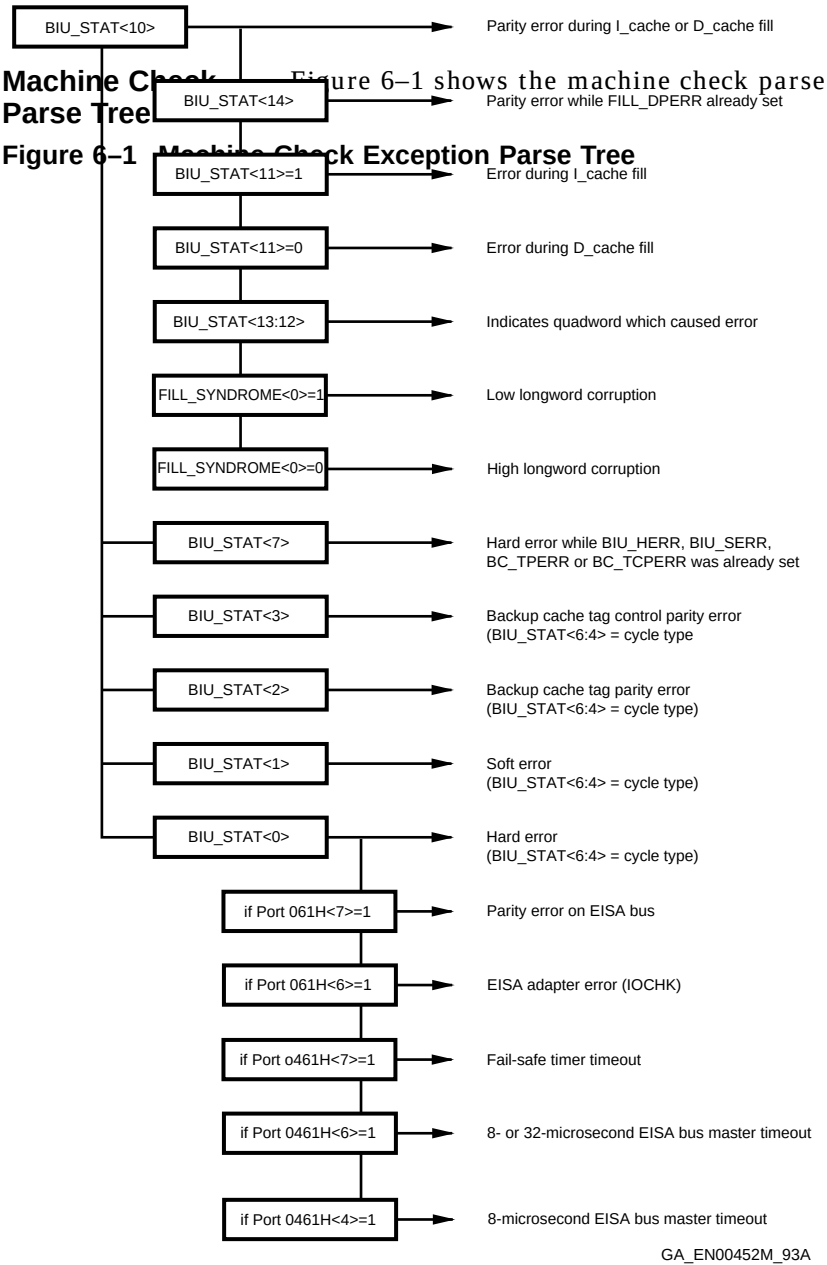
Note

Reading from modifiable memory-like data areas must be avoided because these locations could be dirty in the disabled cache and produce an incoherent access.

Machine Check Parse Tree

Figure 6–1 shows the machine check parse tree.

Figure 6–1 Machine Check Exception Parse Tree



PAL Code Errors

The following sections describe the PAL code errors. See Chapter 9 for more information about error handling.

Parity Error During I_Cache or D_Cache Fill

This error occurs on reads only. The PAL code must determine the following:

- If the error occurred while a previous error was being handled
- If it is an I-stream or D-stream error
- Which quadword resulted in the error
- Which longword within the quadword contains the error
- The address of the error

This error is fatal to the context in which the cached location is referenced, if the error occurred during an I_cache fill. However, if the error occurred during a D_cache fill, the severity of this error depends on the context of the processes that reference it. In user space, the error is process fatal. In system space, the error is system fatal.

Backup Cache Tag Parity Error

This error is restricted to reads only. The PAL code must remove the parity error from the tag store using the standard backup cache initialization procedure. If the DIRTY bit on the cache block in question is set, a system fatal error must be signaled to the system software. Otherwise, only error logging is required.

Backup Cache Tag Control Parity Error

This error is restricted to reads. The PAL code must remove the parity error from the tag control store using the standard backup cache initialization procedure. This error is fatal to the referenced cached location context.

Backup Cache Tag Store Errors

This error is restricted to reads. The PAL code must remove the parity error from the tag store using the standard backup cache initialization procedure. If the dirty bit on the cache block in question was set, a system fatal error must be signaled to the system software. Otherwise only error logging is required.

If a tag parity error occurs when an even number of tag bits change state during victimization of a dirty cache block, it is possible to generate a WRITE_DATA not acknowledged error, which causes a machine check. When this occurs, a system fatal error must be signaled to the system software.

Generally, this type of error indicates a hardware fault. However, if a simple test of the interface passes, the error may be recoverable after removing the affected locations and restarting the failing instruction.

EISA Bus Parity Errors

This error is the result of the parity lines being asserted on the H_bus, which in turn generates an NMI interrupt on the EISA bus to the DECchip 21064 CPU.

EISA Bus Adapter Error (IOCHK)

This error is the result of the IOCHK(L) line being asserted by an EISA adapter. The PAL code must try to determine which adapter on the EISA bus caused the IOCHK(L) line to be asserted and then attempt simple reads or writes to the device to determine the nature of the failure. The PAL code must disable the failing module if possible and restart execution, otherwise it is system fatal.

Fail-Safe Timer Timeout Error

This error occurs when the fail-safe timer in the 82357 ISP chip expires before being reset by the software. This results in the generation of an NMI interrupt. The PAL code must reset the fail-safe timer. This error is generally not system fatal.

8- or 32-Microsecond EISA Bus Master Timeout

The PAL code must determine if the timeout was by an 8-μs or a 32-μs bus master by examining bit 4 of port 0461_{16} and then determine the specific device or adapter. The PAL code must disable the failing module if possible and restart execution, otherwise it is system fatal.

Interrupts

Hardware Interrupts

Hardware interrupts are caused by hardware activity that requests the attention of the processor.

Table 6–4 describes how the various sources are wired to the six interrupt request pins on the DECchip 21064 CPU. Three of the interrupt request pins have multiple sources. In all cases, a unique device driver can be associated with each interrupt request.

The HALT interrupt is usually connected to a reset switch on a system unit. You must decide how your software handles this interrupt. For example, you can use it to pass control to a debugger, to perform a system reset, or ignore it (perhaps based on the setting of a keyboard lock switch for example). A reset function (or switch) is not provided in the hardware.

Table 6–4 DECchip 21064 CPU Interrupt Assignments

IRQ<(*n*)>	Interrupt Source
0	Interval timer from the VL82C106 chip
1	The 82357 ISP chip's programmable interrupt controller (PIC)†
2	NMI interrupts from the 82357 ISP chip
3	Keyboard and mouse interrupts from the VL82C106 chip (IRQK(H) and IRQM(H))
4	Halt switch interrupt (from the front panel)
5	Serial ports 1 and 2 from the VL82C106 chip (IRQA(H) and IRQB(H))

†See Table 17–2 for a description of the 82357 ISP interrupt assignments.

Interrupt Handling

All system interrupts go through the interrupt PAL code entry point, which is defined as the internal processor register (IPR) PAL_BASE + $00E0_{16}$. From this entry point, the appropriate interrupt priority level (IPL) is set and the system is interrogated to determine the cause of the interrupt. When the cause has been determined, the associated SCB offset is added to the SCB base address, and control is passed to that interrupt service routine.

PAL Priority Levels

Table 6–5 lists the interrupt priorities. This list goes from the highest to the lowest priority interrupts.

Note

The information in Table 6–5 applies only to the OpenVMS and OSF/1 operating systems.

Table 6–5 Interrupt Priorities

PAL Priority	Description	SCB Offset (n_{16})	SRM IPL (n_{16})	PAL IPL (n_{10})
1 (second highest)	Hardware 0—Interval timer	600	TBD †	TBD
2 (joint next)	Hardware 1—82357 chip interrupt	various		
3 (joint highest)	Hardware 2—NMI interrupts	660		
4 (joint next)	Hardware 3—Keyboard and mouse	keyboard 980 mouse 990		
5 (joint highest)	Hardware 4—HALT switch	NA ‡		
6 (joint next)	Hardware 5—Serial ports 1 and 2	various		
7	Performance counter 0	0650	14	20
8	Performance counter 1	0650	14	20
9	Software 1-15	0500-05F0	1-0F	20
10	Asynchronous system trap	0240-0270	2	20

†To be done.
‡Not applicable.

PAL Code Entry $00E0_{16}$

PAL Code Entry $00E0_{16}$ Characteristics

Various system status and error conditions are reported using one of the many interrupts that cause entry into the PAL code interrupt entry point. Because of the nature of backup cache errors, the PAL code executed on entry is restricted to read-only behavior. The procedure is as follows:

- Determine if a backup cache error has occurred. If a backup cache error has not occurred, the restrictions are lifted.
- When a backup cache error occurs, the state of the backup cache is frozen. However, memory system coherence is still maintained. This is done by suspending cache allocation when an error is detected.
- The PAL code must disable the backup cache probing by the processor by writing a 0 to the BC_ENA bit of the BIU_CTL register.

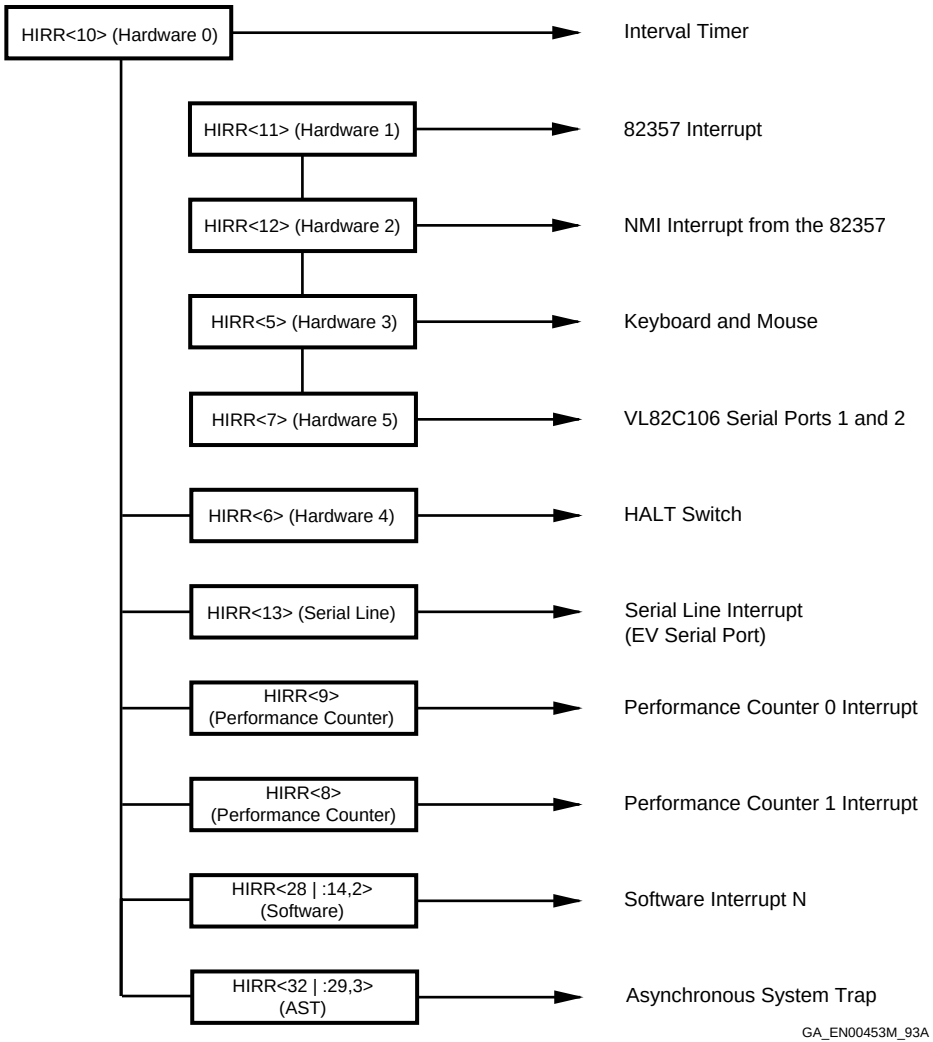
Note

You must avoid reading modifiable memory-like data areas, because these locations may be dirty in the disabled cache and produce an incoherent access.

Interrupt Parse Tree PAL code Entry $00E0_{16}$

Figure 6–2 shows an interrupt parse tree of PAL code entry $00E0_{16}$.

Figure 6–2 Interrupt Parse Tree



Hardware Interrupt Levels

Hardware 0 Interrupt

The VL82C106 chip's interval timer is the source of this interrupt. It occurs at a regular interval enabling the processor to schedule processing time to each process requiring attention.

The interval timer interrupt interrupts the processor every 976.562 μs (periodic interrupt from the VL82C106 chip). The PAL code handling this interrupt must do the following:

- Update its copy of the absolute time
- Copy the updated absolute time to register R4
- Clear the interrupt in the local system interrupt clear register
- Pass control to the interval timer interrupt routine

Hardware 1 Interrupt

Hardware 1 interrupts are generated by the 82357 chip's programmable interrupt controller subsystem. These interrupts are used to signal to the DECchip 21064 CPU that an interrupt request is pending and needs to be serviced.

Hardware 2 Interrupt

Hardware interrupt 2 is caused by the detection of hardware errors on the system module by the 82357 chip. These errors include the following:

- Assertion of IOCHL by EISA adapters
- Assertion of the PARITY(H) signal indicating a parity error has been detected on the EISA bus
- An EISA bus master timeout
- A fail-safe timer timeout
- A software generated NMI

These errors cause a machine check exception and the processing of the error is left to the machine check handler. Generally, these errors are system fatal.

Hardware 3 Interrupt	This interrupt is the result of an interrupt request from either the keyboard or mouse. The source of this interrupt is the VL82C106 chip.
Hardware 4 Interrupt	This interrupt is generated by the halt switch on the front panel of the enclosure.
Hardware 5 Interrupt	This interrupt is the result of an interrupt request from either serial port 1 or 2 located on the VL82C106 chip.
Performance Counter X Interrupt	The performance counter interrupts after a specified number of events have been counted.
Asynchronous System Trap Interrupt	Asynchronous system traps (ASTs) provide a way of notifying a process of events that are not synchronized with its execution, but which must be dealt with in the context of the process.

7

Direct Memory Access

Introduction

This section describes how the PB22H-KB system module implements direct memory access (DMA).

In This Chapter

This chapter contains the following section:

- DMA

DMA

DMA Definition

A direct memory access (DMA) is a memory access by any device other than the DECchip 21064 CPU.

DMA Addresses

DMA addresses are H_BUS addresses with HA<31:27> = 0 (or non-VL82C106 chip addresses). Address bits HA<30:27> are ignored, while address bits HA<26:5> select the 32-byte block in memory, and address bits HA<4:2> select the longword within the block. The programming of DMA transfers is simplified, because the DECchip 21064 CPU memory space address can be converted into an equivalent H_BUS memory address by discarding CA<33:32>.

DMA Cycles

There is full support for 8, 16, and 32-bit cycles with single transfers as well as all the following DMA cycles:

- Compatible
- Type A
- Type B
- Type C (burst mode)

Non 32-bit transfers are predictably slow because of the read, modify, and write nature of the cycle. Speeds up to 25M bytes per second can be achieved from an EISA device to system memory using burst mode with 32-bit transfers.

EISA and ISA Considerations

Host memory does not respond to DMA from the EISA or ISA bus in the upper half (0.5M byte to 1.0M byte) of the first megabyte of EISA and ISA memory. These addresses are assumed to reside on the EISA or ISA bus. This is necessary to allow ISA memory (for example, BIOS ROMs, or shared memory) to exist in the first megabyte of memory. The first megabyte of physical memory is not treated in any special way by the DECchip 21064 CPU, except that DMA transactions cannot occur in or out of this space.

Error Detection

Parity is checked when memory is read by a DMA device and errors are reported by the PARITY(H) signal mechanism in the 82357 chip. When the parity interrupt is enabled in the 82357 chip, it results in the EIRQ<2> pin being asserted at the DECchip 21064 CPU. This interrupt is normally not masked.

8
Local Buses

Introduction This section describes the PB22H-KB system module local buses.

In This Chapter This chapter contains the following section:

- H_BUS
- DECchip 21064 CPU Address Translation
- L_BUS

H_BUS

H_BUS Description

The I/O system is built around the H_BUS, which is essentially a clone of the pin interface of an Intel 80486DX microprocessor. The 25 MHz system clock generated by the DECchip 21064 CPU is used as the CLK(H) signal on this bus. Normally, the H_BUS is controlled by the DECchip 21064 CPU. However, during DMA transactions, the DECchip 21064 CPU is forced off the H_BUS (using the HOLDREQ(H) and HOLDACK(L) signal mechanism) and the H_BUS is controlled by the Intel 82357 integrated system peripheral (ISP) chip. During EISA master to host memory cycles, the H_BUS is controlled by the EISA interface logic.

DECchip 21064 CPU Cycle Mapping

The mapping of the DECchip 21064 CPU cycles into H_BUS cycles is not straightforward. Some of the reasons are as follows:

- Interrupt acknowledge cycles must be generated
- There is a memory versus I/O space distinction
- Byte masks are needed
- The low order address bits are different

DECchip 21064 CPU Address Translation

The DECchip 21064 CPU to H_BUS address translation is as follows:

- DECchip 21064 CPU addresses with CA<33:32> not equal to 00 are H_BUS addresses.
- The DECchip 21064 CPU address bits CA<31:9> are copied directly to H_BUS address bits HA<24:2>.
- Host address extension register bits HAE<6:0> are copied onto H_BUS address bits HA<31:25>.
- The H_BUS read/write signal is determined by examining the DECchip 21064 CPU's command on the bus.
- The target for each DECchip 21064 CPU reference is determined by the DECchip 21064 CPU address bits CA<33:28> as shown in Table 8–1.

Table 8–1 System Address Map

CA<33:32>	CA<31>	CA<30>	CA<29:28>	Effect
00	MBZ	MBZ	MBZ	Local memory CA<31:27> MBZ
01	0	MBZ	MBZ	EISA INTA cycle CA<31:5> MBZ (-> HA<24:2> = 0) CA<4:0> SBZ Vector appears in low byte
01	1	0	0X	FEPROM #0 CA<28:9> = Address for up to 1M byte of ROM CA<8:0> SBZ Data appears in low byte
01	1	0	1X	FEPROM #1 CA<28:9> = Address for up to 1M byte of ROM CA<8:0> SBZ Data appears in low byte

(continued on next page)

Table 8–1 (Cont.) System Address Map

CA<33:32>	CA<31>	CA<30>	CA<29:28>	Effect
01	1	1	00	VL82C106 chip ComboAddr<..0> from CA<..9> CA<8:0> SBZ (byte-wide Bus) Data appears in low byte
01	1	1	01	Host address extension register CA<27:0> SBZ
01	1	1	10	SYSCTL register CA<27:0> SBZ
01	1	1	11	Spare register CA<27:0> SBZ
10	X	X	XX	EISA memory HA<31:25> <- HAE<6:0> HA<24:2> <- CA<31:9> Length or offset from CA<8:5>
11	X	X	XX	EISA I/O HA<31:25> <- HAE<6:0> HA<24:2> <- CA<31:9> Length or offset from CA<8:5>

Table 8–2 shows the EISA and H_BUS byte mask generation.

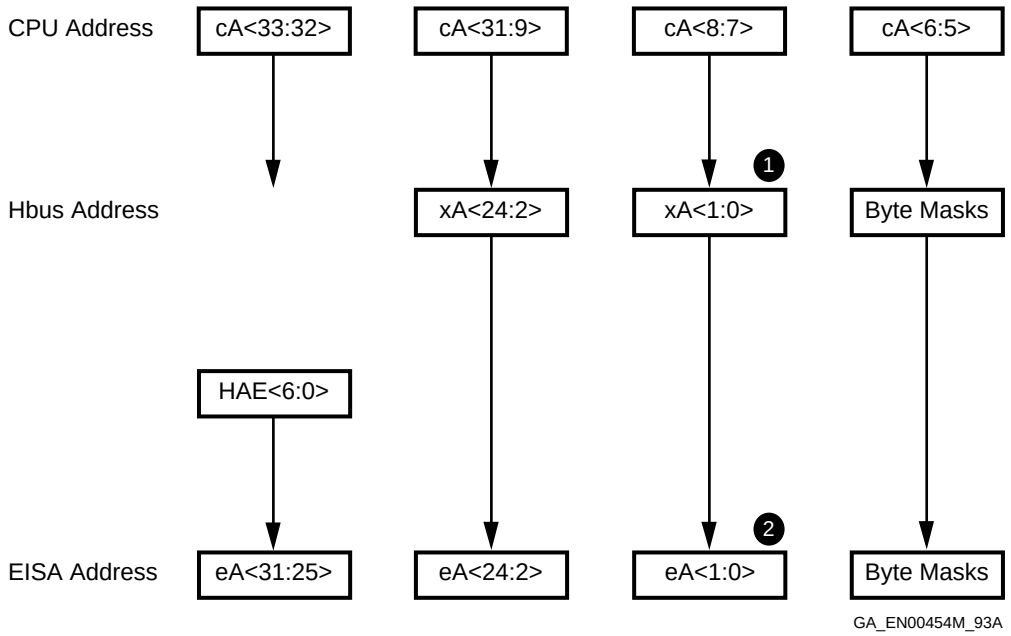
Table 8–2 EISA and H_BUS Byte Mask Generation

CA<6:5> (Length)	CA<8:7> (Offset)	Byte Enables
00	00	FFFT
00	01	FFTF
00	10	FTFF
00	11	TFFF
01	00	FFTT
01	01	FTTF
01	10	TTFF
01	11	Reserved
10	00	FTTT
10	01	TTTF
10	1X	Reserved
11	00	TTTT
11	01	Reserved
10	1X	Reserved

For byte, word, and longword accesses, the data is sent or received on the low-order byte lanes. A write access to a byte at offset 1 has address bits <4:0> equal to 0, and the data is transferred on bits <7:0> of the DECchip 21064 CPU pin interface. Tribytes do not have this feature and must appear in the correct byte lanes. A tribyte access to offset 1 requires that the data is shifted up by 1 byte before writing, and shifted down by 1 byte after reading.

Figure 8–1 shows how DECchip 21064 CPU addresses are translated into EISA bus addresses.

Figure 8–1 H_BUS and EISA Bus Address Translation



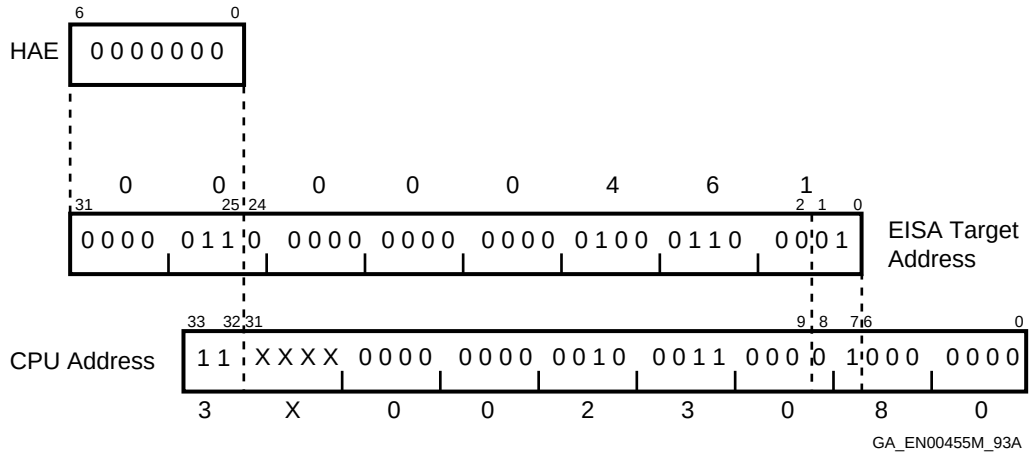
Example H_BUS and EISA Address Translation

To write a byte to EISA I/O address 0461_{16}, do the following (see Table 8–1):

1. Write 0 to the host address extension register ($0x1.D000.0000_{16}$).
2. Set address bits <33:32> to 11 (EISA I/O space).
3. Insert the byte count (0 = 1 byte) into address bits <6:5>.
4. Set the byte offset (1 = 1 byte) into address bits <8:7>.
5. Put the target address bits <24:2> into address bits <31:9>.
6. Send this address onto the CA bus.
7. Put the data bits in byte 1 (bits <15:8>).

The resulting address is $0x3.x002.3080_{16}$. Figure 8–2 shows how the effective CA bus address is obtained. To access an EISA address that uses bits <31:25>, they must first be set in the HAE ($0x1.D000.0000_{16}$).

Figure 8–2 CA and EISA Bus Address Translation



A read with CA<33:31> = 010 generates a H_BUS interrupt acknowledge cycle to H_BUS I/O space. PAL code is used to retrieve interrupt vectors from the interrupt controller. Two IAK(??) cycles are required to retrieve the vector from the 82357 chip. Ensure that your software maintains the minimum time required between IAK(??) cycles to the 82357 chip.

Reads from the H_BUS are not cached and parity is not checked. On reads, the H_BUS data appears on longword <0> of the cache block, and longwords <7:1> are unpredictable. On writes, the data on longword <0> is sent to the H_BUS and is independent from the state of the DECchip 21064 CPU's longword write masks. This implies that address bits <4:0> should be zero (SBZ) when accessing the H_BUS.

L_BUS

Peripheral Selection

The system module contains a number of integrated peripherals. The following DECchip 21064 CPU addresses are used:

- Addresses with CA<33:31> = 011 address the local peripherals.
- Address bits CA<30:28> select the peripheral.
- Address bits CA<28:9> become address bits <19:0> at the system ROMs.
- Address bits CA<24:9> become address bits <15:0> at the VL82C106 chip.
- The VL82C106 chip and the ROMs are wired onto the lower 8 bits of the H_BUS data longword.

Table 8–3 shows the L_BUS address map.

Table 8–3 L_BUS Address Map

Address (CA<8:0> MBZ)	Device
$1.8000.0000\text{-}1.8003.FFFF_{16}$	FEPROM 0 (256K bytes)
$1.8004.0000\text{-}1.9FFF.FFFF_{16}$	Reserved
$1.A000.0000\text{-}1.A00F.FFFF_{16}$	FEPROM 1 (1M byte)
$1.A010.0000\text{-}1.BFFF.FFFF_{16}$	Reserved
$1.C000.0000\text{-}1.C1FF.FE00_{16}$	VL82C106 chip
$1.C1FF.FE01\text{-}1.CFFF.FFFF_{16}$	Reserved
$1.D000.0000_{16}$	Host address extension (HAE) register
$1.D000.0001\text{-}1.DFFF.FFFF_{16}$	Reserved
$1.E000.0000_{16}$	System control (SYSCTL) register
$1.E000.0001\text{-}1.EFFF.FFFF_{16}$	Reserved

9

Error Handling

Introduction

This chapter describes how the system module handles errors.

In This Chapter

This chapter contains the following sections:

- Error Handling Overview
- I/O Error Detection
- Parity Error Detection
- Backup Cache Parity Errors
- Nonmaskable Interrupt Errors

Error Handling Overview

The following types of error handling are supported:

- Hardware error handling consists of parity checking on data longwords in memory and in cache
- Serial port data has parity and framing error checking
- The keyboard has parity checking for data only
- Nonmaskable interrupts (NMIs) are generated for the following:
  - H_bus and EISA bus parity errors
  - EISA adapter errors (IOCHK(L))
  - EISA bus master time-out
  - Fail-safe timer expiration

Errors are generally treated as fatal. The recovery mechanism for fatal errors is to reboot the system. An alternative to rebooting the system, which may be a feature of an operating system, is to ignore the errors and continue processing.

Only one method is implemented on the system board for external logic to signal an error to the DECchip 21064 CPU. This is by an IRQ2(H) interrupt signal to the DECchip 21064 CPU. There are three conditions in the hardware that can cause this interrupt:

- IOCHK(L)
- PARITY(L)
- EISA bus time-out

The NMI control and status register in the 82357 chip can be accessed after an IRQ2(H) interrupt signal has been posted to determine which of the three conditions caused the interrupt. For more information on NMI errors see the section entitled Nonmaskable Interrupt Errors.

I/O Error Detection

There is no error detection mechanism for the EISA and ISA bus data path. Error detection is localized on the I/O devices.

For the 82C106 chip, the serial lines and keyboard have the option of parity data protection from the devices to the controller. The software driver enables and detects the parity protection. The individual operating systems determine the recovery mechanism.

Any EISA or ISA adapter device that detects an internal error can assert the IOCHK(L) signal on the EISA bus. This signal is routed through the 82357 chip and results in an IRQ2(H) interrupt signal to the DECchip 21064 CPU. When an IRQ2(H) interrupt signal occurs, the NMI control and status register in the 82357 chip indicates whether the error was a parity, bus time-out, or IOCHK(L) signal error. There is no indication of which EISA bus adapter asserted the IOCHK(L) signal. This error is fatal (nonrecoverable).

If an EISA bus master does not release the bus 8 μs after the MACK(L) line is deasserted, the bus time-out signal causes an IRQ2(H) interrupt signal as described in the previous paragraph.

Parity Error Detection

The data path between the EISA bus and memory is protected by longword parity. Data from the EISA bus travels along the H_bus, and parity is generated for each longword before being written with the parity information to memory.

Data being accessed from memory by an EISA device travels along the H_bus where parity is checked prior to its placement on the EISA bus. A parity error detected on the H_bus causes the parity input to the 82357 chip to be asserted, resulting in an IRQ2(H) interrupt signal. The PE bit in the NMI status and control register is set to indicate a parity error.

When the DECchip 21064 CPU reads system memory, the longword parity is fetched from memory with the data and checked by the DECchip 21064 CPU. Parity errors result in either a D_stream or I_stream parity error. The I_stream and D_stream error flows are described in the sections entitled I_Stream Parity Error Flow and D_Stream Parity Error Flow.

I_Stream Parity Error Flow

The I_stream parity error flow is as follows:

1. Data is put into the I_cache unchanged and the block is validated.
2. A machine check occurs.
3. In the BIU_STAT register, the FILL_DPERR bit is set and the FILL_IRD bit is set.
4. The FILL_ADDR<33..5> register bits and BIU_STAT[FILL_QW] register bits contain the address of the bad quadwords.
5. The FILL_SYNDROME register identifies the failing longwords.
6. The BIU_ADDR register and the BIU_STAT<6..0> register bits are locked and the contents are unpredictable.
7. The DC_STAT register is locked, and the contents are unpredictable.
8. The BC_TAG register holds the results of the external cache tag probe, if the external cache was enabled for this transaction.

D_Stream Parity Error Flow

The D_stream parity error flow is as follows:

1. Data is put into the D_cache unchanged, and the block is validated.
2. A machine check occurs.
3. In the BIU_STAT register, the FILL_DPERR bit is set, and the FILL_IRD bit is cleared.
4. The FILL_ADDR<33..5> register bits and BIU_STAT[FILL_QW] register bits contain the address of the bad quadwords.
5. The FILL_ADDR<4..2> register bits contain PA bits <4..2> of the location that the failing load instruction attempted to read.
6. The FILL_SYNDROME register identifies the failing longwords.
7. The BIU_ADDR register and the BIU_STAT<6..0> register bits are locked and the contents are unpredictable.
8. The DC_STAT[RA] register bits identify the register that holds the bad data. The DC_STAT[LW], DC_STAT[LOCK], DC_STAT[INT], and DC_STAT[VAX_FP] register bits identify the type of load instruction.
9. The BC_TAG register holds the results of the external cache tag probe, if the external cache was enabled for this transaction.

Backup Cache Parity Errors

Data parity, tag parity, and control parity are stored in the backup cache. Parity errors cannot be detected for DECchip 21064 CPU writes directly to the backup cache.

Backup Cache Data Parity Errors

Data errors detected during reads directly from the backup cache result in either I_stream or D_stream parity errors. The I_stream and D_stream error flows are described in I_Stream Parity Error Flow and D_Stream Parity Error Flow.

Backup Cache Tag and Control Parity Errors

Parity errors on the tag and control bits are detected on backup cache reads only. See the sections entitled Tag Address Parity Error Flow and Tag Control Parity Error Flow for the tag address and control parity error flows.

Tag Address Parity Error Flow

The error flow for backup cache tag address parity errors is as follows:

1. Tag address parity errors are recognized at the end of a tag probe sequence.
2. Cache lookup uses a predicted parity so the transaction misses the external cache.
3. The BC_TAG register holds the results of an external cache tag probe.
4. A machine check occurs.
5. The BIU_STAT[BC_TPERR] bit is set.
6. The BIU_ADDR register holds the physical address of the error.

Tag Control Parity Error Flow

The error flow for backup cache tag control parity errors is as follows:

1. Tag control parity errors are recognized at the end of a tag probe sequence.
2. A transaction is forced to miss an external cache.
3. The BC_TAG register holds the result of an external cache tag probe.

4. A machine check occurs.
5. The BIU_STAT[BC_TCPERR] register bit is set.
6. The BIU_ADDR register holds the physical address of the error.

Nonmaskable Interrupt Errors

EISA bus errors are reported to the DECchip 21064 CPU by the NMI facility on the Intel 82357 chip. The NMI output of the 82357 chip is connected to the IRQ2(H) input of the DECchip 21064 CPU. There are five separate sources for this error, each of which is maskable. The NMI interrupts are controlled by the master enable control bit (bit 7) of the NMI enable and disable register, and the RTC address register at port 0170_{16}.

NMI Error Types

The NMI error types are listed in Table 9–1.

Table 9–1 NMI Error Types

Bit	Name	Description
NMI CSR (Port 061_{16})		
7	PE	System parity error. A parity error occurred during an EISA option DMA. This NMI is initiated by the assertion of the PARITY(L) input to the 82357 chip. It is enabled by bit 2 of the NMI status and control register.
6	IOCHK(L)	EISA option error. This NMI is initiated when an EISA option asserts the IOCHK(L) input to the 82357 chip. It is enabled using bit 3 of the NMI status and control register.

(continued on next page)

Table 9–1 (Cont.) NMI Error Types

Bit	Name	Description
NMI Extended CSR (Port 461_{16})		
7	FST	Software controlled fail-safe timer time-out. This NMI is driven when timer 2, counter 0 in the 82357 chip reaches a terminal count. It is enabled using bit 2 of the NMI extended CSR register.
6	BMT	EISA bus master time-out. There are two sources for this NMI, both of which are driven by the bus arbitration logic within the 82357 chip. This NMI can be caused if a bus master has held the bus for longer than 8 μs or if a slave has not released the bus within 32 μs. Both of these are enabled using bit 3 of the NMI extended CSR register and the status of the 8 μs time-out is shown in bit 4. An indication of the last master to own the bus can be read from the EISA bus master status latch at port 464_{16}.
5	SNMI	Software NMI. Your software can trigger an NMI by writing to the software NMI generation register (port 462_{16}). This is enabled using bit 1 of the NMI extended CSR register.

NMI Error Handling

The PAL code treats all of these errors as uncorrectable; that is, it flags no retry and dispatches in SCB location 660_{16}. The machine check log is then built. At present, the only specific information included for this system is the error ID and the bus master status register contents. You can include other information.

NMI Error IDs

NMI error IDs are listed in Table 9–2.

Table 9–2 Error Identification

Type	ID Number
Parity error	20
I/O check error	21
Bus master time-out	22
Slave disconnect time-out	23
Fail-safe timer time-out	24
Software NMI	25

10

Power-Up Initialization

Introduction

This chapter describes the system power-up initialization sequence.

In This Chapter

This chapter contains the following sections:

- Power-Up Initialization Overview
- Power-Up Initialization Flow
- Power-Up Diagnostics
- Power-Up Initialization Routines
- Map of Memory Following Power-Up Initialization

Power-Up Initialization Overview

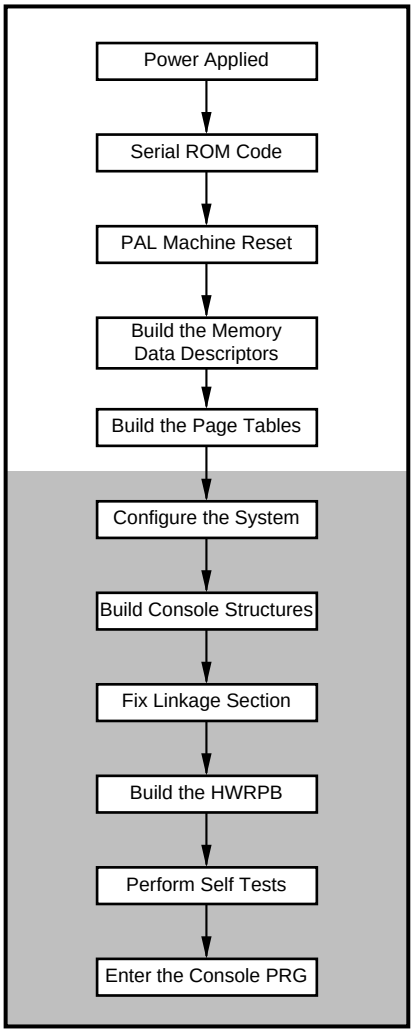
When the system starts from a power-up condition, the serial ROM (SROM) code is loaded into the DECchip 21064 CPU's 8K-byte instruction cache. The SROM code verifies the functionality of the hardware required to load and execute the system ROM code in flash EPROM (FEPROM), which contains the PAL machine initialization code and the console code.

After the I_cache is loaded, the SROM data and clock lines are used to establish a serial line connection to the DECchip 21064 CPU. The SROM diagnostics output is sent to this serial port at 9600 baud and to the LEDs. If the SROM code encounters a fault that stops the loading and executing of the FEPROM code, then power-up execution is halted and a branch is made to the SROM miniconsole routine.

Power-Up Initialization Flow

Figure 10–1 shows the power-up initialization flow of the system.

Figure 10–1 Power-Up Initialization Flow



Power-Up Diagnostics

When the system powers-up, the diagnostic LEDs display codes. You can use the diagnostic LED codes for troubleshooting.

Table 10–1 lists the SROM code execution sequence and corresponding LED codes. In the column labeled Result in Table 10–1, the term *fatal error* means that the console ROM cannot be successfully loaded and an unconditional branch is made to the miniconsole.

Table 10–1 Power-Up Sequence LED Codes

LED Code	Indication	Action
F (1111)		
E (1110)	First instruction executed.	
D (1101)	The 82357 chip and the 82C106 chip are initialized.	
C (1100)	Memory refresh is enabled and sizing is complete.	
B (1011)	A machine check occurred during the memory or cache access.	
A (1010)	Error in memory bank 0 configuration.	
9 (1001)	Error in memory bank 1 configuration.	
8 (1000)	Memory configuration success.	
7 (0111)	Memory test fail with DCACHE disabled.	
6 (0110)	Memory test fail with DCACHE enabled.	
5 (0101)	Reserved.	

(continued on next page)

Table 10–1 (Cont.) Power-Up Sequence LED Codes

LED Code	Indication	Action
4 (0100)	ROM path test fail—Load the fail-safe loader.	
3 (0011)	Console checksum error—Load the fail-safe loader.	
2 (0010)	Failsafe loader checksum error.	
1 (0001)	Control passing to the fail-safe loader.	
0 (0000)	Control passing to console.	

Power-Up Initialization Routines

The following sections describe the power-up initialization routines that reside in the SROM.

SROM$POWERUP

Arguments

- None

Returns

- R20—Memory size in M bytes
- R0—Zero
- pal_base—PAL code base address
- exc_addr—PAL reset vector

Description

Execution of system SROM code begins with the *SROM$POWERUP* routine. This routine is the main dispatcher to the other SROM routines. If the system ROM code has been successfully loaded into memory, information required by this code is placed in registers and a HW_REI is made to the PAL entry point.

The algorithm for the SROM$POWERUP routine is as follows:

1. Create test data patterns
2. Initialize the B_cache tag store and IPRs and CSRs as required
3. Call SROM$357_INIT to initialize the 82357 chip interrupts
4. Call SROM$SIZE_MEMORY to get the size of memory
5. Call SROM$MEM_TEST to test the system memory
6. Call SROM$sySROM_LOAD to load the system ROM into memory
7. Set exit (return) values of the registers as required by PAL code
8. Go to the PAL code entry point

SROM$SIZE_MEMORY

Note

What does This routine do?

SROM$MEM_TEST

Arguments

- R22—Quadword of Hex 5s
- R29—Return Address

Returns

- R21—Top of test memory, up to what is required for the console

Description

This routine starts testing memory at address 0. It tests all of system memory. If there is an error in the first 2M bytes of system memory, a fatal error is flagged. Otherwise, the routine returns the size of good memory in R21. It does an address pattern test followed by an x55 and an xAA test.

SROM$SYSROM_LOAD

Arguments

- R29 —Return Address

Returns

- None

Description

This routine loads the 256K bytes from the FEPROM into memory starting at address 4000_{16}.

SROM$MEM_FILL

Arguments

- R1—data
- R2—base_pointer, Kbytes
- R3—block_size, Kbytes, 0 for one cache block (32 bytes)
- R4—step_size, multiples of 8 bytes (quadwords)

- R30—Return address

Returns

- R15—pointer to last address written

Description

The SROM$MEM_FILL writes fixed quadword data to memory locations from base_pointer for block_size K bytes. The step_size argument is used to increment the pointer.

The call must place the return address in R30, because when this routine finishes it jumps unconditionally to whatever address is in R30.

The base_pointer and block_size arguments are received as K bytes (modulo 2_{10}). Local copies are made and logically shifted left 10 places to get the intended base_pointer and block_size. This simplifies loading the argument registers, because the largest literal that can be placed in an Alpha AXP instruction is 16 bits. Passing a block_size of 0 to this routine causes it to default to a 32-byte block_size (one cache block). The step_size is expected as increments of 8 bytes (quadwords). For example, the integer passed in R4 must be 8, 16, 24, 32, and so on.

SROM$MEM_RDCMP

Arguments

- R1—data_expected
- R2—base_pointer, K bytes
- R3—block_size, K bytes, 0 for one cache block (32 bytes)
- R4—step_size, multiples of 8 bytes (quadwords)
- R30—Return address

Returns

- R11—data_fetched
- R13—bytes, bytes remaining (0, if no errors)
- R15—pointer to last address tested
- R16—error_flag, 0 if no errors, else non-zero

Description

This routine compares the data stored in memory locations with that passed in R1.

Fixed quadword data is read from memory locations from base_pointer for block_size K bytes. The step_size is used to increment the pointer. Data is read into R11 and XORed with the data passed in R1, and the result is stored in R16. The call must place the return address in R30, because when this routine finishes it jumps unconditionally to whatever address is in R30.

The base_pointer and block_size are received as K bytes (modulo 2_{10}). Local copies are made and logically shifted left 10 places for the intended base_pointer and block_size. This simplifies loading the argument registers, because the largest literal that can be placed in an Alpha AXP instruction is 16 bits. Passing a block_size of 0 to this routine causes it to default to a 32 byte block_size (one cache block). The step_size is expected in multiples of 8 bytes; that is, the integer passed in R4 must be 8, 16, 24, 32, and so on.

SROM$MEM_PACKROM

Arguments

- R2—Number of bytes to pack
- R10 —checksum to be updated
- R14—pointer to ROM space
- R30—Return address

Returns

- R10—Updated checksum
- R14—pointer to next byte of ROM space

Description

This routine extracts the low byte from each longword access to ROM space and inserts it into its appropriate byte order in a longword returned in R1. This routine extracts the low byte from each longword access to ROM space and inserts it into its appropriate byte order in a longword returned in R1.

A byte-wide checksum algorithm is also applied to the data as it is read out of the ROM a byte at a time. The checksum algorithm updates the checksum value that is placed in R10 by the call.

SROM$DIAG_REPORT

Arguments

- R13—fatal_flag
- R16—error_flag
- R25—sequence_number (see Table 10–1)

Returns

- None

Description

This routine prints the sequence number to the LEDs and the SROM port.

If the fatal_flag is set, then this routine jumps to the miniconsole.

SROM$CONSOLE

Arguments

- None

Returns

- None

Description

This routine uses the DECchip 21064 CPU SL_XMIT and SL_RCV registers to produce a bit oriented console using a 9600 baud software timing loop.

The following subroutines in this module are global:

- getChar
- putChar
- putString
- putByte
- putLong
- putReg

Map of Memory Following Power-Up Initialization

When the system powers-up, it needs 3M bytes of memory for the following:

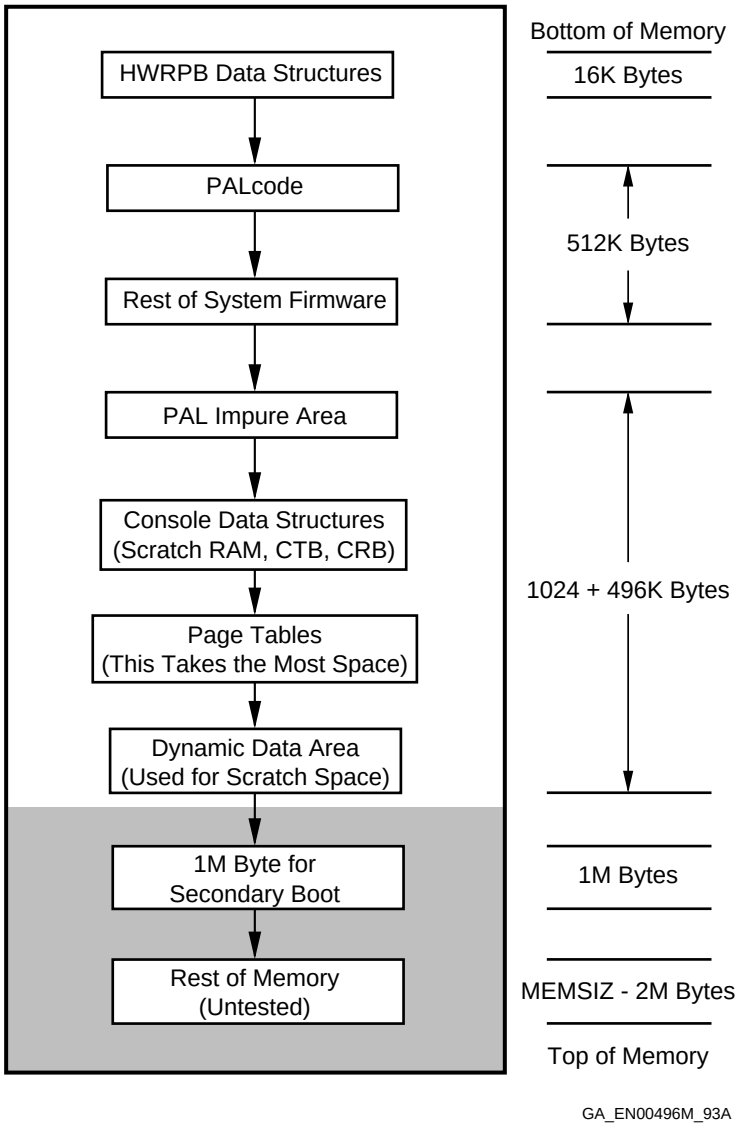
- 2M bytes are needed for the 512K bytes of system firmware and any data that it requires
- 1M byte is used to load in the secondary boot program

The 3M bytes of memory are always at the bottom 3M bytes of good memory. Figure 10–2 shows a map of this memory.

Note

The following map applies to OSF/1 and OpenVMS systems only.

Figure 10–2 Map of Memory Following Power-Up Initialization



Part II

DECchip 21064 CPU Overview

Part II provides an overview of the AXP data types and the DECchip 21064 CPU registers and functions. For more information on the DECchip 21064 CPU see the following manuals:

- *Alpha AXP Architecture Handbook* (EC-H1689-10)
- *Alpha AXP Architecture Reference Manual* (EK-VAXAR-RM)

This part includes the following chapters.

- Chapter 11
- Chapter 12, I-Box Internal Processor Registers
- Chapter 13, A-Box Internal Processor Registers
- Chapter 14, PAL Temporary Registers
- Chapter 15, CPU Cycle Types, Transactions, and Initialization

11

Alpha AXP Architecture

Introduction

This chapter gives an overview of the Alpha AXP architecture.

In This Chapter

This chapter contains the following sections:

- AXP Addressing and Data Types
- Byte and Word Data Types
- Longword and Quadword Data Type
- F_Floating Floating Point Format
- G_Floating Floating Point Format
- D_Floating Floating Point Format
- S_Floating Floating Point Format
- Other Data Type Information
- Alpha AXP Registers
- Alpha AXP Instruction Formats

AXP Addressing and Data Types

Addressing

The Alpha AXP architecture uses the following values for addressing:

- The 8-bit byte is the basic addressable unit in the Alpha AXP architecture.
- Virtual addresses are 64 bits long.
- An implementation of the Alpha AXP architecture can support a virtual address smaller than 64 bits long.
- The minimum virtual address is 43 bits long.
- Virtual addresses used by software are translated into physical memory addresses by the memory management mechanism.

Data types and Floating Point Formats

The following sections describe the data types and floating point formats supported by the Alpha AXP architecture. The data types are:

- Byte
- Word
- Longword
- Quadword

The floating point formats are:

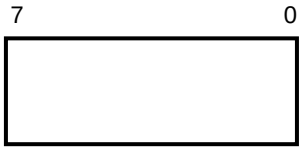
- F_floating
- G_floating
- D_floating
- S_floating

Byte and Word Data Types

Byte

A byte is 8 contiguous bits starting on an addressable byte boundary. The bits are numbered from right to left, 0 to 7 (see Figure 11–1).

Figure 11–1 Byte Data Format



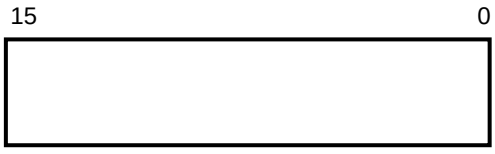
GA_EN00400M_93A

A byte is specified by its address A. A byte is an 8-bit value. The byte is supported in the Alpha AXP architecture only by the EXTRACT, MASK, INSERT, and ZAP instructions.

Word

A word is 2 contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 to 15 (see Figure 11–2).

Figure 11–2 Word Data Format



GA_EN00401M_93A

A word is specified by its address, which is the address of the byte containing bit 0. A word is a 16-bit value. The word is supported in the Alpha AXP architecture only by the EXTRACT and INSERT instructions.

Longword and Quadword Data Type

Longword

A longword is 4 contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left 0 to 31 (see Figure 11–3).

A longword is specified by its address A, which is the address of the byte containing bit 0. A longword is a 32-bit value. When interpreted arithmetically, a longword is a two's-complement integer with bits of increasing significance going from 0 to 30. Bit 31 is the sign bit. The longword is supported in Alpha AXP architecture only by sign-extended load and store instructions, and by longword arithmetic instructions.

Quadword

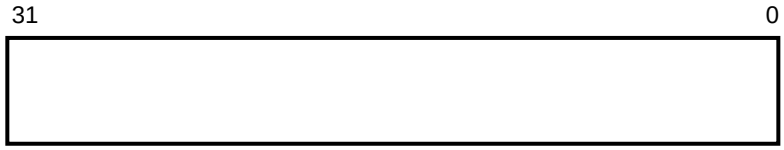
A quadword is 8 contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 to 63 (see Figure 11–4).

A quadword is specified by its address A, which is the address of the byte containing bit 0. A quadword is a 64-bit value. When interpreted arithmetically, a quadword is either a two's-complement integer with bits of increasing significance going from 0 to 62 and bit 63 as the sign bit, or an unsigned integer with bits of increasing significance going from 0 to 63.

Longword Data Format

Figure 11–3 shows the format of the longword data type.

Figure 11–3 Longword Data Format

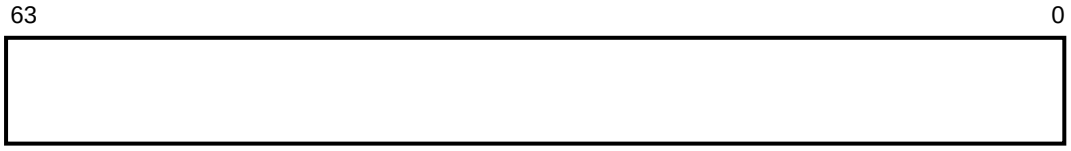


GA_EN00402M_93A

Longword Data Format

<REFERENCE>(qwf_fig) shows the format of the quadword data type.

Figure 11–4 Quadword Data Format



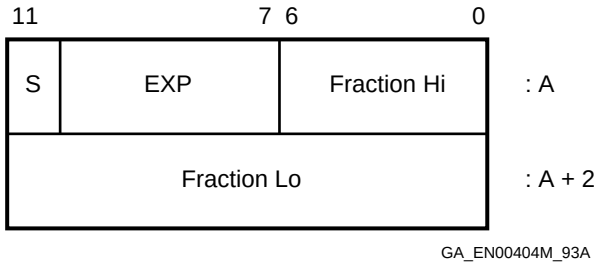
GA_EN00403M_93A

F_Floating Floating Point Format

F_Floating

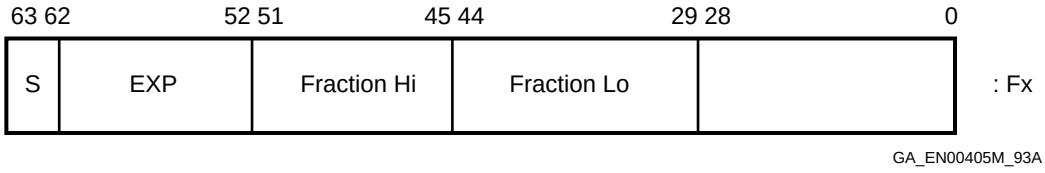
An F_floating datum is 4 contiguous bytes in memory starting on an arbitrary byte boundary. The bits are labeled from right to left, 0 to 31 (see Figure 11–5).

Figure 11–5 F_Floating Data Format



An F_floating operand occupies 64 bits in a floating register, left-justified in the 64-bit register (see Figure 11–6).

Figure 11–6 F_Floating Register



The F_floating load instruction does the following:

- Reorders bits from memory
- Expands the exponent from 8 to 11 bits
- Sets the low-order fraction bits to 0

The F_floating load instruction produces an equivalent G_ floating number in the register, which is suitable for either F_floating or G_floating operations. The mapping from 8-bit memory-format exponents to 11-bit register-format exponents is shown in Table 11–1.

Table 11–1 Alpha AXP F-Floating Load Exponent Mapping

Memory	Register
1 1111111	1 000 1111111
1 xxxxxxx	1 000 xxxxxxx
0 xxxxxxx	0 111 xxxxxxx (xxxxxxx not all 1s)
0 0000000	0 000 0000000 (xxxxxxx not all 0s)

The F_floating load exponent mapping preserves both normal values and exceptional values. The F_floating store instruction reorders register bits to memory and does not check the low-order fraction bits. Register bits <61:59> and <28:0> are ignored by the store instruction.

An F_floating datum is specified by its address A, which is the address of the byte containing bit 0. The memory form of an F_floating datum is as follows:

- Sign magnitude with bit 15 as the sign bit
- Bits <14:7> are an excess-128 binary exponent
- Bits <6:0> and <31:16> are a normalized 24-bit fraction with the redundant, most-significant fraction bit not represented

In the normalized 24-bit fraction, bits of increasing significance go from 16 to 31 and 0 to 6. The 8-bit exponent field encodes the values 0 to 255. An exponent value of 0 with a sign bit of 0 indicates that the F_floating datum has a value of 0.

If the result of a floating-point instruction has a value of 0, the instruction always produces a datum with a sign bit of 0, an exponent of 0, and all fraction bits of 0. Exponent values of 1 to 255 indicate true binary exponents of -127 to 127. An exponent value of 0, with a sign bit of 1, is taken as a reserve operand. Floating-point instructions that process a reserve operand take an arithmetic exception.

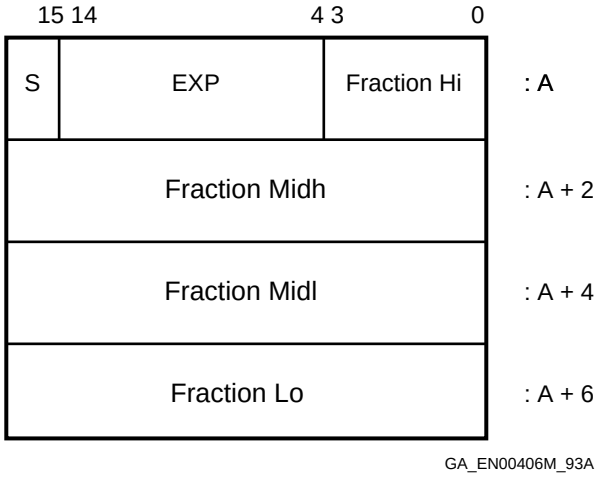
The value of an F_floating datum is in the approximate range of $0.29 * 10^{-38}$. The precision of an F_floating datum is approximately one part in 2^{23}; that is, typically 7 decimal digits.

G_Floating Floating Point Format

G_Floating

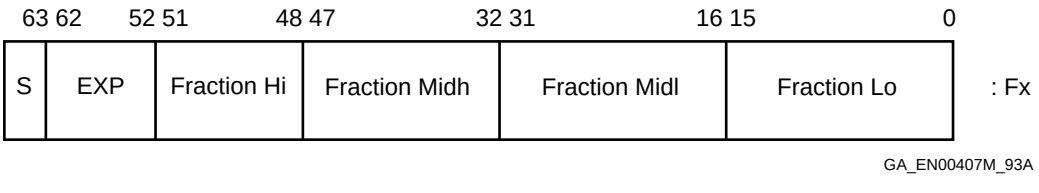
A G_floating datum in memory is 8 contiguous bytes starting on an arbitrary byte boundary. The bits are labeled from right to left, 0 to 15 (see Figure 11–7).

Figure 11–7 G_Floating Operand



A G_floating operand occupies 64 bits in a floating register, arranged as shown in Figure 11–8.

Figure 11–8 G_Floating Data Format



A G_floating datum is specified by its address A, which is the address of the byte containing bit 0. The form of a G_floating datum is as follows:

- Sign magnitude with bit 15 the sign bit
- Bits <14:4> are an excess-1024 binary exponent
- Bits <3:0> and <63:16> are a normalized 53-bit fraction with the redundant most significant fraction bit not represented

In the normalized 53-bit fraction, bits of increasing significance go from 48 to 63, 32 to 47, 16 to 31, and 0 to 3. The 11-bit exponent field encodes the values of 0 to 2047. An exponent value of 0 with a sign bit of 0 indicates that the G_floating datum has a value of 0.

If the result of a floating-point instruction has a value of 0, the instruction always produces a datum with a sign bit of 0, an exponent of 0, and all fraction bits of 0. Exponent values of 1 to 2047 indicate true binary exponents of -1023 to 1023. An exponent value of 0 with a sign bit of 1 is a reserve operand. Floating-point instructions that process a reserve operand have a user-visible arithmetic exception.

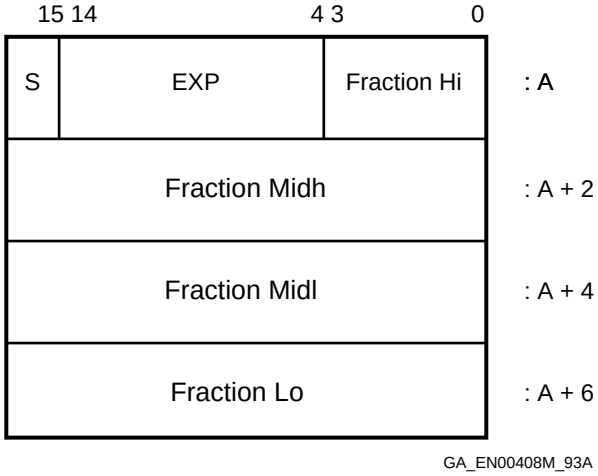
The value of a G_floating datum is in the approximate range of $0.56*10^{-308}$ to $0.9*10^{308}$. The precision of a G_floating datum is approximately one part in 2^{52}; that is, typically 15 decimal digits.

D_Floating Floating Point Format

D_Floating

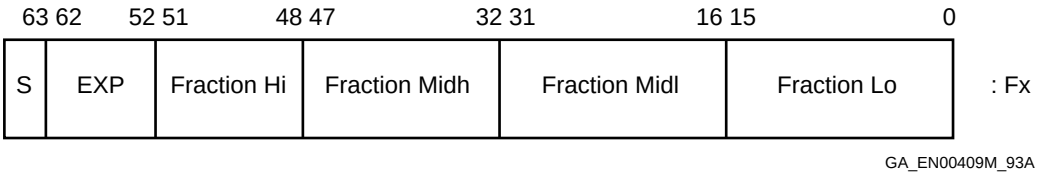
A D_floating datum in memory is 8 contiguous bytes starting on an arbitrary byte boundary. The bits are labeled from right to left, 0 to 63 (see Figure 11–9).

Figure 11–9 D_Floating Data Format



A D_floating operand occupies 64 bits in a floating register, arranged as shown in Figure 11–10.

Figure 11–10 D_Floating Register Format



The reordering of bits that is required for a D_floating load or store instruction is the same as the reordering of bits that is required for a G_floating load or store instruction. The G_

floating load and store instructions are therefore used for loading or storing D_floating data.

A D_floating datum is specified by its address A, which is the address of the byte containing bit 0. The memory form of a D_floating datum is identical to an F_floating datum except for an additional 32 low-significance fraction bits. Within the fraction, bits of increasing significance go from 48 to 63, 32 to 47, 16 to 31, and 0 to 6.

The exponent conventions and approximate range of values are the same for D_floating as they are for F_floating.

The precision of a D_floating datum is approximately one part in 2^{55}; that is, typically 16 decimal digits.

Note

D_floating is not a fully-supported data type; D_floating arithmetic operations are not provided in the Alpha AXP architecture. For backward compatibility, exact D_floating arithmetic can be provided by software emulation. D_floating format compatibility, in which binary files of D_floating numbers may be processed but without the last 3 bits of fraction precision, can be obtained by converting to G_floating, G arithmetic operations, and then converting back to D_floating.

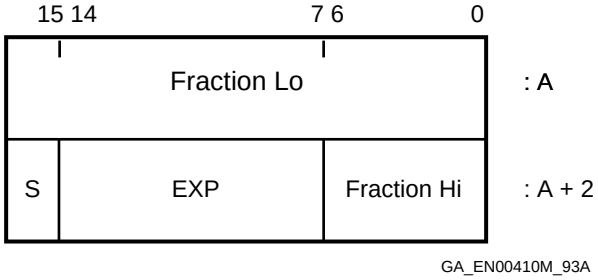
S_Floating Floating Point Format

S_Floating

An IEEE single precision or S_floating datum occupies 4 contiguous bytes in memory starting on an arbitrary byte boundary.

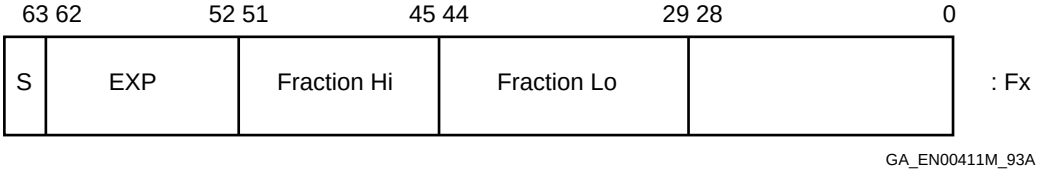
The bits are labeled from right to left, 0 to 31 (see Figure 11–11).

Figure 11–11 S_Floating Operand



An S_floating operand occupies 64 bits in a floating register, left-justified in the 64-bit register, as shown in Figure 11–12.

Figure 11–12 S_Floating Register Format



The S_floating load instruction reorders bits from memory, expanding the exponent from 8 to 11 bits, and sets the low-order fraction bits to 0. This produces an equivalent T_floating number in the register, which is suitable for either S_floating or T_floating operations. The mapping from 8-bit memory-format exponents to 11-bit register-format exponents is shown in Table 11–2.

Table 11–2 S_Floating Load Exponent Mapping

Memory	Register
1 1111111	1 111 1111111
1 XXXXXXX	1 000 XXXXXXX
0 XXXXXXX	0 111 XXXXXXX (XXXXXXX bits are not all 1s)
0 0000000	0 000 0000000 (XXXXXXX bits are not all 0s)

The mapping from 8-bit memory-format exponents to 11-bit register-format exponents preserves both normal values and exceptional values. The mapping for all 1s differs from that of F_floating load, because for S_floating all 1s is an exceptional value and for F_floating all 1s is a normal value.

The S_floating store instruction reorders register bits on the way to memory and does not check the low-order fraction bits. Register bits <61:59> and <28:0> are ignored by the store instruction. The S_floating load instruction does not check the input. The S_floating store instruction does not check the data. The preceding operation must specify an S_floating result.

An S_floating datum is specified by its address A, which is the address of the byte containing bit 0. The memory form of an S_floating datum is as follows:

- Sign magnitude with bit 31 the sign bit
- Bits <30:23> are an excess-127 binary exponent
- Bits <22:0> are a 23-bit fraction

The value *V* of an S_floating number is inferred from its constituent sign *S*, exponent *E*, and fraction *F* fields as follows:

- If $E = 255$ and $F <> 0$, then *V* is NaN, regardless of *S*
- If $E = 255$ and $F = 0$, then $V = (-1)^S$ * infinity
- If $0 < E < 255$, then $V = (-1)^S * 2^{(E-127)} * (1.F)$
- If $E = 0$ and $F <> 0$, then $V = (-1)^S * 2^{(-126)} * (0.F)$
- If $E = 0$ and $F = 0$, then $V = (-1)^S * 0$ (0)

Floating-point operations on S_floating numbers may take an arithmetic exception for a number of different reasons, including invalid operations, overflow, underflow, division by 0, and inexact results.

Other Data Type Information

Data Types with No Hardware Support

The following data types are not supported by the hardware in the Alpha AXP architecture:

- Octaword
- H_floating
- D_floating, except for load and store instructions, and converting to or from G_floating
- Variable length bit field
- Character string
- Trailing numeric string
- Leading separate numeric string
- Packed decimal string

Data Type Performance Penalties

The Alpha AXP architecture imposes a significant performance penalty when accessing certain data-type operands that are not naturally aligned as follows:

- A naturally aligned longword has 0 as the low-order 2 bits of its address.
- A naturally aligned quadword has 0 as the low-order 3 bits of its address.
- A naturally aligned F_floating datum has 0 as the low-order 2 bits of its address.
- A naturally aligned G_floating datum has 0 as the low-order 3 bits of its address.
- A naturally aligned D_floating datum has the low-order 3 bits of its address 0.
- A naturally aligned S_floating datum has 0 as the low-order 2 bits of its address.

Alpha AXP Registers

The following sections describe the Alpha AXP registers.

Program Counter Register

The program counter register is a special register that addresses the instruction stream. As each instruction is decoded, the program counter is advanced to the next sequential instruction. This is referred to as the updated program counter. Any instruction that uses the value of the program counter uses the updated program counter. The program counter includes only bits <63:2>, with bits <1:0> treated as RAZ or IGN. This quantity is a longword-aligned byte address. The program counter is an implied operand on conditional branch and subroutine jump instructions. The program counter is not accessible as an integer register.

Processor Status Register

The processor status register is a special register that contains the current status of the processor. It can be read by all CALL_PAL RD_PS routines. The PS<SW> field can be written to by a CALL_PAL WR_PS SW routine.

Integer Registers

There are 32 integer registers (R0 to R31), each 64 bits wide.

The following registers are assigned a special meaning by the Alpha AXP architecture:

- **R30**– The R30 register is the stack pointer. The stack pointer contains the address of the top of the stack in the current mode.

  Certain PALcode (for example, REI) uses R30 as an implicit operand. During such operations, the address value in R30, interpreted as an unsigned 64 bit integer, decreases (predecrements) when items are pushed onto the stack, and increases (postincrements) when they are popped from the stack. After pushing (writing) an item to the stack, the stack pointer points to that item.

- **R31**— When R31 is specified as a register source operand, a 0-valued operand is supplied. There is one exception: the results of an instruction that specifies R31 as a destination operand is discarded and it is unpredictable whether the other destination operands (implicit and explicit) are changed

by the instructions. In this case, how the instruction is executed when it has been fetched depends on the implementation. Also, it is unpredictable whether exceptions are signaled during the execution of such an instruction. The exceptions associated with the instruction fetch of such an instruction are always signaled. The exception to the above rule is for the following branch instructions when R31 is specified as the Ra operand: the unconditional branch (BR and BSR) and jump to subroutine (JMP, JSR, RET, and JSR_COROUTINE) instructions. These instructions execute normally and update the program counter with the target virtual address when R31 is specified as the Ra operand. No program counter value can be saved in R31. Applying the previous rule, the following are interesting cases involving R31 as a destination:

- STx_C R31, disp(Rb)

  Although this might seem like a good way to write zeros (zero-out) to a shared location and reset the lock_flag, this instruction causes the lock_flag and virtual location {Rbv + SEXT(disp)} to become unpredictable.

- LDx_L R31,disp(Rb)

  This instruction does not produce any useful results because it causes both lock_flag and locked_physical_ address to become unpredictable.

Floating-Point Registers

There are 32 floating-point registers (FO to F31), each 64 bits wide. When F31 is specified as a register source operand, a true 0-valued operand is supplied. The results of an instruction that specifies F31 as a destination operand are discarded, and it is unpredictable whether the other destination operands (implicit and explicit) are changed by the instruction. In this case, how the instruction is executed when it has been fetched depends on the implementation. Also, it is unpredictable whether exceptions are signaled during the execution of such an instruction. The exceptions associated with the instruction fetch of such an instruction are always signaled.

A floating-point instruction that operates on single-precision data reads all bits <63:0> of the source floating-point register. A floating-point instruction that produces a single-precision result writes all bits <63:0> of the destination floating-point register.

Lock Registers

The Alpha AXP architecture specifies two per-processor registers associated with the LDx_L and STx_C instructions, the lock_flag, and the locked_physical_address registers. These registers are not implemented in the system.

Internal Processor Registers

There are a number of internal processor registers with specialized uses that are available only to privileged software that uses the MTPR and MFPR PAL code routines.

Alpha AXP Instruction Formats

The five basic instruction formats in the Alpha AXP architecture are as follows:

- Memory
- Branch
- Operate
- Floating-point operate
- PAL code

All instruction formats are 32 bits long with a 6-bit major opcode field in bits <31:26> of the instruction.

12

I-Box Internal Processor Registers

Introduction

This chapter describes the I-box in the DECchip 21064 CPU.

In This Chapter

This chapter contains the following sections:

- I-Box Functions
- TB_TAG Register
- ITB_PTE Register
- ITB_PTE_TEMP and Other ITB Registers
- ICCSR Register
- EXC_ADDR Register
- EXC_SUM Register
- SL_CLR Register
- SL_RCV Register
- SL_XMIT Register
- Processor Status Register
- PAL_BASE Register
- HIRR Register
- SIRR Register
- ASTRR Register
- HIER Register
- SIER Register
- ASTER Register

I-Box Functions

The primary function of the I-box is to issue instructions to the E-box, A-box, and F-box. The I-box decodes two instructions in parallel and checks that the required resources are available for both instructions. The following sections describe the I-box registers:

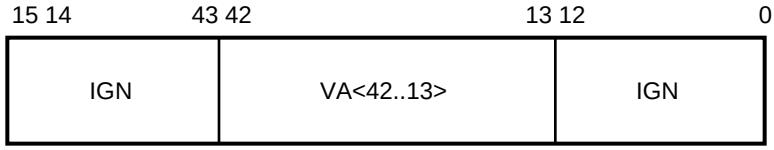
TB_TAG Register

The translation buffer tag register (TB_TAG) is a write-only register that holds the tag for the next translation buffer (TB) update operation in either the instruction translation buffer (ITB) or data translation buffer (DTB). To ensure the integrity of the TB, the tag is written to a temporary register and not transferred to the ITB or DTB until the ITB page table entry (ITB_PTE) or the DTB page table entry (DTB_PTE) register is written to. The entry that is written to is chosen at the time of the ITB_PTE or DTB_PTE write operation by a not-last-used algorithm implemented in the hardware.

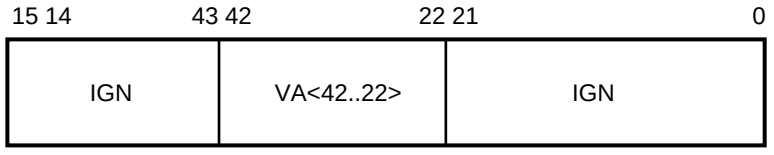
The ITB_TAG register is written to only while in PAL mode regardless of the state of the hardware enable (HWE) bit in the instruction cache control and status register (ICCSR).

Figure 12–1 TB_TAG Register Format

Small Page Format



GH = 11_2 Format (DTB only)



GA_EN00412M_93A

ITB_PTE Register

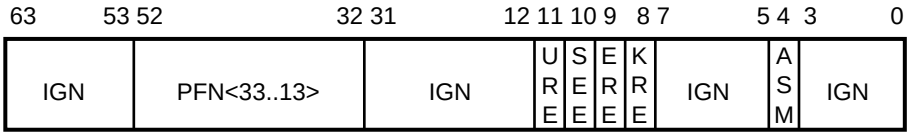
The instruction translation buffer page table entry register (ITB_PTE) is a read/write register representing the eight ITB page table entries. The entry to be written to is chosen by a not-last-used algorithm implemented in hardware. Writes to the ITB_PTE register use the memory format bit positions, which are described in the *Alpha AXP System Reference Manual*, except for some fields that are ignored.

To ensure the integrity of the ITB, the ITB's tag array is updated from the internal tag register when the ITB_PTE register is written to. To read from the ITB_PTE register requires two instructions. First, a read from the ITB_PTE register sends the PTE data to the ITB_PTE_TEMP register. Second, an instruction reading from the ITB_PTE temporary (ITB_PTE_ TEMP) register returns the PTE entry to the register file. Reading from or writing to the ITB_PTE register increments the TB entry pointer, which allows the entire set of eight ITB_PTE register entries to be read from.

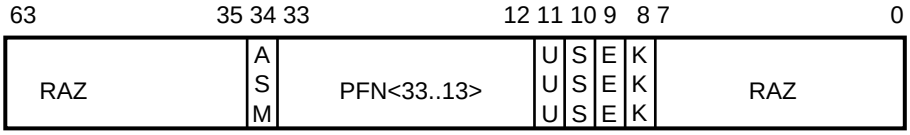
The ITB_PTE register is read from and written to only in PAL mode, regardless of the state of the HWE bit in the ICCSR register.

Figure 12–2 ITB_PTE Register Format

Write Format



Read Format

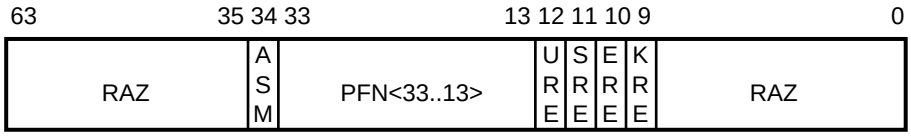


GA_EN00413M_93A

ITB_PTE_TEMP and Other ITB Registers

The instruction translation buffer page table entry temporary register (ITB_PTE_TEMP) is a read-only holding register for read data in the ITB_PTE register. Reads from the ITB_PTE register require two instructions to return the data to the register file. The first reads the ITB_PTE register to the ITB_PTE_TEMP register. The second returns the ITB_PTE_TEMP register to the integer register file. The ITB_PTE_TEMP register is updated on all ITB accesses, both read and write. A read from the ITB_PTE register to the ITB_PTE_TEMP register must be followed closely by a read from the ITB_PTE_TEMP register to the register file. The ITB_PTE_TEMP register is read-only while in PAL mode, regardless of the state of the HWE bit in the ICCSR IPR.

Figure 12–3 ITB_PTE_TEMP Register Format



GA_EN00415M_93A

ITB_ZAP Register

Writing any value to the instruction translation buffer zap register (ITB_ZAP) invalidates all eight ITB entries. It also resets the not-last-used (NLU) pointer to its initial state. The ITB_ZAP register must be written to only in PAL mode.

ITB_ASM Register

Writing any value to the instruction translation buffer ASM ITB_ASM register invalidates all ITB entries in which the ASM bit is equal to 0. The ITB_ASM register must be written to only in PAL mode.

ITB_IS Register

Writing any value to the instruction translation buffer IS register (ITB_IS) invalidates all eight ITB entries. It also resets the not-last-used (NLU) pointer to its initial state. The ITB_IS register must be written to only in PAL mode.

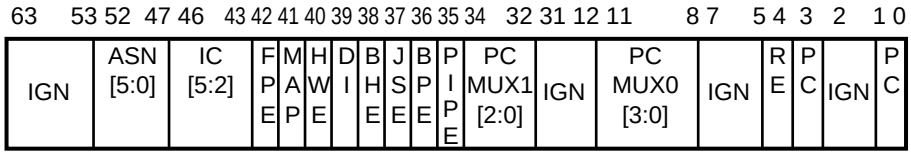
ICCSR Register

The instruction cache control and status register (ICCSR) contains various I-box hardware enables. The only architecturally defined bit in this register is the floating-point enable (FPE) bit, which enables floating-point instruction execution. When clear, all floating-point instructions generate FEN exceptions. This register is cleared by hardware at reset.

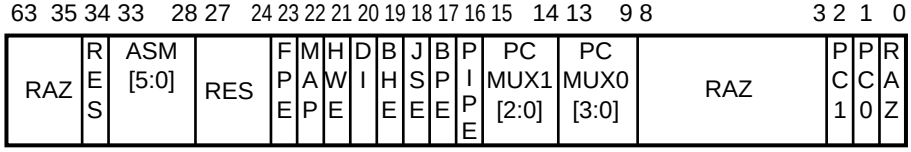
The hardware enable (HWE) bit enables special PAL instructions to execute in kernel mode. This bit is intended for diagnostics or operating system alternative PAL routines only. The HWE bit does not allow access to the ITB registers outside PAL mode. Therefore, some PAL code flows may require the PAL mode environment to execute properly (for example, ITB fill). Figure 12–4 shows the format of the ICCSR register.

Figure 12–4 ICCSR Register Format

Write Format



Read Format



GA_EN00414M_93A

ICCSR Register Fields

Table 12–1 shows the type and description of each ICCSR register field.

Table 12–1 ICCSR Register Fields

Field	Type	Description
FPE	R/W	If set, floating point instructions can be issued. If clear, floating point instructions cause FEN exceptions.
MAP	R/W	If set, allows super-page instruction stream memory mapping of VPC<33:13> directly to physical PC<33:13> essentially bypassing ITB for VPC addresses containing VPC<42:41>=2. Super page mapping is allowed in kernel mode only. The ASM bit is always set.
HWE	R/W	If set, allows the five PALRES instructions to be issued in kernel mode. Using the HW_MTPR instruction to update the EXC_ADDR IPR while in native mode is restricted to values with bit<0> equal to 0. The combination of native mode and EXC_ADDR<0> equal to 1 causes undefined behavior.
DI	R/W	If set, enables dual issue.
BHE	R/W	Used with BPE. See Table 12–2 for programming information.
JSE	R/W	If set, enables the JSR stack to push return addresses.
BPE	R/W	Used with BHE. See table Table 12–2 for programming information.
VAX	R/W	If clear, causes all hardware-interlocked instructions to drain the machine and waits for the write buffer to empty before issuing the next instruction. Examples of instructions that do not cause the pipe to drain include HW_MTPR, HW_REI, conditional branches, and instructions that have a destination register of R31.
PCMUX1	R/W	See Table 12–4 for programming information.
PCMUX0	R/W	See Table 12–3 for programming information.
PC1	R/W	If set, enables a performance counter 1 interrupt request after 2^8 events are counted. If clear, enables a performance counter 1 interrupt request after 2^{12} events are counted.

(continued on next page)

Table 12–1 (Cont.) ICCSR Register Fields

Field	Type	Description
PC0	R/W	If set, enables a performance counter 0 interrupt request after 2^{12} events are counted. If clear, enables a performance counter 0 interrupt request after 2^{16} events are counted.
ASN	R/W	The address space number field is used with the instruction cache (I-cache) in the DECchip 21064 CPU to further qualify cache entries and avoid some cache flushes. The ASN bit field is written to the instruction cache during fill operations and compared with the instruction stream data on fetch operations. Mismatches invalidate the fetch without affecting the I-Cache.
IC	R/W	The IC state bits are unused by hardware.

BHE and BPE Branch Prediction Selection

Table 12–2 shows the BHP and BHE bit values used for branch prediction.

Table 12–2 BHE and BPE Branch Prediction Selection

BPE	BHE	Prediction
0	X	Not taken
1	0	Sign of displacement
1	1	Branch history table

Performance Counters

Performance counters are reset to 0 at power-up, but are otherwise never cleared. They are intended for counting events over a long period relative to the event frequency, so they do not provide a means of extracting intermediate counter values. Because the counters continuously accumulate selected events despite interrupts being enabled, the first interrupt after selecting a new counter input has an error bound as large as the selected overflow range.

Some inputs may overcount events occurring simultaneously with data stream (D-stream) errors that abort the actual event very late in the pipeline. For example, when counting load instructions, attempts to execute a load that result in a DTB miss exception, increment the performance counter after the first aborted execution attempt. The performance counter is incremented again after the TB fill routine when the load instruction reissues and completes.

Performance counter interrupts are reported six cycles after the event that caused the counter to overflow. Additional delay in servicing interrupts may occur if the processor is executing PAL code that always disables interrupts. In either case, events occurring during the interval between counter overflow and interrupt service are counted towards the next interrupt. An interrupt can be missed only when a complete counter wraparound occurs while interrupts are disabled.

Because there are 6 cycles before an interrupt is triggered, up to 12 instructions may have completed before the start of the interrupt service routine. In most cases, it is possible to further isolate trigger events by examining the possible intervening instructions. Two cases always provide a more accurate exception program counter.

When counting instruction cache misses, no intervening instructions can complete and the exception program counter contains the address of the last instruction cache miss. Branch mispredictions allow a maximum of two instructions to complete before the start of the interrupt service routine.

Performance Counter 0

Table 12–3 lists the performance counter 0 input selection.

Table 12–3 Performance Counter 0 Input Selection

MUX0<3:0>	Input	Comment
000X	Total issues/2	Counts the total issues divided by 2; for example, a dual issue increments the count by 1.
001X	Pipeline dry	Counts the cycles when instructions are not being issued because of a shortage of valid instruction stream data. Causes include I-Cache fill, misprediction, branch delay slots, and pipeline drain for an exception.
010X	Load instructions	Counts all load instructions.
011X	Pipeline frozen	Counts the cycles when instructions are not being issued because of a resource conflict.
100X	Branch instructions	Counts all branch instructions, conditional, unconditional, any JSR, and HW_REI.
1010	PAL mode	Counts cycles while executing in PAL mode.
1011	Total cycles	Counts the total cycles.
110X	Total non-issues/2	Counts the total non-issues divided by 2; that is, no issue increments count by 1.
111X	PERF_CNT_H<0>	Counts the external events supplied by a pin at the selected system clock cycle interval.

Performance Counter 2

Table 12–4 lists the performance counter 1 input selection.

Table 12–4 Performance Counter 1 Input Selection

MUX1<2:0>	Input	Comment
000	D-cache miss	Counts the total data cache misses.
001	I-cache miss	Counts total instruction cache misses.
010	Dual issues	Counts the cycles of dual issue instructions.
011	Branch mispredicts	Counts both conditional branch mispredictions and JSR or HW_REI mispredictions. Conditional branch mispredictions cost 4 cycles, and others cost 5 cycles of dry pipeline delay.
100	FP instructions	Counts the total floating-point operate instructions: no FP branch, load, store.
101	Integer operate	Counts the integer operate instructions including LDA and LDAH with a destination other than R31.
110	Store instructions	Counts the total store instructions.
111	PERF_CNT_ H<1>	Counts the external events supplied by a pin at the selected system clock cycle interval.

EXC_ADDR Register

The exception address register (EXC_ADDR) is a read/write register used to restart the machine after exceptions or interrupts. The EXC_ADDR register can be read from and written to by software by using the HW_MTPR instruction, and can be read from and written to directly by hardware. The HW_REI instruction executes a jump to the address contained in EXC_ADDR.

The EXC_ADDR register is written to by hardware after an exception to provide a return address for PAL code. The instruction pointed to by the EXC_ADDR register did not complete execution. Because the program counter is longword aligned, the least significant bit (LSB) of EXC_ADDR is used to indicate PAL mode to the hardware. When the LSB is clear, the HW_REI instruction executes a jump to native (non-PAL) mode, enabling address translation.

The CALL_PAL exceptions load the EXC_ADDR with the program counter of the instruction following the CALL_PAL exception. This function allows CALL_PAL service routines to return without needing to increment the value in EXC_ADDR.

This feature requires careful treatment in PAL code. Arithmetic traps and machine check exceptions can pre-empt CALL_PAL exceptions resulting in an incorrect value being saved in the EXC_ADDR register. In the case of an arithmetic trap or a machine check exception, and only in these cases, EXC_ADDR<1> takes on special meaning. PAL code servicing these two exceptions must interpret a 0 in EXC_ADDR<1> as indicating that the program counter in EXC_ADDR<63:2> is too large by a value of 4 bytes and subtract 4 before executing a HW_REI from this address. PAL code must interpret a 1 in EXC_ADDR<1> as indicating that the program counter in EXC_ADDR<63:2> is correct, and clear the value of EXC_ADDR<1>. All other PAL code entry points, except reset, can expect EXC_ADDR<1> to be 0.

This logic allows the following code sequence to conditionally subtract 4 from the address in the EXC_ADDR register without using an additional register. This code sequence must be present in arithmetic trap and machine check flows only.

Example 12–1 Exception Address Code

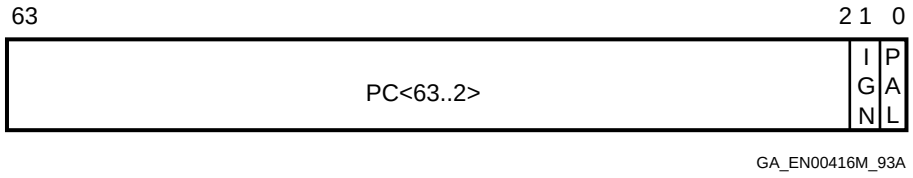
```
HW_MFPR       Rx, EXC_ADDR  ; read EXC_ADDR into GPR
SUBQ          Rx, #2, Rx    ; subtract 2 causing borrow if [1]=0
BIC           Rx, #2, Rx    ; clear
HW_MTPR       Rx, EXC_ADDR  ; write back to EXC_ADDR
```

Note that bit<1> is undefined when the EXC_ADDR is read. The hardware ignores this bit; however, PAL code must explicitly clear this bit before it pushes the exception address on the stack.

EXC_ADDR Format

Figure 12–5 shows the format of the EXC_ADDR register.

Figure 12–5 EXC_ADDR Register Format



GA_EN00416M_93A

EXC_SUM Register

The exception summary register (EXC_SUM) records the various types of arithmetic traps that have occurred since the last time that the EXC_SUM register was written to (cleared). When the result of an arithmetic operation produces an arithmetic trap, the corresponding EXC_SUM register bit is set.

In addition, the register containing the result of that operation is recorded in the exception register write mask as a single bit in a 64-bit field specifying registers F31-F0 and I31-I0. The exception register is visible only through the EXC_SUM register. The EXC_SUM register provides a 1-bit window to the exception register write mask. Each read from the EXC_SUM register shifts 1 bit in the order F31-F0, then I31-I0. The read also clears the corresponding bit. Therefore, the EXC_SUM register must be read 64 times to extract the complete mask and clear the entire register.

Any write to the EXC_SUM register clears bits <8..2> and does not affect the write mask.

The write mask register bit clears three cycles after a read. Therefore, code that reads from the register must allow at least three cycles between reads to allow the clear and shift operation to complete and to ensure that successive bits are read.

Figure 12–6 shows the format of the EXC_SUM register.

Figure 12–6 EXC_SUM Register Format

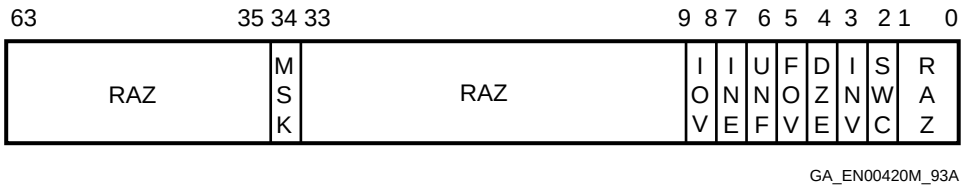


Table 12–5 lists the EXC_SUM register fields.

Table 12–5 EXC_SUM Register Fields

Field	Type	Description
SWC	WA	Indicates that software completion is possible. The bit is set after a floating-point instruction that contains the /S modifier completes with an arithmetic trap, if all previous floating point instructions that trapped since the last MTPR EXC_SUM also contained the /S modifier. The SWC bit is cleared when a floating point instruction without the /S modifier completes with an arithmetic trap. The bit remains cleared regardless of additional arithmetic traps until the register is written to using an MTPR instruction. The bit is always cleared on any MTPR write to the EXC_SUM register.
INV	WA	Indicates an invalid operation
DZE	WA	Indicates a divide by 0
FOV	WA	Indicates a floating point overflow
UNF	WA	Indicates a floating point underflow
INE	WA	Indicates a floating inexact error
IOV	WA	Indicates an F-box convert to integer overflow or integer arithmetic overflow
MSK	RC	Exception register write mask IPR window

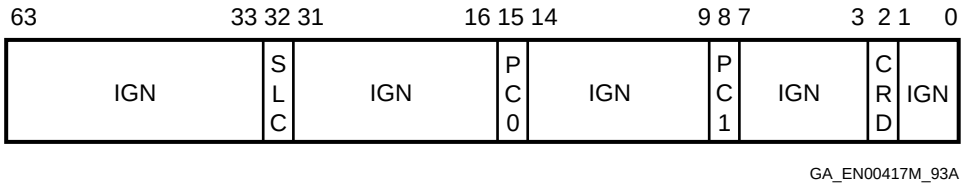
SL_CLR Register

The serial line clear register (SL_CLR) is a write-only register that clears the serial line interrupt request, the performance counter interrupt request, and the correctable read (CRD) interrupt request. Therefore, the write of any data to the SL_ CLR register clears the remaining serial line interrupt request. The DECchip 21064 CPU requires that the indicated bit is written to with a 0 to clear the selected interrupt source.

SL_CLR Format

Figure 12–7 shows the format of the SL_CLR register.

Figure 12–7 SL_CLR Register Format



SL_CLR Fields

Table 12–6 lists the SL_CLR register fields.

Table 12–6 SL_CLR Register Fields

Field	Type	Description
CRD	W0C	Clears the correctable read error interrupt request
PC1	W0C	Clears the performance counter 1 interrupt request
PC0	W0C	Clears the performance counter 0 interrupt request
SLC	W0C	Clears the serial line interrupt request

SL_RCV Register

The serial line receive register (SL_RCV) contains a single read-only bit (RCV) used with the interrupt control registers and the SROMD(H) and SROMCLK(H) pins to provide an on-chip serial line function.

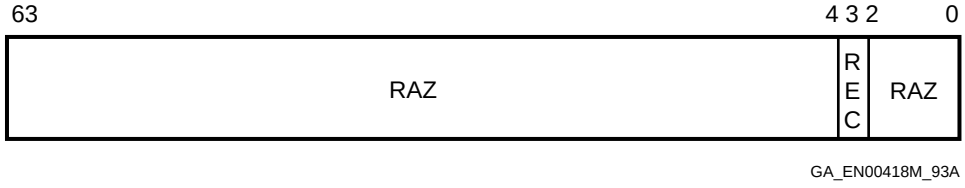
The RCV bit is functionally connected to the SROMD(H) pin after the instruction cache is loaded from the external serial ROM. The RCV bit can be read to receive external data 1 bit at-a-time under a software timing loop.

A serial line interrupt is requested on detection of any receive line transition that sets the serial line request (SL_REQ) bit in the hardware interrupt request register (HIRR). The serial line interrupt can be disabled by clearing the hardware interrupt enable register's (HIER's) serial line enable bit (SL_ENA).

SL_RCV Format

Figure 12–8 shows the format of the SL_RCV register.

Figure 12–8 SL_RCV Register Format

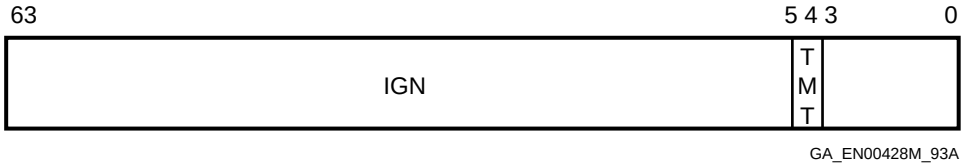


SL_XMIT Register

The serial line transmit register (SL_XMIT) contains a single write-only bit used with the interrupt control registers and the SROMD(H) and SROMCLK(H) pins to provide an on-chip serial line function.

Figure 12–9 shows the format of the SL_XMIT register fields.

Figure 12–9 SL_XMIT Register Format

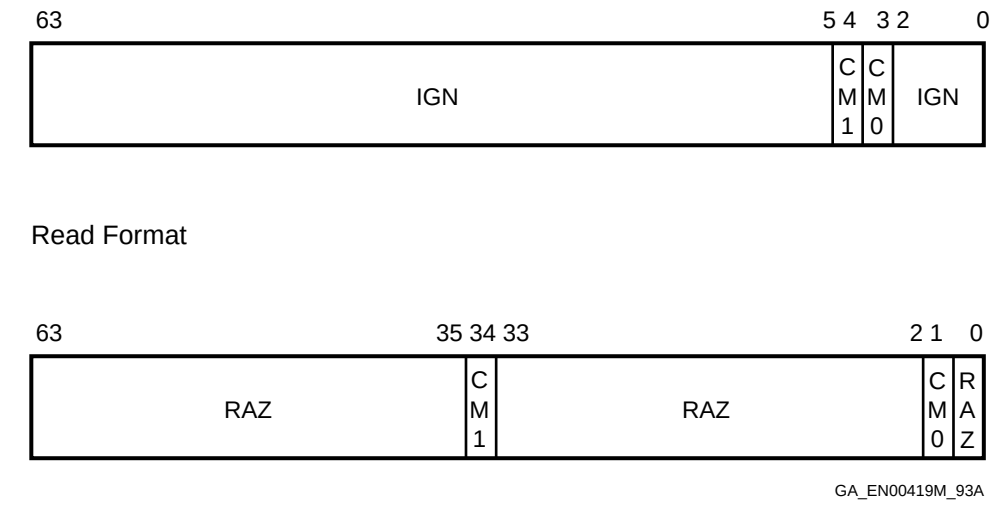


Processor Status Register

The processor status register is a read/write register containing only the current mode bits of the architecturally defined processor status.

Figure 12–10 shows the format of the processor status register.

Figure 12–10 Processor Status Register Format

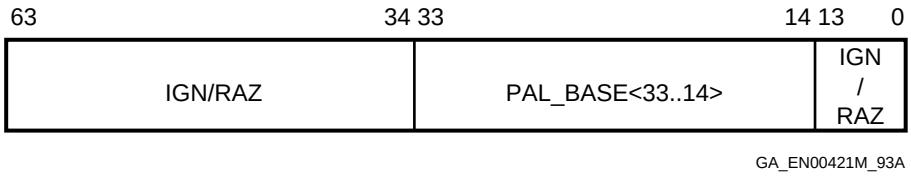


PAL_BASE Register

The privileged architecture library base register (PAL_BASE) is a read/write register containing the base address for PAL code. This register is cleared by hardware at reset.

Figure 12–11 shows the format of the PAL_BASE register.

Figure 12–11 PAL_BASE Register Format



HIRR Register

The hardware interrupt request register (HIRR) is a read-only register that provides a record of all currently outstanding interrupt requests and summary bits at the time of a read. For each bit of the HIRR <5:0>, there is a corresponding bit of the hardware interrupt enable register (HIER) that must be set to request an interrupt. In addition to returning the status of the hardware interrupt requests, a read from the HIRR returns the state of the software interrupt and asynchronous system trap (AST) requests. Note that a read from the HIRR may return a value of 0 if the hardware interrupt was released before the read (passive release). The register guarantees that the HWR bit reflects the status as shown by the HIRR bits. All interrupt requests are blocked while executing in PAL mode.

Figure 12–12 shows the format of the HIRR register.

Figure 12–12 HIRR Register Format

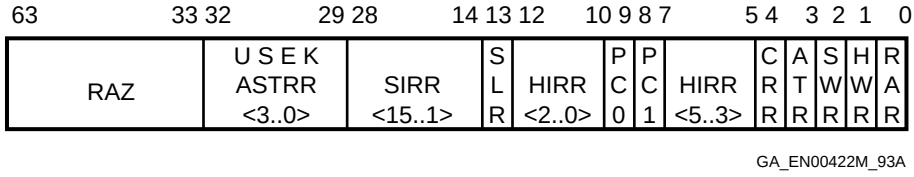


Table 12–7 lists the HIRR register fields.

Table 12–7 HIRR Register Fields

Field	Type	Description
HWR	RO	Is set if any hardware interrupt request and corresponding enable is set.
SWR	RO	Is set if any software interrupt request and corresponding enable is set.

(continued on next page)

Table 12–7 (Cont.) HIRR Register Fields

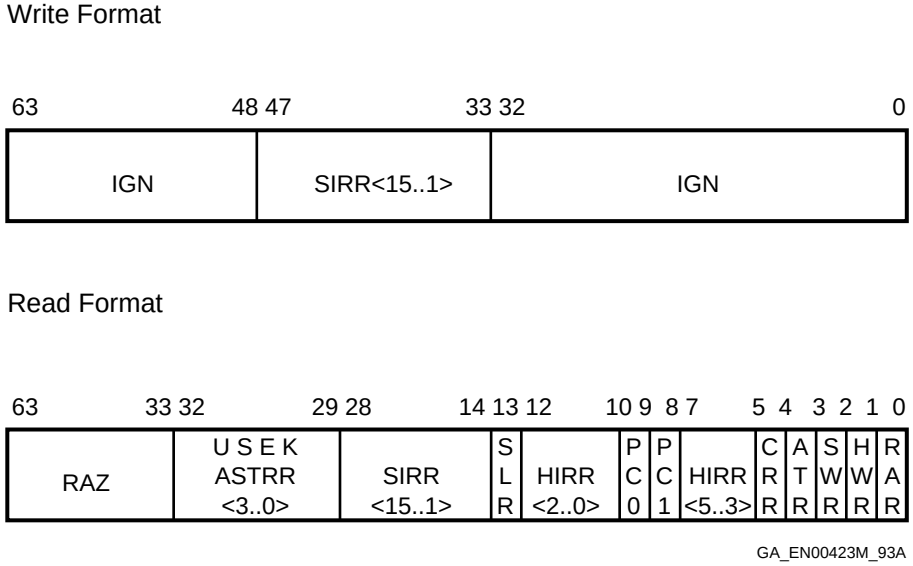
Field	Type	Description
ATR	RO	Is set if any AST request and corresponding enable is set. This bit also requires that the processor mode is equal to or higher than the request mode. In the DECchip 21064 CPU, SIER<2> must also be set to allow AST interrupt requests.
HIRR<5..0>	RO	Corresponds to pins IRQ(H)<5..0>.
SIRR<15..1>	RO	Corresponds to software interrupt requests 15 to 1.
ASTRR<3..0>	RO	Corresponds to AST requests 3 to 0 (USEK).
PC1	RO	Performance counter 1 interrupt request.
PC0	RO	Performance counter 0 interrupt request.
SLR	RO	Serial line interrupt request.
CRR	RO	CRD correctable read error interrupt request.

SIRR Register

The software interrupt request register (SIRR) is a read/write register that is used to control software interrupt requests. For each bit of the SIRR register, there is a corresponding bit of the software interrupt enable register (SIER) that must be set to request an interrupt. Reads from the SIRR register return the complete set of interrupt request registers and summary bits (see HIRR Register for more information). All interrupt requests are blocked while executing in PAL mode.

Figure 12–13 shows the format of the SIRR register fields.

Figure 12–13 SIRR Register Format



ASTRR Register

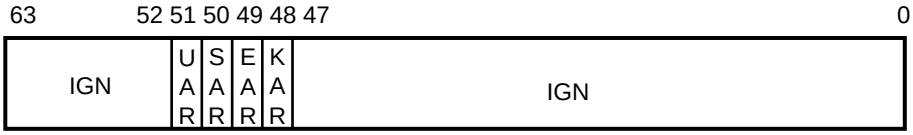
The asynchronous trap request register (ASTRR) is a read /write register. It contains bits to request AST interrupts in each of the processor modes. To generate an AST interrupt, the corresponding enable bit in the asynchronous trap enable register (ASTER) must be set and the processor must be in the selected processor mode or have a higher privilege as described by the current value of the processor status register CM bits (see Processor Status Register). In addition, AST interrupts are enabled in the DECchip 21064 CPU only if SIER<2> is set.

This process provides a mechanism to lock out AST requests over certain interrupt priority levels (IPLs). All interrupt requests are blocked while executing in PAL mode. Reads from the ASTRR register return the complete set of interrupt request registers and summary bits (see HIRR Register for more information).

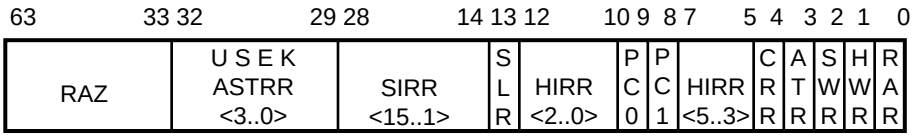
Figure 12–14 shows the format of the ASTRR register fields.

Figure 12–14 ASTRR Register Format

Write Format



Read Format



GA_EN00424M_93A

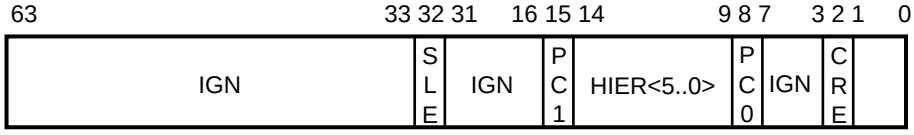
HIER Register

The hardware interrupt enable register (HIER) is a read/write register. It is used to enable corresponding bits of the HIRR register requesting interrupt. The PC0, PC1, SLE, and CRE bits of this register enable the performance counters, serial line, and correctable read interrupts. There is a one-to-one correspondence between the interrupt requests and enable bits. Reads from the HIER register are the same as reads from the interrupt request registers. Both return the complete set of interrupt enable registers (see HIRR Register for more information).

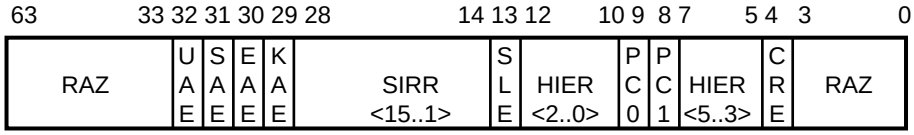
Figure 12–15 shows the format of the HIER register fields.

Figure 12–15 HIER Register Format

Write Format



Read Format



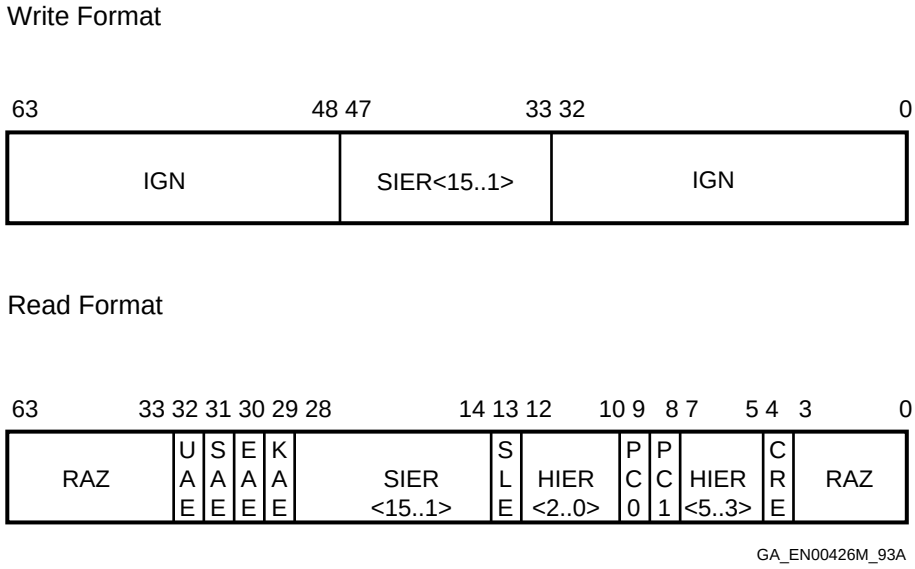
GA_EN00425M_93A

SIER Register

The software interrupt enable register (SIER) is a read/write register. It is used to enable the corresponding bits of the SIRR register that requests interrupts. There is a one-to-one correspondence between the interrupt request and the enable bits. Reads from the SIER register are the same as reads from the interrupt request registers. Both return the complete set of interrupt enable registers (see HIRR Register for more information).

Figure 12–16 shows the format of the SIER register fields.

Figure 12–16 SIER Register Format



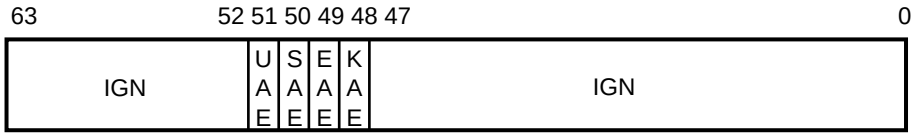
ASTER Register

The asynchronous system trap enable register (ASTER) is a read/write register. It is used to enable corresponding bits of the ASTRR register that requests interrupts. There is a one-to-one correspondence between the interrupt request and the enable bits. Reads from the ASTER register are the same as reads from the interrupt request registers. Both return the complete set of interrupt enable registers (see HIRR Register for more information).

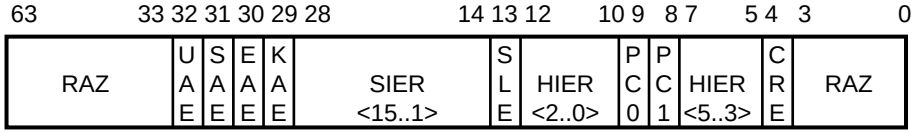
Figure 12–17 shows the format of the ASTER register fields.

Figure 12–17 ASTER Register Format

Write Format



Read Format



GA_EN00427M_93A

13

A-Box Internal Processor Registers

Introduction

This chapter describes the A-box in the DECchip 21064 CPU.

In This Chapter

This chapter contains the following sections:

- A-BOX Sections
- TB_CTL Register
- DTB_PTE Register
- DTB_PTE_TEMP Register
- MM_CSR Register
- ABOX_CTL Register
- ALT_MODE Register
- Cycle Counter Registers
- BIU_CTL Register
- <REFERENCE>(Oth_a_box)

A-BOX Sections

The A-box in the DECchip 21064 CPU contains six major sections:

- Address translation data path
- Load silo
- Write buffer
- Data cache interface
- External bus interface unit (BIU)
- Internal processor registers

The address translation data path has a displacement adder that generates the effective virtual address for the load and store instructions and a pair of translation buffers that generate the corresponding physical address. The following sections describe the registers contained in the DECchip 21064 CPU's A-box unit.

TB_CTL Register

The tag buffer control register (TB_CTL) is a write-only register. Figure 13–1 shows the format of the TB_CTL register fields.

Figure 13–1 TB_CTL Register Format

DTB_PTE Register

The data tag buffer page table entry register (DTB_PTE) is a read/write register representing the 32-entry small-page and 4-entry large-page data tag buffer page table entries. The entry to be written to is chosen by a not-last-used algorithm implemented in hardware and by the value in the data tag buffer control register (DTB_CTL). Writes to the DTB_PTE register use the memory format bit positions as described in the *Alpha AXP System Reference Manual*, except for some fields that are ignored. In particular, the valid bit is not represented in hardware.

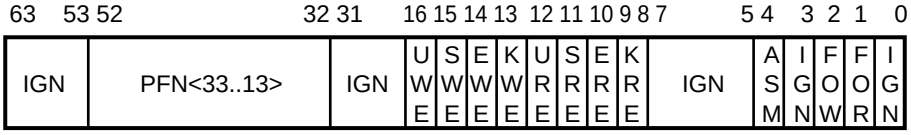
To ensure the integrity of the DTBs, the DTB's tag array is updated from the internal tag register when the DTB_PTE register is written to. Reads from the DTB_PTE register require the following two instructions:

1. The first instruction reads from the DTB_PTE register and sends the PTE data to the DTB_PTE_TEMP register.
2. The second instruction reads from the DTB_PTE_TEMP register and returns the PTE to the register file. Reading from or writing to the DTB_PTE register increments the TB entry pointer of the DTB indicated by the DTB_CTL register. This allows the entire set of DTB_PTE entries to be read.

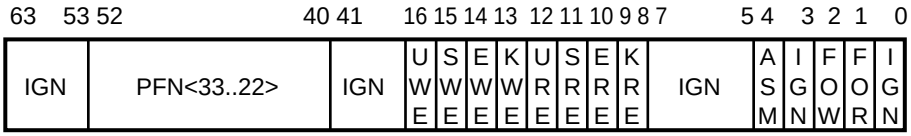
Figure 13–2 shows the format of the DTB_PTE register fields.

Figure 13–2 DTB_PTE Register Format

Write Format



Read Format



GA_EN00430M_93A

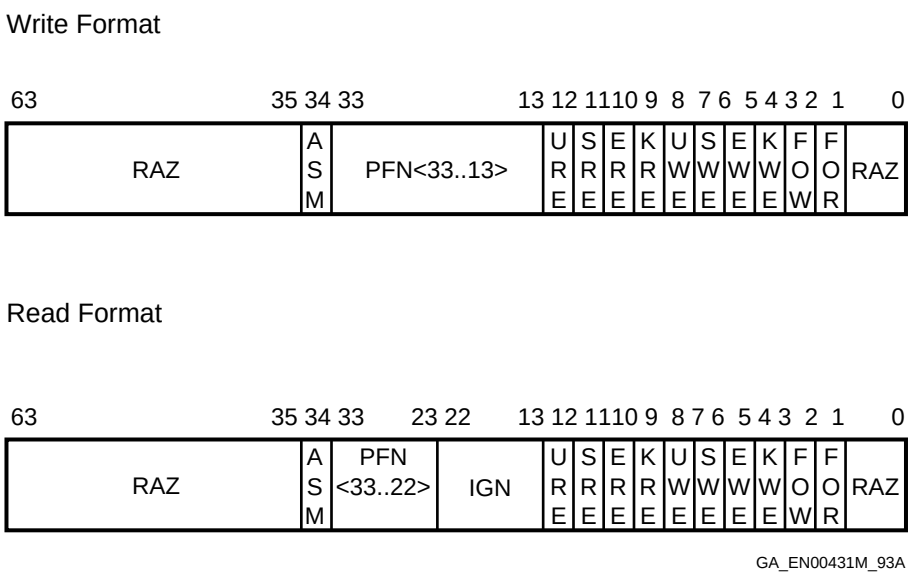
DTB_PTE_TEMP Register

The data tag buffer page table entry temporary register (DTB_PTE_TEMP) is a read-only holding register for the DTB_PTE register read data. Reads from the DTB_PTE register require the following two instructions to return the data to the register file:

1. The first instruction reads from the DTB_PTE register to the DTB_PTE_TEMP register.
2. The second instruction returns the DTB_PTE_TEMP register to the integer register file.

Figure 13–3 shows the format of the DTB_PTE_TEMP register fields.

Figure 13–3 DTB_PTE_TEMP Register Format



GA_EN00431M_93A

MM_CSR Register

When data-stream faults occur, the information about the fault is latched and saved in the memory management control and status register (MM_CSR). The virtual address (VA) register and the MM_CSR register are locked against further updates until the software reads from the VA register. PAL code must explicitly unlock the MM_SCR register when its entry point was higher in priority than a DTB miss. The MM_CSR register bits are modified only by hardware when the register is not locked and a memory management error or a DTB miss occurs. The MM_CSR register is unlocked after a reset.

Figure 13–4 shows the format of the MM_CSR register fields.

Figure 13–4 MM_CSR Register Format

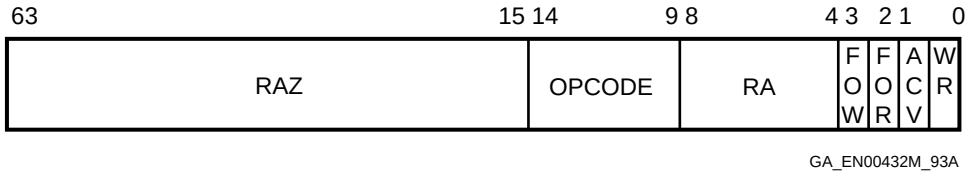


Table 13–1 lists the MM_CSR register fields.

Table 13–1 MM_CSR Register Fields

Field	Type	Description
WR	RO	This bit is set if the reference that caused the error was a write.
ACV	RO	This bit is set if the reference caused an access violation.
FOR	RO	This bit is set if the reference was a read and the PTE's FOR bit was set.
FOW	RO	This bit is set if the reference was a write and the PTE's FOW bit was set.
RA	RO	The Ra field of the faulting instruction.
OPCODE	RO	The opcode field of the faulting instruction.

ABOX_CTL Register

The A-box control register directs the actions of the DECchip 21064 CPU's A-box unit. Figure 13–5 shows the format of the ABOX_CTL register fields.

Figure 13–5 ABOX_CTL Register Format

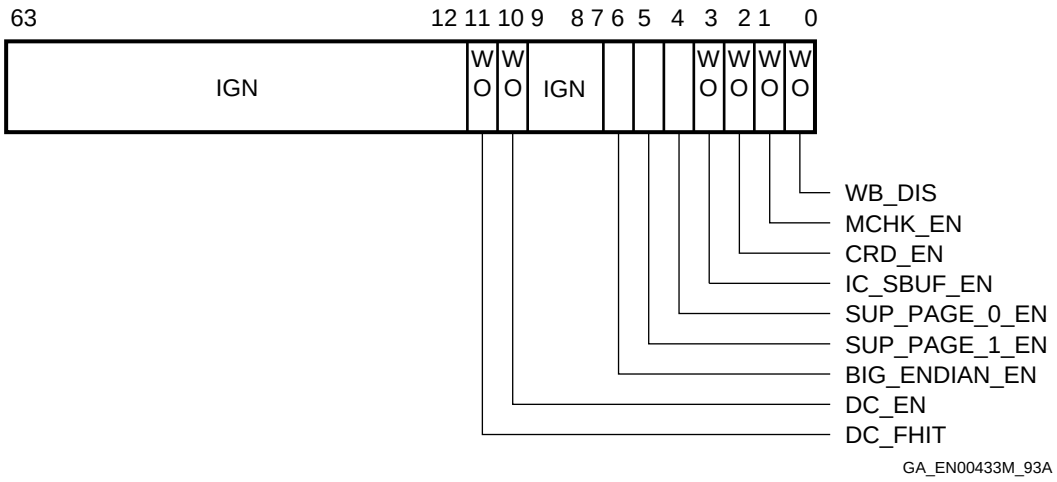


Table 13–2 lists the ABOX_CTL register fields.

Table 13–2 ABOX_CTL Register Fields

Field	Type	Description
WB_DIS	WO	Write buffer unload disable. When set, this bit prevents the write buffer from sending write data to the DECchip 21064 CPU's internal bus interface unit (BIU). The WB_DIS bit must be set for diagnostics only.

(continued on next page)

Table 13–2 (Cont.) ABOX_CTL Register Fields

Field	Type	Description
MCHK_EN	WO	Machine check enable. When this bit is set, the A-box generates a machine check when errors that are not correctable by the hardware are encountered. When this bit is cleared, uncorrectable errors do not cause a machine check, but the BIU_STAT, DC_STAT, BIU_ADDR, FILL_ADDR, and DC_ADDR registers are updated and locked when the errors occur.
CRD_EN	WO	Corrected read data interrupt enable. When this bit is set, the A-box generates an interrupt request when a pin bus transaction is terminated with a CACK(H) code of SOFT_ERROR.
IC_SBUF_EN	WO	Instruction cache (I-Cache) stream buffer enable. When set, this bit enables the operation of a single-entry instruction cache stream buffer.
SUP_PAGE_0_EN	WO	This bit, when set, enables one-to-one super-page mapping of data stream virtual addresses with VA<42:30> = $1FFE_{16}$ to physical addresses with PA<33:30> = 0_{16}. Access is allowed only in kernel mode.
SUP_PAGE_1_EN	WO	This bit, when set, enables one-to-one super-page mapping of data stream virtual addresses with VA<33:13>, if virtual address bits VA<42:41> = 2. Virtual address bits VA<40:34> are ignored in this translation. Access is allowed only in kernel mode.
BIG_ENDIAN_EN	WO	Hardware support for big endian data formats is supported by bit <6> of the ABOX_CTL register. This bit, when set, inverts physical address bit <2> for all data stream references. It is intended that chip endian mode be selected during initialization PAL code only.
DC_EN	WO	D-cache enable. When clear, this bit disables and flushes the DECchip 21064 CPU's data cache. When set, this bit enables the D-cache.

(continued on next page)

Table 13–2 (Cont.) ABOX_CTL Register Fields

Field	Type	Description
DC_FHIT	WO	D-cache force hit. When set, this bit forces all data stream references to hit in the data cache. This bit takes precedence over DC_EN; for example, when DC_FHIT is set and DC_EN is clear all data stream references hit in the data cache.

ALT_MODE Register

The alternate mode register (ALT_MODE) is a write-only register. The AM field specifies the alternate processor mode used by the hardware load (HW_LD) and hardware store (HW_ ST) instructions that have their ALT bit (bit 14) set.

Figure 13–6 shows the format of the ALT_MODE register.

Figure 13–6 ALT_MODE Register Format

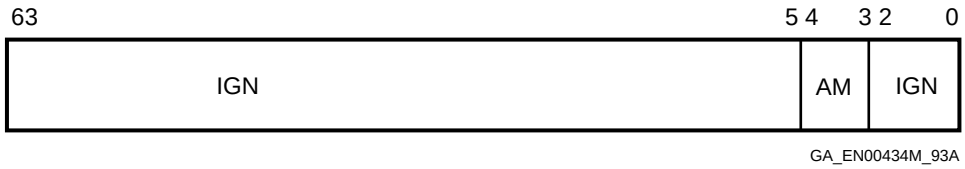


Table 13–3 lists the ALT_MODE register fields.

Table 13–3 ALT_MODE Register Fields

ALT_MODE<4..3>	Mode
0 0	Kernel
0 1	Executive
1 0	Supervisor
1 1	User

Cycle Counter Registers

CC Register

The DECchip 21064 CPU supports a cycle counter as described in the *Alpha AXP System Reference Manual*. This counter, when enabled, increments once for each CPU cycle. The HW_MTPR RN,CC instruction writes to CC<63..32> with the value held in RN<63..32>, and CC<31..0> are not changed. This register is read by the RCC instruction defined in the *Alpha AXP System Reference Manual*.

CC_CTL Register

The cycle counter register (CC_CTL) is a write-only register. The HW_MTPR RN,CC_CTL instruction writes to CC<31:0> with the value held in RN<31:0>, and the contents of bits CC<63:32> are not changed. The CC<3:0> bits must be written to with 0. If the RN<32> bit is set, then the counter is enabled, otherwise the counter is disabled.

BIU_CTL Register

Figure 13–7 shows the format of the bus interface unit control register (BIU_CTL).

Figure 13–7 BIU_CTL Register Format

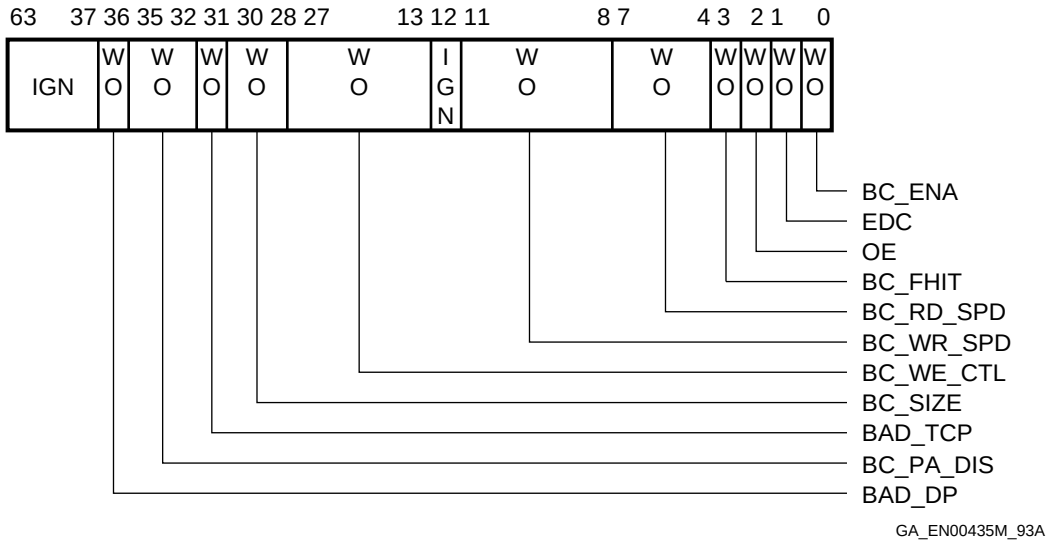


Table 13–4 lists the BIU_CTL register fields.

Table 13–4 BIU_CTL Register Fields

Field	Type	Description
BC_ENA	WO	External cache enable. When clear, this bit disables the external cache. When the external cache is disabled, the BIU does not probe the external cache tag store for read and write references. It initiates a request on the CREQ(H) line immediately.

(continued on next page)

Table 13–4 (Cont.) BIU_CTL Register Fields

Field	Type	Description
EDC	WO	Error detection and correction. When this bit is set, the DECchip 21064 CPU generates or expects EDC on the CHECK(H) pins. When this bit is clear the DECchip 21064 CPU generates or expects parity on four of the CHECK(H) pins. The memory subsystem is protected only by parity and therefore the EDC bit must be clear.
OE	WO	Output enable. When this bit is set, the DECchip 21064 CPU does not assert its chip enable pins during RAM write cycles, thus enabling these pins to be connected to the output enable pins of the cache RAMs.
BC_FHIT	WO	Backup cache force hit. When this bit and the BC_ENA bit are set, all pin bus READ_BLOCK and WRITE_BLOCK transactions are forced to hit in the backup (external to DECchip 21064 CPU) cache. Tag and tag control parity are ignored when the BIU operates in this mode. The BC_ENA bit takes precedence over the BC_FHIT bit. When the BC_ENA bit is clear and the BC_FHIT bit is set, tag probes do not occur and external requests are directed to the memory subsystem on the CREQ(H) pins. The BC_PA_DIS field takes precedence over the BC_FHIT bit.

(continued on next page)

Table 13–4 (Cont.) BIU_CTL Register Fields

Field	Type	Description
BC_RD_SPD	WO	Backup (external) cache read speed. This field indicates to the bus interface unit the read access time of the RAMs used to implement the off-chip external cache, measured in CPU cycles. This field must be written to with a value equal to one less than the read access time (in cycles) of the external cache RAMs. Access times for reads must be in the range 16..3 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..2. The BC_RD_SPD field is not initialized on reset and must be explicitly written to before enabling the external cache. For the PB22H-KB system module, the BC_RD_SPD field is written to with a value of 3 by the initialization program, which corresponds to a backup cache read access time of four cycles.
BC_WR_SPD	WO	Backup cache write speed. This field indicates to the bus interface unit the write cycle time of the RAMs used to implement the off-chip external cache, (backup cache on the CPU module) measured in CPU cycles. It must be written to with a value equal to one less than the write cycle time of the external cache RAMs. Access times for writes must be in the range 16..2 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..1. The BC_WR_SPD field is not initialized on reset and must be explicitly written to before enabling the external cache. For the PB22H-KB system module, the BC_WD_SPD field is written to with a value of 4 by the initialization program, which corresponds to a backup cache write access time of five cycles.

(continued on next page)

Table 13–4 (Cont.) BIU_CTL Register Fields

Field	Type	Description
BC_WE_CTL	WO	Backup cache write enable control. This field controls the timing of the write enable and chip enable pins during writes to the data and tag control RAMs. It consists of 15 bits. Each bit determines the value placed on the write enable and chip enable pins during a given CPU cycle of the RAM write access. When a given bit of the BC_WE_CTL field is set, the write enable and chip enable pins are asserted during the corresponding CPU cycle of the RAM access. Bit BC_WE_CTL<0> (bit 13 in BIU_CTL) corresponds to the second cycle of the write access, bit BC_WE_CTL<1> (bit 14 in BIU_CTL) to the third CPU cycle, and so on. The write enable pins are never asserted in the first CPU cycle of a RAM write access. Unused bits in the BC_WE_CTL field must be written to with 0s. The BC_WE_CTL field is not initialized on reset and must be written to before enabling the external backup cache.
BC_SIZE	WO	Backup cache size. This field is used to indicate the size of the external cache. The BC_SIZE field is not initialized by a reset and must be written to before enabling the backup cache (see Table 13–5 for the encodings). For the PB22H-KB system module, the BC_SIZE field is written to with a value of 010_2, which corresponds to a backup cache size of 512K bytes.
BAD_TCP	WO	Bad tag control parity. When set, the BAD_TCP bit causes the DECchip 21064 CPU to write bad parity to the tag control RAM when it does a fast external RAM write.

(continued on next page)

Table 13–4 (Cont.) BIU_CTL Register Fields

Field	Type	Description
BC_PA_DIS	WO	Backup cache physical address disable. This 4-bit field can be used to prevent the DECchip 21064 CPU from using the external cache to service reads and writes based on the quadrant of the physical address space that they reference. Table 13–6 shows the correspondence between this bit field and the physical address space. When a read or write reference is presented to the bus interface unit (BIU), the values of BC_PA_DIS, BC_ENA, and physical address bits <33:32> determine whether the external cache is used to satisfy the reference. If the external cache is not to be used for a given reference, the bus interface unit does not probe the tag store and makes the appropriate system request immediately. The value of the BC_PA_DIS field has no effect on which portions of the physical address space can be cached in the primary caches. System components control this through the RDACK field of the pin bus. The BC_PA_DIS field is set to a value of 0001_2. This enables external caching for quadrant 0 (cA<33:32> = 0) only. The BC_PA_DIS field is not initialized by a reset.
BAD_DP	WO	Bad data parity. When set, BAD_DP causes the DECchip 21064 CPU to invert the value placed on bits <0>,<7>,<14> and <21> of the CHECK(H)<27:0> field during off-chip writes. This produces bad parity when the DECchip 21064 CPU is in parity mode, and bad check bit codes when the CPU is in EDC mode.

Table 13–5 lists the bit values of the BC_SIZE field and the corresponding backup cache sizes.

Table 13–5 BC_SIZE Bits and Cache Sizes

BC_SIZE(n_2)	Backup Cache Size
0 0 0	128K bytes
0 0 1	256K bytes
0 1 0	512K bytes (Value used on the PB22H-KB system module.)
0 1 1	1M byte
1 0 0	2M bytes
1 0 1	4M bytes
1 1 0	8M bytes

Table 13–6 lists the bits of the BC_PA_DIS field and the corresponding physical addresses.

Table 13–6 BC_PA_DIS Bits and Physical Addresses

BC_PA_DIS Bit	Physical Address
<32>	PA<33..32> = 0
<33>	PA<33..32> = 1
<34>	PA<33..32> = 2
<35>	PA<33..32> = 3

Table 13–7 shows the BIU_CTL field initialization value.

Table 13–7 BIU_CTL Initialization Values

Init Value	Note
$C30006435_{16}$	512K bytes 4-cycle read, 5-cycle write cache, 2-cycle write strobe

Other A-BOX Registers

Virtual Address Register

When data-stream faults or DTB misses occur, the effective virtual address associated with the fault or miss is latched in the read-only virtual address register (VA). The VA and MM_CSR registers are locked against further updates until software reads from the VA register. The VA register is unlocked after reset. PAL code must explicitly unlock the VA register when its entry point is higher in priority than a DTB miss.

DTB_ZAP Register

A write of any value to the data translation buffer zap register (DTB_ZAP) invalidates all 32 small-page and 4 large-page DTB entries. A write also resets the NLU pointer to its initial state.

DTB_ASM Register

A write of any value to the data translation buffer invalidates all 32 small-page and 4 large-page DTB entries in which the ASM bit is equal to 0.

DTB_IS Register

If the virtual address in the RB field is mapped in either the small-page or large-page DTB, then those entries are invalidated.

FLUSH_IC Register

A write of any value to this pseudo-IPR flushes the entire instruction cache.

FLUSH_IC_ASM Register

In the DECchip 21064 CPU, a write of any value to this pseudo-IPR invalidates all instruction cache blocks in which the ASM bit is clear.

14

PAL Temporary Registers

Introduction

This chapter describes the 32 registers contained in the DECchip 21064 CPU that provide temporary storage for PAL code. These registers are accessible through HW_MXPR instructions.

In This Chapter

This chapter contains the following sections:

- BIU_STAT Register
- DC_STAT Register
- BIU_ADDR Register
- DC_ADDR Register
- FILL_ADDR Register
- FILL_SYNDROME Register
- BC_TAG Register

BIU_STAT Register

The bus interface unit status register (BIU_STAT) is a read-only register. When the BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR bit is set, the BIU_STAT<6:0> register bits are locked against further updates. The address associated with the error is latched and locked in the BIU_ADDR register.

The BIU_STAT<6:0> register bits and the BIU_ADDR register are also spuriously locked when FILL_EDC or BIU_DPERR bit is set.

The BIU_STAT<7:0> bits and BIU_ADDR register are unlocked when the BIU_ADDR register is read.

When FILL_EDC or BIU_DPERR bit is set, the BIU_ STAT<13:8> bits are locked against further updates. The address associated with the error is latched and locked in the FILL_ADDR register. The BIU_STAT<14:8> register bits and FILL_ADDR register are unlocked when the FILL_ADDR register is read.

The BIU_STAT register is not unlocked or cleared by a reset and must be cleared by PAL code.

Figure 14–1 shows the format of the BIU_STAT register.

Figure 14–1 BIU_STAT Register Format

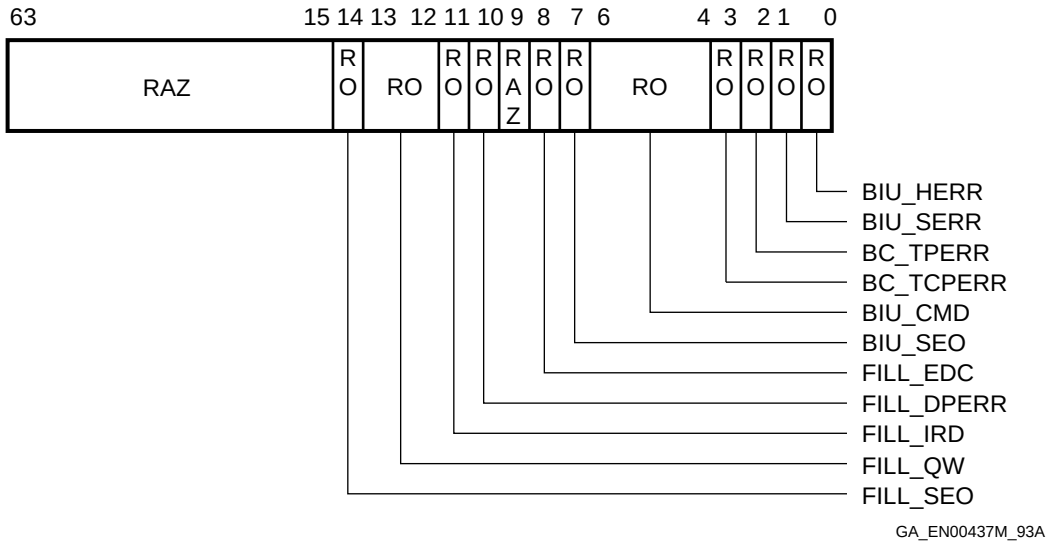


Table 14–1 lists the BIU_STAT register fields.

Table 14–1 BIU_STAT Register Fields

Field	Type	Description
BIU_HERR	RO	Hard error. When set, indicates that an external cycle was terminated with the CACK(H) pins indicating a HARD_ERROR.
BIU_SERR	RO	Soft error. When set, indicates that an external cycle was terminated with the CACK(H) pins indicating a SOFT_ERROR.
BC_TPERR	RO	Backup cache tag parity error. When set, indicates that an external cache tag probe encountered bad parity in the tag address RAM.
BC_TCPERR	RO	Backup cache tag control parity error. When set, indicates that an external cache tag probe encountered bad parity in the tag control RAM.

(continued on next page)

Table 14–1 (Cont.) BIU_STAT Register Fields

Field	Type	Description
BIU_CMD	RO	Bus interface unit CMD. This field latches the cycle type on the CREQ(H) pins when a BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR error occurs.
BIU_SEO	RO	Bus interface unit SEO. When set, indicates one of the following: • An external cycle was terminated with the CACK(H) pins indicating a HARD_ERROR. • An external cache tag probe encountered bad parity in the tag address RAM or the tag control RAM while BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR was set.
FILL_EDC	RO	EDC error. When set, indicates that the primary cache fill data received from outside the DECchip 21064 CPU contained an EDC error. The backup cache implemented on the PB22H-KB system module is protected by longword parity and therefore the FILL_EDC bit remains cleared.
FILL_DPERR	RO	Fill parity error. When set, indicates that the bus interface unit received data with a parity error from outside the DECchip 21064 CPU while performing either a data cache or instruction cache fill. The FILL_DPERR bit is used only when the DECchip 21064 CPU is in parity mode, which is the case for the PB22H-KB system module.
FILL_IRD	RO	This bit is used only when either the FILL_EDC bit or FILL_DPERR bit is set. When set, the FILL_IRD bit indicates that the error that caused the FILL_EDC bit or the FILL_DPERR bit to set occurred during an instruction cache fill. When cleared, the FILL_IRD bit indicates that the error occurred during a data cache fill.
FILL_QW	RO	This field is used only when the FILL_DPERR bit is set. The FILL_QW field identifies the quadword in the hexaword primary cache fill block that caused the error. You can use the FILL_QW field with the FILL_ADDR<33:5> bits to get the complete physical address of the bad quadword.

(continued on next page)

Table 14–1 (Cont.) BIU_STAT Register Fields

Field	Type	Description
FILL_SEO	RO	Fill SEO. When set, indicates one of the following: • A primary cache fill operation resulted in either an uncorrectable EDC error or in a parity error while the FILL_EDC bit was set • The FILL_DPERR bit was already set

DC_STAT Register

The data cache status register (DC_STAT) is a read-only register. When an external parity error is recognized during a primary cache fill operation, the DC_STAT register is locked against further updates. The DC_STAT register is unlocked when the DC_ADDR register is read.

Figure 14–2 shows the format of the DC_STAT register.

Figure 14–2 DC_STAT Register Format

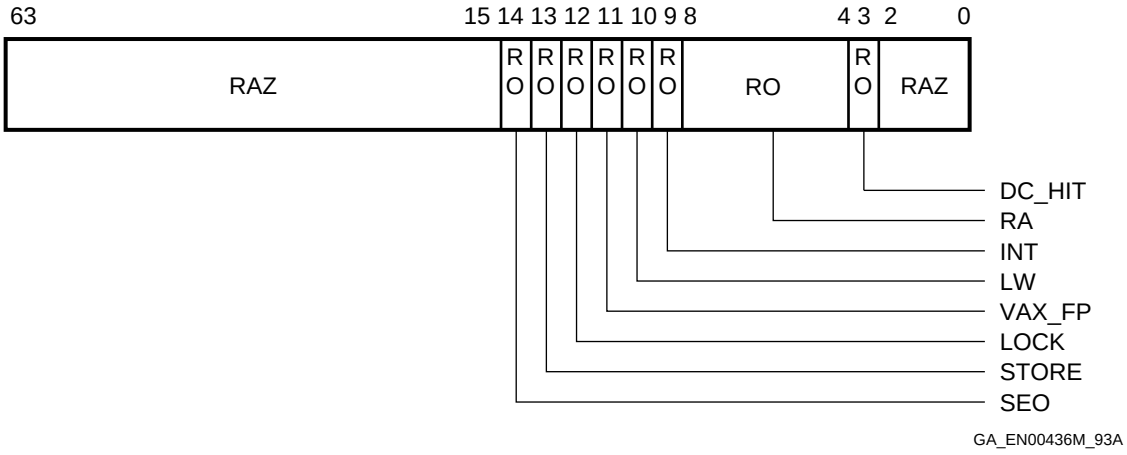


Table 14–2 lists the DC_STAT register fields.

Table 14–2 Data Cache Status Register

Field	Type	Description
DC_HIT	RO	Data cache hit. This bit indicates whether the last load or store instruction processed by the A-box hit (DC_HIT set) or missed (DC_HIT clear) the data cache. In the DECchip 21064 CPU, the loads that miss the data cache may be completed without requiring external reads; that is, pending fill or pending store hits.
SEO	RO	Second error occurred. Set when an error that normally locks the DC_STAT register occurs while the DC_STAT register is already locked.

The other DC_STAT register bits (see Table 14–3) are used only if the FILL_EDC or FILL_DPERR bit in the BIU_STAT register is set.

Table 14–3 Data Cache Status Error Modifiers

Field	Type	Description
RA	RO	The RA field of the instruction that caused the error.
INT	RO	Integer. When set, indicates an integer load or store.
LW	RO	Longword. When set, indicates that the data length of the load or store was a longword.
VAX_FP	RO	VAX floating-point. When INT is clear, this bit is set to indicate that a VAX floating-point format load or store caused the error.
LOCK	RO	Lock. This bit is set to indicate that the error was caused by an LDLL, LDQL, STLC, or STQC instruction.
STORE	RO	Store. This bit is set to indicate that the error was caused by a store instruction.

BIU_ADDR Register

The bus interface unit address register (BIU_ADDR) is a read-only register that contains the physical address associated with errors reported by the BIU_STAT<7:0> bits. The BIU_ADDR register's contents are used only when BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR is set. Reading the BIU_ADDR register unlocks both BIU_ADDR and BIU_STAT<7:0>.

The BIU_ADDR<33:5> bits contain the values of ADR(H)<33:5> associated with the pin bus transaction that resulted in the error indicated in BIU_STAT<7:0>.

If the BIU_CMD field of the BIU_STAT register indicates that the transaction that received the error was a READ_BLOCK or LDx/L transaction, the state of BIU_STAT<4:2> is unpredictable. If the BIU_CMD field of the BIU_STAT register encodes any pin bus command other than a READ_BLOCK or LDx/L transaction, then BIU_ADDR<4:2> contains 0s. The BIU_ADDR<63:34> register bits and the BIU_ADDR<1:0> register bits are always read as 0.

DC_ADDR Register

In the DECchip 21064 CPU, the DC_ADDR register is a pseudo-register used for unlocking the DC_STAT register. The DC_STAT and DC_ADDR registers are unlocked when the DC_ADDR register is read.

FILL_ADDR Register

The fill address register (FILL_ADDR) is a read-only register that contains the physical address associated with the errors reported by the BIU_STAT<14:8> bits. Its contents are used only when the FILL_EDC bit or FILL_DPERR bit is set. Reading the FILL_ADDR register unlocks the FILL_ADDR register, the BIU_STAT<14:8> bits, and the FILL_SYNDROME register.

The FILL_ADDR<33:5> bits identify the 32-byte cache block that the CPU was attempting to read when the error occurred.

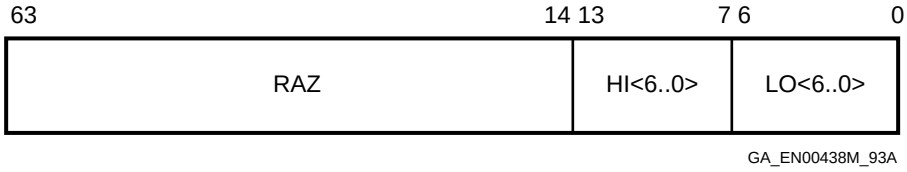
If the FILL_IRD bit of the BIU_STAT register is clear, which indicates that the error occurred during a data stream cache fill, then the FILL_ADDR<4:2> bits contain bits <4:2> of the physical address generated by the load instruction that triggered the cache fill. If the FILL_IRD bit is set, then the state of FILL_ADDR<4:2> is unpredictable. The FILL_ADDR<63:34> bits and the FILL_ADDR<1:0> bits are read as 0.

FILL_SYNDROME Register

The fill syndrome register (FILL_SYNDROME) is a 14-bit read-only register. If a parity error is recognized during a primary cache fill operation, the FILL_SYNDROME register indicates which longword in the quadword has bad parity. The FILL_SYNDROME<0> bit is set to indicate that the low longword was corrupted, and the FILL_SYNDROME<7> bit is set to indicate that the high longword was corrupted. The FILL_SYNDROME<13:8> bits and <6:1> bits are read as zero in parity mode.

Figure 14–3 shows the format of the FILL_SYNDROME register.

Figure 14–3 FILL_SYNDROME Register Format



BC_TAG Register

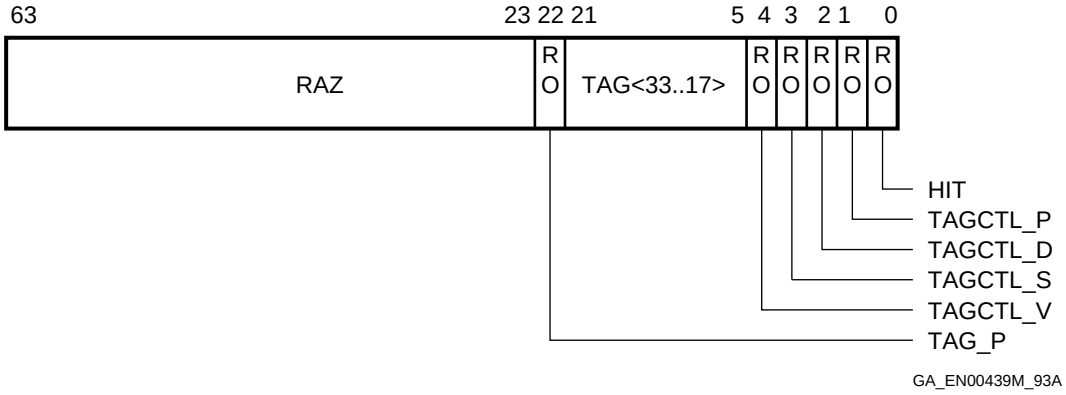
The backup cache tag register (BC_TAG) is a read-only register.

Unless locked, the BC_TAG register is loaded with the results of every backup cache tag probe. When a tag or tag control parity error or primary fill data error (parity or EDC) occurs, this register is locked against further updates. Your software can read the LSB of this register by using the HW_MFPR instruction. Each time an HW_MFPR instruction from BC_TAG completes, the contents of BC_TAG are shifted one bit position to the right. The entire register can then be read using a sequence of HW_MFPR instructions. Your software can unlock the BC_TAG register using a HW_MTPR instruction to BC_TAG.

Successive HW_MFPR instructions from the BC_TAG register must be separated by at least one null cycle.

Figure 14–4 shows the format of the BC_TAG register.

Figure 14–4 BC_TAG Register Format



Unused tag bits in the TAG field of this register are always cleared, based on the size of the external cache, which is determined by the BC_SIZE field of the BIU_CTL register.

15

CPU Cycle Types, Transactions, and Initialization

Introduction

This chapter describes the DECchip 21064 CPU Cycle Types, Transactions, and Initialization

In This Chapter

This chapter contains the following sections:

- DECchip 21064 CPU Cycle Types
- DECchip 21064 CPU Transactions
- Fast External Cache Read Hit Transaction
- Fast External Cache Write Hit Transaction
- READ_BLOCK Transaction
- WRITE_BLOCK Transaction
- LDxL and STxC Transactions
- BARRIER Transaction
- FETCH and FETCHM Transactions
- Initialization

DECchip 21064 CPU Cycle Types

The DECchip 21064 CPU requests an external cycle when it determines that the cycle it wants to perform requires system module level action. It has the following cycle types:

- A **BARRIER** cycle is generated by the MB instruction. Because an external write buffer does not exist between the processor and an error detection point in the system, the PB22H-KB system module acknowledges the cycle.
- The **FETCH** and **FETCHM** cycles are generated by the FETCH and FETCHM instructions respectively. The backup cache controller acknowledges the instruction and no other action takes place.
- The **READ_BLOCK** cycle is generated on read misses. The backup cache controller reads the addressed block from memory and supplies it, 128 bits at a time, to the DECchip 21064 CPU on the data bus. The backup cache location that missed is written to memory if the dirty bit is set and updated with the new cache entry.
- The **WRITE_BLOCK** cycle is generated on write misses to the backup cache. Data is read from memory to the backup cache and data is then written from the DECchip 21064 CPU to the backup cache with longword granularity.
- The **LDxL** cycle is generated by the LDLL and LDQL instructions. The cycle works the same as a READ_BLOCK cycle, except that the backup cache is not probed by the processor. The backup cache controller performs the backup cache probe, and if the reference is to cacheable address space, the lock flag is set.
- The **STxC** cycle is generated by the STLC and STQC instructions. The cycle works the same as a WRITE_BLOCK cycle, except that the backup cache is not probed by the processor. The backup cache controller performs the backup cache probe, and the cycle is acknowledged with a completion status. The STxC cycle completes successfully only if the lock flag is set.

Table 15–1 lists the processor initiated transactions.

Table 15–1 Processor Initiated Transactions

Transaction	Activity
P-cache read	Not visible outside the processor.
P-cache write	The backup cache is written to if a hit occurs. A write block is generated if a miss occurs.
P-cache masked write	The backup cache is written to if a hit occurs. A write block is generated if a miss occurs.
Fast backup cache read hit	Backup cache data is read.
Fast backup cache write hit	Data is written to the backup cache data store and the dirty bit is set.
Fast backup cache masked write hit	Data is written to the backup cache data store and the dirty bit is set.
Read block[1]	The system bus is read or an exchange cycle is generated.
Write block[2]	The system bus is read or an exchange and possibly write or null cycles are generated.
LDxL—Load lock	The system bus is read and an exchange or null cycle is generated. The following conditions apply: • If it is a cacheable address space reference, the address is latched and the lock bit is set. • If it is a noncacheable address space, then there is no change to the address lock or lock bit.

[1]Generated as a result of a fast backup cache read miss.

[2]Generated as a result of a fast backup cache write miss.

(continued on next page)

Table 15–1 (Cont.) Processor Initiated Transactions

Transaction	Activity
STxC—Store conditional	The system bus is read, and an exchange and possibly null or write cycles are generated. The following conditions apply: • If it is a cacheable address space reference, then the previously set lock bit is cleared and the store completes. • If it is noncacheable address space, there is no change to the lock bit and the store fails if the responder asserts UC_ERR(L) during the cycle.
Barrier[3]	All data buffers are flushed to the system coherence point.
FETCH/FETCHM[4]	The request is acknowledged and there is no other module level activity.

[3]Generated as a result of the execution of a memory barrier instruction.

[4]Generated as a result of the execution of a FETCH or FETCHM instruction.

DECchip 21064 CPU Transactions

The following sections describe the DECchip 21064 CPU transactions. The DECchip 21064 CPU transactions are as follows:

- Fast external cache read hit transaction
- Fast external cache write hit transaction
- READ_BLOCK transaction
- WRITE_BLOCK transaction
- LDxL transaction
- STxC transaction
- BARRIER transaction
- FETCH transaction
- FETCHM transaction

Fast External Cache Read Hit Transaction

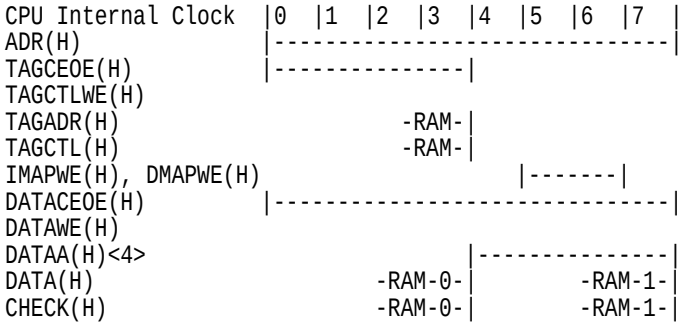
A fast external cache read consists of a probe read overlapped with the first data read, followed by the second data read if the probe hits. Example 15–1 shows a fast external cache read that selects 4 CPU cycle reads (BC_RD_SPD = 3), 5 CPU cycle writes (BC_WR_SPD = 4), and chip enable control (the OE bit of the BIU_CTL register is set to 1).

If the probe misses, the cycle aborts at the end of clock cycle 3. If the probe hits and the miss address has bit 4 set, then the two data reads are swapped. The DATAA(H)<4> signal is true in cycles 0, 1, 2, 3, and is false in cycles 4, 5, 6, 7.

Example

Example 15–1 shows an example of a fast external cache read hit transaction.

Example 15–1 Fast External Cache Read Hit Transaction



Fast External Cache Write Hit Transaction

A fast external cache write consists of a probe read, followed by 1 or 2 data writes. Example 15–2 shows that the external cache transaction is using 4 CPU cycle reads (BC_RD_SPD = 3), 5 CPU cycle writes (BC_WR_SPD = 4), chip enable control (OE = 1), and a 2-cycle write pulse beginning in cycle 3 (BC_WE_CTL <15..1> = 000000000000110_2).

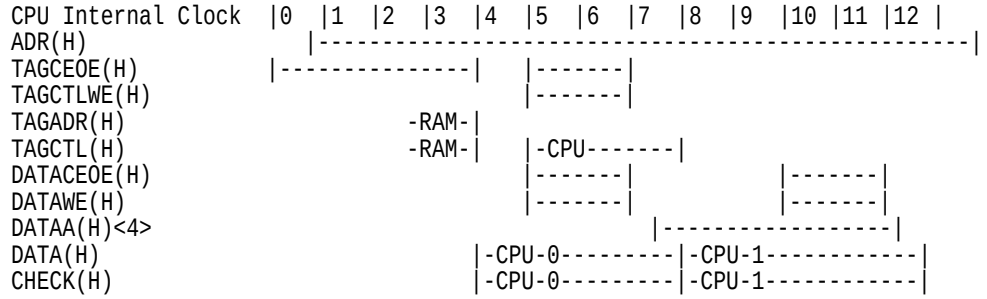
The DECchip 21064 CPU drives the TAGCTL(H) pins one CPU cycle later than it drives the DATA(H) and CHECK(H) pins relative to the start of the write cycle. This is because, unlike DATA(H) and CHECK(H), the TAGCTL(H) field must be read during the tag probe that precedes the write cycle.

Because the DECchip 21064 CPU can switch its pins to a low-impedance state much more quickly than most RAMs can switch their pins to a high-impedance state, the DECchip 21064 CPU waits one CPU cycle before driving the TAGCTL(H) pins to minimize tristate driver overlap. If the probe misses, the cycle aborts at the end of clock cycle 3.

Example

Example 15–2 shows an example of a fast external cache write hit transaction.

Example 15–2 Fast External Cache Write Hit Transaction



READ_BLOCK Transaction

A READ_BLOCK transaction appears at the external interface on external cache read misses, either because it was a miss, or because the external cache has not been enabled. The READ_ BLOCK transaction sequence is as follows:

1. The CREQ(H) pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The ADR(H) pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
2. When the READ_BLOCK transaction begins, the DECchip 21064 CPU does the following:
   - Has already placed the address of the block containing the miss on the ADR(H) line
   - Places the quadword-within-block and the I/D indication on the CWMASK(H) line
   - Places a READ_BLOCK command code on the CREQ(H) line
   - Clears the RAM control pins (DATAA(H)<4..3>, DATACEOE(H)<3..0>, and TAGCEOE(H)) no later than one CPU cycle after the system clock edge where the transaction begins
3. The external logic obtains the first 16 bytes of data. Although a single stall cycle is shown in Example 15–3, there may be no stall cycles or many stall cycles.
4. The external logic has the first 16 bytes of data and places it on the DATA(H) and CHECK(H) buses. It asserts the DRACK(H) line to indicate to the DECchip 21064 CPU that the data and check bit buses are valid. The DECchip 21064 CPU detects the DRACK(H) signal at the end of this cycle and reads in the first 16 bytes of data at the same time.
5. The external logic obtains the second 16 bytes of data. Although a single stall cycle is shown in Example 15–3, there may be no stall cycles or many stall cycles.

6. The external logic has the second 16 bytes of data. It places it on the DATA(H) and CHECK(H) buses. It asserts the DRACK(H) line to indicate to the DECchip 21064 CPU that the data and check bit buses are valid. The DECchip 21064 CPU detects the DRACK(H) signal at the end of this cycle and reads in the second 16 bytes of data at the same time. In addition, the external logic places an acknowledge code on the CACK(H) line to indicate to the DECchip 21064 CPU that the READ_BLOCK cycle is completed. The DECchip 21064 CPU detects the acknowledge code at the end of this cycle and changes the address.
7. Everything is idle. The DECchip 21064 CPU can start a new external cache cycle at this time.

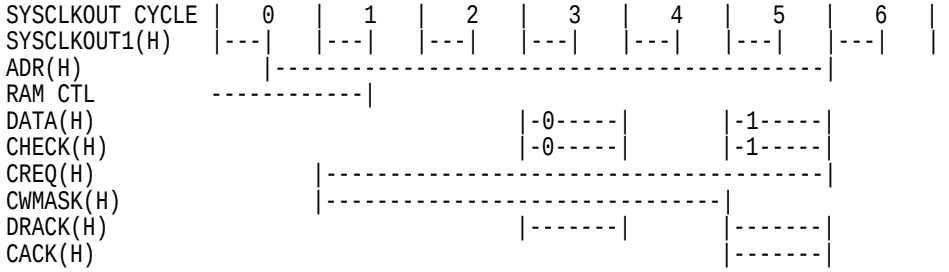
When the DECchip 21064 CPU deasserts its RAM control signals at the beginning of a READ_BLOCK transaction, control of RAM passes to the external logic. Because the external logic has control of RAM, it can cache the data by asserting its write pulses on the external cache during cycles 3 and 5.

The DECchip 21064 CPU performs parity checking on the data supplied to it through the data and check buses, if requested by the acknowledge code. It is not necessary to place data in the external cache to get checking.

Example

Example 15–3 shows an example of a READ_BLOCK transaction.

Example 15–3 READ_BLOCK Transaction



WRITE_BLOCK Transaction

A WRITE_BLOCK transaction appears at the external interface on either external cache write misses (either because it was a miss or because the external cache has not been enabled) or on external cache write hits to shared blocks.

The WRITE_BLOCK transaction sequence is as follows:

1. The CREQ(H) pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The ADR(H) pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
2. The WRITE_BLOCK cycle begins. The DECchip 21064 CPU does the following:
   - Places the address of the block on the ADR(H) line
   - Places the longword valid masks on the CWMASK(H) line and a WRITE_BLOCK command code on the CREQ(H) line
   - Clears the DATAA(H)<4..3> signals and TAGCEOE(H) signal no later than one CPU cycle after the system clock edge at which the transaction begins
   - Clears the DATACEOE(H)<3..0> signals at least one CPU cycle before the system clock edge where the transaction begins
3. The external logic detects the command and asserts the DOE(L) line to indicate to the DECchip 21064 CPU to drive the first 16 bytes of the block onto the data bus.
4. The DECchip 21064 CPU drives the first 16 bytes of write data onto the DATA(H) and CHECK(H) buses, and the external logic writes it into the destination. Although a single stall cycle is shown in Example 15–4, there may be no stall cycles or many stall cycles.
5. The external logic asserts the DOE(L) and DWSEL(H) lines to indicate to the DECchip 21064 CPU to drive the second 16 bytes of data onto the data bus.

6. The DECchip 21064 CPU drives the second 16 bytes of write data onto the DATA(H) and CHECK(H) buses, and the external logic writes it into the destination. Although a single stall cycle is shown in Example 15–4, there may be no stall cycles or many stall cycles. In addition, the external logic places an acknowledge code on the CACK(H) line to indicate to the DECchip 21064 CPU that the WRITE_BLOCK cycle is completed. The DECchip 21064 CPU detects the acknowledge code at the end of this cycle, and changes the address and command to the next values.
7. Everything is idle. When the DECchip 21064 CPU deasserts its RAM control signals at the beginning of a READ_BLOCK transaction, control of RAM passes to the external logic. Because the external logic has control of RAM, it can cache the data by asserting its write pulses on the external cache during cycles 3 and 5.

The DECchip 21064 CPU performs parity generation on data it drives onto the data bus.

Although in Example 15–4 external logic cycles through both 128-bit blocks of potential write data, this is not always the case. External logic must extract from the DECchip 21064 CPU only those 128-bit blocks of data that contain valid longwords as specified by the CWMASK(H) signals. However, if both halves are extracted from the DECchip 21064 CPU, and the lower half must be extracted before the upper half.

Example

Example 15–4 shows an example of a WRITE_BLOCK transaction.

Example 15–4 WRITE_BLOCK Transaction

LDxL and STxC Transactions

LDxL Transaction

An LDxL transaction appears at the external interface as a result of an LDQL or LDLL instruction being executed. The external cache is not probed. With the exception of the command code output on the CREQ(H) pins, the LDxL transaction is the same as a READ_BLOCK transaction (see READ_BLOCK Transaction).

StxC Transaction

An STxC transaction appears at the external interface as a result of STLC and STQC instructions. The external cache is not probed. The STxC transaction is the same as the WRITE_ BLOCK transaction, with the following exceptions:

- The code placed on the CREQ(H) pins by an STxC transaction is different from the code placed on the CREQ(H) pins by a WRITE_BLOCK transaction.
- The CWMASK field never validates more than a single longword or quadword of data.
- External logic has the option of making the transaction fail by using the CACK(H) line code of STxC_FAIL. It can do so without asserting either the DOE(L) or the DWSEL(H) line.

BARRIER Transaction

The BARRIER transaction appears on the external interface as a result of an MB instruction. The acknowledgment of the BARRIER transaction indicates to the DECchip 21064 CPU that all invalidations have been supplied to it, and that any external write buffers have been pushed out to the coherence point. Any errors detected during these operations can be reported to the DECchip 21064 CPU when the BARRIER transaction is acknowledged. This transaction is acknowledged immediately because it does not buffer WRITE_BLOCK transactions.

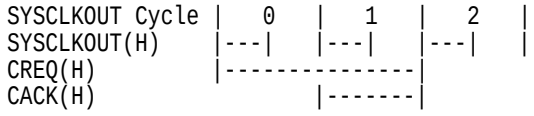
The BARRIER transaction sequence is as follows:

1. The BARRIER transaction begins. The DECchip 21064 CPU places the command code for the BARRIER transaction onto the CREQ(H) outputs.
2. The external logic detects the BARRIER transaction command code, and because it has completed processing the command (the external logic does not act on the command), it places an acknowledge code on the CACK(H) inputs.
3. The DECchip 21064 CPU detects the acknowledge code on the CACK(H) line and removes the command. The external logic removes the acknowledge code from the CACK(H) line. The cycle is finished.

Example

Example 15–5 shows an example of a BARRIER transaction

Example 15–5 BARRIER Transaction



FETCH and FETCHM Transactions

FETCHM Transaction

A FETCH transaction appears on the external interface as a result of a FETCH instruction. The transaction supplies an address to the external logic that the DECchip 21064 CPU ignores and responds with an immediate acknowledge.

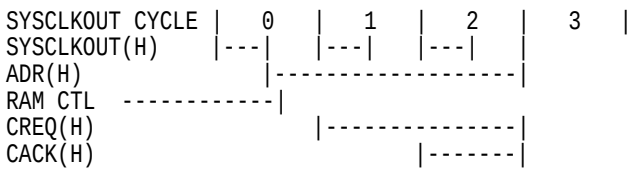
The FETCH transaction sequence is as follows:

1. The CREQ(H) pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The ADR(H) pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
2. When the FETCH transaction begins, the DECchip 21064 CPU does the following:
   - Places the effective address of the FETCH command code on the address outputs
   - Places the command code for FETCH on the CREQ(H) outputs
   - Clears the RAM control pins (DATAA(H)<4..3>, DATACEOE(H)<3..0>, and TAGCEOE(H)) no later than one CPU cycle after the system clock edge that begins the transaction
3. The external logic detects the FETCH command, and because it has completed processing the command (the external logic does not act on the command), it places an acknowledge code on the CACK(H) inputs.
4. The DECchip 21064 CPU detects the acknowledge on the CACK(H) line, and removes the address and the command. The external logic removes the acknowledge code from the CACK(H) line. The cycle is finished.

Example

Example 15–6 shows an example of a FETCH transaction.

Example 15–6 FETCH Transaction



FETCHM Transaction

A FETCHM transaction appears on the external interface as a result of a FETCHM instruction. The transaction supplies an address to the external logic, which the system ignores and responds with an immediate acknowledge. With the exception of the command code placed on the CREQ(H) line, the FETCHM transaction is the same as the FETCH transaction (see FETCH and FETCHM Transactions).

Initialization

Following the deassertion of the reset signal to the DECchip 21064 CPU, an initialization program is loaded into the I-cache from a Xilinx XC1765 serial ROM chip, which is on the system module. The program counter is set to location 0.0000.0000 and instruction execution is started.

Part III

Intel 82357 Integrated System peripheral Chip Functions

Part III provides an overview of the functions of the Intel 82357 integrated system Peripheral (ISP) chip.

This section includes the following chapters:

- Chapter 16, DMA Controller
- Chapter 17, Interrupt Controller
- Chapter 18, Nonmaskable Interrupt Ports
- Chapter 19, Interval Timer

16

DMA Controller

Introduction

This chapter describes the functions of the Intel 82357 ISP chip direct memory access (DMA) controller.

In This Chapter

This chapter contains the following sections:

- Overview
- DMA Controller Transfer Modes
- DMA Transfer Types
- Autoinitialization
- Master and Slave Modes
- DMA Controller Registers
- Stop Registers
- DMA Controller Memory Low-Page Register
- DMA Controller Memory High-Page Register
- Current Address Register
- Current Word Register
- Base Page, Base Address, and Base Word Count Registers
- Command Register
- Mode Register
- Extended Mode Registers
- Request Register
- Mask Register
- DMA Controller Status Register

- Set Chaining Mode Register
- Set Chaining Mode Status Register
- Channel Interrupt Status Register
- Chain Buffer Expiration Control Register
- DMA Controller Software Commands
- Terminal Count and EOP Summary
- EISA Bus Master Status Latch

Overview

This section gives an overview of the main functions of the Intel 82357 DMA controller.

Programmable Channels

The Intel 82357 DMA controller circuitry combines the functions of two DMA controllers with seven independently programmable channels: channels 0-3 and channels 5-7. DMA controller channel 4 is used to cascade the two controllers together and defaults to cascade mode in the mode register.

In addition to accepting requests from DMA slaves, the Intel 82357 DMA controller also responds to requests from your software. Your software can initiate a DMA service request by setting any DMA controller channel request register bit to 1.

DMA Controller Device Sizes and ISA Modes

You can program DMA controller channels for 8, 16, or 32-bit DMA device sizes, and you can program a DMA controller channel for the following ISA compatible modes:

- Type A
- Type B
- Type C burst modes

The device provides the timing controls, while the Intel 82358 **EISA Bus Controller (EBC)** does the data size translations that are necessary for the DMA transfer. The DMA controller memory addressing circuitry supports full 32-bit addresses for DMA devices. Each channel includes the following:

- A 16-bit, ISA-compatible **Current Register** that holds the 16 least-significant bits of the 32-bit address
- A **Low-Page Register** that contains the eight second most significant bits
- A **High-Page Register** that contains the eight most significant bits of the 32-bit address

DMA Controller Transfer Modes

You can program the DMA controller channels for the following four transfer modes; single, block, demand, and cascade. Each of the three active transfer modes—single, block, and demand—can perform three different types of transfers: read, write, or verify.

Additional DMA Controller Functions

The DMA controller also does the following:

- Refresh address generation
- Buffer chaining
- Autoinitialization
- Provides support for a ring buffer data structure in memory

  Stop registers are used to support data communications or devices that work from a ring buffer in memory.

DMA Controller Master and Slave Modes

The DMA controller is either in master mode or slave mode. In master mode, the DMA controller does one of the following:

- Services a DMA slave's request for DMA cycles
- Generates refresh cycles
- Allows a 16-bit ISA master to use the bus by a cascaded DREQ(H) signal

In slave mode, the DMA controller does the following:

- Monitors the bus
- Decodes I/O read and write commands that address its registers
- Responds to I/O read and write commands that address its registers

When the DMA controller is in master mode and servicing a DMA slave, it works with the Intel 82358 EISA bus controller to create bus cycles on the system bus. The DMA controller places the addresses and the memory read/write HW/R(L) signal on the host CPU bus. It uses the ST0(H) and ST1(H) lines to instruct the Intel 82358 EBC when to start and what type of bus cycle to run. The Intel 82358 EBC uses the DRDY(H) signal to inform the DMA controller when to place a new address on the bus.

DMA Controller Transfer Modes

Note

Memory to memory transfers are not supported by the Intel 82357 ISP.

Single Transfer Mode

In single transfer mode, the device is programmed to make one transfer only. The word count is decremented, and the address is decremented or incremented following each transfer. When the word count goes from zero to $FFFFFF_{16}$ or an external end of process (EOP) is encountered, a terminal count (TC) causes an autoinitialization to occur if the channel has been programmed for autoinitialization. If chaining is enabled, the next chain buffer is enabled if available.

To be recognized, the DREQ(H) signal must be held active until the DACK(L) signal also becomes active.

If the DREQ(H) signal is held active during a single transfer, the bus is released to the CPU after a single transfer. The bus is immediately requested again, and after winning the bus another single transfer is done. The single transfer process gives other devices a chance to execute cycles if they require the bus.

Block Transfer Mode

In block transfer mode, a device is activated by the DREQ(H) signal and continues making transfers during the service until one of the following occurs:

- A terminal count caused by the word count going to $FFFFFF_{16}$.
- An external EOP is encountered.

The DREQ(H) signal must be held active until the DACK(L) signal becomes active. An autoinitialization occurs at the end of the service if the channel has been programmed for it. In this mode, it is possible to lock out other devices for a period (including refresh) if the terminal count is programmed to a large number.

Demand Transfer Mode

In demand transfer mode, the device is programmed to continue making transfers until a terminal count is encountered, or until the DREQ(H) signal goes inactive. Transfers can continue until the I/O device has exhausted its data capacity. When the I/O device requires further service, the DMA service is re-established using the DREQ(H) signal. During the time between services, when the system is allowed to operate, the intermediate values of address and word count are stored in the DMA controller current address and current word count registers. A terminal count can cause an autoinitialization at the end of the service, if the channel has been programmed for autoinitialization.

Cascade Mode

This mode is used to cascade more than one DMA controller for simple system expansion. This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. In this architecture, channel 0 of the second controller (ch4) is used to cascade the first controller to provide a total of seven channels.

Cascade mode is also used to allow direct access of the system by 16-bit ISA bus masters. These devices use the DREQ(H) and DACK(L) signals to arbitrate for the system bus and then drive the address and command lines to control the bus.

In cascade mode, the DMA controller responds to a DREQ(H) signal with a DACK(L) signal but the HW/R(L) signal, address, and ST(H)<3:0> outputs are disabled.

Channel 4 is used to connect the second half of the DMA controller system. This channel is not available for any other purpose.

DMA Transfer Types

Each of the three active transfer modes can perform three different types of transfers. The transfer types are read, write, and verify.

Read Transfer

Read transfers move data from memory to an I/O device starting with the DMA controller deactivating the HW/R(L) signal and activating the ST(H)<3:0> lines. The bus controller activates the IOWC(L) signal and the appropriate EISA or ISA control signals to indicate a memory read, depending on the bus that the memory is on.

Write Transfer

Write transfers move data from an I/O device to memory starting with the DMA controller activating the HW/R(L) signal and activating the ST(H)<3:0> lines. The bus controller activates the IORC(L) signal and the appropriate EISA or ISA control signals to indicate a memory write, depending on the bus that the memory is on.

Verify Transfer

Verify transfers are pseudo-transfers. The DMA controller operates the same as it does in read or write transfers, generating addresses, and producing a terminal count, and so on. However, the ST(H)<3:0> signals are not activated, and therefore, the bus controller does not activate the memory and I/O control lines. Only the DACK(L) signal lines go active. Because EISA cycles are not broadcast in this mode, the LA bus is not copied to the SA bus. Internally, the DMA controller counts BCLK(H) cycles so that the DACK(L) signal lines have a defined pulse width. This pulse width is 9 BCLK(H) cycles long. If verify transfers are repeated during block or demand DMA requests, each additional pseudo-transfer adds 8 BCLK(H) cycles. The DACK(L) signal lines are not toggled for repeated transfers.

Autoinitialization

You can program a single bit in the mode register to configure a channel as an autoinitialization channel.

During autoinitialization, the original values of the current page, current address, and current word count registers are automatically restored from the base page, address, and word count registers or the channel following a terminal count.

The base registers are loaded simultaneously with the current registers by the DECchip 21064 CPU and remain unchanged during the DMA service. The mask bit is not set when the channel is in the autoinitialization state. Following an autoinitialization, the channel is ready to perform another DMA service, without CPU intervention, as soon as a valid DREQ(H) signal is detected.

Master and Slave Modes

The 82357 chip is either a slave device or a master device.

In slave mode, the 82357 chip monitors the address lines and decodes all bus cycles attempting to read from or write to any of its internal registers. In slave mode, either an EISA master or the host CPU can read from or write to any of the 82357 chip's internal registers. The 16-bit ISA masters can read from or write to any of the 82357 chip's compatible registers.

The registers that an ISA master cannot access are located in the I/O space of 00H-$0F_{16}$ and 0C0H-ODF_{16}. The 82357 chip disables these registers when granting the bus to an ISA master. In slave mode, the 82357 chip also detects and responds to interrupt acknowledge cycles.

In master mode, the 82357 chip becomes the master of the bus system. It can perform either DMA cycles or refresh cycles.

The 82357 chip's arbiter determines which mode the device is in.

DMA Controller Registers

The following sections describe the various registers relevant to the DMA controller operations:

- Stop Registers
- DMA Controller Memory Low-Page Register
- DMA Controller Memory High-Page Register
- Current Address Register
- Current Word Register
- Base Page, Base Address, and Base Word Count Registers
- Command Register
- Mode Register
- Extended Mode Registers
- Request Register
- Mask Register
- DMA Controller Status Register
- Set Chaining Mode Register
- Set Chaining Mode Status Register
- Channel Interrupt Status Register
- Chain Buffer Expiration Control Register

Stop Registers

To support a common data communications data structure (the ring buffer), a set of DMA controller registers are provided. These registers are called stop registers. Each channel has 22 bits of register location associated with it. The 22 bits are distributed between three different registers: one 6-bit register and two 8-bit registers. You can enable or disable the stop registers by writing to the channels's corresponding extended mode register.

The ring buffer data structure reserves a fixed portion of memory on doubleword boundaries that is used for a DMA controller channel. Frames that are received consecutively or other data structures are stored sequentially in the boundaries of the ring buffer memory.

The beginning of the ring buffer area is defined in the base address register. The end of the ring buffer area is defined in the base address register and the base byte or terminal count. The incoming frames (data) are deposited in sequential locations of the ring buffer. When the DMA controller reaches the end of the ring buffer, indicating the byte count has expired, the DMA controller (if programmed) is autoinitialized. When autoinitialization occurs, the current address register is restored from the base address register, taking the process back to the start of the ring buffer. The DMA controller is then available to begin depositing the incoming bytes in the ring buffer's sequential locations, provided that the host CPU has read the data that was previously placed in those locations. The DMA controller determines that the CPU has read certain data by the value that the CPU writes into the stop register.

When the data of a frame is read by the CPU, the memory location it occupies becomes available for other incoming frames. The stop register prevents the DMA controller from overwriting data that has not yet been read by the CPU. After the CPU has read a frame from memory, it updates the stop register to point to the location that was last read. The DMA controller does not deposit data into any location beyond the location pointed to by the stop register. The last address that is transferred before the channel is masked is the first address that matches the stop

register. The stop register stores values to compare only with A<23:2>, so the size of the ring buffer is limited to 16M bytes.

Table 16–1 shows the last three transfers if the stop register is set to a value of $00001C_{16}$.

Table 16–1 DMA Controller Address and Stop Register Correlation

Operation	By Bytes	By Words	By Doublewords
Increment	$XX000001A_{16}$	$XX000018_{16}$	$XX000014_{16}$
	$XX000001B_{16}$	$XX00001A_{16}$	XX000018 $_{16}$
	$XX000001C_{16}$	$XX00001C_{16}$	XX00001C $_{16}$
Decrement	$XX0000021_{16}$	XX000023 $_{16}$	XX000027 $_{16}$
	$XX0000020_{16}$	$XX000021_{16}$	XX000023 $_{16}$
	$XX000001F_{16}$	$XX00001F_{16}$	XX00001F $_{16}$

The bus controller provides I/O recovery for back-to-back CPU to 8-bit I/O cycles. For EISA master accesses, the software must provide I/O recovery of at least 1 BCLK(H) cycle.

Note

I/O writes must match the I/O slave size; that is, 8-bit writes must be used to program the ISP registers. When writing to the DMA controller registers, the DMA controller channels also must be masked.

DMA Controller Memory Low-Page Register

The DMA controller memory low-page register is a read/write register.

Each channel has an 8-bit memory low-page register associated with it. The DMA controller memory low-page register contains the 8 second most-significant bits of the 32-bit address (16-23). This register has the following features:

- It works with the DMA controller's high-page register and current address register to define the complete (32-bit) address for the DMA controller channels.
- It corresponds to the current address register for each channel.

This 8-bit register is read from or written to directly by the processor or bus master. This register can also be reinitialized by an autoinitialization to its original value. An autoinitialization takes place only after a terminal count or EOP.

DMA Controller Memory High-Page Register

The DMA controller memory high-page register is a read/write register.

Each channel has an 8-bit memory high-page register associated with it. The DMA controller memory high-page register contains the 8 most-significant bits of the 32-bit address (24-31). This register has the following features:

- It works with the DMA controller's low-page register and current address register to define the complete (32-bit) address for the DMA controller channels.
- It corresponds to the current address register for each channel.

This 8-bit register is read from or written to directly by the processor or bus master. This register can also be reinitialized by an autoinitialization to its original value. An autoinitialization occurs only after a terminal count or EOP.

This register is reset to 00_{16} during the programming of both the low-page register and the current address register. If this register is not programmed after the low-page register and other address registers are programmed, then its value is zero. In this case, the DMA controller channel operates the same as an 82C37 (from an addressing standpoint), and this operation is called address compatibility mode.

If the high 8-bits of the address are programmed after the other addresses, the channel modifies its operation to increment (or decrement) the entire 32-bit address. This differs from the 82C37 page register that was used in the original PCs. The 82C37 page register increments to a 64K-byte boundary (for 8-bit channels) or 128K-byte boundaries (for 16-bit channels). This mode is called extended address mode. In this mode, the 82358 EISA bus controller generates the MRDC(L) and MWTC(L) signals only for addresses below 16M bytes.

Address Compatibility Mode

When the DMA controller is operating in address compatibility mode, the addresses do not increment or decrement through the high- and low-page registers, and the high-page register is set to 00_{16}. This is compatible with the 82C37 and page register implementations used in the PC/AT. This mode is set when any of the lower three address bytes of a channel are programmed. If the upper byte of a channel's address is programmed last, the channel enters extended address mode. In this mode, the high byte can be any value and the address increments or decrements through the entire 32 bits. When programming the page register in address compatibility mode, the current address must also be programmed.

After reset, all channels are set to address compatibility mode. The master clear command also resets the proper channels to address compatibility mode. The mode bits are stored in individual flip-flops on a per-channel basis.

Current Address Register

The current address register is a read/write register.

Each channel has a 16-bit current address register. This address holds the value of the 16 least significant bits of the full 32-bit address used during DMA transfers. The address is automatically incremented or decremented after each transfer. The intermediate values of the address are stored in the current address register during the transfer. This register is written to or read from by the CPU or bus master in successive 8-bit bytes. It can also be reinitialized by an autoinitialization to its original value. An autoinitialization takes place only after a terminal count or EOP.

Address Shifting When Programmed for 16-Bit I/O Count by Words

To maintain compatibility with the DMA controller in the 82C37 chip used in the PC/AT, the DMA controller shifts the addresses when the extended mode register is programmed for or defaulted to transfers to or from a 16-bit device in count by words format.

Table 16–2 shows how the addresses are shifted when the extended mode register is programmed for 16-bit I/O in count by words format. Note that the least significant bit of the low-page register is not used in 16-bit shifted mode.

Table 16–2 Address Shifting When Programmed for 16-Bit I/O Count by Words

Output Address	8-Bit I/O Programmed Address	16-Bit I/O Programmed Address	32-Bit I/O Programmed Address	16-Bit I/O Programmed Address (No Shift)
A0	A0	0	A0	A0
A<16:1>	A<15:0>	A<16:1>	A<16:1>	A<16:1>
A<31:17>	A<31:17>	A<31:17>	A<31:17>	A<31:17>

Current Word Register

The current word register is a read/write register.

Each channel has a 24-bit current word count register. This register determines the number of transfers to be performed. The actual number of transfers is one more than the number programmed in the current word count register; that is, programming a count of 100 results in 101 transfers. The word count is decremented after each transfer. The intermediate value of the word count is stored in the register during the transfer. When the value in the register goes from zero to $0FFFFFF_{16}$, a terminal count is generated.

Following the end of a DMA service, the word count register can also be reinitialized by an autoinitialization to its original value. An autoinitialization can occur only when a terminal count occurs. If the word count register is not autoinitialized, it has a count of $FFFFFF_{16}$ after terminal count.

To maintain compatibility with the 82C37 chip, programming either the low byte (bits<7:0>) or the middle byte (bits <15:8>) clears the high byte (bits<23:16>). This enables you to use software that was written for the 82C37 chip that cannot access the upper byte of the word count in the current word count register.

When the extended mode register is programmed for or defaulted to transfers to or from an 8-bit I/O, the word count indicates the number of bytes to be transferred.

When the extended mode register is programmed for or defaulted to transfers to or from a 16-bit I/O, with a shifted address, the word count indicates the number of 16-bit words to be transferred.

When the extended mode register is programmed for or defaulted to transfers to or from a 16 or 32-bit I/O device, the word count indicates the number of bytes that you want to transfer. In this case, the number of bytes that you want to transfer need not be a multiple of two or four.

Base Page, Base Address, and Base Word Count Registers

The base page, base address, and base word count registers are read/write registers.

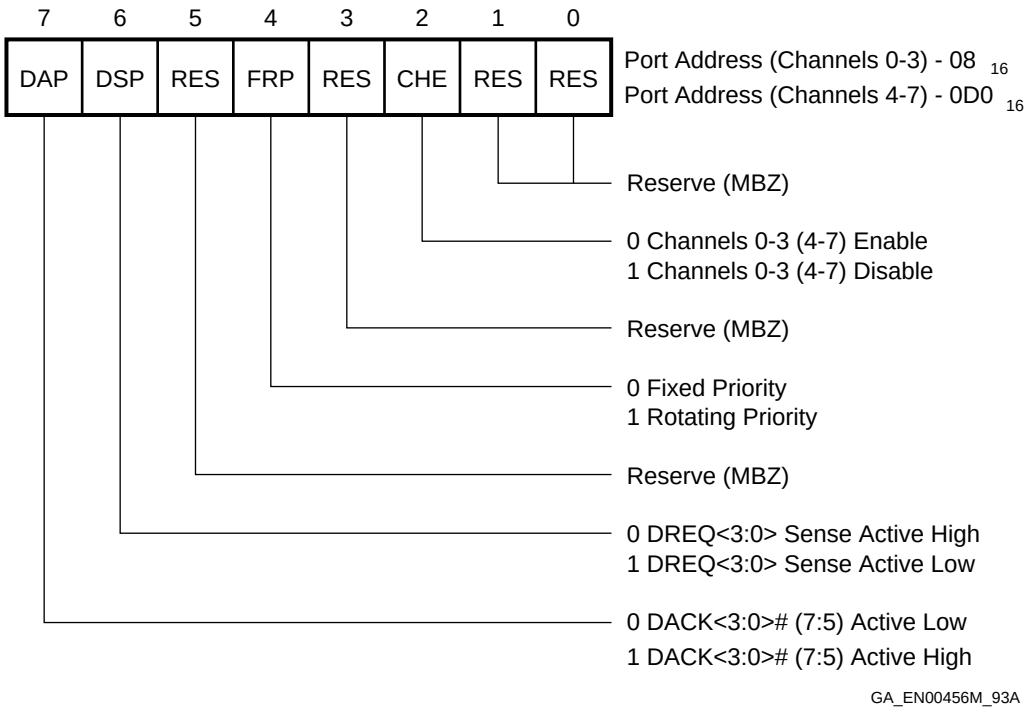
Each channel has a set of base page, base address, and base word count registers. These registers store the original value of their associated current registers. During an autoinitialization, these values are used to restore the current registers to their original values. The base registers are written to simultaneously by the DECchip 21064 CPU with their corresponding current register in 8-bit bytes in the program condition. The DECchip 21064 CPU cannot read these registers.

During chaining mode, these registers can be programmed to store the information about the next buffer in the chain.

Command Register

The command register is a write-only register. This 8-bit register controls the operation of the DMA controller. It is programmed by the DECchip 21064 CPU in the program condition and is cleared by a reset or a master clear instruction. Figure 16–1 shows the format of the command register.

Figure 16–1 Command Register Bits



Note

Disabling channels 4-7 also disables channels 0-3, because channels 0-3 are logically cascaded into channel 4.

Mode Register

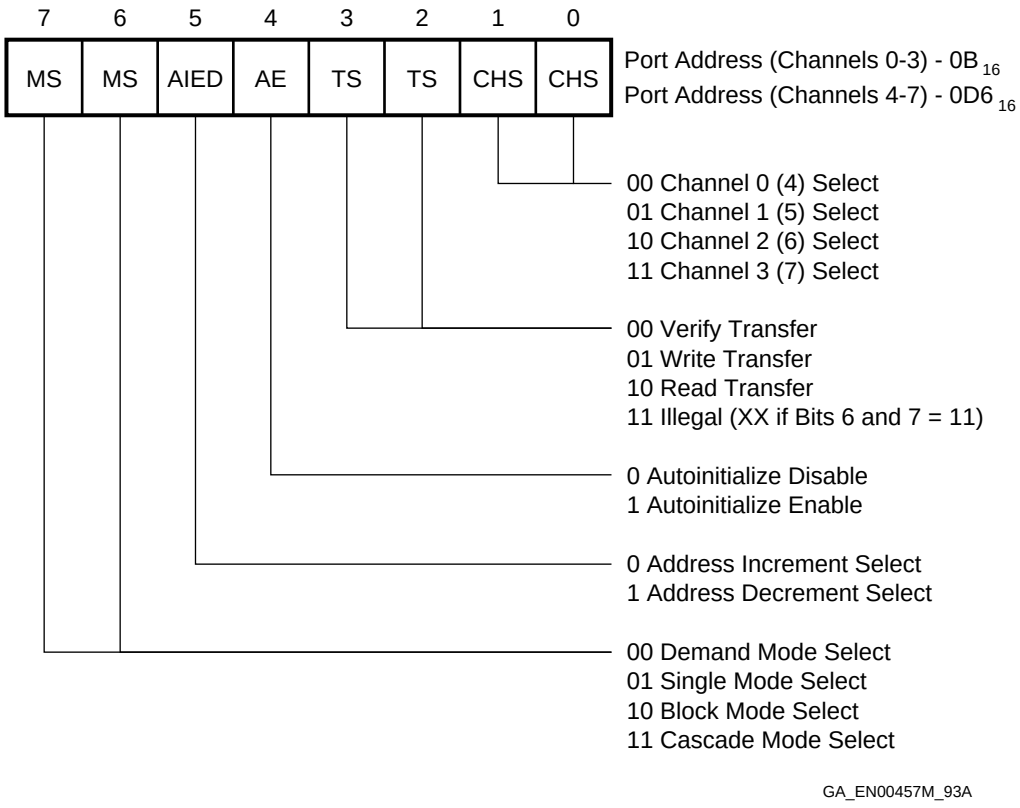
The mode register is a write-only register.

Each channel has an 8-bit mode register associated with it. When the register is being written to, bits 0 and 1 determine which channel is selected. This register is reset when it receives an RST(H) signal and a master clear instruction. Its reset value is as follows:

- Verify transfer
- Autoinitialize disable address increment
- Demand mode

Channel 4 defaults to cascade mode. Figure 16–2 shows the format of the mode register.

Figure 16–2 Mode Register Bits



Note

Channel 4 defaults to cascade mode and you can program for only cascade mode.

Extended Mode Registers

The extended mode register is a write-only register.

Each channel has a 16-bit extended mode register associated with it. This register is used to program the DMA device data size and timing mode. When the register is being written to, bits 0 and 1 determine which channel is selected.

The default programmed values for channels 0-3 are as follows:

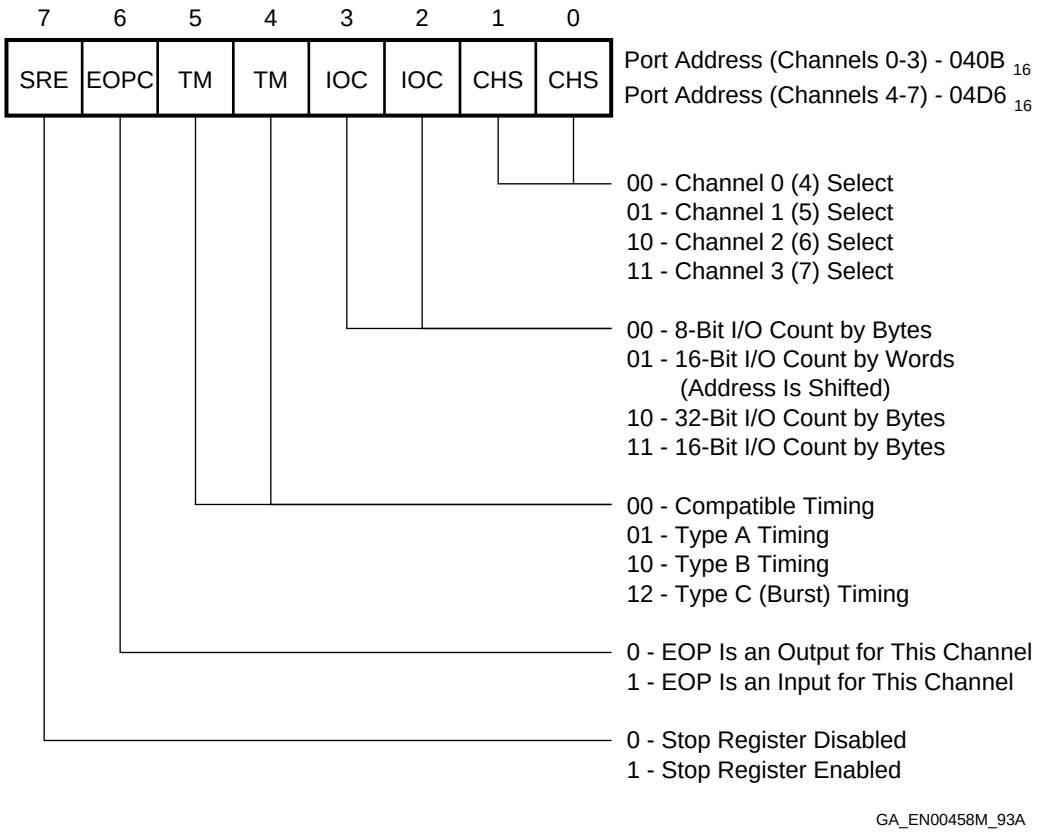
- 8-bit I/O count by bytes
- Compatible timing
- EOP output
- Stop registers disabled

The default values for channels 4-7 are as follows:

- 16-bit I/O count by words with shifted address
- Compatible timing EOP output
- Stop register disabled

The default is selected when reset with an RST(H) signal. It is not selected by a master clear instruction or any other programming sequence. Figure 16–3 shows the format of the extended mode register.

Figure 16–3 Extended Mode Register Bits



8-Bit I/O Count by Bytes Mode

In 8-bit count by bytes mode, you can program the address counter to any address. The count register is programmed with the number of bytes to transfer minus 1. In this mode, byte assembly or disassembly is neither available nor necessary. Therefore, the timing used when 8- or 16-bit memory is sensed is compatible with the original ISA products.

16 Bit I/O Count by Words Mode

In count by words mode (address shifted), you can program the address counter to any even address, but you must program with the address value shifted right by 1 bit.

The page registers are not shifted. This results in the least significant bit of the low-page register being ignored. In this mode, burst timing and byte assembly or disassembly are not available. Therefore, the timing used when 8- or 16-bit memory is sensed is compatible with the original ISA products. The count register is programmed with the number of words to be transferred minus 1.

16 Bit I/O Count by Bytes Mode

In 16-bit count by bytes mode, you can program the address counter to any byte address. For most DMA devices, however, you must program the address counter only to even addresses. If the address is programmed to an odd address, the DMA controller does a partial word transfer during the first and last transfer, if necessary. The bus controller logic does the byte or word assembly necessary to read from or write to any size of memory device. Both the DMA controller and bus controllers support burst for this mode. In this mode, the address register is incremented or decremented by two and the byte count is decremented by the number of bytes transferred during each bus cycle. The count register is programmed with the number of bytes to be transferred minus 1.

32 Bit I/O Count by Bytes Mode

In 32-bit count by bytes mode, you can program the address counter to any byte address. For most DMA devices, however, you must program the address counter only to addresses that are evenly divisible by 4. If you program the address to a value that is not divisible by 4, the DMA controller does partial transfers for the first and last transfer, if necessary. The bus controller logic does the byte or word assembly necessary to read from or write to any size of memory device. Both the DMA controller and bus controllers support burst for this mode. In this mode, the address register is incremented or decremented by 4 and the byte count is decremented by the number of bytes transferred during each bus cycle. The count register is programmed with the number of bytes to be transferred minus 1.

EOP Input or Output Selection

Bit 6 of the extended mode register selects whether the EOP signal is used as an input or an output during DMA transfers. The EOP I/O selection is programmable on a channel-by-channel basis. An EOP is generally used as an output, which was available on the PC/AT. The input function exists to allow data communications and other devices that want to trigger an autoinitialization when a collision or some other event occurs. The direction of EOP is switched when the DACK(L) signal is changed. The EOP and DMA slave signals that are generated by the ISP might overlap. However, during this overlap, both devices send the signal to a low level (inactive).

Stop Register Selection

Bit 7 of the extended mode register selects whether the stop registers associated with this channel are used. Normally, the stop registers are not used. This function exists to support data communications or other devices that work from a ring buffer in memory.

Summary of DMA Transfer Sizes

Table 16–3 lists each of the DMA device transfer sizes. The column labeled *word count register* indicates that the register contents represent either the number of bytes to transfer or the number of 16-bit words to transfer. The column labeled *current address register increment or decrement* indicates the number added to or taken from the current address register after each DMA transfer cycle. The mode register determines if the current address register is incremented or decremented.

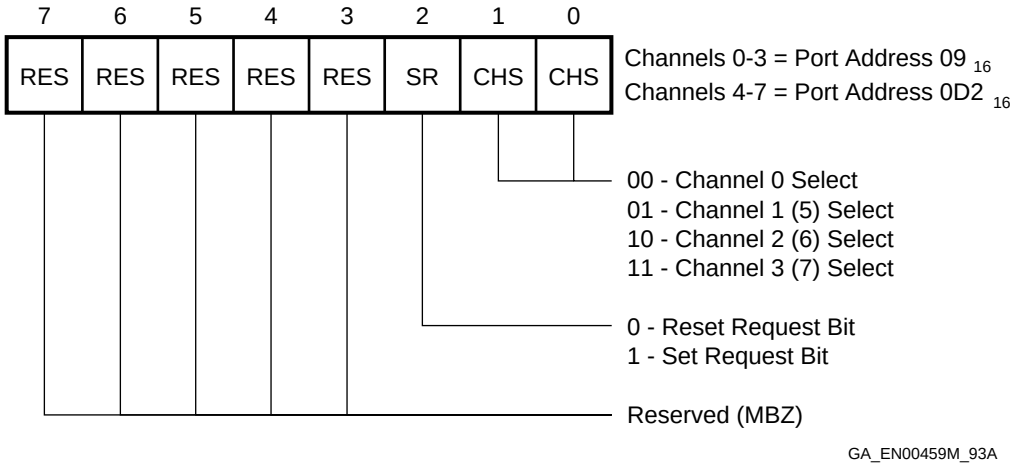
Table 16–3 DMA Device Transfer Sizes

DMA Device Data Size and Word Count	Word Count Register	Current Address Register Increment or Decrement
8-Bit I/O count by bytes	Bytes	1
16-Bit I/O count by words (address shifted)	Words	1
16-Bit I/O count by bytes	Bytes	2
32-Bit I/O count by bytes	Bytes	4

Request Register

Each channel has a request bit associated with it in one of the two 4-bit request registers. The request register is used by software to initiate a DMA request. These requests are nonmaskable and subject to prioritization by the priority encoder network. Each register bit is set or reset separately under software control or is cleared when a terminal count is generated. The register is cleared when a reset signal is received (RST(H)). It is not cleared when an RSTDRV(H) signal is received. To set or reset a bit, your software must load the proper form of the data word. When the register is being written to, bits 0 and 1 determine which channel is selected. To make a software request, the channel must be in block mode (see Figure 16–4).

Figure 16–4 Request Register



Mask Register

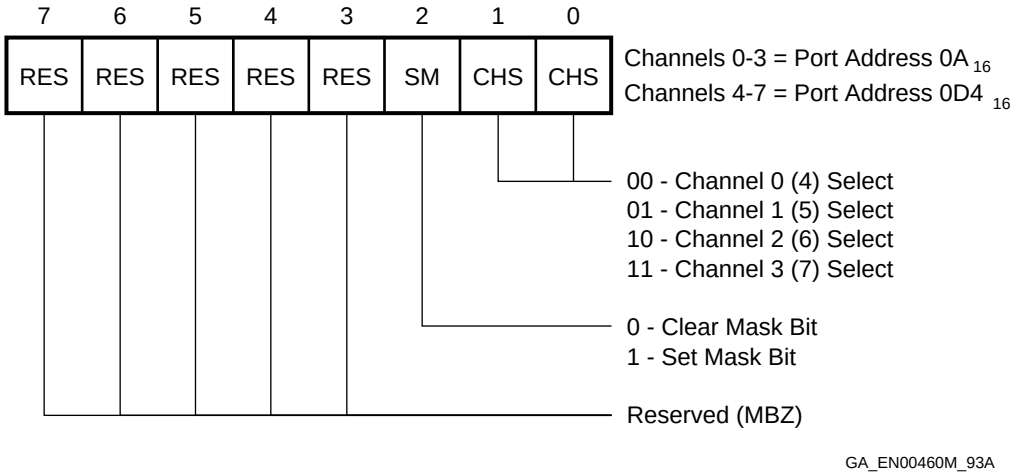
Each channel has associated with it a mask bit that can be set to disable the incoming DREQ(H) signal. Each mask bit is automatically set when the current word count register reaches terminal count, unless the channel is programmed for autoinitialization or chaining mode.

You can set or clear each bit of the two 4-bit registers under software control. The register is also set by a RESET and master clear. This disables all DMA requests until a clear mask register instruction allows them to occur. The instruction to separately set or clear the mask bits is similar to the instruction used with the request register (see Figure 16–5).

Note

If the channel 4 mask bit is set, the channels that are logically cascaded into it are also masked.

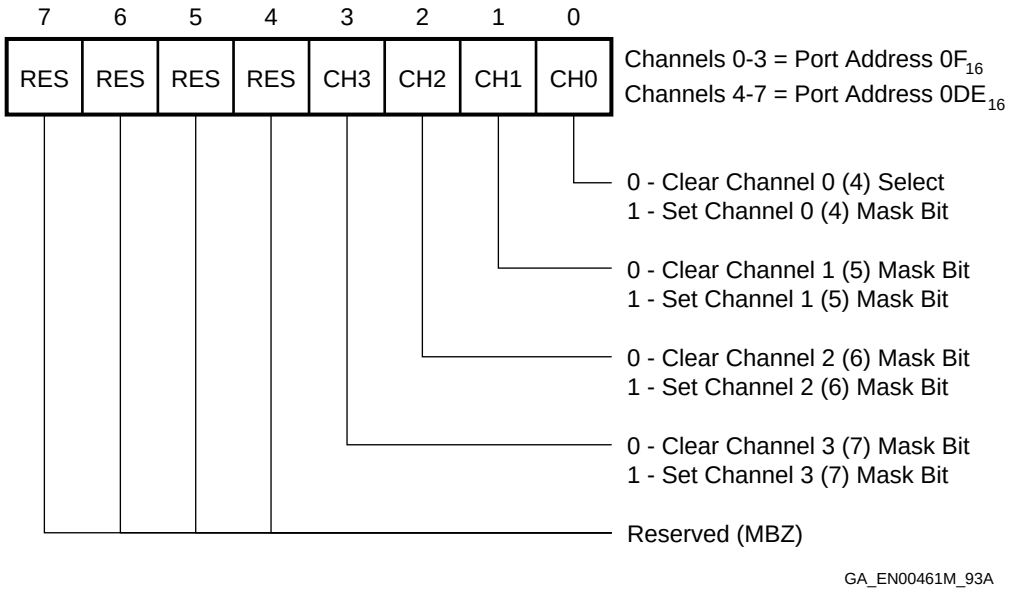
Figure 16–5 Write Single Mask Register



Mask Register

All 4 bits of the mask register can also be read from or written to with a single command (see Figure 16–6).

Figure 16–6 Write All Mask Register

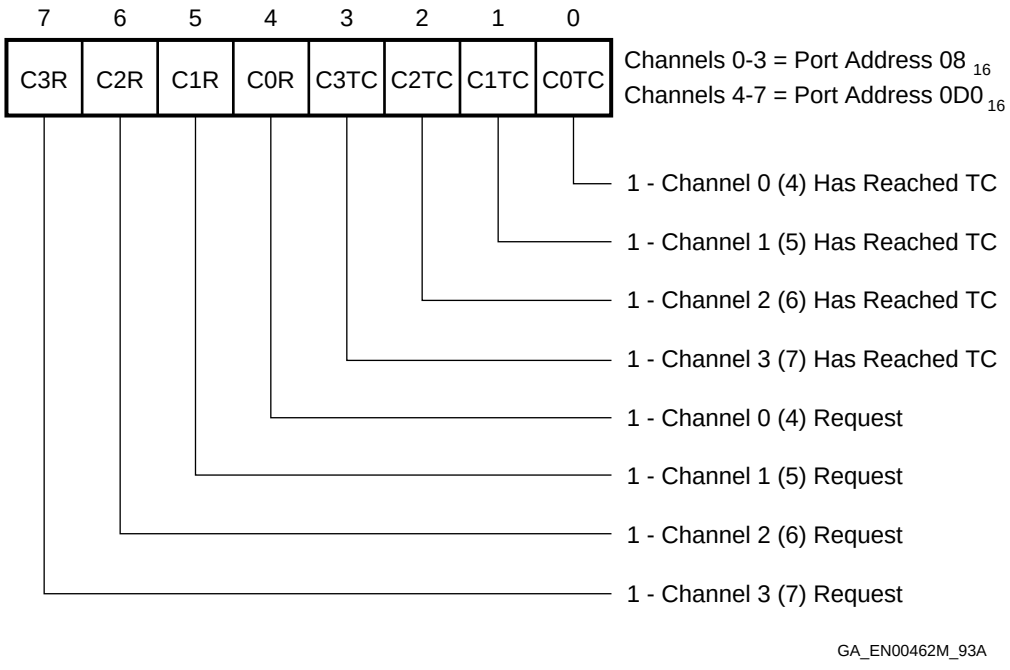


DMA Controller Status Register

The DMA controller status register is a read-only register.

The status register contains information about the status of the devices that can be read from by the DECchip 21064 CPU. This information includes which channels have reached a terminal count and which channels have pending DMA requests. Bits 0-3 are set every time a terminal count is reached by that channel. These bits are cleared on a Reset and on each status read. Bits 4-7 are set when their corresponding channel is requesting a service (see Figure 16–7).

Figure 16–7 DMA Controller Status Register



Note

EISA masters that access the DMA controller registers are not allowed to read from the status registers. Only the host DECchip 21064 CPU can read from the status registers. This is because the terminal count bits are cleared when these registers are read.

Set Chaining Mode Register

The set chaining mode register is a write-only register.

Each channel has a chaining mode register associated with it. The chaining mode register is used to enable or disable DMA buffer chaining and to indicate when the DMA controller base registers are being programmed. When the register is being written to, bits 0 and 1 determine which channel is selected. The chaining status and interrupt status for all channels can be determined by reading the following registers:

- The set chaining mode status register
- The channel interrupt status register
- The chain buffer expiration control register

The chaining mode register is reset to 0 when one of the following events occurs:

- A reset (RST)
- Access (read or write) of a channels's mode register or extended mode register
- A master clear

The values when reset are as follows (see Figure 16–8):

- Disable chaining mode
- Generate IRQ13(H)

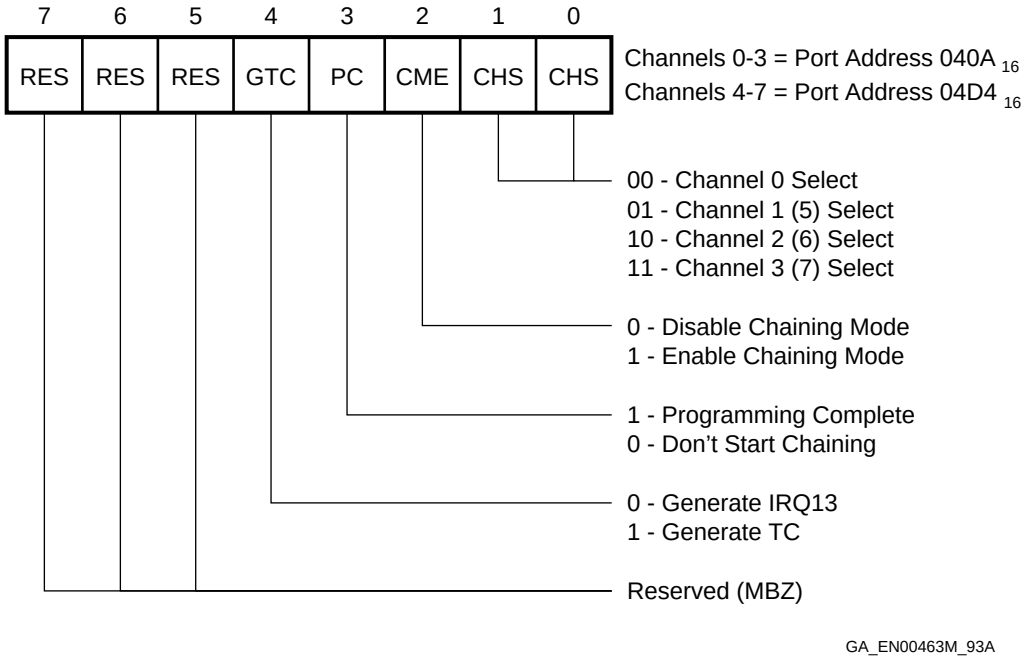
The enable chaining mode bit (CME) is used to control the chaining mode logic. If you program the bit to a 1 after the initial DMA address and count are programmed, then the base address and count registers become available for programming the next chain buffer.

After you program the base registers (as indicated), both the CME bit and the programming complete bit are set to begin a DMA chaining sequence. The DMA channel is then ready to begin transferring data (assuming that the mask bit is cleared).

When a chaining mode interrupt or terminal count (for EISA programming masters) occurs, you must program the next address and count registers and you must set the programming complete bit to prepare for the next transfer. When you set the programming complete bit, the enable chaining mode (CME) bit and the generate IRQ bit both need to be written to the correct state. When this is done, the interrupt request for that channel is reset, if it was active.

Bit 4 of the set chaining mode register is used to determine the response to the next set of base registers.

Figure 16–8 Set Chaining Mode Register

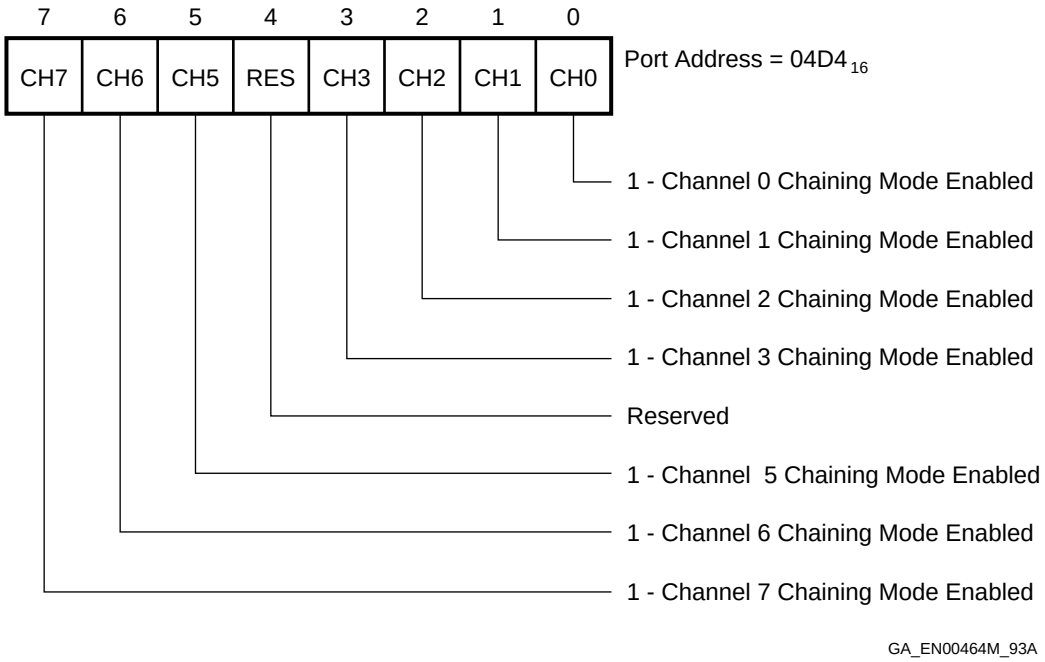


If an EISA bus master is using the DMA controller to assist in data transfer, then bit 4 can be set to a 1 to generate an EOP (terminal count). The EISA master can then use the EOP (terminal count) in the same way that the DECchip 21064 CPU uses an interrupt. In this mode, the EOP signal is active only while the channel that caused it is running, which is determined by the DACK(L) lines.

Set Chaining Mode Status Register

The set chaining mode status register is a read-only register. It is used to determine if chaining mode for a particular channel is enabled or disabled. A 1 read from this register indicates that the channel's chaining mode is enabled. A 0 indicates that chaining mode is disabled. All chaining mode bits are disabled after a reset. After using the DMA channel in chaining mode, the DECchip 21064 CPU must clear the chaining mode enable bit to enable nonchaining mode. This bit is programmed in bit 2 of the set chaining mode register (see Figure 16–9).

Figure 16–9 Set Chaining Mode Register Status

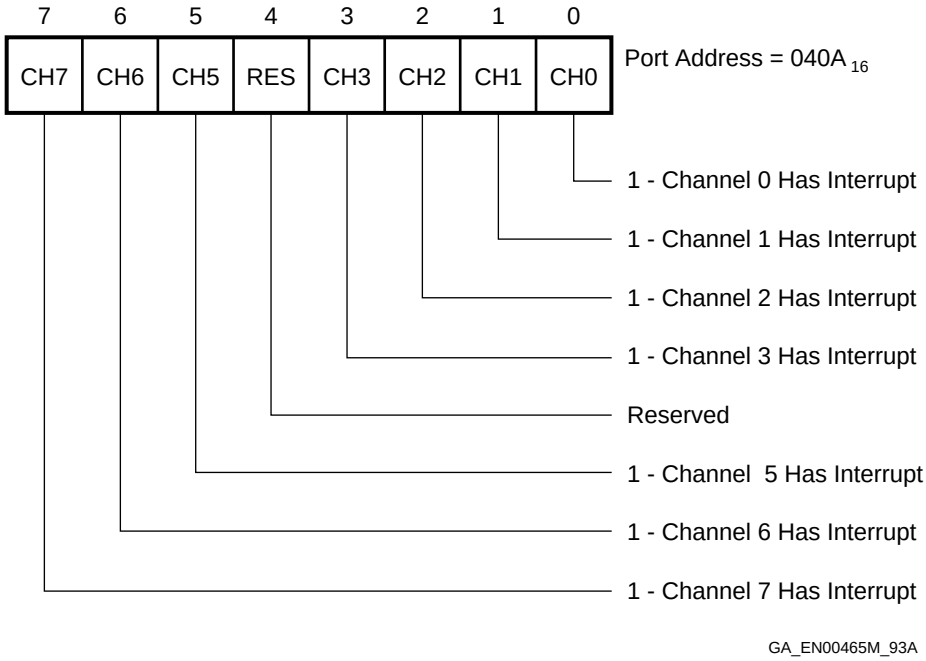


Channel Interrupt Status Register

The channel interrupt status register is a read-only register.

The channel interrupt status register is used to indicate the source (channel) of a DMA chaining interrupt on the IRQ13(H)(H) line. The DMA controller drives the IRQ13(H) line active after reaching terminal count, with chaining mode enabled. It does not drive the IRQ13(H) line active during the initial programming sequence that loads the base registers (see Figure 16–10).

Figure 16–10 Channel Interrupt Status Register

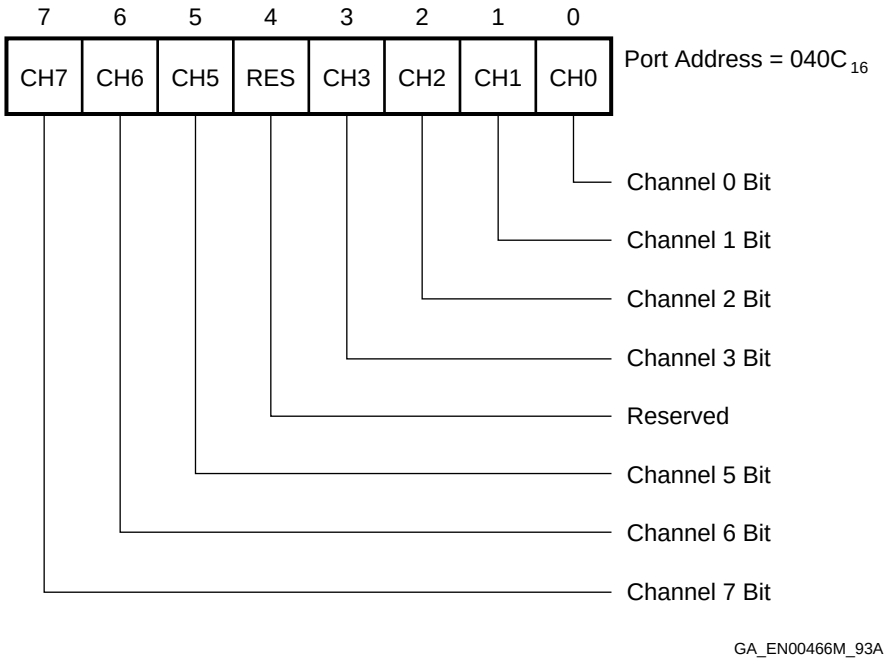


Chain Buffer Expiration Control Register

The chain buffer expiration control register is a read-only register.

The chain buffer expiration control register reflects the outcome of the expiration of a chain buffer. If a channel bit is set (1), a terminal count is issued. This bit is programmed in bit 4 of the set chaining mode register (see Figure 16–11).

Figure 16–11 Chain Buffer Expiration Control Register



DMA Controller Software Commands

These are additional special software commands that can be executed in the program condition. They do not depend on any specific bit pattern on the data bus. The three software commands are as follows; clear byte pointer flip-flop, master clear, and clear mask register. These commands are described in the following sections.

Clear Byte Pointer Flip-Flop Command

This command is executed before writing or reading new address or word count information to the DMA controller. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the DECchip 21064 CPU address upper and lower bytes in the correct sequence.

When the DECchip 21064 CPU is reading from or writing to DMA registers, two byte-pointer flip-flops are used: one for channels 0-3 and one for channels 4-7. Both of these act independently. There are separate software commands for clearing each of them ($0C_{16}$ for channels 0-3, and $0D8_{16}$ for channels 4-7).

An additional byte-pointer flip-flop exists for use when EISA masters are reading from and writing to DMA registers. The arbiter state is used to determine the current master of the bus. This flip-flop is cleared when an EISA master performs a write to either $0C_{16}$ or $0D8_{16}$. There is only one flip-flop for all eight DMA channels. This byte-pointer exists to eliminate the problem of the DECchip 21064 CPU's byte-pointer getting out of synchronization if an EISA master takes the bus during the DECchip 21064 CPU's DMA programming.

Master Clear Command

This software command has the same effect as the hardware reset signal (RST). The command, status, request, and internal first or last flip-flop registers are cleared and the mask register is set. The DMA controller enters the idle cycle.

There are two independent master clear commands: $0D_{16}$ which acts on channels 0-3, and $0DA_{16}$ which acts on channels 4-7.

Clear Mask Register Command

This command clears the mask bits of all four channels, enabling them to accept DMA requests. The I/O port $0E_{16}$ is used for channels 0-3, and the I/O port $0DC_{16}$ is used for channels 4-7.

Terminal Count and EOP Summary

Table 16–4 summarizes the events that happen as a result of a terminal count or external EOP when running DMA cycles in various modes.

Example

Read Table 16–4 vertically. If an event occurs and certain conditions apply, then the results listed occur. If you read the first column vertically in Table 16–4, the following occur:

- Event

  The Word Counter Expired event has the value *yes*.

- Conditions

  The AUTOINIT condition is set to the value *no*, and the Chain and Base Loaded condition has the value *no*.

- Result

  Given the event and conditions described, the Status TC is set, the Mask is set, and the Software Request is cleared.

Table 16–4 Terminal Count and EOP Summary

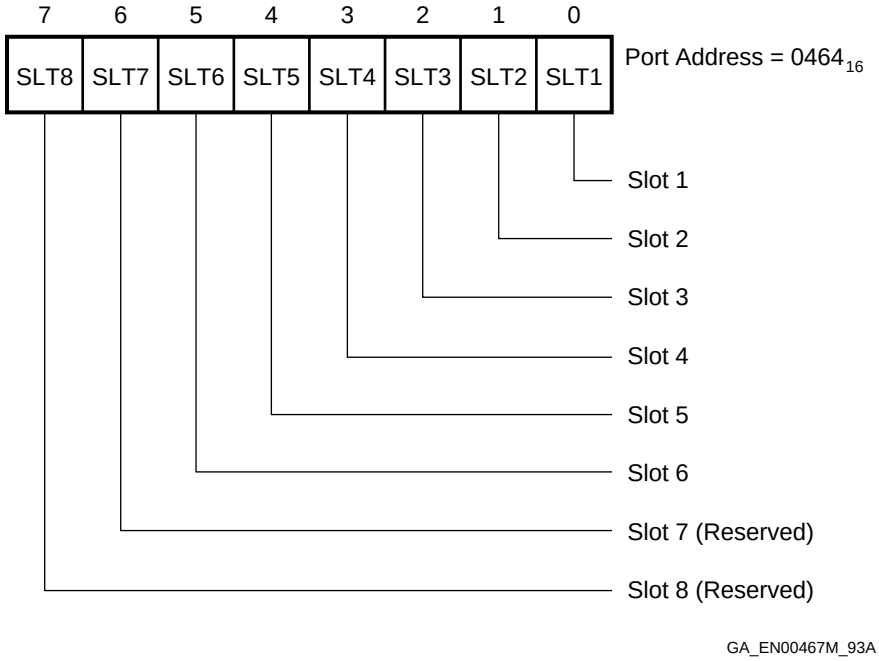
Event								
Word Counter Expired	Yes	X	Yes	X	Yes	X	X	X
Stop Register Limit Reached	X	X	X	X	X	X	Yes	Yes
EOP Input	X	Asserted	X	Asserted	X	Asserted	X	X
Conditions								
Autoinit	No	No	Yes	Yes	No	No	X	X
Chain and Base Loaded	No	No	X	X	Yes	Yes	X	X
Result								
Status TC	Set	Set	Set	Set				
Mask	Set	Set					Set	Set
Software Request	Clear	Clear	Clear	Clear				
Current Register			Load	Load	Load	Load		

EISA Bus Master Status Latch

To simplify testing EISA bus master operations, a DECchip 21064 CPU readable status latch contains information about which EISA bus master most recently had control of the bus. This latch is located at port address 046_{16} and is read-only (see Figure 16–12). A read value of 0 indicates that the slot was most recently granted the bus. An NMI service routine can read this latch to determine which bus master controlled the bus when a bus pre-empt timeout occurred.

Port 0465_{16} is reserved for an additional eight bus master status latch slots, but is not implemented. The bits for slots 7 and 8 are driven by the 82357 chip, though they are always inactive (high).

Figure 16–12 EISA Bus Master Status Latch



17

Interrupt Controller

Introduction

This chapter describes the functions of the Intel 82357 ISP chip interrupt controller.

In This Chapter

This chapter contains the following sections:

- Interrupt Controller Overview
- Interrupt Controller I/O Address Map
- Interrupt Assignments
- Interrupt Details and Registers
- Interrupt Sequence
- 80x86 Mode
- Programming the Interrupt Controller
- Initialization Command Words
- Initialization Command Words 1 and 2
- Initialization Command Word 3 (ICW3)
- Initialization Command Word 4 (ICW4)
- Operation Command Words
- Operation Command Word 1 (OCW1)
- Operation Command Word 2 (OCW2)
- Operation Command Word 3 (OCW3)
- End of Interrupt Operation
- Modes of Operation
- Interrupt Masks

- Reading the Interrupt Controller Status

Interrupt Controller Overview

The interrupt controller consists of two separate 82C59 cores. Interrupt controller 1 (CNTRL-1) and interrupt controller 2 (CNTRL-2) are initialized separately and can be programmed to operate in different modes. The default settings are as follows:

- 80x86 mode
- Edge sensitive (IRQ0-15(H))
- Normal EOI
- Non-buffered mode
- Special fully nested disabled
- Cascade mode

Controller 1 (CNTRL-1) is connected as the master interrupt controller and controller 2 (CNTRL-2) is connected as the slave interrupt controller.

Interrupt Controller I/O Address Map

Table 17–1 lists the I/O port address map for the interrupt registers.

Table 17–1 Interrupt Controller I/O Address Map

Interrupts IRQ*n*(H)	I/O Address n_{16}	Number of Bits	Register
<7:0>	0020	8	CNTRL-1 control register
<7:0>	0021	8	CNTRL-1 mask register
<7:0>	04D0	8	CNTRL-1 edge/level control register
<15:8>	00A0	8	CNTRL-2 control register
<15:8>	00A1	8	CNTRL-2 mask register
<15:8>	04D1	8	CNTRL-2 edge/level control register

Interrupt Assignments

Table 17–2 lists the interrupt inputs to the 82357 chip. The IRQ0(H) and IRQ2(H) signal lines are connected to the interrupt controller internally. The arrangement of EISA interrupt sources is the same as the arrangement of the EISA (ISA) interrupt sources of the 8259A chip's of the PC/AT. The VL82C106 parallel port interrupt is attached to the IRQ1 signal line. In nonrotating priority mode, this is nearly the highest priority interrupt into the 82357 chip.

Digital recommends that you consider writing your code to use specific rotation mode or automatic rotation mode to avoid the printer port locking out other interrupts.

Table 17–2 82357 Interrupt Assignments

Priority	Label	Controller	Interrupt Source
1	IRQ0	1	Interval timer, counter 0 OUT
2	IRQ1	1	Line printer
3-10	IRQ2	1	Interrupt from controller 2 (not used)
3	IRQ8(L)	2	Real-time clock
4	IRQ9	2	EISA bus pin B04
5	IRQ10	2	EISA bus pin D03
6	IRQ11	2	EISA bus pin D04
7	IRQ12	2	EISA bus pin D05
8	IRQ13	2	(Not used)
9	IRQ14	2	EISA bus pin D07 (EISA SCSI host adapter)
10	IRQ15	2	EISA bus pin D06
11	IRQ3	1	EISA bus pin B25
12	IRQ4	1	EISA bus pin B24

(continued on next page)

Table 17–2 (Cont.) 82357 Interrupt Assignments

Priority	Label	Controller	Interrupt Source
13	IRQ5	1	EISA bus pin B23
14	IRQ6	1	EISA bus pin B22 (Diskette controller)
15	IRQ6	1	EISA bus pin B21

Interrupt Details and Registers

Interrupt Request Register and In-Service Register

The interrupts at the IRQ(H) signal input lines are handled by two registers in cascade: the interrupt request register (IRR) and the in-service register (ISR). The IRR is used to store all the interrupt levels that are requesting service. The ISR is used to store all the interrupt levels that are being serviced.

Priority Resolver

The priority resolver determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during interrupt acknowledge cycles.

Interrupt Mask Register

The interrupt mask register (IMR) stores the bits that mask the interrupt lines to be masked. The IMR operates on the IRR. The masking of a higher priority input does not affect the interrupt request lines of lower priority.

Interrupt (INT)

This INT(H) output goes directly to the DECchip 21064 CPU interrupt input IRQ1(H).

Interrupt Acknowledge (INTA)

The interrupt acknowledge INTA (H) signal pulses cause the interrupt controller system to release vectoring information onto the data bus. The format of this data depends on the system mode (μPM) of the interrupt controller (programmed for 80x86 mode in EISA systems). The 82357 chip uses the ST2(H) signal input as the interrupt acknowledge line.

Interrupt Sequence

The powerful feature of the interrupt controller in a microcomputer system is its programming capability and the interrupt routine addressing capability. The interrupt routine addressing capability enables direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The interrupt sequence for the system (8080 mode must never be selected by EISA software) is as follows:

1. One or more of the interrupt request IRQ(H) lines are asserted high, setting the corresponding IRR bits.
2. The interrupt controller evaluates these requests and sends an interrupt (INT(H)) to the DECchip 21064 CPU.
3. The DECchip 21064 CPU acknowledges the interrupt (INT(H)) and responds with an INTA(L) signal pulse.
4. When receiving an INTA(L) signal from the DECchip 21064 CPU, the highest priority ISR bit is set and the corresponding IRR bit is cleared. The interrupt controller does not drive the data bus during this cycle.
5. The DECchip 21064 CPU initiates a second INTA(L) signal pulse. During this pulse, the interrupt controller releases an 8-bit pointer onto the data bus where it is read by the DECchip 21064 CPU.
6. This completes the interrupt cycle. In the AEOI mode, the ISR bit is cleared at the end of the second INTA(L) signal pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step four; that is, the request was too short in duration, the interrupt controller issues an interrupt level 7.

80x86 Mode

In the 80x86 mode, the processor produces only two interrupt acknowledge cycles. The interrupt controller uses the first interrupt acknowledge cycle to internally freeze the state of the interrupts for priority resolution. The first controller (CNTRL-1), as a master, issues the interrupt code on the cascade lines (internal to the 82357) at the end of the INTA(L) signal pulse. On this first cycle, the first controller does not issue any data to the processor and leaves its data bus buffers disabled. On the second interrupt acknowledge cycle, the master (CNTRL-1) or slave (CNTRL-2) sends a byte of data to the processor with the acknowledged interrupt code composed (see Table 17–3). The state of the ADI mode control is ignored and A5-A11 are unused in 80x86 mode.

Note

In Table 17–3, the values T7-T3 represent the interrupt vector address. For more information, see the section entitled Initialization Command Words.

Table 17–3 Content of Interrupt Vector Byte for 80x86 System Mode

Source	D7	D6	D5	D4	D3	D2	D1	D0
IRQ7, 15	T7	T6	T5	T4	T3	1	1	1
IRQ6, 14	T7	T6	T5	T4	T3	1	1	0
IRQ5, 13	T7	T6	T5	T4	T3	1	0	1
IRQ4, 12	T7	T6	T5	T4	T3	1	0	0
IRQ3, 11	T7	T6	T5	T4	T3	0	1	1
IRQ2, 10	T7	T6	T5	T4	T3	0	1	0
IRQ1, 9	T7	T6	T5	T4	T3	0	0	1
IRQ0, 8	T7	T6	T5	T4	T3	0	0	0

Programming the Interrupt Controller

The interrupt controller accepts the following two types of command words generated by the DECchip 21064 CPU or bus master:

- Initialization command words (ICWs)
- Operation command words (OCWs)

These two types of command words are described in the following sections.

Initialization Command Words

Before normal operation can begin, each interrupt controller must be brought to a starting point by a sequence of 2-4 bytes timed by I/O write pulses.

Note

Interrupt controller 2 (CNTRL-2) must be initialized before interrupt controller 1 (CNTRL-1).

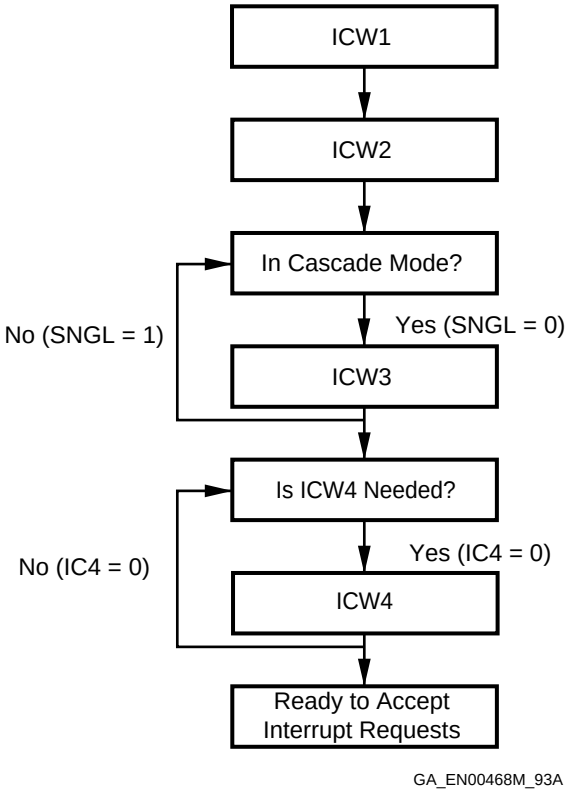
An I/O write to CNTRL-1 or CNTRL-2 base address that has D4 = 1 and A0 = 0 is interpreted as an ICW1. Because the system is an EISA system, two I/O writes to the base address + 1 must follow the ICW1. The first write to the base address + 1 performs ICW2. The second write to the base address + 1 performs ICW3, and a third write performs ICW4. The base address for CNTRL-1 is 020_{16}, and the base address for CNTRL-2 is $0A0_{16}$.

An ICW1 starts the initialization sequence during which the following occur automatically:

1. The edge sense circuit is reset; that is, after an initialization, an interrupt request IRQ(H) signal input must make a low-to-high transition to generate an interrupt.
2. The interrupt mask register is cleared.
3. An IRQ7(H) signal input is assigned priority 7.
4. The slave mode address is set to 7.
5. The special mask mode is cleared and status read is set to IRR.
6. If IC4=0, then all functions selected in ICW4 are set to zero (nonbuffered mode).

Figure 17–1 shows the sequence that the software must follow to load the interrupt controller initialization command words (ICWs). The sequence must be executed for CNTRL-1 and CNTRL-2.

Figure 17–1 Initialization Sequence



Initialization Command Words 1 and 2

In EISA systems, the interrupt controllers are programmed for 80x86 mode. In an 80x86 system, the A15-A11 local bus addresses are inserted in the five most significant bits of the vectoring byte, and the interrupt controller sets the 3 least significant bits according to the interrupt level. The A10-A5 local bus addresses are ignored and the address interval (ADI) has no effect. Table 17–4 describes the ICW1 bits and their function in initializing the interrupt controller.

Table 17–4 ICW1 Bit Definitions

Bit	Description
LTIM	This bit is disabled in EISA systems. Its function is replaced by the edge/level triggered control register (ELCR). It enables you to program each interrupt to either edge or level mode on a channel-by-channel basis.
ADI	Ignored for EISA.
SNGL	This bit is set to 0 for EISA systems. It indicates that there is more than one interrupt controller in the system.
IC4	If this bit is set, ICW4 must be read. If ICW4 is not needed, set IC4=0.

The ICW2 initializes the interrupt controller with the 5 most significant bits of the interrupt vector address. Figure 17–2 and Figure 17–3 illustrate the ICW1 and ICW2 command words.

Initialization Command Words 1 and 2

Figure 17–2 ICW1

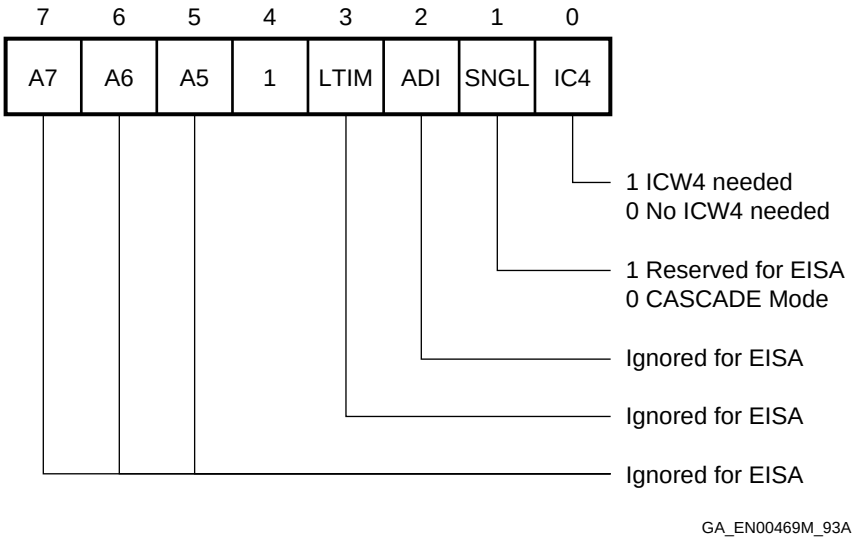
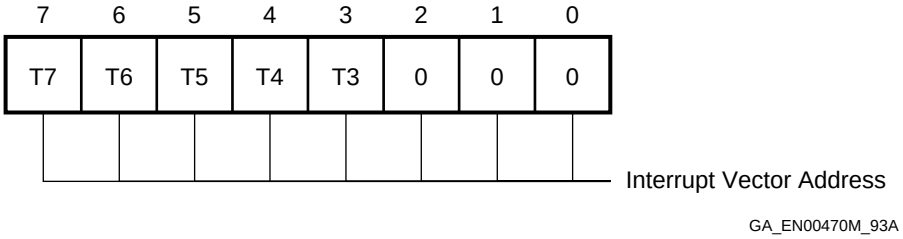


Figure 17–3 ICW2



Initialization Command Word 3 (ICW3)

This initialization command word (ICW3) is read only in EISA systems. An interrupt request on the IRQ2(H) line causes CNTRL-1 to enable CNTRL-2 to present the interrupt vector address during the second interrupt acknowledge cycle (see Figure 17–4 and Figure 17–5).

Figure 17–4 ICW3 (Master Device)

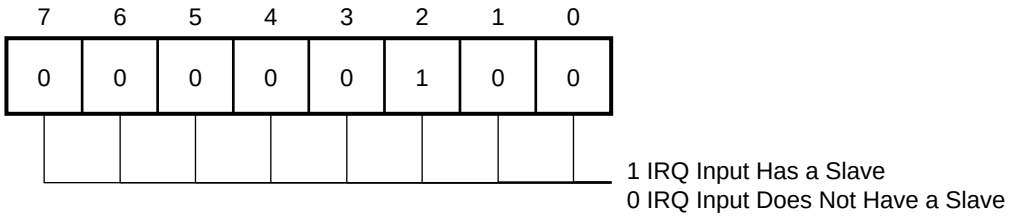
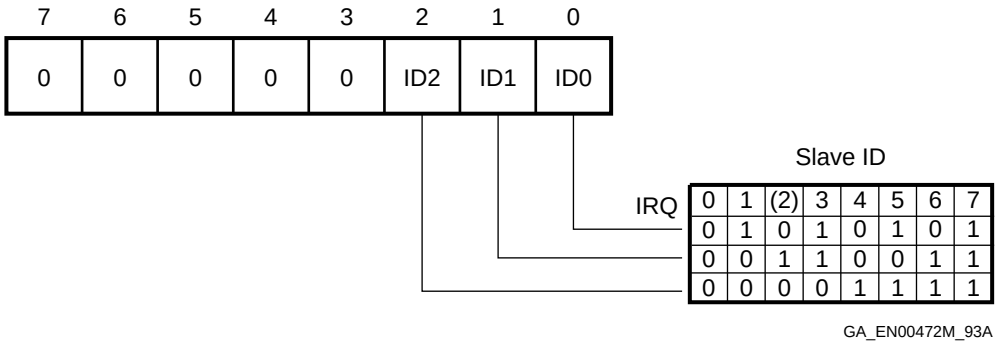


Figure 17–5 ICW3 (Slave Device)



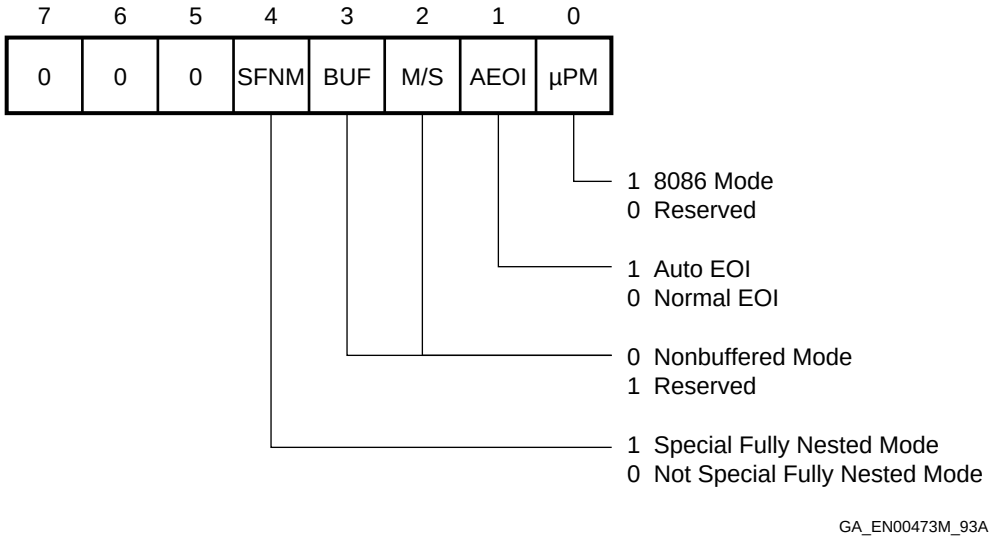
Initialization Command Word 4 (ICW4)

The ICW4 bits are defined in Table 17–5.

Table 17–5 ICW4 Bit Definitions

Bit	Description
SFNM	If SFNM = 1, the special fully nested mode is programmed.
BUF	Programmed to 0 for EISA systems.
M/S	Ignored for EISA systems.
AEOI	If AEOI = 1, the automatic end of interrupt mode is programmed.
μPM	Microprocessor mode. The μPM bit must be set to 1 for EISA systems. This sets the interrupt controller for 80x86 system operation.

Figure 17–6 ICW4 021H (CNTRL-1) or 0A1H (CNTRL-2)



Operation Control Words

After the initialization command words (ICWs) are programmed into the interrupt controller, the chip is ready to accept interrupt requests at its input lines. However, during the interrupt controller operation, a selection of algorithms can command the interrupt controller to operate in various modes through the operation command words (OCWs). The following sections describe each of the operation command words.

Operation Command Words

The OCWs are the command words that command the interrupt controller to operate in various interrupt modes. These modes are as follows:

- Fully nested mode
- Rotating priority mode
- Special mask mode
- Polled mode

You can write the OCWs into the interrupt controller at any time after initialization. Table 17–6 lists the values of the interrupt controller parameters that are initialized by firmware at power-up.

Table 17–6 Initial Interrupt Controller Values

Port n_{16}	Value n_{16}	Description of Contents
020	11	CNTRL-1, ICW1
021	08	CNTRL-1, ICW2 vector address for 000020
021	04	CNTRL-1, ICW3 indicates slave connection
021	01	CNTRL-1, ICW4 8086 mode
021	B8	CNTRL-1, interrupt mask
04D0	00	CNTRL-1, edge/level control register

(continued on next page)

Table 17–6 (Cont.) Initial Interrupt Controller Values

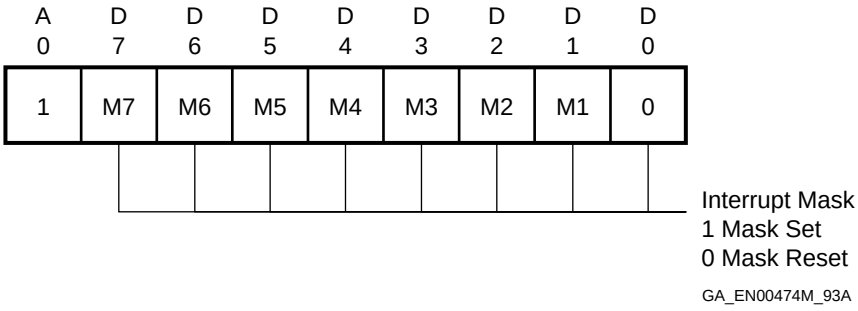
Port n_{16}	Value n_{16}	Description of Contents
0A0	11	CNTRL-2, ICW1
0A1	70	CNTRL-2, ICW2 vector address for 00001C
0A1	02	CNTRL-2, ICW3 indicates slave ID
0A1	01	CNTRL-2, ICW4 8086 mode
04D1	00	CNTRL-2, edge/level control register
021	BD	CNTRL-2, interrupt mask

Operation Command Word 1 (OCW1)

The OCW1 is a read/write control word.

The OCW1 sets and clears the mask bits in the interrupt mask register (IMR). The M7-M0 bits represent the 8 mask bits. When the M bit is set to 1, the channel is masked (inhibited). When the M bit is set to 0, the channel is enabled (see Figure 17–7).

Figure 17–7 OCW1

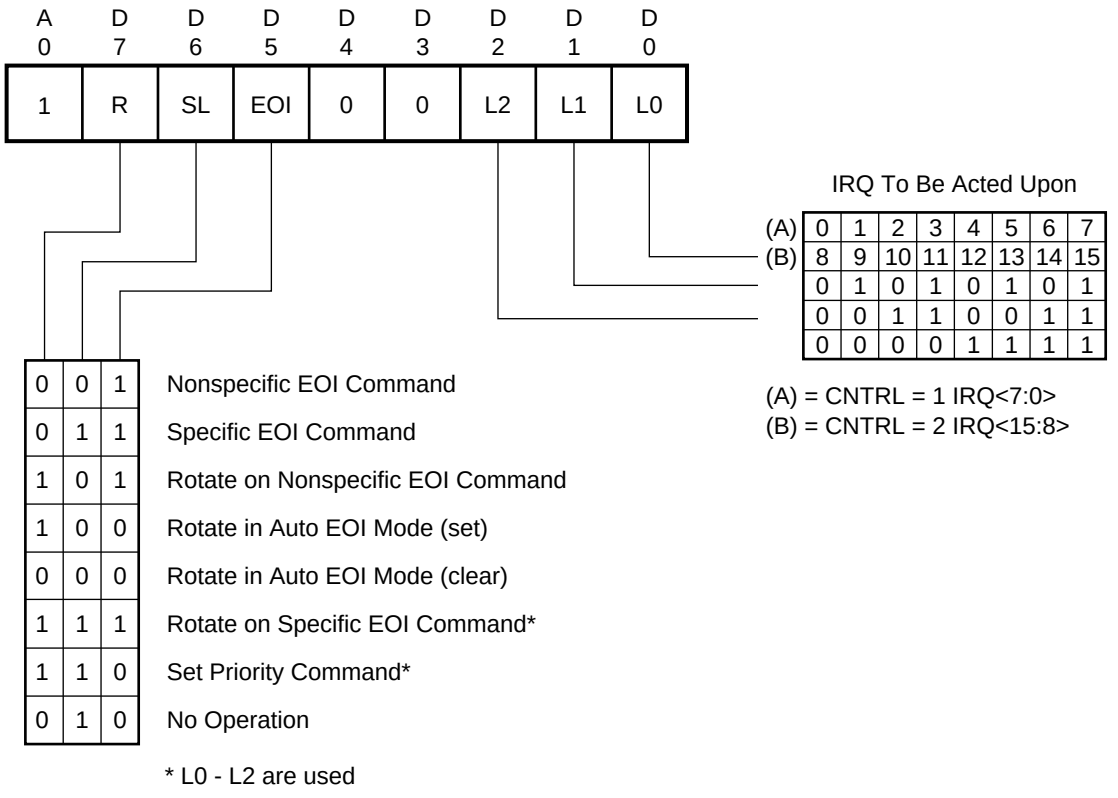


Operation Command Word 2 (OCW2)

The R, SL and EOI bits control the rotate and end of interrupt modes and combinations of the two. These combinations are shown in Figure 17–8.

The L2, L1 and L0 bits determine the interrupt level acted on when the SL bit is active (see Figure 17–8).

Figure 17–8 OCW2

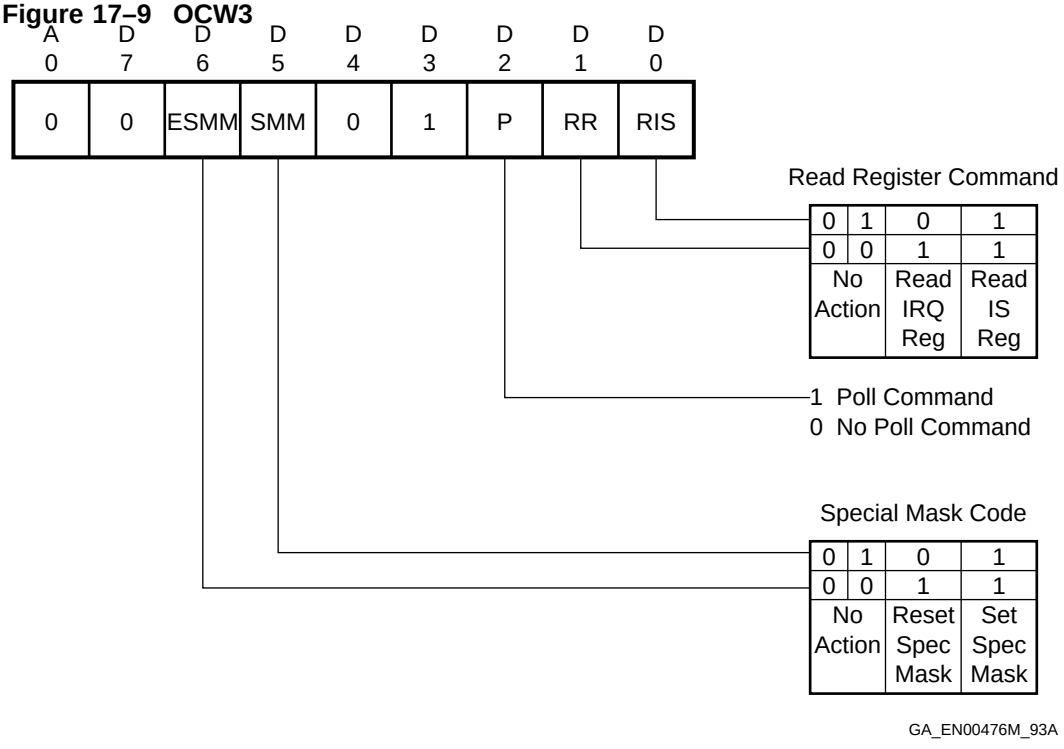


Operation Command Word 3 (OCW3)

When the enable special mask mode bit (ESMM) is set (1), it enables the special mask mode bit (SSM) to set or reset the special mask mode. When the ESMM bit is clear, the SMM bit becomes a don't care bit.

If the enable special mask mode bit is set (1), and the special mask mode bit is set (1), then the interrupt controller enters special mask mode. If the ESMM bit = 1 and the SSM bit = 0, then the interrupt controller reverts to normal mask mode. When the ESMM bit = 0, the SMM bit has no effect (see Figure 17–9).

Figure 17–9 OCW3



End of Interrupt Operation

End of Interrupt (EOI)

The in service (IS) bit can be reset in either of the following ways:

- Automatically following the trailing edge of the last in service INTA(L)(L) pulse, when the AEOI bit in ICW1 is set.
- By a command word that must be issued to the interrupt controller before returning from a service routine (EOI command).

You must issue an EOI command twice if the interrupt controller is in the cascade mode: once for the master and once for the slave. You can use the following two types of EOI commands:

- Specific
- Nonspecific

When the interrupt controller operates in modes that preserve the fully nested structure, it can determine which IS bit to reset on end of interrupt (EOI). When a nonspecific EOI command is issued, the interrupt controller automatically resets the highest IS bit of those that are set. It does this because in the fully nested mode, the highest IS level was not necessarily the last level acknowledged and serviced. You can issue a nonspecific EOI command with an OCW2 command (EOI=1, SL=0, R=0).

When a mode is used that disturbs the fully nested structure, the interrupt controller may no longer be able to determine the last level acknowledged. In this case, you must issue a specific EOI command that includes, as part of the command, the IS level to be reset. You can issue a specific EOI command with an OCW2 command (EOI=1, SL=1, R=0 with L0-L2 representing the binary level of the IS to be reset).

Note

An IS bit that is masked by an IMR bit is not cleared by a nonspecific EOI command if the interrupt controller is in the special mask node.

Automatic End of Interrupt (AEOI)

If automatic end of interrupt (AEOI) bit in ICW4 is set to 1, then the interrupt controller operates in AEOI mode until reprogrammed by ICW4. In this mode, the interrupt controller automatically performs a nonspecific EOI operation at the trailing edge of the last interrupt acknowledge pulse. This mode must be used in the system only when a nested multilevel interrupt structure is not required with a single interrupt controller. The AEOI mode can only be used in a master interrupt controller and not a slave interrupt controller.

Modes of Operation

The interrupt controller has the following seven modes of operation:

- Fully nested mode
- Special fully nested mode
- Automatic rotation
- Specific rotation
- Poll command
- Cascade mode
- Edge and level trigger modes

The seven operating modes are described in the following sections.

Fully Nested Mode

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered from 0-7 (0 = highest). When an interrupt is acknowledged, the highest priority request is determined and its vector placed on the bus. Additionally, a bit in the interrupt service register is set. This bit remains set until the microprocessor issues an EOI command immediately before returning from the service routine, or if the AEOI bit is set, until the trailing edge of the last INTA(L)(L) signal. While the IS bit is set, all further interrupts of the same or lower priority are inhibited. Higher priority interrupts generate an interrupt that is acknowledged by the DECchip 21064 CPU, if interrupts are enabled.

After the initialization sequence, the IRQ0 signal has the highest priority and the IRQ7(H) signal has the lowest priority. You can change the priorities in the rotating priority mode.

Special Fully Nested Mode

You can use this mode when cascading is used, and the priority has to be conserved within each slave. In this case, the fully nested mode is programmed to the master using ICW4. This mode is similar to the normal nested mode with the following exceptions:

- When an interrupt request from a certain slave is in service, this slave is not locked out from the master's priority logic. Further interrupt requests from higher priority IRQs within the slave are recognized by the master and initiate interrupts to the processor.

  In normal nested mode, a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.

- When exiting from the interrupt service routine, your software must check whether the interrupt service was the only one from that slave. Do this by sending a nonspecific EOI command to the slave and then reading its in-service register and checking for zero. If the register is empty, send a nonspecific EOI to the master also. If it is not empty, do not send an EOI.

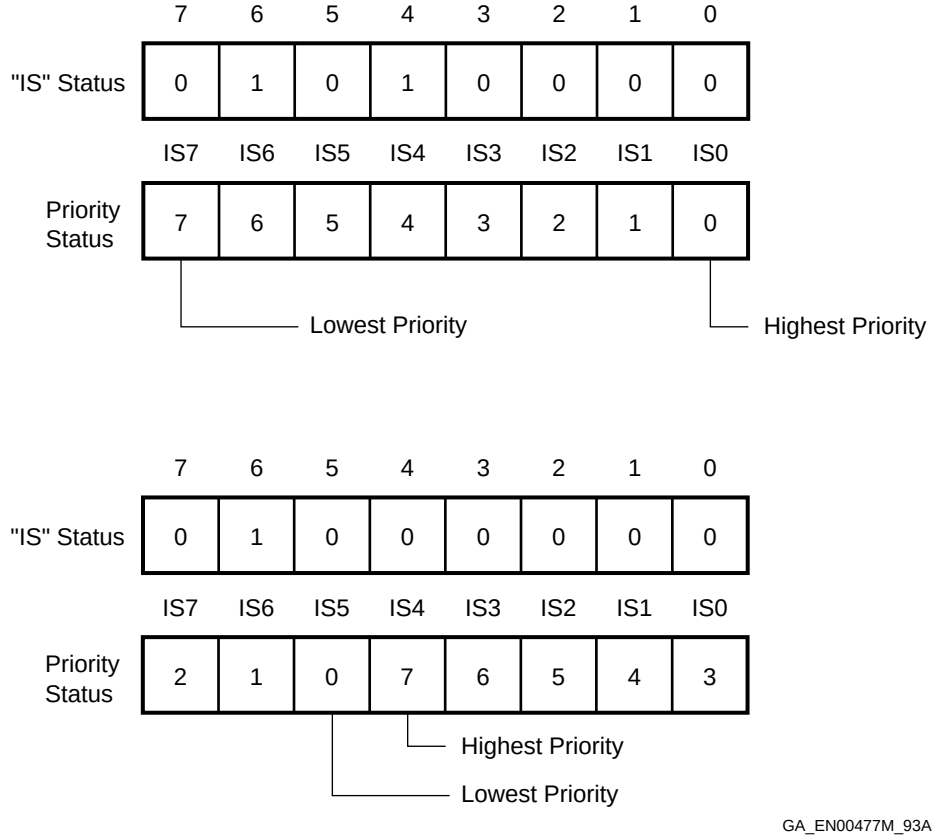
Automatic Rotation (Equal Priority Devices)

In some cases, there are a number of interrupting devices of equal priority. In this mode, a device receives the lowest priority after being serviced. Therefore, this device has to wait in the worst case until each of the seven other devices are serviced at least once (see Figure 17–10).

There are two ways to accomplish automatic rotation using OCW2, as follows:

- Rotation on nonspecific EOI command (R=1, SL=0, EOI=1)
- Rotate in automatic EOI mode, which is set by (R=1, SL=0, EOI=0) and cleared by (R=0, SL=0, EOI=0).

Figure 17–10 Automatic Rotation



Specific Rotation (Specific Priority)

You can change priorities by programming the lowest priority, which sets all other priorities. That is, if you program the IRQ5(H) device as the lowest priority device, then the IRQ6(H) device has the highest priority.

You can issue the set priority command in OCW2 by setting R to 1, SL to 1, and L0-L2 to the binary priority level code of the lowest priority device.

In this mode, the internal status is updated by software control during OCW2. However, it is independent of the EOI command (also executed by OCW2). You can change priorities during an EOI command by using the rotate on specific EOI command in OCW2 (R=1, SL=1, EOI=1 and L0-L2=IRQ level to receive the lowest priority).

Poll Command

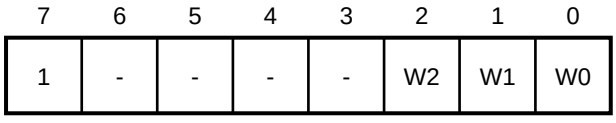
In this mode, the INT(H) signal output of the 82357 chip is not used and the microprocessor internal interrupt enable flip-flop is reset, disabling its interrupt input. Use a poll command service devices.

You can issue the poll command by setting the P bit in OCW3. The interrupt controller does the following:

- Treats the next I/O read pulse to the interrupt controller as an interrupt acknowledge
- Sets the appropriate IS bit if there is a request
- Reads the priority level

The interrupt is frozen from the I/O write to the I/O read. The word that is enabled onto the data bus during the I/O read is shown in Figure 17–11.

Figure 17–11 Word Format for Polling Command I/O Read



GA_EN00478M_93A

Bits W0-W2 represent the binary code of the highest priority level requesting service. Bit D7 is set (1), if there is an interrupt.

This mode is useful if there is a routine command that is common to several levels, in which case, the INTA(L) sequence is not needed.

Cascade Mode

The interrupt controllers in EISA systems are interconnected in a scheme of one master with one slave to handle up to 15 priority levels.

In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master enables the corresponding slave to release the device routine address during byte 2 of INTA(L).

Each interrupt controller in the system must follow a separate initialization sequence, and you can program them to work in different modes. You must issue an EOI command twice: one for the master and one for the slave.

Edge- and Level-Triggered Modes

In ISA systems, you can program this mode using bit 3 in ICW1. In EISA systems, the LTIM bit is disabled and a new register for edge- and level-triggered mode selection per interrupt input is included. This new register is the edge and level control register (ELCR). The default programming is equivalent to programming the LTIM bit (ICW1, bit 3) to 0.

If an ELCR bit is clear (0), an interrupt request is recognized by a low-to-high transition on the corresponding IRQ(H) input. The IRQ(H) input can remain high without generating another interrupt.

If an ECLR bit is set (1), an interrupt request is recognized by a low level on the corresponding IRQ(H) input, and there is no need for an edge detection. For level-triggered interrupt mode, you must remove the interrupt request signal before issuing the EOI command, or you must disable the CPU interrupt. This is necessary to prevent a second interrupt from occurring.

In both the edge and level-triggered modes, the IRQ(H) inputs must remain active until after the falling edge of the first INTA(L). If the IRQ(H) input goes inactive before this time, a default IRQ7(H) signal occurs when the CPU acknowledges the interrupt. This is a useful safeguard for detecting interrupts caused by spurious noise glitches on the IRQ(H) signal inputs. To implement this

feature, the IRQ7(H) routine is used for clean-up by executing a return instruction, and so ignoring the interrupt.

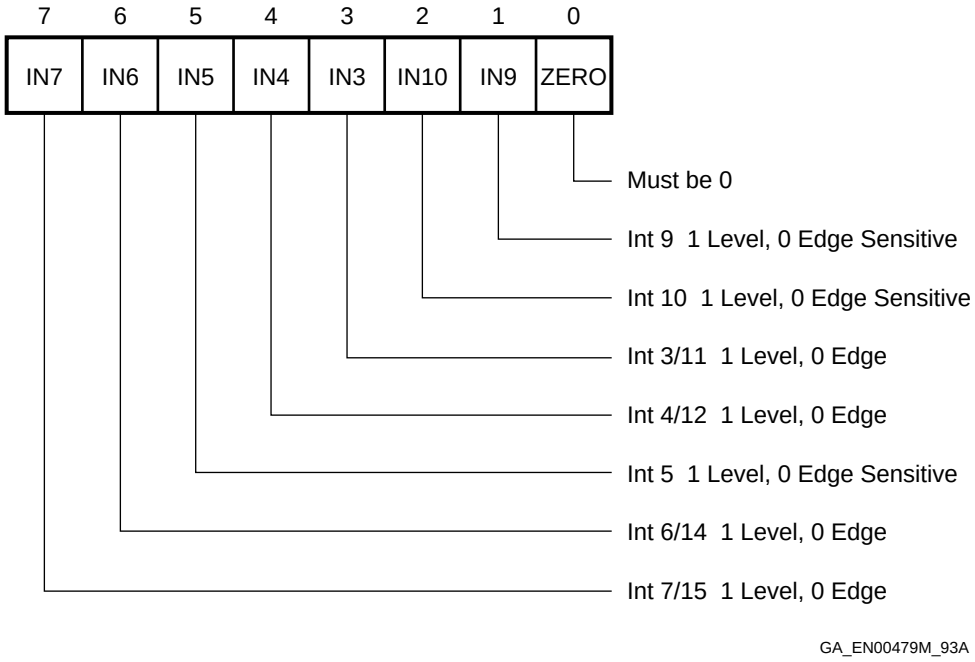
If you need to use the IRQ7(H) interrupt for other purposes, you can still detect a default IRQ7(H) interrupt by reading the ISR. A normal IRQ7(H) interrupt sets the corresponding ISR bit, while a default IRQ7(H) interrupt does not. If a default IRQ7(H) routine occurs during a normal IRQ7(H) routine, the ISR remains set. In this case, you need to keep track of whether the IRQ7(H) routine was previously entered. If another IRQ7(H) interrupt occurs, it is a default.

Edge and Level-Triggered Control Register

The edge and level-triggered control register is a read/write register.

There are two edge and level-triggered control registers (ELCRs), one for each 82C59 bank. They are located at I/O ports $04D0_{16}$ (for the master bank, IRQ<0:1,3:7>) and $04D1_{16}$ (for the slave bank, IRQ<8(L):15>). They enable you to select edge and level-sense on an interrupt-by-interrupt basis, instead of on a complete bank. You can program for level sensitivity only the interrupts that connect to the EISA bus. That is, you must program IRQS 0, 1, 2, 8(L), and 13 for edge sensitive operation (see Figure 17–12).

Figure 17–12 ECLR Register Format



Interrupt Masks

This section describes the 82357 chip's interrupt masks. There are two types of interrupt masks, as follows:

- Masking on an individual interrupt request basis
- Special mask mode

The interrupt masks are described in the following sections.

Masking on an Individual Interrupt Request Basis

You can mask each interrupt request input by using the interrupt mask register (IMR), which you can program using OCW1. Each bit in the IMR masks one interrupt channel if it is set (1). Bit 0 masks IRQ0(H), bit 1 masks IRQ1(H), and so on. Masking an IRQ channel does not affect the way other channels operate.

Special Mask Mode

Your application may require an interrupt service routine to dynamically alter the system priority structure during its execution. For example, you may want your application to inhibit lower priority requests for a portion of its execution, but enable some of them for another portion.

A problem occurs if an interrupt request is acknowledged and an EOI command does not clear its IS bit while executing a service routine. The interrupt controller inhibits all lower priority requests and it is difficult for the service routine to enable them. You can use the special mask mode to solve the problem.

In the special mask mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level and enables interrupts from all other lower and higher levels that are not masked. You can selectively enable interrupts by loading the mask register.

You can set the special mask mode in OCW3 by setting the ESMM bit to 1 and the SMM bit to 1. You can clear the special mask mode in OCW3 by setting the ESMM bit to 1 and the SMM bit to 0.

Reading the Interrupt Controller Status

You can read the interrupt status of several internal registers to update the user information on the system. Table 17–7 lists the registers that can be read by OCW3 (IRR and ISR) or OCW1 ([IMR]).

Table 17–7 Reading Registers for Interrupt Controller Status

Name	Description
Interrupt request register (IRR)	This is an 8-bit register that contains the levels requesting an interrupt to be acknowledged. The highest request level is reset from the IRR when an interrupt is acknowledged. (Not affected by IMR.)
In-Service register (ISR)	This is an 8-bit register that contains the priority levels that are being serviced. The ISR is updated when an EOI command is issued.
Interrupt mask register (IMR)	This is an 8-bit register that contains the interrupt request lines that are masked.

You can read the IRR when you issue a read register command with OCW3 (RR = 1, RIS = 0) before an RD pulse.

You can read the ISR when you issue a read register command with OCW3 (RR = 1, RIS = 1) before an RD pulse.

You do not need to write an OCW3 command before every status read operation, provided that the status read corresponds to the previous one; that is, the interrupt controller detects whether the IRR or the ISR has been previously selected by the OCW3 command. However, this is not true when you use a poll command.

After initialization, the interrupt controller is set to IRR.

You do not need to issue an OCW3 command to read the IMR. The output data bus contains the IMR when an I/O read is active and the address is 021_{16} or 061_{16} (OCW1).

Polling overrides a status read when P=1 and RR=1 in OCW3.

18

Nonmaskable Interrupt Ports

Introduction

This chapter describes nonmaskable interrupts (NMIs).

In This Chapter

This chapter contains the following sections:

- Overview
- NMI Status and Control Register
- NMI Extended Status and Control Register
- Software NMI Generation
- NMI Enable and Disable and Real-Time Clock Address

Overview

An NMI is an interrupt that requires immediate attention and that has priority over the normal IRQ interrupt lines. An NMI indicates errors that are caused by either the hardware or the software.

Causes of Nonmaskable Interrupts

An NMI is caused by the following conditions:

- Parity errors on the system module memory. The system module uses the PARITY(L) line to indicate memory errors.
- Parity errors on the add-in memory boards on the ISA expansion bus. The IOCHK(L) signal is driven low when this error occurs.
- Time-out of the fail-safe timer counter 0 on the interval timer 2, which is used to prevent the system from locking up. This NMI is sensed with a rising-edge detect latch.
- Time-out of an 8 μs 32-bit bus master timeout. If a 32-bit bus master retains the bus for more than 8 μs after a MACK(L) signal goes inactive, the 82357 chip drives the NMI(H) and RESDRV(H) signals active. The RESDRV(H) signal remains active until the NMI is reset.

  Note

  An NMI is not generated if the CPU keeps the bus longer than the 8 μs timeout.

- Timeout of the 32 μs CMD(L) signal active timer.
- Software writing to the NMI I/O interrupt port (0462_{16}). This is a special port that when written to causes an immediate NMI, provided that port 070_{16} is enabled.

Table 18–1 lists the NMI sources and the enable or disable bits and the status read bits for each source.

Table 18–1 NMI Source Enable or Disable and Status Bits

NMI Source	I/O Port Bit for Status Reads	I/O Port Bit for Enable or Disable
PARITY(L)	Port 061_{16} bit <7>	Port 061_{16} bit <2>
Fail-safe timer	Port 0461_{16} bit <7>	Port 0461_{16} bit <2>
IOCHK(L)	Port 061_{16} bit <6>	Port 061_{16} bit <3>
Bus timeout	Port 0461_{16} bit <6>	Port 0461_{16} bit <3>
Write to port 0462_{16}	Port 0461_{16} bit <5>	Port 0461_{16} bit <1>

Nonmaskable Interrupt Registers

The NMI logic consists of four 8-bit registers. The NMI registers are addressed as ports 061_{16}, 070_{16}, 0461_{16}, and 0462_{16}. The status of ports 0461_{16} and 061_{16} are read by the CPU to determine which source caused the NMI. Bits set to 1 in these ports indicate which source requested an NMI. Bits in these ports are cleared by software when the interrupt routine has processed the NMI. The bits are cleared setting the corresponding enable and disable bit high.

Port 070_{16} is the mask register for the NMIs. This register can mask the NMI signal and can also enable or disable all NMI sources.

If you want to reset the system bus without resetting other devices in the system (standard system module devices are not reset), you must set port 0462_{16} bit <0> to 1. Keep bit <0> set to 1 for the length of time that you want the RSTDRV(H) signal to remain active, then reset bit <0> to 0, its normal state.

If a 32-bit bus master tries to keep the bus longer than 8 μs, or if the CMD(L) signal is active for more than 32 μs, the 82357 chip drives the NMI(H) and RSTDRV(H) signals active. The RSTDRV(H) signal remains active until the NMI is reset by resetting port 0461_{16} bit <3> to 0.

Nonmaskable Interrupt Service Routines

To service all NMI requests, your software must meet the requirements for the edge detect circuitry used in 80386 and 80486 systems. Your software must handle the following conditions:

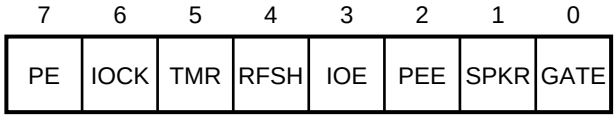
- The CPU detects an NMI on the rising edge of an NMI input signal.
- The CPU reads the status bits stored in ports 061_{16} and 0461_{16} to determine the causes of the NMI. When the CPU has read the status bits of the active sources, it can reset them. When the CPU is resetting the status bits of the active sources, a new NMI might occur from another source. The level of the new NMI remains active and the CPU does not detect this new NMI because it cannot detect the edge of the signal.
- To detect a new NMI, the CPU must disable all NMIs by setting the port 070_{16} bit <7> high and then enable all NMIs by setting the port 070_{16} bit <7> low. This causes the NMI output signal to go from low to high if there are any NMI sources pending. The CPU's NMI input logic can then detect a new NMI.

NMI Status and Control Register

The NMI status and control register is at port 061_{16}, which is a read/write port.

The following sections describe the individual bits of the NMI status and control register. The value on reset is 00x00000 (see Figure 18–1).

Figure 18–1 NMI Status and Control Register



GA_EN00480M_93A

Table 18–2 describes the format of the NMI status and control register.

Table 18–2 NMI Status and Control Register

Bit	Type	Definition
GATE	R/W	The gate signal for interval timer 1, counter 2 (speaker). When set (1), counter 2 is enabled. When cleared (0), counter 2 is disabled.
SPKR	R/W	Speaker data. This bit is ANDed with the timer 1, counter 2 OUT bit to produce the SPKR output.
PEE	R/W	Parity error enable. When set (1), the system board parity error NMI interrupt is disabled and cleared. When cleared (0), the system board parity error NMI interrupt is enabled.
IOE	R/W	IOCHK(L) NMI enable. When set (1), the IOCHK(L) signal NMI interrupt is disabled and cleared. When cleared (0), the IOCHK(L) signal NMI interrupt is enabled.
RFSH	R/W	This bit toggles with every refresh cycle. MBZ for writes.

(continued on next page)

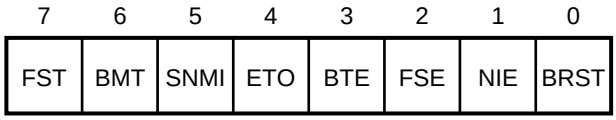
Table 18–2 (Cont.) NMI Status and Control Register

Bit	Type	Definition
TMR	R/W	State of interval timer 1, counter 2 OUT (speaker). This bit is the OUT bit of the counter and not the state of the SPKR bit. MBZ for writes.
IOCK	R/W	I/O Check. This bit set if an expansion board drives the IOCHK(L) signal active (low) on the EISA bus. An NMI is requested. This interrupt is enabled by programming bit 3 to 0. To reset this interrupt, bit 3 must be set (1), and then cleared (0). MBZ for writes.
PE	R/W	Parity error from system memory. This bit is set (1) if the system board drives the PARITY(L) signal active. An NMI is requested. This interrupt is enabled by programming bit 2 to 0. To reset the parity error, bit 2 must be set (1) and then cleared (0). MBZ for writes.

NMI Extended Status and Control Register

The following sections describe the individual bits of the NMI extended status and control register (port 0461_{16} R/W). The value on reset is 00000000 (see Figure 18–2).

Figure 18–2 NMI Extended Status and Control Register



GA_EN00481M_93A

Table 18–3 describes the format of the NMI extended status and control register.

Table 18–3 NMI Extended Status and Control Register

Bit	Type	Definition
BRST	R/W	Bus Reset. You can use this bit to perform a system bus reset without resetting other devices in the system. A system bus reset is done by setting (1) bit 0, which drives the RSTDRV(H) signal active on the EISA bus. Bit 0 must be set long enough for the system bus devices to be properly reset (8 BCLK(H) cycles), and then bit 0 must be cleared to continue normal operation.
NIE	R/W	NMI I/O port enable. When set (1), the NMI I/O port is enabled. When cleared (0), the NMI I/O port is disabled.
FSE	R/W	Fail-safe timer NMI enable. When set (1), the fail-safe timer NMI is enabled. When cleared (0), the fail-safe timer NMI is disabled.
BTE	R/W	EISA bus master timeout NMI enable. When set (1), the EISA bus master timeout NMI is enabled. When cleared (0), the EISA bus master timeout NMI is disabled.

(continued on next page)

Table 18–3 (Cont.) NMI Extended Status and Control Register

Bit	Type	Definition
ETO	R/W	An 8 μs EISA bus master timeout. This bit indicates whether an 8 μs bus timeout occurred. For example, if bit 6 = 1 and bit 4 = 0, a 32 μs bus timeout occurred. If bit 6 = 1 and bit 4 = 1, an 8 μs bus timeout occurred. MBZ for writes.
SNMI	R/W	Software generated NMI. Set (1) if an I/O write access occurred to NMI I/O port 0462_{16}. This interrupt is enabled by setting (1) bit 1 of 0461_{16}. The interrupt is reset by clearing (0) bit 1 of port 0461_{16} and then setting it. MBZ for writes.
BMT	R/W	Bus master timeout. Set (1) if either a 64 BCLK(H) or a 256 BCLK(H) (8 μs or 32 μs bus timeout) occurs. The bus timeout interrupt is enabled by setting (1) bit 3, and disabled by clearing (0) bit 3. To clear the bus timeout interrupt, clear bit 3 (0), and then set it. The 82357 chip drives RSTDRV(H) line active when a bus timeout occurs. Clearing the bus timeout status bit causes the 82357 chip to negate the RSTDRV(H) signal. MBZ for writes.
FST	R/W	Fail-safe timer timeout. Set (1) if the fail-safe timer count has expired before being reset by a software routine. This interrupt is enabled by setting bit 2. To reset this interrupt, clear bit 2, and then set bit 2. MBZ for writes.

82357 B-Stepping

Bit 4 of port 0461_{16} (see Figure 18–2) has a new definition in the 82357 b-stepping. The 82357 b-stepping can be determined at system reset (see Table 18–4).

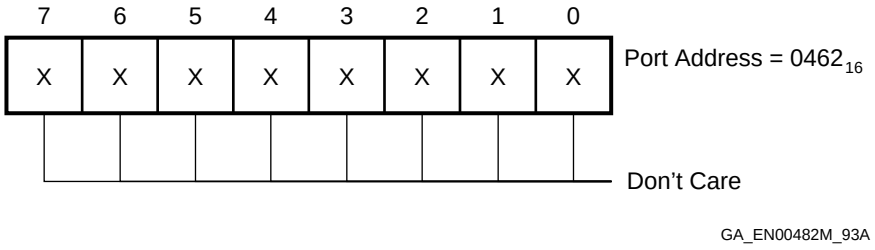
Table 18–4 82357 (ISP) Stepping

Bit	Value	Definition
4	1	A-step
4	0	B-step

Software NMI Generation

A write to this port (port 0462_{16} WO) with any data causes an NMI. This port provides a software mechanism to cause an NMI if interrupts are enabled (see Figure 18–3).

Figure 18–3 Port 0462_{16} Bit Map



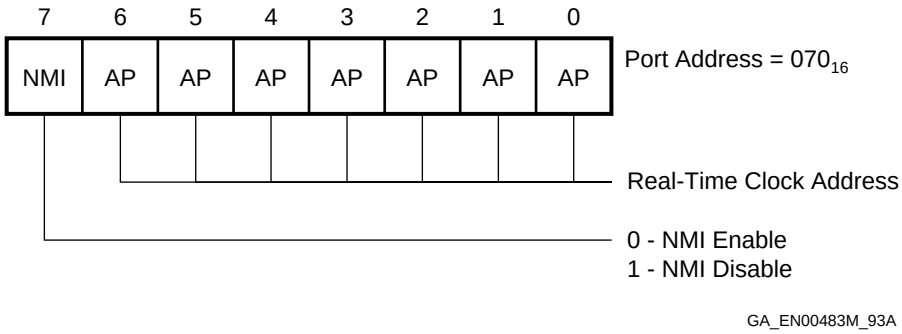
NMI Enable and Disable and Real-Time Clock Address

The mask register for the NMI interrupt is at I/O address 070_{16} (see Figure 18–4). The most-significant bit enables or disables all NMI sources including the following:

- IOCHK(L)
- Fail-safe timer
- PARITY(L)
- Bus timeout
- The NMI port

Masking the NMI signal is done by writing a value of 80_{16} to port 070_{16}. This port is shared with the real-time clock. The real-time clock uses the lower 6 bits of this port to address memory locations. Writing to port 070_{16} sets (1) both the enable and disable bit and the memory address pointer. The contents of this register must not be modified without considering the effects on the state of the other bits (see Figure 18–4).

Figure 18–4 Port 070_{16} Bit Map



19
Interval Timer

Introduction

This chapter describes the of the Intel 82357 ISP chip interval timer functions.

In This Chapter

This chapter contains the following sections:

- Interval Timer Overview
- Programming the Interval Timer
- Interval Timer Control Word Format
- Interval Timer Counter Latch Command
- Interval Timer Read Back Command

Interval Timer Overview

Interval Timer

The 82357 chip contains five counter-timers that are the same as the counter-timers in the 82C54 programmable interval timer chip. The 82357 counter-timers are addressed as though they are contained in two 82C54 counter-timers. Timer 1 contains three counters and timer 2 contains two counters. Counter 1 of timer 2 is not implemented in EISA systems. Table 19–1 shows the I/O address map of the interval timer counter-timers.

Table 19–1 Interval Timer and Counter-Timer I/O Address Map

I/O Port Address n_{16}	Register Description
040	Timer 1, system timer (counter 0)
041	Timer 1, refresh request (counter 1)
042	Timer 1, speaker tone (counter 2)
043	Timer 1, control word Register
048	Timer 2, fail-safe timer (counter 0)
049	Timer 2, reserved
04A	Timer 2, CPU speed control (counter 2)
04B	Timer 2, control word register

Timer Frequencies

Each timer provides three frequencies or counters for the system. The 8 MHz counters use the BCLK(H) signal as a clock source. The other counters divide the 14.31818 MHz OSC(H) signal input by either 12 or 48 to get their frequencies.

Timer 1 Functions

Timer 1 counters have the following functions:

- Counter 0 is connected to the interrupt controller's IRQ0 line, and provides the following functions:
  - System timer interrupt for time-of-day
  - System timer interrupt for diskette timeouts
  - Other system timer functions

- Counter 1 generates a refresh request signal
- Counter 2 generates a tone for the speaker

Timer 2 Functions

Timer 2 counters have the following functions:

- Counter 0 is the fail-safe timer that regularly generates NMIs to prevent the system locking-up.
- Counter 1 is not implemented.
- Counter 2 is connected to the SLOWH(L) output signal and is used to slow down the CPU using pulse-width modulation.

  You must program the counter to use this function. If you do not program the counter, the SLOWH(L) output signal does not become active.

  Counter 2 is placed in the one-shot mode and is triggered by the refresh request signal that is generated by timer 1 counter 1. If you have programmed counter 2, the SLOWH(L) output signal stops the CPU for the period of the one-shot every time a refresh request occurs. To enable one-shot mode, you must write a value of 92_{16} to port $4B_{16}$, which selects mode 1 in the interval timer control word register.

  Note

  Refresh cycles are not necessarily generated when the SLOWH(L) output signal is active. The bus arbiter determines when the refresh cycle is placed on the bus.

  Because the slow function depends on the refresh request frequency of another counter, chaining the refresh request frequency affects the period of the counter 2 SLOWH(L) output signal. Timer 2 counter 2 is not configured for one-shot mode and you must not program it for a counter value until you require a speed reduction in the system. The value you program depends on the system speed that you want.

Programming the Interval Timer

The counter-timers are programmed by I/O accesses and are addressed as though they are contained in two separate 82C54 interval timers. Timer 1 contains three counters. Timer 2 contains two counters (EISA systems do not implement the middle counter of timer 2).

The interval timer is an I/O mapped device. Several commands are available and are as follows:

- Control word operations—These can be used to specify the following:
  - Which counter to read or write
  - The operating mode
  - The count format (binary or BCD)
- Counter latch—Latches the current count so that the system can read it. The countdown process continues.
- Read back—Reads the count value, programmed mode, the current state of the OUT pins, and the state of the null count flag of the selected counter.

Table 19–2 lists the six operating modes for the interval counters.

Table 19–2 Interval Timer Counter Operating Modes

Mode	Function
0	Out signal on end of count (= 0)
1	Hardware retriggerable one-shot
2	Rate generator (divide by *n* counter)
3	Square wave output
4	Software triggered strobe
5	Hardware triggered strobe

Because the counter-timers are in an unknown state after power-up, multiple refresh requests can be queued up. To avoid possible multiple refresh cycles after power-up, you must

program the counter-timers immediately after power-up. Follow these steps to program the interval timer:

1. Write a control word.
2. Write an initial count for each counter-timer.
3. Load the least-significant bytes, the most-significant bytes, or both (as required by control word bits 4 and 5) of the 16-bit counter.

Interval Timer Control Word Format

The control word specifies the following:

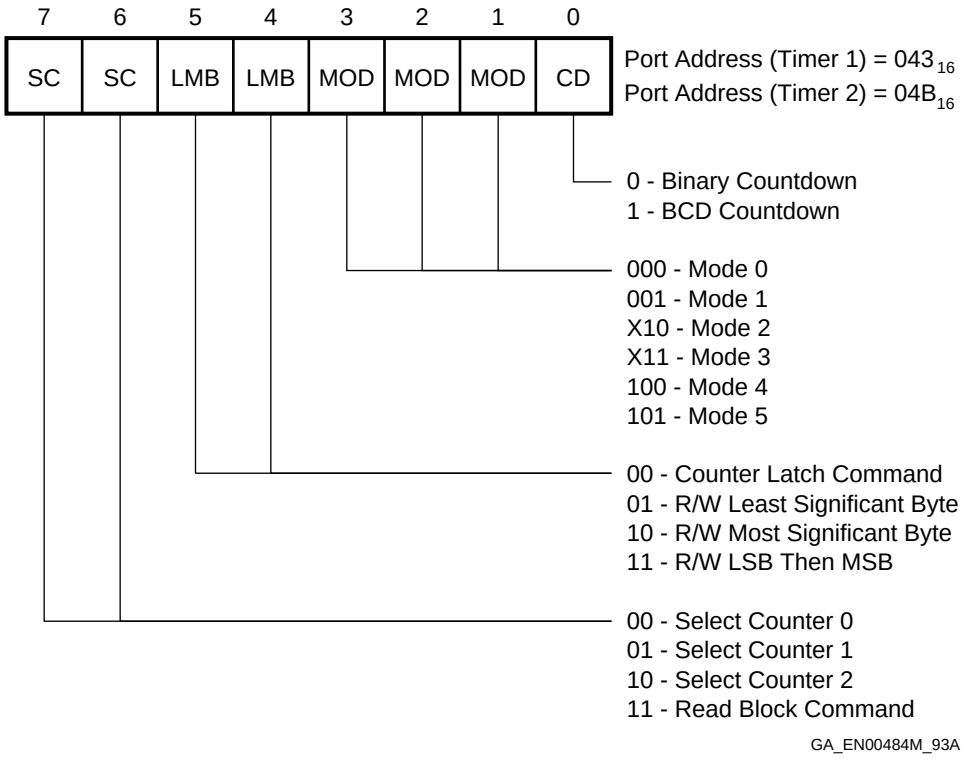
- A counter
- The operating mode
- The order and size of the count value
- Whether it counts down in a 16-bit or BCD format

After writing the control word, you can write a new count at any time. The new value takes effect according to the programmed mode.

Caution

If you program a counter-timer to read or write 2-byte counts, your program must not transfer control between writing the first and second bytes to another routine that also writes into the same counter-timer. Otherwise, the counter-timer is loaded with an incorrect count. The count must always be loaded with both bytes (see Figure 19–1).

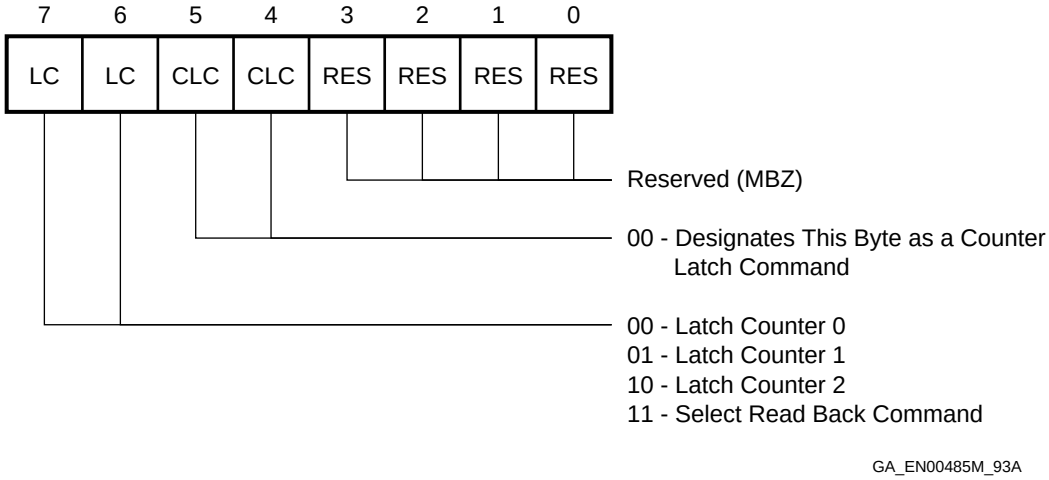
Figure 19–1 Interval Timer Control Word Format



Interval Timer Counter Latch Command

The counter latch command latches the count at the time the command is received. This command is used to ensure that the count read from the counter is accurate, particularly when reading a 2-byte count. The count value is then read from each counter's count register as programmed by the control register (see Figure 19–2).

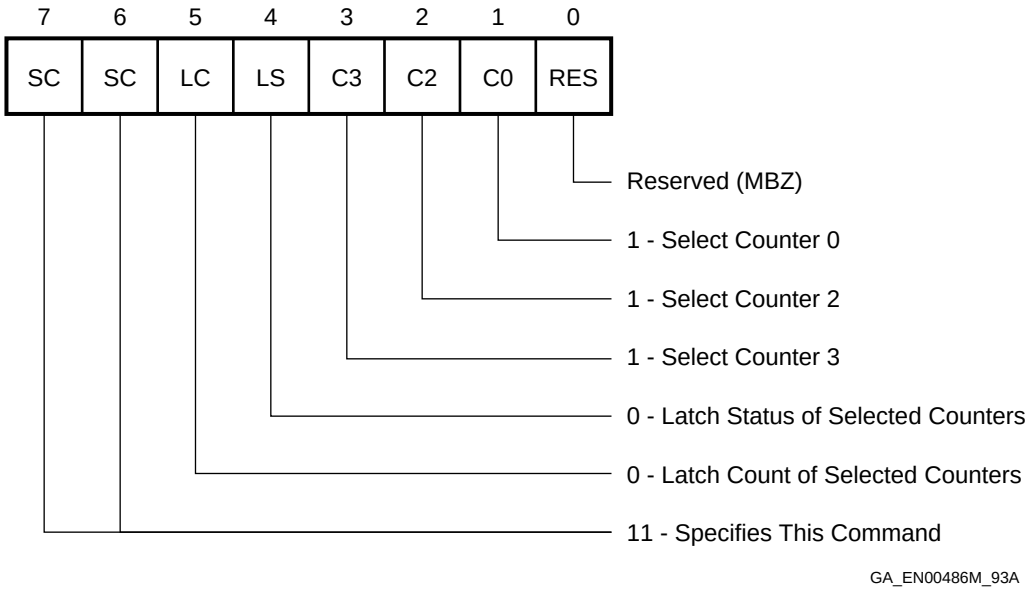
Figure 19–2 Interval Timer Counter Latch Command Format



Interval Timer Read Back Command

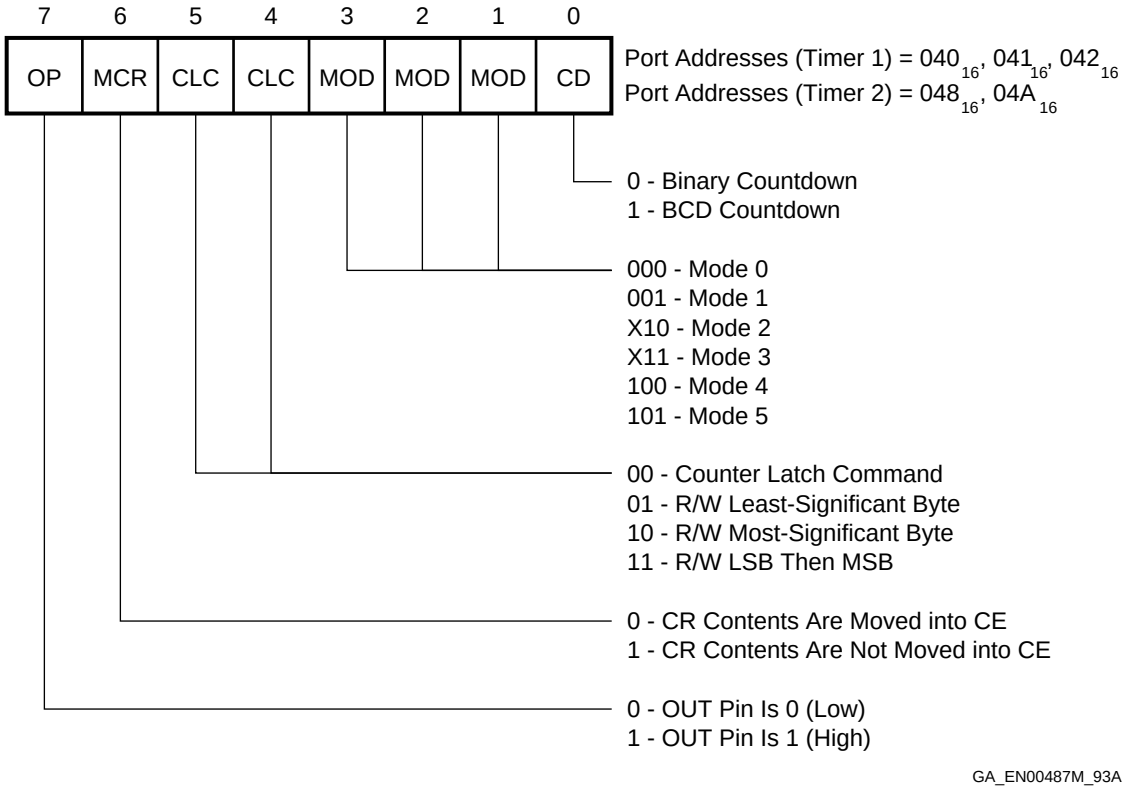
You can use the read back command to determine the count value, programmed mode, and the current states of the OUT pin and null flag of the selected counter or counters. The read back command is written to the control word register, which latches the current states of the OUT pin and null flag. You can read the value of the counter and its states by an I/O access to the counter address. Figure 19–3 and Figure 19–4 show the formats for the read back command and the status byte.

Figure 19–3 Interval Timer Read Back Command Format



Interval Timer Read Back Command

Figure 19–4 Interval Timer Status Byte Format



Part IV

VLSI Technology VL82C106 Combination Chip Functions

Part IV provides an overview of the functions of the VLSI Technology VL82C106 combination chip.

This section includes the following chapters:

- Chapter 20, Serial Communications Ports
- Chapter 21, Line Printer Port
- Chapter 22, Real-Time Clock
- Chapter 23, Keyboard Controller
- Chapter 24, Chip Select Registers

20

Serial Communications Ports

Introduction

This chapter describes the VLSI Technology VL82C106 chip serial communications ports functions and registers.

In This Chapter

This chapter contains the following sections:

Serial Communications Port Overview

The VL82C106 combination chip contains two universal asynchronous receiver transmitters (UARTs) that are based on the VL16C450B chip megacell core. Each of these UARTs share a common baud-rate clock, which is the XTAL1 input (18.432 MHz) divided by 10. The 18.432 MHz signal is shared with the keyboard controller, which divides it by 3 to get an approximately 6 MHz reference clock (see the *VLSI Technology VL16C452B Data Sheet* for the register descriptions).

Asynchronous Communications Registers

The following three types of internal registers are used for each of the two serial channels:

- Control registers
  - Bit rate select
  - Divisor latch least-significant byte (DLL)
  - Divisor latch most-significant byte (DLM)
  - Line control
  - Interrupt enable
  - Modem
- Status registers; line and modem.
- Data registers; receiver buffer and transmitter holding

The address, read, and write inputs are used with the divisor latch access bit (DLAB) to select the register to be read from or written to. The divisor latch access bit is bit 7 in the line control register (see Table 20–1).

Table 20–1 Serial Channel Internal Registers

DLAB	Address Bit 2	Address Bit 1	Address Bit 0	Mnemonic	Register
0	0	0	0	RBR	Receiver buffer register (read only)
0	0	0	0	THR	Transmitter holding register (write only)
0	0	0	1	IER	Interrupt enable register
X	0	1	0	IIR	Interrupt identification register (read only)
X	0	1	1	LCR	Line control register
X	1	0	0	MCR	Modem control register

(continued on next page)

Table 20–1 (Cont.) Serial Channel Internal Registers

DLAB	Address Bit 2	Address Bit 1	Address Bit 0	Mnemonic	Register
X	1	0	1	LSR	Line status register
X	1	1	0	MSR	Modem status register
X	1	1	1	SCR	Scratch register
1	0	0	0	DLL	Divisor latch (LSB)
1	0	0	1	DLM	Divisor latch (MSB)

The transmitter buffer register and receiver buffer register are data registers that hold from 5-8 bits of data. If less than 8 bits are transmitted, data is right justified to the LSB. Bit 0 of a data word is always the first serial data bit received and transmitted. The serial channel data registers are double buffered so that read and write operations can be performed during the parallel-to-serial or serial-to-parallel conversion.

Line Control Registers

The format of the data character is controlled by the line control register (LCR). The contents of the LCR can be read from (see Figure 20–1).

Figure 20–1 Line Control Register

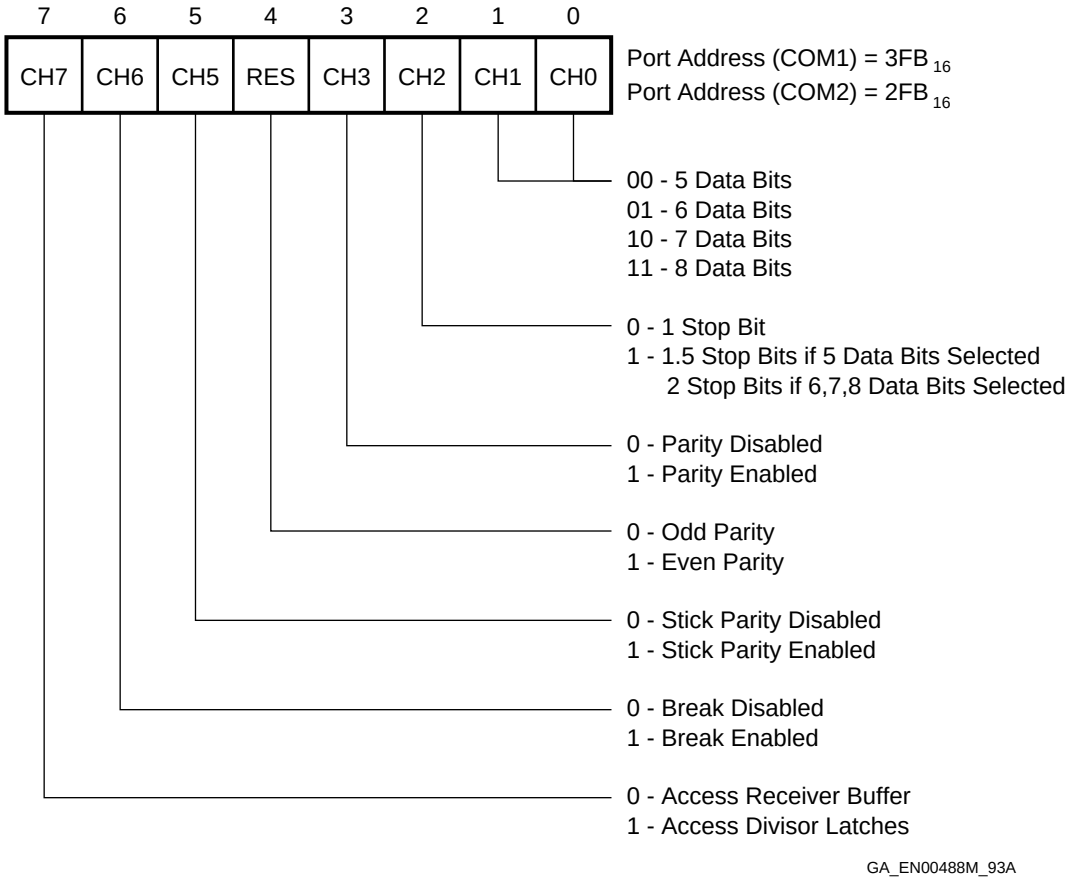


Table 20–2 describes each bit in the line control register.

Table 20–2 Line Control Register

Bit	Description
CH0 and CH1	The word length select bits select the number of bits in each serial character and are programmed (see Figure 20–1).
CH2	The stop bit select bit specifies the number of stop bits in each transmitted character. If this bit is set (1) when a 5-bit word length is selected, 1.5 stop bits are generated. If this bit is set when either a 6-, 7-, or 8-bit word length is selected, 2 stop bits are generated. The receiver always checks for 1 stop bit.
CH3	When the parity enable bit is set (1), a parity bit is generated and checked between the last data word bit and stop bit.
RES	When parity is enabled (bit 3 = 1), the even parity enable bit selects odd parity when clear (0) and even parity when set (1).
CH5 (SP)	When parity is enabled (bit 3 = 1), setting the stick parity (SP) bit causes the transmission and reception of a parity bit to be in the opposite state from the value of bit 4 (EP). This allows forced parity to a known state and the receiver to check the parity bit in a known state.
CH6 (BC)	When the break control bit is set (1), the serial output (SOUT) is forced to the spacing (logic 0) state. The break control is disabled by clearing bit 6. The break control bit acts only on SOUT and does not affect the transmitter logic. Use the following procedure to prevent invalid characters being transmitted because of the break: 1. Load pad characters (all 0s) in response to a transmitter holding register empty (THRE). 2. Set the break in response to the next THRE. 3. Wait for the transmitter to be idle (transmitter empty [TEMT] bit = 1), then clear the break when the normal transmission has to be restored.
CH7 (DLAB)	The divisor latch access bit (DLAB) must be set (1) to access the baud rate generator's divisor latches (DLL and DLM) during a read or write operation. Bit 7 must be clear (0) to access the receiver buffer, the transmit holding, or the interrupt enable registers.

Line Status Registers

The line status register (LSR) is a single register that provides status indications. The line status register is shown in Figure 20–2. Bits 1-4 are the error conditions that produce a receiver line status interrupt (priority 1 interrupt in the interrupt identification register [IIR]), when any of the conditions are detected. This interrupt is enabled by setting bit 2 of the interrupt enable register (IER). The following sections describe the bits.

Line Status Registers

Figure 20–2 Line Status Register

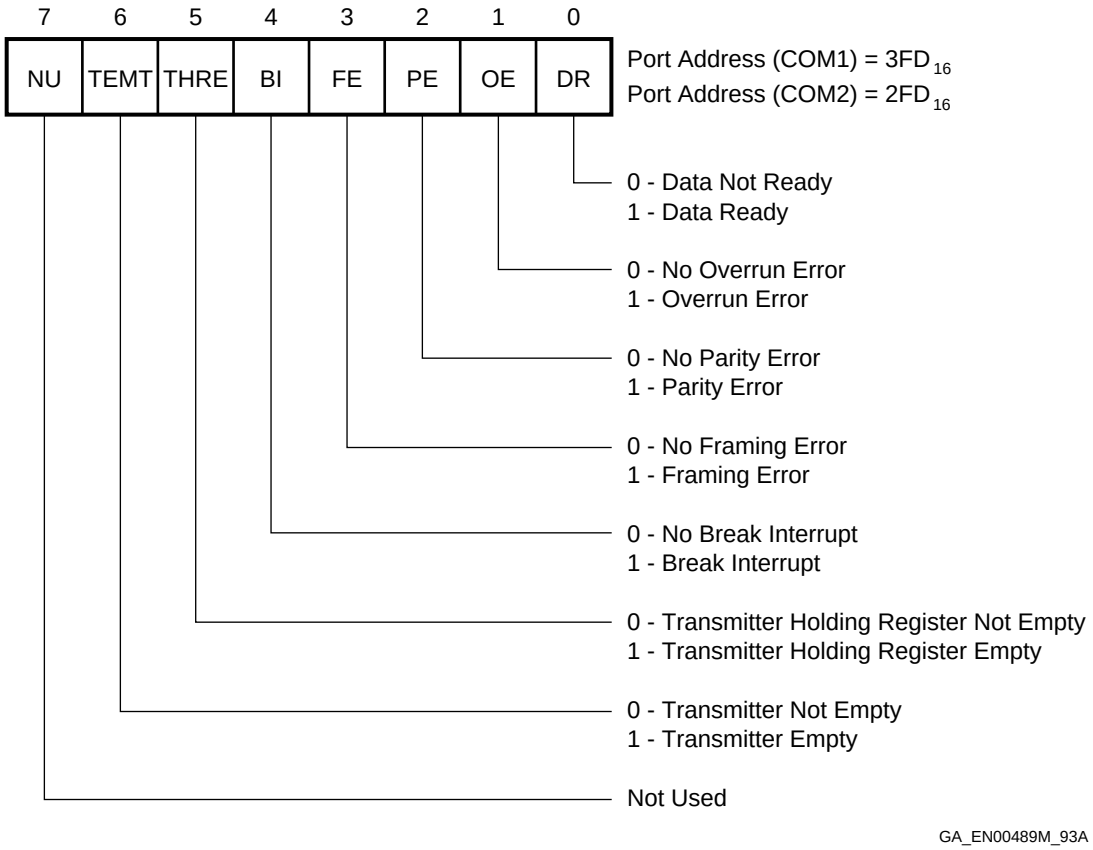


Table 20–3 describes each bit in the line status register.

Table 20–3 Line Status Register

Bit	Description
DR	The data ready (DR) bit is set (1) when an incoming character has been received and transferred into the receiver buffer register. The DR bit is cleared by a CPU read of the data in the receiver buffer register.
OE	The overrun error (OE) bit indicates that data in the receiver buffer register was not read by the CPU before the next character was transferred to the receiver buffer register, overwriting the previous character. The OE indicator is cleared when the CPU reads the contents of the line status register.
PE	The parity error (PE) bit indicates that the received data character does not have the correct even or odd parity, as selected by the even parity select bit. The PE bit is set (1) when a parity error is detected and cleared when the CPU reads the contents of the line status register (LSR).
FE	The framing error (FE) bit indicates that the received character did not have a valid stop bit. The FE bit is set (1) when the stop bit following the last data bit or parity bit is detected as a 0 (spacing level). The FE bit is cleared when the CPU reads the contents of the LSR.
BI	The break interrupt (BI) bit is set (1) when the received data is held in the spacing (logic 0) state for longer than a full word transmission time, which is the time taken to transmit the following sequence of bits: • Start bit • Data bits • Parity • Stop bits The BI indicator is cleared when the CPU reads the contents of the line control register.

(continued on next page)

Table 20–3 (Cont.) Line Status Register

Bit	Description
THRE	The transmitter holding register empty (THRE) bit indicates that the serial channel is ready to accept a new character for transmission. The THRE bit is set (1) when a character is transferred from the transmitter register into the transmitter shift register. The THRE bit is cleared when the CPU loads the transmitter holding register. The THRE bit is not cleared by a CPU read of the LSR. When the THRE interrupt is enabled (bit 1 of the IER is set), THRE causes a priority 3 interrupt in the IIR. If THRE is the interrupt source indicated in the IIR, INTRPT(H)is cleared by a read of the IIR.
TEMT	The transmitter empty (TEMT) bit is set (1) when the transmitter holding register (THR) and the transmitter shift register (TSR) are both empty. The TEMT bit is cleared when a character is loaded into the THR and remains cleared until the character is transferred out of SOUT. The TEMT bit is not cleared by a CPU read from the LSR.

Modem Control Registers

The modem control register (MCR) controls the interface with the modem or data set (see Figure 20–3). The MCR can be written to or read from. The RTS(L) and DTR(L) signal outputs are directly controlled by their control bits in this register. A high input asserts a low (true) at the output pins. The MCR register bits 0, 1, 3, and 4 are described in the following sections.

Figure 20–3 Modem Control Register

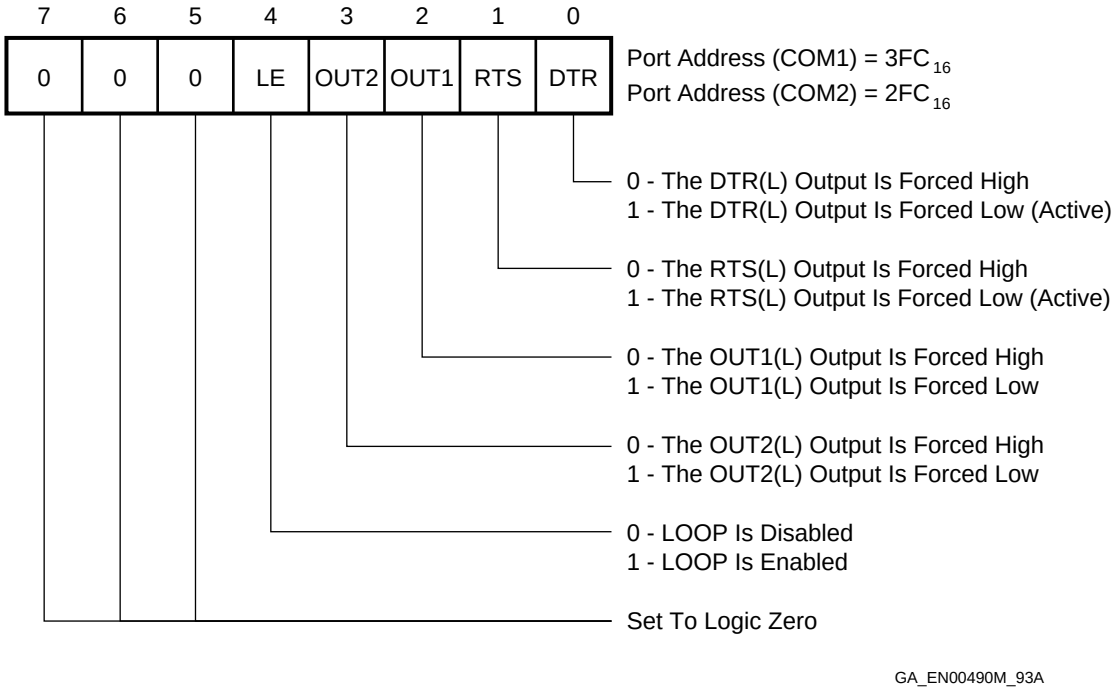


Table 20–4 describes each bit in the modem control register.

Table 20–4 Modem Control Register

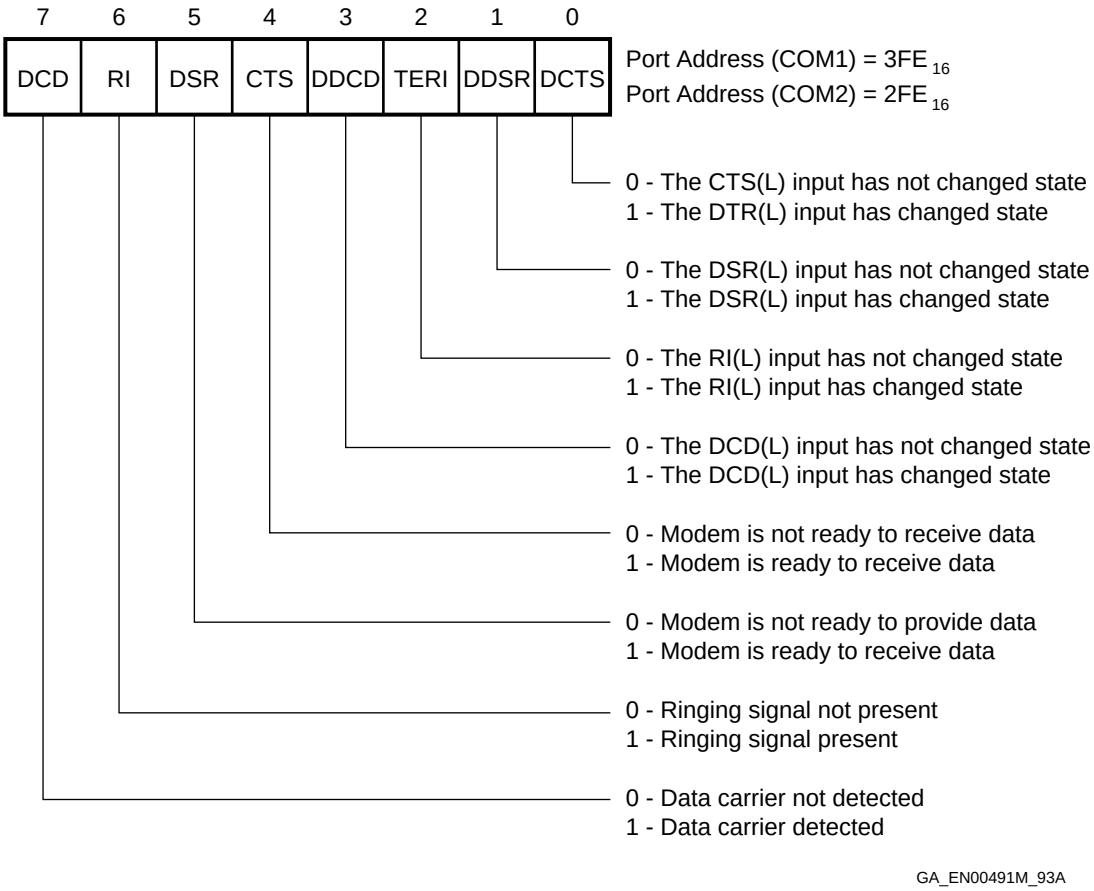
Bit	Description
DTR	When the data terminal ready (DTR) bit is set (1), the DTR(L) signal output is forced low. When the DTR bit is cleared (0), the DTR(L) signal output is forced high.
RTS	When the request to send (RTS) bit is set (1), the RTS(L) signal output is forced low. When the RTS bit is cleared, the RTS(L) signal output is forced high.
OUT1	When OUT1 is set (1), the OUT1(L) signal output is forced low.
OUT2	When the OUT2 bit is set (1), the OUT2 signal output is forced low.
LE	The loop enable (LE) bit provides a local loopback feature for diagnostic testing of the channel. When LE is set (1), serial output (SOUT) is set to the marking (logic 1) state, and the receiver data input, serial input (SI) is disconnected. The output of the transmitter shift register is looped back into the receiver shift register input. The 4 modem-control input signals (CTS(L), DSR(L), DCD(L) (RLSD), and RI(L)) are disconnected. The modem control output signals (DTR(L), RTS(L), OUT1(L), and OUT2(L)) are internally connected to the four modem control inputs. The modem control output pins are forced to their inactive state (high) on the VL82C106 chip. In diagnostic mode, data being transmitted is immediately received. This enables the DECchip 21064 CPU to verify the transmit and receive data paths of the selected serial channel.

Modem Status Registers

The modem status register (MSR) provides the CPU with the status of the modem input lines from the modem or peripheral devices. The MSR enables the CPU to read the serial channel modem signal inputs by accessing the data bus interface. In addition to the current status information, 4 bits of the MSR indicate whether the modem inputs have changed since the last reading of the MSR. The delta status bits are set (1) when a control input from the modem changes state. They are cleared when the CPU reads the MSR.

The modem input lines are CTS(L), DSR(L), RI(L), and DCD(L) (RLSD). Bits 4 to 7 of the MSR are the status indications of these lines. A set (1) status bit indicates that the input is low, while a cleared status bit indicates that the input is high. If the modem status interrupt in the interrupt enable register is enabled (bit 3 of the IER is set), an interrupt is generated when bits 0 to 3 of the MSR are set (1). The MSR is a priority 4 interrupt. The contents of the modem status register are described in Figure 20–4.

Figure 20–4 Modem Status Register



Reading the MSR register clears the delta modem status indications, but has no effect on the other status bits.

For LSR and MSR, the setting of status bits is prohibited during status register read operations. If a status condition is generated during a read of the DISTR signal, the status bit is not set until the trailing edge of the read.

If a status bit is set during a read operation and the same status condition occurs, that status bit is cleared at the trailing edge of the read instead of being set again.

Table 20–5 describes each bit in the modem status register.

Table 20–5 Modem Status Register

Bit	Description
DCTS	The data clear to send (DCTS) bit indicates that the CTS signal input to the serial channel has changed state since it was last read by the CPU.
DDSR	The delta data set ready (DDSR) bit indicates that the DSR signal input to the serial channel has changed state since it was last read by the CPU.
TERI	The trailing edge of ring indicator (TERI) bit indicates that the RI signal input to the serial channel has changed state since it was last read by the CPU. High to low transitions on the RI line do not activate the TERI bit.
DDCD	The delta data carrier detect (DDCD) bit indicates that the DCD (RLSD) signal input to the serial channel has changed state since it was last read by the CPU.
CTS	The clear to send bit is the compliment of the CTS(L) signal input from the modem, indicating to the serial channel that the modem is ready to receive data from the serial channel's transmitter output signal SOUT. If the serial channel is in loop mode (bit 4 of the MCR is set), the CTS bit reflects the value of RTS in the MCR.
DSR	The data set ready (DSR) bit is the compliment of the DSR(L) signal input from the modem, indicating to the serial channel that the modem is ready to provide received data to the serial channel's receiver circuitry. If the serial channel is in loop mode (bit 4 of the MCR is set), the DSR bit reflects the value of DTR in the MCR.
RI	The ring indicator (RI) bit is the compliment of the RI(L) signal input. If the serial channel is in loop mode (bit 4 of the MCR is set), the RI bit reflects the value of OUT1(L) signal in the MCR.
DCD	The data carrier detect (DCD) and receive line signal detect (RLSD) bit indicates the status of the data carrier detect and receive line signal detect (DCD(L) and RLCD) signal input. If the serial channel is in loop mode (bit 4 of the MCR is set), the DCD and RLCD bit reflects the value of OUT2(L) signal in the MCR.

Divisor Latches

Each serial channel contains a programmable baud rate generator (BRG) that divides the clock (DC to 3.1 MHz) by any divisor from 1 to 2^{16-1}. The output frequency of the baud rate generator is 16 times the data rate $[divisornumber = clock + (baudrate * 16)]$. Two 8-bit divisor latch registers must be loaded during initialization. When loading either of the divisor latches, a 16-bit baud counter is immediately loaded. This prevents long counts on an initial load.

Receive Buffer Registers

The receiver circuitry in the serial channels is programmable for 5, 6, 7, or 8 data bits per character. For words of less than 8 bits, the data is right justified to the least significant bit (LSB = data bit 0). Data bit 0 of a data word is the first data bit received. The unused bits in a character that is less than 8 bits long are 0s.

Received data at the SIN input pin is shifted into the receiver shift register by the 16X clock provided at the CLK signal input. This clock is synchronized to the incoming data on the position of the start bit. When a complete character is shifted into the receiver shift register, the assembled data bits are parallel-loaded into the receiver buffer register and the DR flag in the LSR is set.

Double buffering of the received data permits continuous reception of data without losing received data. While the receiver shift register is shifting a new character into the serial channel, the receiver buffer register is holding a previously received character for the CPU to read. Failure to read the character in the receive buffer register before complete reception of the next character results in the loss of the character in the receiver buffer register. The OE flag in the LSR is set, indicating an overrun condition.

Transmitter Holding Registers and Scratchpad Registers

Transmitter Holding Registers

The transmitter holding register (THR) holds character data until the transmitter shift register is empty and ready to accept a new character. The transmitter and receiver word length are the same. If the character is less than 8 bits, unused bits are ignored by the transmitter.

Data bit 0 of the THR is the first serial data bit transmitted. The THRE flag (bit 5 of the LSR) reflects the status of the THR. The TEMT flag (bit 6 of the LSR) indicates whether both the THR and TSR are empty.

Scratchpad Registers

The scratchpad register is an 8-bit read/write register that has no effect on either channel. Use it to hold data temporarily.

Interrupt Identification Registers

To minimize software overhead during data character transfers, the serial channels prioritize interrupts into four levels. The four levels of interrupt conditions are as follows:

- Receiver line status (priority 1)
- Received data ready (priority 2)
- Transmitter holding register empty (priority 3)
- Modem status (priority 4)

The interrupt identification register (IIR) stores two pieces of information that specify the following :

- That a prioritized interrupt is pending
- The type of interrupt pending

The IIR register indicates the highest priority interrupt pending. When bit 0 of the IIR is clear (0), an interrupt is pending. Bits 1 and 2 are used to identify the highest priority interrupt pending (see Table 20–6).

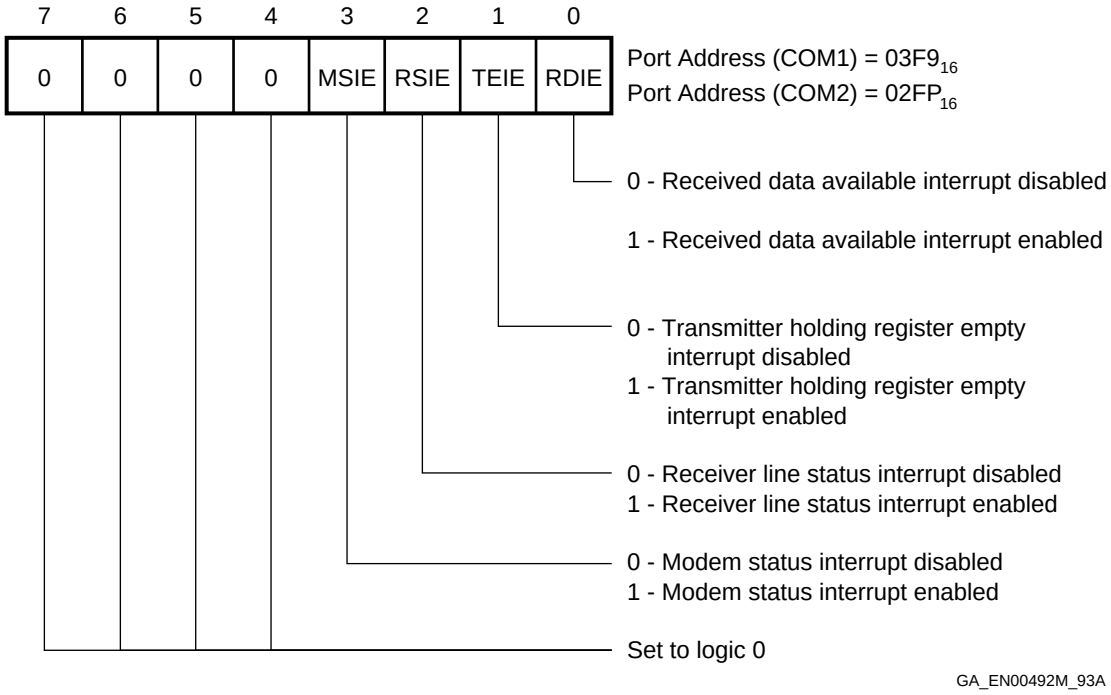
Table 20–6 Serial Channel Internal Identification Registers

Interrupt Identification				Interrupt Set and Reset Functions		
Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Flag	Interrupt Source	Interrupt Reset Control
X	X	1		None	None	
1	1	0	First	Receiver line status	OE, PE, FE, or BI	LSR Read
1	0	0	Second	Received data available	Received data available	RBR Read
0	1	0	Third	THRE	THRE	IIR read if THRE is the interrupt source or THR write
0	0	0	Fourth	Modem status	CTS, DSR, RI, DCD (RLSD)	MSR read

Interrupt Enable Registers

The interrupt enable register (IER) is used to independently enable the four serial channel interrupt sources that activate the interrupt INTRPT(H) output. All interrupts are disabled by clearing bits 1 to 3 of the IER. Interrupts are enabled by setting the appropriate bits of the IER. Disabling the interrupt system inhibits the interrupt identification register and the active INTRPT(H) output. All other system functions operate normally, including the setting of the line status and modem status registers. The contents of the interrupt enable register are listed in Figure 20–5.

Figure 20–5 Interrupt Enable Register



Serial Transmission Process

The serial transmitter consists of the following:

- A transmitter holding register (THR)
- Transmitter shifting register (TSR)
- Associated control logic

The serial transmission process is as follows:

- The transmitter holding register empty (THRE) bit and the transmitter empty (TEMT) bit are two bits in the line status register that indicate the status of the THR and TSR registers.
- The CPU must perform a write operation to the THR only if the THRE bit is set (1). This causes the THRE bit to be cleared (0).
- The THRE bit is set when the word is automatically transferred from the THR register to the TSR register during the transmission of the start bit.
- The TEMT bit remains cleared during the transmission of the data word, because the data word cannot be transmitted from the THR to the TSR until the TSR has completed sending the word.

Serial Reception Process

The serial reception process is as follows:

- Serial asynchronous data is entered into the SIN(?) pin.
- The serial channel continually searches for a high to low transition from the idle state.
- When a transition is detected, a counter is reset and counts the 16X clock to 7.5, which is the center of the start bit.
- The start bit is valid if the SIN signal is still low.

  Verifying the start bit prevents the receiver from assembling a false data character because of a low-going noise spike on the SIN signal input.
- The line control register (bits 0 and 1) determines the following:
  - The number of data bits in a character.
  - If parity is used (by bit 3)
  - The polarity of parity (by bit 4).
- Status for the receiver is provided in the line status register when a full character is received, including parity and stop bits, by the data ready (DR) bit being set (1).
- The CPU reads the receiver buffer register, which clears the data ready bit.
- If the character is not read before a new character transfer from the RSR to the RBR, the overrun error status bit (OE) is set (1).
- If there is a parity error, the parity error bit (PE) is set (1).
- If a stop bit is not detected, the framing error bit (FE) is set (1).
- If the data arriving on the SIN line is a symmetrical square wave, the center of the data cells occur within +/- 3.125% of the actual center, providing an error margin of 46.875%. The start bit can begin as much as one 16X clock cycle before being detected.

Baud Rate Generator

The baud rate generator (BRG) generates the clocking for the UART function, providing standard ANSI and CCITT bit rates.

The data rate is determined by the divisor latch registers and the external frequency. The bit rate is selected by programming the following two divisor latches:

- The divisor latch most significant byte (DLM)
- The divisor latch least significant byte (DLL)

Setting DLL = 1 and DLM = 0 selects the divisor to divide by 1, giving the maximum baud rate for a given input frequency at the CLK signal input.

The BRG can use any of the following three frequencies to provide standard baud rates:

- 1.8432 MHz
- 2.4576 MHz
- 3.072 MHz

With these frequencies, standard bit rates from 50 to 38.5K bits per second (bits/s) are available. The divisors that are required to obtain standard rates using these three input frequencies are listed in the following tables.

Table 20–7 Serial Channel Baud Rates (1.8432 MHz Clock)

Required Baud Rate	Divisor Used	Percent Error Difference Between Required and Actual
50	2304	
75	1536	
110	1047	0.026
134.5	857	0.058
150	768	

(continued on next page)

Table 20–7 (Cont.) Serial Channel Baud Rates (1.8432 MHz Clock)

Required Baud Rate	Divisor Used	Percent Error Difference Between Required and Actual
300	384	
600	192	
1200	96	
1800	64	
2000	58	0.69
2400	48	
3600	32	
4800	24	
7200	16	
9600	12	
19200	6	
38400	3	
56000	2	2.68

Table 20–8 Serial Channel Baud Rates (2.4576 MHz Clock)

Required Baud Rate	Divisor Used	Percent Error Difference Between Required and Actual
50	3072	
75	2048	
110	1396	0.026
134.5	1142	0.0007
150	1024	
300	512	
600	256	
1200	128	

(continued on next page)

Table 20–8 (Cont.) Serial Channel Baud Rates (2.4576 MHz Clock)

Required Baud Rate	Divisor Used	Percent Error Difference Between Required and Actual
1800	85	0.392
2000	77	0.260
2400	64	
3600	43	0.775
4800	32	
7200	21	1.587
9600	16	
19200	8	
38400	4	

Table 20–9 Serial Channel Baud Rates (3.072 MHz Clock)

Required Baud Rate	Divisor Used	Percent Error Difference Between Required and Actual
50	3840	
75	2560	
110	1745	0.026
134.5	1428	0.034
150	1280	
300	640	
600	320	
1200	160	
1800	107	0.312
2000	96	
2400	80	
3600	53	0.628

(continued on next page)

Table 20–9 (Cont.) Serial Channel Baud Rates (3.072 MHz Clock)

Required Baud Rate	Divisor Used	Percent Error Difference Between Required and Actual
4800	40	
7200	27	1.23
9600	20	
19200	10	
38400	5	

Master Reset

After power-up, the master reset (MR(?)) signal input must be held high for 1 microsecond to reset the serial communications circuits to an idle mode until initialization. A high on the MR line does the following:

- Initializes the transmitter and receiver internal clock counters.
- Clears the line status register (LSR), except for the transmitter shift register empty (TEMT) and transmitter holding register empty (THRE) bits, which are set. The modem control register (MCR) is also cleared.

  All the discrete lines, memory elements and miscellaneous logic associated with these register bits are also cleared or turned off. The following are not affected:

  - The line control register (LCR)
  - The divisor latches
  - The receiver buffer register
  - The transmitter register

Following the removal of the reset condition, the serial channels remain in the idle mode until they are programmed.

A hardware reset of the serial channels sets the THRE and TEMT status bits in the line status register. When interrupts are subsequently enabled, an interrupt occurs because of THRE. A summary of the effects of a hardware reset on the serial channels is shown in Table 20–10.

Table 20–10 Effects of a Master Reset on the Serial Channels

Register or Signal	Reset Control	Reset
Interrupt enable register	Reset	All bits low (0-3 forced and 4-7 permanent)
Interrupt identification register	Reset	Bit 0 is high, and bits 1 and 2 are low
Line control register	Reset	All bits low
Modem control register	Reset	All bits low
Line status register	Reset	All bits low
Line control register	Reset	All bits low, except bits 5 and 6, which are high
Modem status register	Reset	Bits 0-3 low, bits 4-7 input signal
SOUT	Reset	High
Interrupt (RCVR errors)	Read LSR/Reset	Low
Interrupt (RCVR Data Ready)	Read RBR/Reset	Low
Interrupt (THRE)	Read IIR/Write THR/Reset	Low
Interrupt (modem status changes)	Read MSR/Reset	Low
OUT2	Reset	High
RTS	Reset	High
DTR	Reset	High
OUT1	Reset	High

Programming the Serial Channels

You can program the serial channels using the following registers:

- Line control register (LCR)
- Interrupt enable register (IER)
- Divisor latch least significant byte (DLL)
- Divisor latch most significant byte (DLM)
- Modem control register (MCR)

These control words define the character length, number of stop bits, parity, baud rate, and modem status.

You can write to the control registers in any order, but you must write to the IER last because it controls the interrupt enable. When the serial channels are programmed and operational, these registers can be updated any time the serial channels are not transmitting or receiving data.

Software Reset of the Serial Channels

A software reset of the serial channels is a useful method for returning to a known state without a system reset. To issue a software reset, you must write to the following:

- The line control register (LCR)
- The divisor latches
- The modem control register (MCR)

The LSR and RBR registers must be read before enabling interrupts to clear any residual data or status bits that might be invalid for subsequent operations.

21

Line Printer Port

Introduction

This chapter describes the VLSI Technology VL82C106 chip line printer port functions and registers .

In This Chapter

This chapter contains the following sections:

- Line printer Port Overview
- Line Printer Port Data Register (Register 0)
- Line Printer Port Status Register (Register 1)
- Line Printer Port Control Register (Register 2)

Line printer Port Overview

The line printer port (LPT) contains the same functions as the port included in the VL16C452B chip, but offers a software programmable extended mode, which includes a direction control bit and an interrupt status bit.

These features are disabled during power-up, but you can turn them on by clearing the EMODE bit of control register 0 (RTC register 69_{16} or I/O Port 102_{16}). When the EMODE bit is set (1), the port works the same as a PC/AT compatible printer port.

The line printer port is accessed by an internally generated programmable chip select (CS3).

Line Printer Port Data Register (Register 0)

The line printer port data register, located at port $3BC_{16}$, is either unidirectional or bidirectional, depending on the state of the extended mode and data direction control bits (see Table 21–1 for the line printer port data register bit definitions).

Compatibility Mode

In compatibility mode, the EMODE bit is set to 1. Read operations to this register return the last data that was written to the LPT port. Write operations immediately output data to the LPT port.

Extended Mode

In extended mode, the EMODE bit is set to 0. Read operations return either the data last written to the LPT data register if the direction bit is set to output (0) or the data that is present on the pins of the LPT port if the direction bit is set to input (1). Write operations latch data into the output register, but only drive the LPT port when the direction bit is set to output (0).

Table 21–1 Line Printer Port Data Register (Register 0)

Bit	Name	Description
0	PD0	Printer data bit 0
1	PD1	Printer data bit 1
2	PD2	Printer data bit 2
3	PD3	Printer data bit 3
4	PD4	Printer data bit 4
5	PD5	Printer data bit 5
6	PD6	Printer data bit 6
7	PD7	Printer data bit 7

Line Printer Port Status Register (Register 1)

The LPT status register is a read-only register, located at port $3BD_{16}$. It contains the interrupt status and the real-time status information of the LPT connector pins (see Table 21–2 for the line printer port status register bit definitions).

Table 21–2 Line Printer Port Status Register (Register 1)

Bit	Name	Description
0	Reserved	Read as 1
1	Reserved	Read as 1
2	IRQ	Interrupt request status bit. This bit is enabled or disabled by bit 4 of the printer control register. When enabled, it is latched low when the ACK signal is deasserted, indicating that the printer has acknowledged the previous transfer. The IRQ status bit is cleared to a high level when the LPT port status register is read. When in PC /AT-compatible mode (bit 1 of RTC register $6A_{16}$ is set to 1), the IRQP signal output follows the ACK signal input, if enabled. The IRQP signal is set during the inactive transition of the ACK signal input, if enabled, and cleared following a read of the LPT status register.
3	ERROR	Error status bit. A 0 indicates that a printer error occurred. A 1 indicates normal operation. This bit follows the state of the ERR pin.

(continued on next page)

Table 21–2 (Cont.) Line Printer Port Status Register (Register 1)

Bit	Name	Description
4	SLCT	Select status bit. Indicates the current status of the SLCT signal from the printer. A 0 indicates that the printer is not selected (off-line). A 1 indicates that the printer is selected (on-line).
5	PE	Paper empty status bit. A 0 indicates normal operation. A 1 indicates that the printer is out of paper. This bit follows the state of the PE pin.
6	ACK	Acknowledge status bit. A 0 indicates that the printer has received a character and is ready to accept another. A 1 indicates that the last operation to the printer has not completed. This bit follows the state of the ACK pin.
7	BUSY	Busy status bit. A 0 indicates that the printer is busy and cannot receive data. A 1 indicates that the printer is ready to accept data. This bit is the inverse of the BUSY pin.

Line Printer Port Control Register (Register 2)

The line printer port control register is a read/write port, located at port $3BE_{16}$. It is used to control the LPT direction and the printer control lines driven from the port. Write operations set or clear these bits, while read operations return the status of the last write operation to this register (see Table 21–3 for the line printer port control register bit definitions).

Table 21–3 Line Printer Port Control Register (Register 2)

Bit	Name	Description
0	STB	Printer strobe control bit. When set (1), the STB signal is asserted on the LPT interface, causing the printer to latch the current data. When reset (0), the signal is negated.
1	AFD	Autofeed control bit. When set (1), the AFD signal is asserted on the LPT interface, causing the printer to automatically generate a line feed at the end of each line. When reset (0), the signal is negated.
2	INIT	Initialize printer control bit. When set (1), the INIT signal is negated (high). When reset (0), the INIT signal is asserted to the printer, forcing a reset.
3	SLIN	Select input control bit. When set (1), the SLIN signal is asserted, causing the printer to go online. When reset (0), the signal is negated.
4	IRQ EN	Interrupt request enable control bit. When set (1), enables interrupts from the LPT port when the ACK signal is asserted by the printer. When reset (0), interrupts are disabled.

(continued on next page)

Table 21–3 (Cont.) Line Printer Port Control Register (Register 2)

Bit	Name	Description
5	DIR	Direction control bit. When set (1) and the EMODE bit = 0, the output buffers in the LPT port are disabled, enabling data from external sources to be read from the LPT port. When reset (0), the output buffers are enabled, forcing the LPT buffers to drive the LPT pins. The power-up reset value of this bit is cleared (0). When the EMODE bit = 1, this write-only bit does not have any effect and must be read as 1.
6	Reserved	Read as a 1.
7	Reserved	Read as a 1.

22

Real-Time Clock

Introduction

This chapter describes the VLSI Technology VL82C106 chip real-time clock (RTC) functions and registers.

In This Chapter

It contains the following sections:

- RTC Overview
- RTC Programmer's Model
- Time of Day Registers
- RTC Control Registers
- RTC Control Register A
- RTC Control Register B
- RTC Control Register C
- RTC Control Register D
- General RTC Notes

RTC Overview

The VL82C106 chip real-time clock (RTC) is the equivalent of the Motorola MC146818A RTC. It is also compatible with the Dallas Semiconductor DS1287A RTC. The RTC functions include the following:

- Time of day clock
- Alarm function
- 100-year calendar function
- Programmable periodic interrupt output
- Programmable square wave output
- 50 bytes of user RAM
- User RAM preset feature

Note

The RTC address and data ports are at port 170_{16}, because the RTCMAP pin (121) has been connected to ground. Therefore, the RTC is accessed by the internally decoded port 170_{16} RTC register address 0 and port 171_{16} (RTC data read/write).

RTC Programmer's Model

The RTC memory consists of 10 RAM bytes, which contain the following:

- Time
- Calendar
- Alarm data
- Four control and status bytes
- 50 general purpose RAM bytes

To read the RTC, you must write an index to I/O address 170_{16} and then read from I/O address 171_{16}. To write to the RTC, you must first write an index to I/O address 170_{16} and then write to I/O address 171_{16}. The address indexes of the real-time clock are shown in Table 22–1.

Table 22–1 Real-Time Clock Address (Index) Map

Address (Index) n_{16}	Function	Range
00	Time registers	0-99
0A	RTC register A	(R/W)
0B	RTC register B	(R/W)
0C	RTC register C	(RO)
0D	RTC register D	(RO)
0E-3F	User RAM (standby)	(R/W)
40-4F	Additional standby RAM	(R/W)
50-68	Reserved—no RAM	
69-7F	Chip select control registers	(W)

All 64 bytes are directly readable and writable by the process program, except for the following:

- Registers C and D are read-only.
- Bit 7 of register A is read-only.

The RTC address map also includes the following:

- Additional standby RAM
- Control registers for combination chip configuration
- Chip select control

The RAM and chip select control registers are powered by the VBAT power supply for battery-backed operation.

The processor program obtains time and calendar information by reading the appropriate locations. The program can initialize the time, calendar, and alarm by writing to these RAM locations. The contents of the time, calendar, and alarm bytes can be either binary or binary-coded decimal (BCD).

Time of Day Registers

The contents of the time of day registers can be either in binary or BCD format. The address map of these registers is shown in Table 22–2.

Table 22–2 Time of Day Registers Address Map

Address (Index)	Function	Range BCD	Range Binary
0	Seconds (time)		0-59 in BCD mode
1	Seconds (alarm)		0-59 in BCD mode
2	Minutes (time)		0-59 in BCD mode
3	Seconds (alarm)		0-59 in BCD mode
4	Hours (time)		1-12 in BCD mode (AM) 81-92 in BCD mode (PM)
5	Hours (alarm)		1-12 in BCD mode (AM) 81-92 in BCD mode (PM)
6	Day of week		1-7 in BCD mode
7	Date of month		1-31 in BCD mode
8	Month		1-12 in BCD mode
9	Year		1-99 in BCD mode

RTC Control Registers

The RTC has four registers that are accessible to the processor program. The four registers are also fully accessible during the update cycle and are listed in Table 22–3.

Table 22–3 Real-Time Clock Control Registers

Address (Index) n_{16}	Function	Access
0A	RTC register A	R/W
0B	RTC register B	R/W
0C	RTC register C	RO
0D	RTC register D	RO
0E-3F	User RAM (standby)	R/W

RTC Control Register A

This register contains control bits for the selection of periodic interrupt, input divisor, and the update in progress status bit. The individual bits are listed in Table 22–4.

Table 22–4 Bit Definitions of Real-Time Clock Control Register A

Bit	Description	Abbreviation
0	Rate select bit 0	RS0
1	Rate select bit 1	RS1
2	Rate select bit 2	RS2
3	Rate select bit 3	RS3
4	Divisor bit 0	DV0
5	Divisor bit 1	DV1
6	Divisor bit 2	DV2
7	Update in progress	UIP

Rate-Selection Bits

Bits 0-3, the four rate-selection bits (RS0-RS3), select one of 15 taps on the 22-stage divider or disable the divider output. You can use the tap selected to generate a periodic interrupt. These 4 bits are read/write bits and are not affected by a reset. The periodic interrupt rates that result from the selection of various tap values are listed in Table 22–5.

Table 22–5 Periodic Interrupt Rates

RS Value n_{16}	Periodic Interrupt Rate	Unit
0	None	
1	3.90625	ms†
2	7.8125	ms
3	122.070	μs‡
4	244.141	μs
5	488.281	μs
6	976.562	μs
7	1.953125	ms
8	3.90625	ms
9	7.8125	ms
0A	15.625	ms
0B	31.25	ms
0C	62.5	ms
0D	125	ms
0E	250	ms
0F	500	ms

†ms = milliseconds
‡ μs = microseconds

Note

The 976.562 μs periodic interrupt rate meets the Alpha AXP architectural requirement. Therefore, bits RS<3:0> of the RTC control register A must be set to 6 to generate the 976.562 μs periodic interrupt. The periodic interrupt is connected to a dedicated pin on the DECchip 21064 CPU, so that PAL code can always take the interrupt, as required by the Alpha AXP architecture.

Divisor-Selection Bits

Bits 4-6 are the three divisor-selection bits (DV0 to DV2). These bits are fixed to provide for only a five-state divider chain, which is used with a 32-kHz external crystal. You can change only bit 6 of this register, allowing control of the reset for the divisor chain. When the divider reset is removed, the first update cycle begins 0.5 seconds later. These bits are not affected by a power-up reset (external pin). The divider conditions associated with a divisor value are listed in Table 22–6.

Table 22–6 Divider Conditions

Divisor Value	Divider Condition
2	Operation mode, divider running
6	Reset mode, divider in reset state

Update in Progress Bit

The update in progress (UIP) bit is a status flag that can be monitored by a program. When the UIP bit is set (1), the update cycle is in progress or will soon begin. When the UIP bit is clear (0), the update cycle is not in progress and will not be for at least 244 μs. The time, calendar, and alarm information in RAM is available to the program when the UIP bit is clear (0). The UIP is a read-only bit. It is not affected by a reset. Setting the SET bit to 1 in register B inhibits any update cycle and then clears the UIP status bit.

RTC Control Register B

RTC control register B contains command bits to control various modes of operation and interrupt enables for the RTC. The bits in register B are listed in Table 22–7.

Table 22–7 Real-Time Clock Control Register B Bit Definitions

Bit	Description	Abbreviation
0	Daylight savings enable	DSE
1	24/12 mode	24/12
2	Data mode (binary or BCD)	DM
3	Not used	
4	Update end interrupt enable	UIE
5	Alarm interrupt enable	AIE
6	Periodic interrupt enable	PIE
7	Set command	SET

Daylight Savings Enable Bit

Bit 0 of control register B is the daylight savings enable (DSE) bit. It is a read/write bit that when set (1) allows a program to enable two special updates. On the last Sunday in April, the time increments from 1:59:59 AM to 3:00:00 AM. On the last Sunday in October, when the time first reaches 1:59:59 AM, it changes to 1:00:00 AM. These special updates do not occur when the DSE bit is clear (0). DSE is not changed by any internal operations or reset.

24/12 Control Bit

Bit 1 of register B is the 24/12 control bit and establishes the format of the hours bytes as either the 24-hour mode when set (1), or the 12-hour mode when clear (0). This is a read/write bit that is affected only by software.

Data Mode Bit

Bit 2 of register B is the data mode (DM) bit and indicates whether time and calendar updates are to use binary or BCD formats. The processor program writes to the DM bit and the bit can be read by the program but is not modified by any internal functions or by a reset. When the DM bit is set (1), binary data is signified. When the DM bit is clear (0), BCD data is signified.

Bit 3

This bit is not used in this version of the RTC but is used for square wave enable in the Motorola MC146818 chip.

RTC Update End Interrupt Enable Bit

Bit 4 of register B is the update end interrupt enable bit. It is a read/write bit that enables the update end interrupt flag (UF) bit in register C to assert an IRQ(?)<?> signal. Assertion of the reset pin or the SET bit going high clears the UEI bit.

RTC Alarm Interrupt Enable Bit

Bit 5 of register B is the alarm interrupt enable bit. It is a read/write bit that when set (1) permits the alarm interrupt flag (AF) bit in register C to assert an IRQ signal. An alarm interrupt occurs for each second that the three time-bytes equal the three alarm-bytes (including a don't care alarm code of 11XXXXXXB). When the AIE bit is clear (0), the AF bit does not initiate an IRQ signal. The AIE bit is cleared with assertion of the reset pin, and it is not affected by internal functions.

RTC Periodic Interrupt Enable Bit

Bit 6 of register B is the periodic interrupt enable bit. It is a read/write bit that allows the periodic interrupt flag (PF) bit in register C to cause the IRQ pin to be driven low. A program writes a 1 to the PIE bit to receive periodic interrupts at the rate specified by the RS3, RS2, RS1, and RS0 bits in register A. A 0 in the PIE bit stops the IRQ signal from being initiated by a periodic interrupt, but the periodic interrupt flag bit is still set at the periodic rate. The PIE bit is not modified by any internal functions, but is cleared by a reset.

Set Command Bit

Bit 7 of register B is the set command (SET) bit. When the SET bit is 0, the update cycle functions normally by advancing the counts once-per-second. When the SET bit is set (1), any update cycle in progress is aborted and the program may initialize the time and calendar bytes without an update occurring during initialization. The SET bit is a read/write bit that is not modified by a reset or internal functions.

RTC Control Register C

Register C contains status information about interrupts and internal operation of the RTC. The bits in this register are shown in Table 22–8.

Table 22–8 Real-Time Clock Control Register C Bit Definitions

Bit	Description	Abbreviation
0	Not used, read as 0	
1	Not used, read as 0	
2	Not used, read as 0	
3	Not used, read as 0	
4	Update end interrupt flag	UF
5	Alarm interrupt flag	AF
6	Periodic interrupt flag	PF
7	IRQ pending flag	IRQF

Bits 0 to 3

The unused bits of register C are read as 0s and cannot be written to.

RTC Update Ended Interrupt Flag Bit

Bit 4 of register C is the update ended interrupt flag (UF) bit. This bit is set (1) after each update cycle. When the UIE bit is 1, the 1 in UF causes the IRQF bit to be a 1, and this asserts an IRQ signal. The UF bit is cleared by reading from register C or by a reset.

RTC Alarm Interrupt Flag Bit

Bit 5 of register C is the alarm interrupt flag (AF) bit. This bit indicates that the current time has matched the alarm time. A 1 in the AF bit causes the IRQ pin to assert (transition low), and sets (1) the IRQF bit when the AIE bit is set (1). The AF bit is cleared by read from register C or by a reset.

RTC Periodic Interrupt Flag Bit

Bit 6 of register C is the periodic interrupt flag (PF) bit. This bit is a read-only bit that is set (1) when a particular edge is detected on the selected tap of the divider chain. The RS3-RS0 bits establish the periodic rate. The PF bit is set independently of the PIE bit. When the PF bit is set, an IRQ signal is initiated and the IRQF bit is set, if the PIE bit is also set. The PF bit is cleared by reading from register C or by a reset.

RTC Interrupt Request Pending Flag Bit

Bit 7 of register C is the interrupt request pending flag (IRQF) bit. The IRQF bit is set (1) when one or more of the following are true:

- PF = PIE = 1
- AF = AIE = 1
- UF = UIE = 1

The logic can be expressed in equation form as follows:

IRQF = (PF AND PIE) OR (AF AND AIE) OR (UF AND UIE)

Any time the IRQF bit is set, the IRQ pin is asserted. The IRQF bit is cleared by reading from register C or by a reset.

RTC Control Register D

Register D contains a bit that indicates the status of the on-chip standby RAM. The contents of the register are shown in Table 22–9.

Table 22–9 Bit Definitions of Real-Time Clock Control Register D

Bit	Description	Abbreviation
0	Not used, read as 0	
1	Not used, read as 0	
2	Not used, read as 0	
3	Not used, read as 0	
4	Not used, read as 0	
5	Not used, read as 0	
6	Not used, read as 0	
7	Valid RAM data and time	VRT

Bits 0 to 6

Bits 0-6 of register D are not used. These bits cannot be written to and are always read as 0s.

Valid RAM Data and Time Bit

Bit 7 of register D is the valid RAM data and time (VRT) bit. This bit indicates the condition of the contents of the RAM, provided that the power sense (PS(?)) pin is satisfactorily connected. The VRT bit is clear when the power sense pin is low. The processor program can set the VRT bit when the time and calendar are initialized to indicate that the RAM and time are valid. The VRT bit is a read-only bit that is not modified by the reset pin. The VRT bit can be set only by reading from register D.

Note

Pulling the PS(?) pin low for a minimum of 2 μs also sets all RAM bytes from address $0E_{16}$-$3F_{16}$ to 1s.

General RTC Notes

The following sections contain general information about the RTC.

Set Operation

Before initializing the internal registers, the SET bit in register B must be set to prevent time or calendar updates from occurring. The program initializes the 10 locations in the selected format (BCD or binary), and then indicates the format in the data mode (DM) bit in register B. All 10 time, calendar, and alarm bytes must use the same data mode of either binary or BCD. The SET bit can be cleared to allow updates. When initialized, the RTC makes all updates in the selected data mode. The data mode cannot be changed without reinitializing the 10 data bytes.

BCD Versus Binary

The 24/12 bit in register B establishes whether the hour locations represent 1 to 12 or 0 to 23. The 24/12 bit cannot be changed without reinitializing the hour locations. When the 12-hour format is selected, the high-order bit of the hours-byte represents PM when it is set (1).

RTC Update Operation

The time, calendar, and alarm bytes are not always accessible by the processor program. The 10 bytes are switched once each second to the update logic. The update logic advances the time by 1 second and checks for an alarm condition. If any of the 10 bytes are read at this time, the data outputs are undefined. The update lockout time is 1948 μs for the 32.768 kHz time base. The section entitled RTC Update Cycle for information on how to accommodate the update cycle in the processor program.

RTC Alarm Operation

The three alarm bytes are used in the following two ways:

- First, when the program inserts an alarm time in the appropriate hours, minutes, and seconds alarm locations, the alarm interrupt is initiated at the specified time each day if the alarm interrupt enable (AIE) bit in register B is set.

- Second, a don't care state is inserted in one or more of the three alarm bytes. The don't care code is any byte from $0C0_{16}$-$0FF_{16}$. An alarm interrupt each hour is created with a don't care code in the hours alarm location. Similarly, an alarm is generated every minute with don't care codes in the hours and minutes alarm bytes. The don't care codes in all three alarm bytes create an interrupt every second.

RTC Interrupts

The RTC, including RAM, has three separate, fully automatic sources of interrupts to the processor. You can program the alarm interrupt to occur at a rate of from 1 per second to 1 per day. You can select the periodic interrupt for a rate of from 0.5 s to 30.517 μs. You can use the update ended interrupt to indicate to the program that an update cycle has completed.

The processor program selects which interrupts, if any, it wants to receive. Three bits in register B enable the three interrupts. Writing a 1 to an interrupt enable bit permits that interrupt to be initiated when the event occurs. A 0 in the interrupt enable bit prohibits the IRQ pin from being asserted because of the event.

If an interrupt flag is already set when the interrupt becomes enabled, the IRQ pin is immediately activated. However, the event that initiated the interrupt may have occurred much earlier. There are cases when the program must clear such earlier interrupts before enabling new interrupts.

When an interrupt event occurs, a flag bit is set in register C. Each of the three interrupt sources has separate flag bits in register C that are set independently of the state of the corresponding enable bits in register B. The flag bit can be used with or without enabling the corresponding enable bits.

Divider Control

The divider control bits are fixed for only 32.768 kHz operation. The divider chain can be held in reset, which allows precision setting of the time. When the divider is changed from reset to an operating time base, the first update cycle is 0.5 seconds later. The divider control bits are also used to test the RTC.

RTC Periodic Interrupt Selection

The periodic interrupt enables the IRQ pin to be triggered from once every 500 ms to once every 30.157 μs. The periodic interrupt is separate from the alarm interrupt, which can generate output from between 1 per second to 1 per day.

RTC Update Cycle

The RTC executes an update cycle once each second, based on the following assumptions:

- One of the proper time bases is in place
- The DV2-DV0 divider is not clear
- The SET bit in register B is clear

When set, the SET bit permits the program to initialize the time and calendar bytes by stopping an existing update and preventing a new one from occurring.

The primary function of the update cycle is to increment the second byte, check for overflow, increment the minutes byte when appropriate, and so on, up to the year-of-the-century byte. The update cycle also compares each alarm byte with the corresponding time byte and issues an alarm if a match or if a don't care code (11XXXXXX) is present in all three positions.

With a 32.768 kHz time base, an update cycle takes 1984 μs, during which, the time, calendar, and alarm bytes are not accessible by the processor program, protecting the program from reading transitional data. This protection is provided by switching off the microprocessor bus the time, calendar, and alarm portion of the RAM during the entire update cycle. If the processor reads these RAM locations before the update is complete, the output is undefined. The update in progress (UIP) status bit is set during the interval.

Three methods of accommodating nonavailablity during an update are usable by the program. In describing the three methods, it is assumed that at random points, your programs are able to call a subroutine to obtain the time of day.

The first method of avoiding the update cycle uses the update ended interrupt. If enabled, an interrupt occurs after every update cycle that indicates that over 999 ms are available to read the valid time and date information. Before leaving the interrupt service routine, the IRQF bit in register C must be cleared.

The second method uses the update in progress bit (UIP) in register A to determine whether the update cycle is in progress. The UIP bit pulses once each second. After the UIP bit goes high, the update cycle begins 244 μs later. Therefore, if a low is read in the UIP bit, your code has at least 244 μs before the time or calendar data is changed. If a 1 is read in the UIP bit, the time or calendar data may not be valid. You must not write or use interrupt service routines that cause the time needed to read the valid time or calendar data to exceed 244 μs.

The third method uses a periodic interrupt to determine whether an update cycle is in progress. The UIP bit in register A is set high between the setting of the PF bit in register C.

To properly set up the internal counters for daylight savings time operation, you must set the time at least 2 seconds before the roll-over occurs. Also, the time must be set at least 2 seconds before the end of the 29th or 30th day of the month.

23

Keyboard Controller

Introduction

This chapter describes the VLSI Technology VL82C106 chip keyboard controller functions and registers.

In This Chapter

This chapter contains the following sections:

- Keyboard Controller Overview
- Keyboard Port Interface Protocol
- Keyboard Controller Programmer Interface
- PS/2 Mode Register
- PS/2 Status Register
- Keyboard Controller Command Set

Keyboard Controller Overview

This section gives an overview of the keyboard controller.

PS/2 Command Set and Conversion Code

The keyboard controller has on-chip ROM that contains the code to support the PS/2 command set and 128 bytes of conversion code.

Keyboard Serial I/O

Keyboard serial I/O is handled with hardware implementations of the receiver and transmitter. Both functions depend on an 8-bit timer for time-out detection. Enhanced status reporting is provided in hardware to simplify error handling in software. This logic is duplicated for the mouse interface.

User RAM

User RAM support is provided. Your program can write commands $20\text{-}3F_{16}$ (read) or $60\text{-}7F_{16}$ (write) with the lower 5 bits representing the RAM address. Data from a read or for a write is accessed through port 60_{16}, the data bus buffer (DBB).

Keyboard Parallel Ports

Parallel port 1 (input) is provided and parallel port 2 (output) has defined functions depending on whether the controller is in PC/AT or PS/2 Mode.

Port 60_{16} and Status Register Support

Support is provided in hardware for the port 60_{16} data bus buffer (reads and writes) and a status register (reads and writes) to act as an interface to the PC host.

Keyboard Port Interface Protocol

Data transmission between the controller, the keyboard, and the mouse consists of a synchronous bit stream over the data and clock lines. The bits are defined in Table 23–1.

Table 23–1 Keyboard Port Interface Protocol

Bit	Function
1	Start bit (always 0)
2	Data bit 0 (LSB)
3-8	Data bits 1-1
9	Data bit 7 (MSB)
10	Parity bit (Odd)
11	Stop bit (always 1)

Keyboard Controller Programmer Interface

The programmer interface to the keyboard controller is simple, consisting of four registers (see Table 23–2).

Table 23–2 Keyboard Controller Registers

Register	Access Type	I/O n_{16}
Status	R	64
Command	W	64
Output buffer	R	60
Input buffer	W	60

Note

The behavior of these registers depends on the PS/2 mode of operation.

The keyboard controller registers are described in the following sections.

PS/2 Mode Register

Figure 23–1 shows the format of the PS/2 mode register.

Figure 23–1 PS/2 Mode Register (Read Port 60_{16} After Writing Command 20_{16} to Port 64_{16})

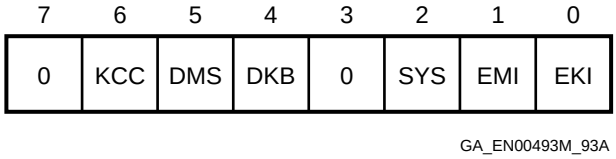


Table 23–3 describes each bit in the PS/2 mode register.

Table 23–3 PS/2 Mode Register

Bit	Description
EKI	Bit 0 is the enable keyboard interrupt (EKI) bit. When set (1), the EKI bit causes the controller to generate a keyboard interrupt when keyboard or command data is written to the output buffer.
EMI	Bit 1 is the enable mouse interrupt (EMI) bit. When set (1), the EMI bit enables the controller to generate a mouse interrupt when mouse data is available in the output register.
SYS	Bit 2 is the system flag (SYS) bit. When set (1), the SYS bit sets the system flag bit of the status register to 1. When set, this bit indicates a switch from virtual to real mode.
Bit 3	Bit 3 is reserved and read as 0.
DKB	Bit 4 is the disable keyboard (DKB) bit. When set (1), the DKB bit disables the keyboard by holding the KCKOUT signal low.
DMB	Bit 5 is the disable mouse (DMS) bit. When set (1), the DMS bit disables the mouse by deasserting the mouse clock signal.
KCC	Bit 6 is the keycode conversion (KCC) bit. When set (1), the KCC bit causes the controller to convert the PS/2 keyboard scan codes to PC/AT format. When reset, the PS/2 keyboard scan codes are passed along unconverted.
Bit 7	Bit 7 is reserved and read as 0.

PS/2 Status Register

Figure 23–2 shows the format of the PS/2 status register.

Figure 23–2 PS/2 Status Register (Read-Only—Port 64H)

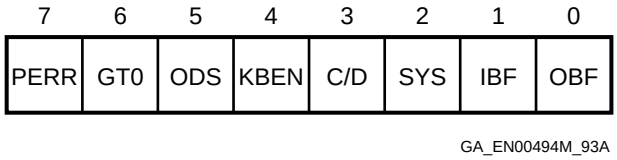


Table 23–4 describes each bit in the PS/2 status register.

Table 23–4 PS/2 Status Register

Bit	Description
OBF	Bit 0 is the output buffer full (OBF) bit. When set (1), the OBF bit indicates that data is available in the controller data bus buffer (DBB), and that the CPU has not yet read the data. The CPU reads from port 60_{16} to reset the state of this bit.
IBF	Bit 1 is the input buffer full (IBF) bit. When set (1), the IBF bit indicates that data has been written to port 60_{16} or port 64_{16}, and that the controller has not read the data.
SYS	Bit 2 is the system flag (SYS) bit. When set (1), the SYS bit indicates that the CPU has changed from virtual to real mode.
C/D	Bit 3 is the command and data (C/D) bit. When set (1), the CD bit indicates that a command has been placed in the input data buffer of the keyboard controller. The keyboard controller uses this bit to determine if the byte written is a command to be executed. This bit is not reset until the command has completed.
KBEN	Bit 4 is the keyboard enable (KBEN) bit. The KBEN bit indicates the state of the keyboard inhibit switch input port (KKSW [KI7]). This general purpose input port has no effect in PS/2 Mode.

(continued on next page)

Table 23–4 (Cont.) PS/2 Status Register

Bit	Description
ODS	Bit 5 is the output buffer data source (ODS) bit. When set (1), the ODS bit indicates that the data in the output buffer is mouse data. When reset, it indicates the data is from the keyboard.
TERR	Bit 6 is the time-out error (TERR) bit. When set (1), the TERR bit indicates that a transmission was started and that it did not complete within the normal time taken (approximately 11 KCKIN cycles). If the transmission originated from the keyboard controller, the value FE_{16} is placed in the output buffer. If the transmission originated from the keyboard, the value FF_{16} is placed in the output buffer.
PERR	Bit 7 is the parity error (PERR) bit. When set (1), the PERR bit indicates that a parity error (even parity = error) occurred during the last transmission from the keyboard. When a parity error is detected, the output buffer is loaded with the value FF_{16}, the output buffer full (OBF) status bit is set, and the KIRQ pin is set to 1 if the EKI bit in the PS/2 mode register is set to 1.

Keyboard Controller Command Set

The command set supported by the keyboard controller is implemented by writing the command byte to port 64_{16}. Any subsequent data is read from port 60_{16} (see the description of command 20) or written to port 60_{16} (see the description of command 60). The commands are shown in Table 23–5.

Table 23–5 Keyboard Controller Commands

Command n_{16}	Description
20	Read mode register
60	Write mode register
AA	Self test
AB	KBD interface test
AC	Diagnostic dump
AD	Disable keyboard
AE	Enable keyboard
C0	Read input port (P10-P17)
D0	Read output port (P20-P27)
D1	Write output port
E0	Read test inputs (T0, T1)
F0-FF	Pulse output port (P20-P27)
21-3F	Read keyboard controller RAM (byte 1-31)
61-7F	Write keyboard controller RAM (byte 1-31)
A4	Test password
A5	Load password
A6	Enable password
A7	Disable mouse
A8	Enable mouse

(continued on next page)

Table 23–5 (Cont.) Keyboard Controller Commands

Command n_{16}	Description
A9	Mouse interface test
C1	Poll-in port low (P10-P13 -> S4-S7)
C2	Poll-in port high (P14-P17 -> S4-S7)
D1	Write output port
D2	Write keyboard output buffer
D3	Write mouse output buffer
D4	Write to mouse

Note

If data is written to the data bus buffer port (port 60_{16}) and the command preceding it does not use the data from the port (port 60_{16}), the data is transmitted to the keyboard.

The keyboard controllers commands are described in the following sections.

Read Keyboard Controller RAM Command (20)

This command reads the keyboard controller's mode register. The keyboard controller sends its current mode byte to the output buffer, which is accessed by reading from port 60_{16}.

Write Mode Register Command (60)

This command writes to the keyboard controller's mode register. The next byte of data written to the keyboard data port (port 60_{16}) is placed in the controller's mode register.

Read Keyboard Controller RAM (Byte 1-31) Command (21-3F)

This command reads the internal keyboard controller RAM. Bits D4-D0 specify the address.

Write Keyboard Controller RAM (Byte 1-31) Command (61-7F)

This command writes to the internal keyboard controller RAM with the address specified in bits D4-D0.

Test Password Command (A4)

This command checks whether there is a password installed in the controller. The test result is placed in the output buffer, the output buffer full (OBF) bit is set, and the KIRQ signal is asserted if the EKI bit is set. A test result with a value of FA_{16} means that a password is installed. A test result with a value of $F1_{16}$ means that a password is not installed.

Load Password Command (A5)

This command initiates the password load procedure. Following this command, the controller takes the data from the input buffer port (port 60_{16}) until a data byte with a value of 00_{16} is detected or a full 8-byte password is loaded into the password latches. Password data bytes are untranslated make scan codes. Breal scan codes are not checked as part of the password validation.

Enable Password Command (A6)

This command enables the security feature. The A6 command is valid only when the A5 load password command has been properly executed. All incoming keyboard make scan codes are compared for a match. All keyboard data and mouse data is discarded until a proper scan code sequence is entered from the keyboard. The keyboard controller does not accept and execute commands from the system while the password security is enabled.

Disable Mouse Command (A7)

This command sets bit 5 of the mode register, which disables the mouse clock signal (MCKOUT).

Enable Mouse Command (A8)

This command resets bit 5 of the mode register, which enables the mouse clock signal (MCKOUT).

Mouse Interface Test Command (A9)

This command causes the keyboard controller to test the mouse clock and data lines. The results are placed in the output buffer, the OBF bit is set, and the MIRQ line is asserted if the EMI bit is set. The results are shown in Table 23–6.

Table 23–6 Mouse Interface Test Result Definitions

Data n_{16}	Meaning
00	No error
01	Mouse clock line stuck low
02	Mouse clock line stuck high
03	Mouse data line stuck low
04	Mouse data line stuck high

Keyboard Controller Self-Test Command (AA)

This command causes the keyboard controller to perform internal diagnostic tests. A value of 55_{16} is placed in the output buffer if no errors are detected. The OBF bit is set, and the KIRQ line is asserted if the EKI bit is set.

Keyboard Interface Test (AB)

This command causes the keyboard controller to test the keyboard clock and data lines. The test result is placed in the output buffer, the OBF bit is set, and the KIRQ line is asserted if the EKI bit is set. The results are shown in Table 23–7.

Table 23–7 Keyboard Interface Test Result Definitions

Data n_{16}	Meaning
00	No error
01	Keyboard clock line stuck low
02	Keyboard clock line stuck high
03	Keyboard data line stuck low
04	Keyboard data line stuck high

Keyboard Controller Diagnostic Dump Command (AC)

This command is not used on PS/2 keyboard implementations.

Keyboard Disable Command (AD)

This command sets bit 4 of the mode register to 1. It disables the keyboard by disabling the keyboard clock line. Data is not sent or received.

Keyboard Enable Command (AE)

This command clears bit 4 of the mode register. It enables the keyboard clock and enables the keyboard.

Read P1 Input Port Command (C0)

This command reads the keyboard input port and places the contents of the port in the output buffer. This command overwrites the data in the output buffer.

Poll Input Port Low Command (C1)

P1 bits 0-3 are written to the status register bits 4-7 until a new command is issued to the keyboard controller.

Poll Input Port High Command (C2)

P1 bits 4-7 are written to the status register bits 4-7 until a new command is issued to the keyboard controller.

Read Output Port Command (D0)

This command causes the keyboard controller to read the P2 output port and place the data from the port in its output buffer. The bit definitions are shown in Table 23–8.

Table 23–8 P2 Output Port Bit Definitions

Bit	Pin	Definition
0	P20	RC_L
1	P21	A20 Gate
2	P22	MDOUT_L
3	P23	MCKOUT_L
4	P24	KIRQ
5	P25	MIRQ
6	P26	KCKOUT_L
7	P27	KDOUT_L

Write Output Port Command (D1)

The next byte of data written to the keyboard data port (port 60_{16}) is written to the keyboard controller's output port (see Table 23–8 for the bit definitions). The following pins are not modified:

- P22
- P23
- P24
- P25
- P26
- P27

Write Keyboard Output Buffer Command (D2)

The next byte written to the data buffer (port 60_{16}) is written to the output buffer (60_{16}) as if it was initiated by the keyboard. The OBF bit is set and the KIRQ line is asserted if the EKI bit is set.

Write Mouse Output Buffer Command (D3)

The next byte written to the data buffer (port 60_{16}) is written to the output buffer as if it was initiated by the mouse. The OBF bit is set and the MIRQ line is asserted if the EMI bit is set.

Write to Mouse Command (D4)

The next byte written to the data buffer (port 60_{16}) is transmitted to the mouse.

Read Test Inputs Command (E0)

This command causes the keyboard controller to read the T0 and T1 input bits. The data is placed in the output buffer with the definitions shown in Table 23–9.

Table 23–9 T0 and T1 Data Definitions

Bit n_{16}	Definition
0	Keyboard clock
1	Mouse clock
3-7	Read as 0s

Pulse Output Port Command (F0-FF)

Bits 0-3 of the keyboard controller's output port can be pulsed low for approximately 6 μs. Bits 0-3 of the command specify which bit is pulsed. A 0 indicates that the bit is pulsed. A 1 indicates that the bit is not pulsed. A value of FF_{16} is treated as a special case (pulse null port). The following bits are not pulsed:

- P22
- P23
- P26
- P27

24

Chip Select Registers

Introduction

This chapter describes the function of the VLSI Technology VL82C106 chip chip select registers.

In This Chapter

This chapter contains the following sections:

- Chip Select Registers Overview
- Chip Select Base Address Register (LSB) Bit Descriptions
- Chip Select Base Address Register (MSB) Bit Descriptions
- Chip Select Range Register Bit Descriptions
- Default Chip Selects
- Chip Control Registers
- Control Register 0
- Control Register 1

Chip Select Registers Overview

The VL82C106 chip contains a set of 26 registers that are used for programming the following:

- Peripheral chip select base addresses
- Chip select address ranges
- Enabling options

Each base address register is a 16-bit register with bits corresponding to address bits $A15_{16}$-$A0_{16}$.

In addition to the base address registers, there is an address range register that you can use to set bits ($A0_{16}$-$A4_{16}$) to a don't care status. These bits are used in the address range comparison. The address range then controls the address space occupied by the chip select from 1-32 bytes.

There are also programmable bits to selectively generate wait states and assert the IOCS16(L) line when the corresponding address range is present. These registers are used in groups of three for each chip select and are defined in Table 24–1, Table 24–2, and Table 24–3.

Chip Select Base Address Register (LSB) Bit Descriptions

Table 24–1 shows the base address register least-significant byte bit descriptions.

Table 24–1 Chip Select Base Address Register (LSB) Bit Descriptions

Bit	Description
0	Base address bit A0
1	Base address bit A1
2	Base address bit A2
3	Base address bit A3
4	Base address bit A4
5	Base address bit A5
6	Base address bit A6
7	Base address bit A7

Chip Select Base Address Register (MSB) Bit Descriptions

Table 24–2 shows the base address register most-significant byte bit descriptions.

Table 24–2 Chip Select Base Address Register (MSB) Bit Descriptions

Bit	Description
0	Base address bit A8
1	Base address bit A9
2	Base address bit A10
3	Base address bit A11
4	Base address bit A12
5	Base address bit A13
6	Base address bit A14
7	Base address bit A15

Chip Select Range Register Bit Descriptions

Table 24–3 Chip Select Range Register Bit Descriptions

Bit	Description
0	Don't care bit A0
1	Don't care bit A1
2	Don't care bit A2
3	Don't care bit A3
4	Don't care bit A4
5	Wait state 0
6	Wait state 1
7	8/16 bit I/O

The following sections describe the bits contained in the range register.

Chip Select Range Register Bits 0-4

Bits 0-4 of the range register are don't care bits. When set (1), these bits cause the corresponding bit to be ignored during the chip select generation, enabling the chip select signals to correspond to a range of addresses in the space from base address + 0 to base address + 31.

Chip Select Range Register Bits 5 and 6

Bits 5 and 6 of the range register are the wait state 0 and 1 bits. These bits determine the number of wait states that are generated when the corresponding chip select signal is generated. They generate the wait states shown in Table 24–4.

Table 24–4 Wait State Bit Descriptions

WS1	WS0	Wait States
0	0	0
0	1	1
1	0	3
1	1	7

Note

The number of wait states is equal to the number of SYSCLK(?) cycles that the IOCHRDY(?) line is forced inactive (low) by the VL82C106 chip. Programmed wait states can extend only the I/O cycle set by the system architecture.

Chip Select Range Register Bit 7

Bit 7 of the range register is the 8-bit or 16-bit I/O bit. This bit is used to selectively assert IOCS16(L) line when the corresponding chip select signal is generated. When set (1), the access is defined as an 8-bit access, and the I0CS16(L) line is not asserted.

Default Chip Selects

The VL82C106 chip also has several hard-wired default chip selects for the serial ports and line printer port. These default chip selects are used after a reset until the battery-backed programmable values are enabled by bit 3 of the second control register (RTC register $6A_{16}$). The wait state and IOCS16(L) values are also disabled in this mode.

The default chip selects enable the VL82C106 chip to function normally without the need for programming. The default chip selects are shown in Table 24–5.

Table 24–5 Default Chip Select Descriptions

Select/Device	Address
COM1	$3F8_{16}$-$3FF_{16}$ (bit 3 of RTC register 69_{16} = 1)
	$2F8_{16}$-$2FF_{16}$ (bit 3 of RTC register 69_{16} = 0)
COM2	$2F8_{16}$-$2FF_{16}$ (bit 3 of RTC register 69_{16} = 1)
	$3F8_{16}$-$3FF_{16}$ (bit 3 of RTC register 69_{16} = 0)
LPT	$03BC_{16}$-$03BF_{16}$ (bits 5 and 6 of RTC register 69_{16} = 0, 0)
	0378_{16}-$037B_{16}$ (bits 5 and 6 of RTC register 69_{16} = 1, 0)
	0278_{16}-$027B_{16}$ (bits 5 and 6 of RTC register 69_{16} = 0, 1)
-CS4	Not used
-CS5	Not used
-CS6	Not used
-CS7	Not used

Note

When a reset occurs, COM1, COM2, and LPT are enabled and set to the hard-wired values.

Chip Control Registers

The VL82C106 chip contains a number of programmable options, including peripheral base address and chip select hole size. The registers used to provide this control are located in the upper bytes of the RTC address space. They are defined in Table 24–6.

Table 24–6 Chip Control Register Definitions

Address n_{16}	Usage
69	Control register 0
6A	Control register 1
6B	CS1 COM1 base address LSB
6C	CS1 COM1 base address MSB
6D	CS1 COM1 range
6E	CS2 COM2 base address LSB
6F	CS2 COM2 base address MSB
70	CS2 COM2 range
71	CS3 LPT base address LSB
72	CS3 LPT base address MSB
73	CS3 LPT range

Note

Control registers 0 and 1 are not battery-backed by the VBAT supply.

Control Register 0

Control register 0 corresponds to the RTC register 69_{16} or I/O port 102_{16}.

This register contains bits that enable or disable the components of the VL82C106 chip. The bits of this register are defined to be consistent with the definitions used in the PS/2-50 family.

For PS/2 compatibility, this register can also be accessed at address 102 (RTC Register 69_{16} or I/O port 102_{16}). The contents of this register are shown in Table 24–7.

Table 24–7 Control Register 0 Bit Definitions

Bit	Use	Value After Reset
0	SYS BD EN	Enabled (1)
1		
2	CS1 (COM1 EN)	Enabled (1)
3	COM1 DEF	COM1 (1)
4	CS3 (LPT EN)	Enabled (1)
5	LPT DEF 0	Parallel port 1 (0)
6	LPT DEF 1	Disabled (0)
7	-EMODE	Compatible mode (1)

The control register 0 bits are described in the following sections.

System Board Enable Control Bit

Bit 0 of control register 0 is the system board enable (SYS BD EN) control bit. When set (1), the SYS BD EN bit allows bits 2 and 4 to enable and disable their respective devices. When clear, CS1 (COM1 EN) and CS3 (LPT EN) are disabled regardless of the contents of bits 2 and 4.

Communications Port 1 Enable Control Bit

Bit 2 of control register 0 is the communications port A enable (COM1 EN) control bit. When set (1), the COM1 EN bit allows the internal COM1 (CS1) port to be accessed. When clear, COM1 is disabled.

Communications Port 1 Default Address Control Bit

Bit 3 of control register 0 is the communications port A default address (COM1 DEF) control bit. When set (1), the COM1 DEF bit forces the hardwired default base address of COM1 to correspond to $3F8_{16}$-$3FF_{16}$ and COM2 to $2F8_{16}$-$2FF_{16}$. When clear, the COM1 hardwired address corresponds to $2F8_{16}$-$2FF_{16}$ and COM2 to $3F8_{16}$-$3FF_{16}$. The base address is the programmed values, if bit 3 of control register 1 (RTC Register $6A_{16}$) is set.

Line Printer Port Enable Control Bit

Bit 4 of control register 0 is the line printer port enable (LPT EN) control bit. When set (1), the LPT EN bit enables the LPT Port (CS3). When clear, the LPT port is disabled.

Line Printer Port Default 0 and 1 Control Bits

Bits 5 and 6 of control register 0 are the line printer port defaults 0 and 1 (LPT DEF 0 and 1) control bits. The line printer hardwired base address defaults are determined by assigning appropriate values to bits 5 and 6. The base address default assignments are shown in Table 24–8.

Table 24–8 LPT Base Address Default Assignments

Bit 5	Bit 6	Address Range n_{16}
0	0	03BC-03BF
0	1	0378-037B
1	0	0278-027B
1	1	Reserved

Setting bit 3 of RTC register $6A_{16}$ changes the base address to that set in the program address registers for LPT (CS3).

Line Printer Extended Mode Control Bit

Bit 7 of control register 0 is the line printer extended mode (EMODE) control bit. When set (1), it disables the extended mode and forces PC/AT compatibility. When clear, the extended mode is enabled, allowing the printer port direction to be controlled.

Control Register 1

This register corresponds to RTC register $6A_{16}$. It is used to control peripheral chip selects that are not included in control register 0. The bits in control register 1 are defined in Table 24–9.

Table 24–9 Control Register 1 Bit Definitions

Bit	Use	Value After Reset
0	COM2 EN	Enabled (1)
1	AT/PS2 KBD	AT (1)
2	PRIV EN	Enabled (1)
3	CS MODE	Hardwire (0)
4	Not used	
5	Not used	
6	Not used	
7	Not used	

The control register 1 bits are described in the following sections.

Communications Port 2 Enable Bit

Bit 0 of control register 1 is the communications port 2 enable (COM2 EN) bit. When set (1), the COM2 port (CS2) is enabled. When clear, the COM2 port is disabled.

PC/AT or PS/2 Compatible Keyboard Bit

Bit 1 of control register 1 is the PC/AT or PS/2 compatible keyboard (AT/PS2 KBD) bit. When set (1), PC/AT type keyboard functions are selected. When clear, PS/2 type keyboard functions are selected.

Note

The system supports only PS/2 type keyboards. Therefore, your software must clear the AT/PS2 KBD bit after a reset.

Private Controls Enable Bit

Bit 2 of control register 1 is the private controls (PRIV EN) enable bit. When the AT/PS2 KBD bit is set, the PRIV EN bit is used to latch the values of the keyboard controller's output signals KHSE(?), KSRE(?), and IRQM(?) to the VL82C106 chip's output pins. When set (1), these outputs follow the keyboard controller's outputs. When clear, these outputs are held at that value regardless of the keyboard controller's outputs.

When in PS/2 mode (AT/PS2 KBD = 0), this bit has no affect on the KHSE(?), KSRE(?), and IRQM(?) output pins. The VL82C106 chip outputs follow the keyboard controller's outputs.

Chip Select Decode Mode Bit

Bit 3 of control register 1 is the chip select decode mode (CS MODE) bit. When clear (0), CS1-CS3 decodes revert to the hardwired address decoding. When set, the address decoding, wait state generation, and 8- or 16-bit operation function according to the values that are programmed in the RTC registers 69_{16}-$7F_{16}$ (see the sections entitled Default Chip Selects and Chip Control Register for more information.)

Bits 4-7

Bits 4-7 of control register 1 are not implemented on the system module; therefore, your software must clear these bits after a reset.

Part V
Appendixes

Part V provides additional technical and other information about the PB22H-KB system module.

This section includes the following appendixes:

- Appendix A, System I/O Map
- Appendix B, Connector Pin Specifications

A

System I/O Map

Introduction

This appendix describes the system I/O map.

In This Appendix

This appendix contains the following sections:

- System I/O Map
- ISA Expansion Address Aliases for 0100—03FF
- EISA Slot-Specific Addresses

System I/O Map

The following tables describe the EISA I/O space addresses (addr) for the PB22H-KB system module. Any addresses that are not listed are undefined. Table A–1 includes the effective cA (CPU) bus address for the target EISA I/O address. All addresses are in hexadecimal format.

Table A–1 System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
0000	3.x000.0000	DMA controller 1 channel 0 base or current address	R/W	
0001	3.x000.0080	DMA controller 1 channel 0 base or current word count	R/W	
0002	3.x000.0100	DMA controller 1 channel 1 base or current address	R/W	
0003	3.x000.0180	DMA controller 1 channel 1 base or current word count	R/W	
0004	3.x000.0200	DMA controller 1 channel 2 base or current address	R/W	
0005	3.x000.0280	DMA controller 1 channel 2 base or current word count	R/W	
0006	3.x000.0300	DMA controller 1 channel 3 base or current address	R/W	
0007	3.x000.0380	DMA controller 1 channel 3 base or current word count	R/W	
0008	3.x000.0400	DMA controller 1 command	W	
0008	3.x000.0480	DMA controller 1 status	R	
0009	3.x000.0500	DMA controller 1 DMA request	W	
000A	3.x000.0580	DMA controller 1 write single mask bit	W	

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
000B	3.x000.0600	DMA controller 1 mode	W	
000C	3.x000.0680	DMA controller 1 clear byte pointer flip-flop	W	
000D	3.x000.0700	DMA controller 1 master	W	
000E	3.x000.0780	DMA controller 1 clear mask register bits	W	
000F	3.x000.0800	DMA controller 1 write all mask register bits	W	
000F	3.x000.0800	DMA controller 1 mask status	R	
0020	3.x000.1000	Interrupt controller 1 ICW1, OCW2, OCW3	R/W	
0021	3.x000.1080	Interrupt controller 1 ICW2, OCW1	R/W	
0021	3.x000.1080	Interrupt controller 1 ICW3 (master device)	R/W	
0021	3.x000.1080	Interrupt controller 1 ICW4	R/W	
0040	3.x000.2000	Timer 1 counter 0 system clock	R/W	
0041	3.x000.2080	Timer 1 counter 1 refresh request	R/W	
0042	3.x000.2100	Timer 1 counter 2 speaker tone	R/W	
0043	3.x000.2180	Timer 1 control register	W	
0048	3.x000.2400	Timer 2 counter 0 fail-safe timer	R/W	
0049	3.x000.2480	Timer 2 reserved		
004A	3.x000.2500	Timer 2 counter 2	R/W	
004B	3.x000.2580	Timer 2 control register	W	
0060	1.C000.3000	Keyboard output buffer	R	
0060	1.C000.3000	Keyboard input buffer	W	

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
0061	3.x000.3080	NMI status register (port B)	R/W	Bits 0-3 = W/R , bits 4-7 = R
0064	1.C000.3200	Keyboard status register	R	
0064	1.C000.3200	Keyboard control and command register	W	
0070	3.x000.3800	NMI enable register	W	
0081	3.x000.4080	DMA channel 2 low page segment register	R/W	
0082	3.x000.4100	DMA channel 3 low page segment register	R/W	
0083	3.x000.4180	DMA channel 1 low page segment register	R/W	
0087	3.x000.4380	DMA channel 0 low page segment register	R/W	
0089	3.x000.4480	DMA channel 6 low page segment register	R/W	
008A	3.x000.4500	DMA channel 7 low page segment register	R/W	
008B	3.x000.4580	DMA channel 5 low page segment register	R/W	
008F	3.x000.4780	DMA low page register refresh page	R/W	
0092		Gate A20 control		
00A0	3.x000.5000	Interrupt controller 2 ICW1, OCW2, OCW3	R/W	
00A1	3.x000.5080	Interrupt controller 2 ICW2, OCW1	R/W	
00A1	3.x000.5080	Interrupt controller 2 ICW3 (slave device)	R/W	

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
00A1	3.x000.5080	Interrupt controller 2 ICW4	R/W	
00C4	3.x000.6200	DMA controller 2 channel 5 base or current address	R/W	
00C6	3.x000.6300	DMA controller 2 channel 5 base or current word count	R/W	
00C8	3.x000.6400	DMA controller 2 channel 6 base or current address	R/W	
00CA	3.x000.6500	DMA controller 2 channel 6 base or current word count	R/W	
00CC	3.x000.6600	DMA controller 2 channel 7 base or current address	R/W	
00CE	3.x000.6700	DMA controller 2 channel 7 base or current word count	R/W	
00D0	3.x000.6800	DMA controller 2 command	W	
00D0	3.x000.6800	DMA controller 2 status	R	
00D2	3.x000.6900	DMA controller 2 DMA request	W	
00D4	3.x000.6A00	DMA controller 2 write single mask bit	W	
00D6	3.x000.6B00	DMA controller 2 mode	W	
00D8	3.x000.6C00	DMA controller 2 clear byte pointer flip-flop	W	
00DA	3.x000.6D00	DMA controller 2 master	W	
00DC	3.x000.6E00	DMA controller 2 clear mask register bits	W	
00DE	3.x000.6F00	DMA controller 2 write all mask register bits	W	
00DE	3.x000.6F00	DMA controller 2 mask status	R	

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
0170	1.C000.B800	Real-time clock register address	W	
0171	1.C000.B880	Real-time clock data	R/W	
01F0-01FF		Hard drive port 1	R/W	Not used
0200-0207		Game I/O		Not used
0242 (bit 0)		Scroll button		Not used
0278-027F		LPT3 parallel port		Not used
02E8-02EF		COM4		Not used
02F8	1.C001.7C00	COM2 receiver buffer (DLAB = 0)	R	Serial port 2
02F8	1.C001.7C00	COM2 transmitter holding (DLAB = 0)	W	Serial port 2
02F8	1.C001.7C00	COM2 divisor latch LSB (DLAB = 1)	R/W	Serial port 2
02F9	1.C001.7C80	COM2 interrupt enable (DLAB = 0)	R/W	Serial port 2
02F9	1.C001.7C80	COM2 divisor latch MSB (DLAB = 1)	R/W	Serial port 2
02FA	1.C001.7D00	COM2 interrupt identification	R	Serial port 2
02FB	1.C001.7D80	COM2 line control	W	Serial port 2
02FC	1.C001.7E00	COM2 modem control	W	Serial port 2
02FD	1.C001.7E80	COM2 line status	R	Serial port 2
02FE	1.C001.7F00	COM2 modem status	R	Serial port 2
02FF	1.C001.7F80	COM2 scratch		Serial port 2

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
0350-0352	3.x001.A800-3.x001.A900	SCSI host adapter		ASC mode only
0370-0377		Diskette drive port 2		Not used
0378-037F		LPT2 (parallel port)		Not used
0380-038F		SDLC bisynchronous port 2		Not used
03A0-03AF		Bisynchronous port 1		Not used
03B0-03BB		Monochrome display adapter		Not used
03BC	1.C001.DE00	LPT1 parallel port data	R	
03BD	1.C001.DE80	LPT1 parallel port status register	R	
03BE	1.C001.DF00	LPT1 parallel port control register	R/W	
03C0-03CF		EGA display adapter		Not used
03D0-03DF		CGA display adapter		Not used
03F0-03F7		Diskette drive port 1		Not used
03F8	1.C001.FC00	COM1 receiver buffer (DLAB = 0)	R	Serial port 1
03F8	1.C001.FC00	COM1 transmitter holding (DLAB = 0)	W	Serial port 1
03F8	1.C002.FC00	COM1 divisor latch LSB (DLAB = 1)	R/W	Serial port 1
03F9	1.C001.FC80	COM1 interrupt enable (DLAB = 0)	R/W	Serial port 1

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
03F9	1.C002.FC80	COM1 divisor latch MSB (DLAB = 1)	R/W	Serial port 1
03FA	1.C001.FD00	COM1 interrupt identification	R	Serial port 1
03FB	1.C001.FD80	COM1 line control	W	Serial port 1
03FC	1.C001.FE00	COM1 modem control	W	Serial port 1
03FD	1.C001.FE80	COM1 line status	R	Serial port 1
03FE	1.C001.FF00	COM1 modem status	R	Serial port 1
03FF	1.C001.FF80	COM1 scratch		Serial port 1
0401	3.x002.0080	DMA channel 0 extended word count register	R/W	
0403	3.x002.0180	DMA channel 1 extended word count register	R/W	
0405	3.x002.0280	DMA channel 2 extended word count register	R/W	
0407	3.x002.0380	DMA channel 3 extended word count register	R/W	
040A	3.x002.0500	DMA controller 1 set chaining mode register	W	
040A	3.x002.0500	DMA controller 1 channel interrupt status register	R	
040B	3.x002.0580	DMA controller 1 extended mode register	R/W	
040C	3.x002.0600	DMA controller 1 chain buffer expiration control register	R	
0461	3.x002.3080	Extended NMI and reset control register	R/W	Bits 0-3 = W /R,Bits 4-7 = R
0462	3.x002.3100	NMI I/O interrupt port	W	
0464	3.x002.3200	EISA Bus Master	R	

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
0481	3.x002.4080	DMA controller 2 channel 2 high page segment register	R/W	
0482	3.x002.4100	DMA controller 2 channel 3 high page segment register	R/W	
0483	3.x002.4180	DMA controller 2 channel 1 high page segment register	R/W	
0487	3.x002.4380	DMA controller 2 channel 0 high page segment register	R/W	
0489	3.x002.4480	DMA controller 2 channel 6 high page segment register	R/W	
048A	3.x002.4500	DMA controller 2 channel 7 high page segment register	R/W	
048B	3.x002.4580	DMA controller 2 channel 5 high page segment register	R/W	
04C6	3.x002.6300	DMA controller 2 channel 1 extended word count	R/W	
04CA	3.x002.6500	DMA controller 2 channel 2 extended word count	R/W	
04CE	3.x002.6700	DMA controller 2 channel 3 extended word count	R/W	
04D0	3.x002.6800	Interrupt controller 1 edge and level control register	R/W	
04D1	3.x002.6880	Interrupt controller 2 edge and level control register	R/W	
04D4	3.x002.6A00	DMA controller 2 set chaining mode	W	
04D4	3.x002.6A00	DMA controller 2 channel interrupt status	R	
04D6	3.x002.6B00	DMA controller 2 extended mode	W	

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
04E0	3.x002.7000	DMA channel 0 stop register bits <7:2>	R/W	
04E1	3.x002.7080	DMA channel 0 stop register bits <15:8>	R/W	
04E2	3.x002.7100	DMA channel 0 stop register bits <23:16>	R/W	
04E4	3.x002.7200	DMA channel 1 stop register bits <7:2>	R/W	
04E5	3.x002.7280	DMA channel 1 stop register bits <15:8>	R/W	
04E6	3.x002.7300	DMA channel 1 stop register bits <23:16>	R/W	
04E8	3.x002.7400	DMA channel 2 stop register bits <7:2>	R/W	
04E9	3.x002.7480	DMA channel 2 stop register bits <15:8>	R/W	
04EA	3.x002.7500	DMA channel 2 stop register bits <23:16>	R/W	
04EC	3.x002.7600	DMA channel 3 stop register bits <7:2>	R/W	
04ED	3.x002.7680	DMA channel 3 stop register bits <15:8>	R/W	
04EE	3.x002.7700	DMA channel 3 stop register bits <23:16>	R/W	
04F4	3.x002.7A00	DMA channel 5 stop register bits <7:2>	R/W	
04F5	3.x002.7A80	DMA channel 5 stop register bits <15:8>	R/W	
04F6	3.x002.7B00	DMA channel 5 stop register bits <23:16>	R/W	

(continued on next page)

Table A–1 (Cont.) System I/O Map

EISA I/O Addr	cA Bus Address	Description	Access	Notes
04F8	3.x002.7C00	DMA channel 6 stop register bits <7:2>	R/W	
04F9	3.x002.7C80	DMA channel 6 stop register bits <15:8>	R/W	
04FA	3.x002.7D00	DMA channel 6 stop register bits <23:16>	R/W	
04FC	3.x002.7E00	DMA channel 7 stop register bits <7:2>	R/W	
04FD	3.x002.7E80	DMA channel 7 stop register bits <15:8>	R/W	
04FE	3.x002.7F00	DMA channel 7 stop register bits <23:16>	R/W	
	1.D000.0000	Host address extension register	R/W	
	1.E000.0000	System control register	R/W	
	1.F000.0000	Spare register	R/W	

ISA Expansion Address Aliases for 0100—03FF

Any I/O address that contains a 1 in either bit 8 or bit 9, or a 1 in both bit 8 and bit 9, is an alias for the ISA expansion address range 0100—03FF. Table A–2 lists the aliases of 0100 – 03FF.

Table A–2 ISA Expansion Address Aliases for 0100—03FF

0500-07FF
0900-0BFF
0D00-0FFF
1100-13FF
1500-17FF
1900-1BFF
1D00-1FFF
.
.
.
x100-x3FF
x500-x7FF
x900-xBFF
xD00-xFFF

EISA Slot-Specific Addresses

EISA expansion cards use the slot-specific addresses listed in Table A–3.

Table A–3 EISA Slot-Specific Addresses

Address Range	Reserved for
1000-1FFF	Slot 1
2000- 2FFF	Slot 2
.	.
.	.
.	.
0z000-0zFFF	Slot z

Slot-specific addresses 0zC80-0zC83 are reserved for the product ID. Address 0zC84 is reserved for control bits. All other addresses are available to the EISA expansion board for configuration registers and general I/O.

B

Connector Pin Specifications

Introduction

This appendix lists the pin specifications of the standard system connectors.

In This Appendix

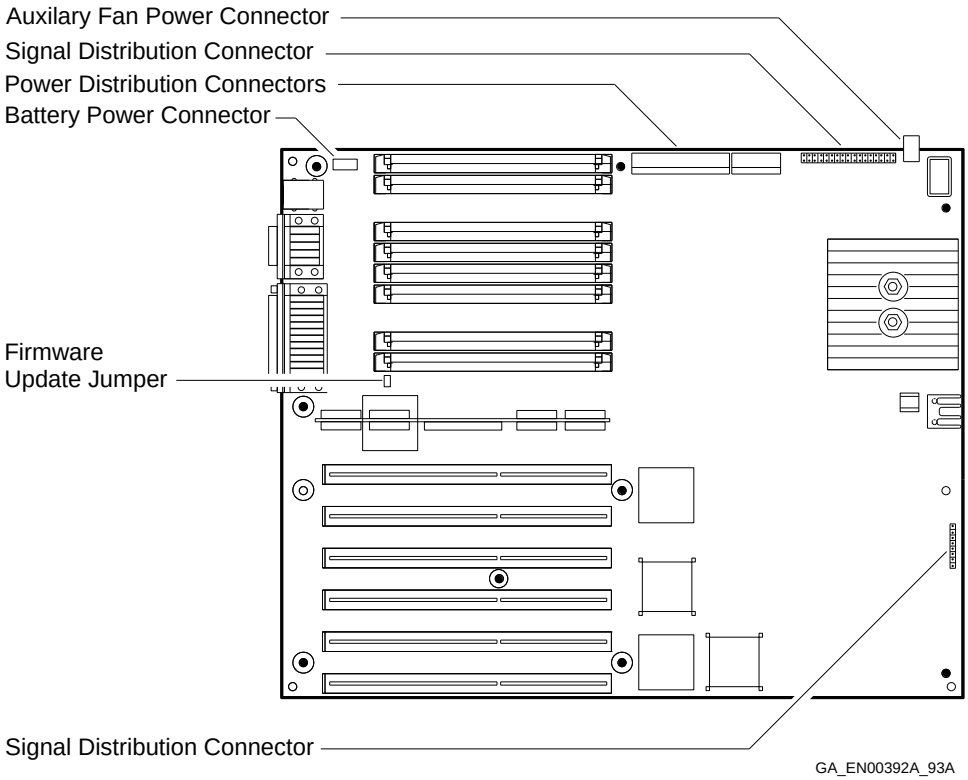
This appendix contains the following sections:

- Keyboard and Mouse Connector Pin Specifications
- Serial Port Pin Specifications
- Parallel Port Pin Specifications

Internal Connector Locations

Figure B–1 shows the locations of the internal connectors on the system module.

Figure B–1 Internal Connector Locations



Power Connectors J22 and J23

J22

Power connector J22 is a 12-pin, 0.156-inch, single row PC power connector.

J23

Power connector J23 is a 6-pin, 0.156-inch, single row PC power connector.

J22 and J23 Pin Specifications

Table B–1 describes the functions of the pins on the J22 and J23 power connectors.

Table B–1 J22 and J23 Pin Specifications

J22		J23	
Pin	**Function**	**Pin**	**Function**
1	Power OK	1	+5 V
2	+5 V	2	+5 V
3	+12 V	3	+5 V
4	-12 V	4	GND
5	GND	5	GND
6	GND	6	GND
7	GND		
8	GND		
9	-5 V		
10	+5 V		
11	+5 V		
12	+5 V		

Battery Power Connector (J25)

The battery power connector (J25) is a 4-pin, 0.1-inch, single row connector.

Battery Power Connector (J25)

Table B–2 describes the functions of the pins on the battery power connector (J25).

Table B–2 Battery Power Connector Pin Specifications

Pin	Function
1	+V battery
2	Key (no pin)
3	Not connected
4	Ground

Front Panel Connector (J24)

The front panel connector is a 34-pin connector.

Front Panel Connector (J24)

Table B–3 describes the functions of the pins on the Front panel connector (J24).

Table B–3 Front Panel Connector Pin Specifications

Pin	Function	Pin	Function
1	Speaker +	18	
2	+5 V	19	
3	Not connected	20	
4	Ground	21	
5	-Reset	22	
6	Ground	23	
7	Not connected	24	
8	Ground	25	
9	Hard disk LED +	26	
10	Hard disk LED -	27	
11	Not connected	28	
12	Not connected	29	
13	-Keylock	30	
14	Ground	31	
15		32	
16		33	
17		34	

Auxiliary Fan Power Connector (J8)

The auxiliary fan power connector (J8) is a 3-pin, single row connector.

Auxiliary Fan Power Connector (J8)

Table B–4 describes the functions of the pins on the auxiliary fan power connector (J8).

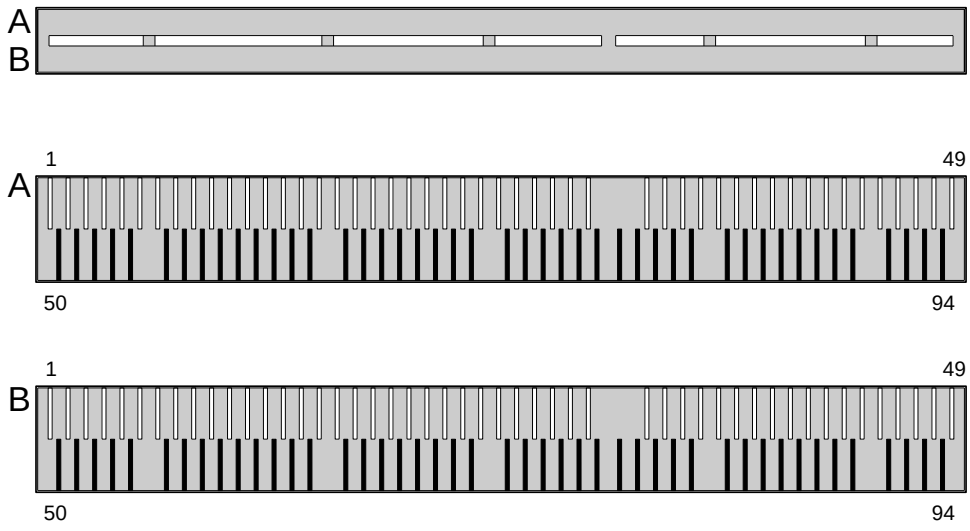
Table B–4 Auxiliary Fan Power Connector Pin Specifications

Pin	Function
1	Ground
2	+12 V
3	Not connected

EISA Connector Pin Specifications

The EISA connectors are standard 188-pin EISA bus slot connectors. Figure B–2 shows the pin numbers on both sides of the EISA connector.

Figure B–2 EISA Connector Pin Numbers



EISA Connectors

Table B–5 describes the functions of the pins on the EISA connector.

Table B–5 EISA Connector Pin Specifications

Pin	Function	Pin	Function	Pin	Function	Pin	Function
A1	GND	A48	MASTER16(L)	B1	IOCHK(L)	B48	D<14>

(continued on next page)

Table B–5 (Cont.) EISA Connector Pin Specifications

Pin	Function	Pin	Function	Pin	Function	Pin	Function
A2	RESDRV	A49	GND	B2	D<7>	B49	D<15>
A3	+5 V	A50	GND	B3	D<6>	B50	CMD(L)
A4	IRQ<9>	A51	+5 V	B4	D<5>	B51	START(L)
A5	-5 V	A52	+5 V	B5	D<4>	B52	EXRDY
A6	DRQ<2>	A53	RESERVED	B6	D<3>	B53	EX32(L)
A7	-12 V	A54	RESERVED	B7	D<2>	B54	GND
A8	NOWS(L)	A55	RESERVED	B8	D<1>	B55	EX16(L)
A9	+12 V	A56	RESERVED	B9	D<0>	B56	SLBURST(L)
A10	GND	A57	+12 V	B10	CHRDY	B57	MSBURST(L)
A11	SMWTC(L)	A58	M-10	B11	AENx	B58	W-R
A12	SMRDC(L)	A59	LOCK(L)	B12	SA<19>	B59	GND
A13	IOWC(L)	A60	RESERVED	B13	SA<18>	B60	RESERVED
A14	IORC(L)	A61	GND	B14	SA<17>	B61	RESERVED
A15	DAK(L)<3>	A62	RESERVED	B15	SA<16>	B62	RESERVED
A16	DRQ<3>	A63	BE(L)<3>	B16	SA<15>	B63	GND
A17	DAK(L)<1>	A64	BE(L)<2>	B17	SA<14>	B64	BE(L)<1>
A18	DRQ<1>	A65	BE(L)<0>	B18	SA<13>	B65	LA(L)<31>
A19	REFRESH(L)	A66	GND	B19	SA<12>	B56	GND
A20	BCLK	A67	+5 V	B20	SA<11>	B67	LA(L)<30>
A21	IRQ<7>	A68	LA(L)<29>	B21	SA<10>	B68	LA(L)<28>
A22	IRQ<6>	A69	GND	B22	SA<9>	B69	LA(L)<27>
A23	IRQ<5>	A70	LA(L)<26>	B23	SA<8>	B70	LA(L)<25>
A24	IRQ<4>	A71	LA(L)<24>	B24	SA<7>	B71	GND
A25	IRQ<3>	A72	LA<16>	B25	SA<6>	B72	LA<15>
A26	DAK(L)<2>	A73	LA<14>	B26	SA<5>	B73	LA<13>
A27	T-C	A74	+5 V	B27	SA<4>	B74	LA<12>

(continued on next page)

Table B–5 (Cont.) EISA Connector Pin Specifications

Pin	Function	Pin	Function	Pin	Function	Pin	Function
A28	BALE	A75	+5 V	B28	SA<3>	B75	LA<11>
A29	+5 V	A76	GND	B29	SA<2>	B76	GND
A30	OSC	A77	LA<10>	B30	SA<1>	B77	LA<9>
A31	GND	A78	LA<8>	B31	SA<0>	B78	LA<7>
A32	M16(L)	A79	LA<6>	B32	SBHE(L)	B79	GND
A33	IO16(L)	A80	LA<5>	B33	LA<23>	B80	LA<4>
A34	IRQ<10>	A81	+5 V	B34	LA<22>	B81	LA<3>
A35	IRQ<11>	A82	LA<2>	B35	LA<21>	B82	GND
A36	IRQ<12>	A83	D<16>	B36	LA<20>	B83	D<17>
A37	IRQ<15>	A84	D<18>	B37	LA<19>	B84	D<19>
A38	IRQ<14>	A85	GND	B38	LA<18>	B85	D<20>
A39	DAK(L)<0>	A86	D<21>	B39	LA<17>	B86	D<22>
A40	DRQ<0>	A87	D<23>	B40	MRDC(L)	B87	GND
A41	DAK(L)<5>	A88	D<24>	B41	MWTC(L)	B88	D<25>
A42	DRQ<5>	A89	GND	B42	D<8>	B89	D<26>
A43	DAK(L)<6>	A90	D<27>	B43	D<9>	B90	D<28>
A44	DRQ<6>	A91	D<29>	B44	D<10>	B91	GND
A45	DAK(L)<7>	A92	+5V	B45	D<11>	B92	D<30>
A46	DRQ<7>	A93	+5 V	B46	D<12>	B93	D<31>
A47	+5 V	A94	MAKx(L)	B47	D<13>	B94	MREQx(L)

Keyboard and Mouse Connector Pin Specifications

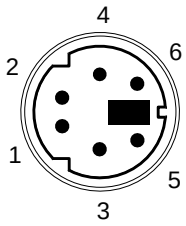
Summary

This section lists the pin specifications for the keyboard and mouse connectors.

Keyboard and Mouse Connector Illustration

Figure B–3 shows the pin numbers on the keyboard and mouse connectors.

Figure B–3 Keyboard and Mouse Connector



GA_EN00286A_93A

Keyboard and Mouse Connector Pin Specifications

Table B–6 describes the functions of the pins on the keyboard and mouse connectors.

Table B–6 Keyboard and Mouse Connector Pin Specifications

Pin	Function
1	Data
2	Unused
3	Ground
4	+5 Volts dc
5	Clock
6	Unused

Serial Port Pin Specifications

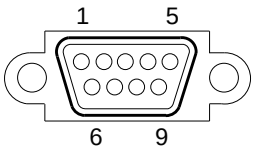
Summary

This section lists the pin specifications for the serial port.

Serial Port Illustration

Figure B–4 shows the pin numbers on the serial port.

Figure B–4 Serial Port



GA_EN00287A_93A

Serial Port Pin Specifications

Table B–7 describes the functions of the pins on the serial port.

Table B–7 Serial Port Pin Specifications

Pin	Function
1	Carrier detect
2	Receive data
3	Transmit data
4	Data term ready
5	Signal ground
6	Data set ready
7	Request to send
8	Clear to send
9	Ring indicator

Parallel Port Pin Specifications

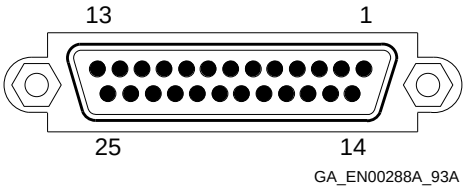
Summary

This section lists the pin specifications for the parallel port.

Parallel Port Illustration

Figure B–5 shows the pin numbers on the parallel port.

Figure B–5 Parallel Port



Parallel Port Pin Specifications

Table B–8 describes the functions of the pins on the parallel port.

Table B–8 Parallel Port Pin Specifications

Pin	Function	Pin	Function
1	Strobe	14	Auto linefeed
2	Data bit 0	15	Error
3	Data bit 1	16	Initialize printer
4	Data bit 2	17	Select in
5	Data bit 3	18	Signal ground
6	Data bit 4	19	Signal ground
7	Data bit 5	20	Signal ground
8	Data bit 6	21	Signal ground
9	Data bit 7	22	Signal ground
10	Acknowledge	23	Signal ground
11	Busy	24	Signal ground
12	Paper end	25	Signal ground
13	Select		

Glossary

The glossary defines some of the technical terms used in this manual.

arbiter

The entity responsible for controlling a bus. It controls bus mastership and may field bus interrupt requests.

assert

To cause a signal to change to its logical true state.

AST

See asynchronous system trap.

asynchronous system trap (AST)

A software-simulated interrupt to a user-defined routine. ASTs enable a user process to be notified asynchronously, with respect to that process, of the occurrence of a specific event. If a user process has defined an AST routine for an event, the system interrupts the process and executes the AST routine when that event occurs. When the AST routine exits, the system resumes execution of the process at the point where it was interrupted.

backup cache

A second, very fast memory that is used in combination with slower large-capacity memories.

bandwidth

Bandwidth is often used to express *a high rate of data transfer* in an I/O channel. This usage assumes that a wide bandwidth may contain a high frequency, which can accommodate a high rate of data transfer.

baud rate

The speed at which data is transmitted over a data line; baud rates are measured in bits per second.

bit

Binary digit. The smallest unit of data in a binary notation system, designated as 0 or 1.

BIU

See bus interface unit.

buffer

An internal memory area used for temporary storage of data records during input or output operations.

bus

A group of signals that consists of many transmission lines or wires. It interconnects computer system components to provide communications paths for addresses, data, and control information.

Some of the buses used on the system include the EISA bus, the serial control bus, L_bus, H_bus, and cA Bus.

bus interface unit

Logic designed to provide an interface from internal logic, from a module or a chip, to a bus.

byte

Eight contiguous bits starting on an addressable byte boundary. The bits are numbered right to left, 0-7.

cache

See cache memory.

cache block

The fundamental unit of manipulation in a cache. Also known as cache line.

cache interference

The result of an operation that adversely affects the mechanisms and procedures used to keep frequently used items in a cache. Such interference may cause frequently used items to be removed from a cache or incur significant overhead operations to ensure correct results. Either action hampers performance.

cache line

The fundamental unit of manipulation in a cache. Also known as cache block.

cache memory

A small, high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed. It contains copies of data recently used by the processor and fetches several bytes of data from memory when it expects that the processor will access the next sequential series of bytes.

central processing unit (CPU)

The unit of the computer that is responsible for interpreting and executing instructions.

channel

A path along which digital information can flow in a computer.

checksum

A sum of digits or bits that is used to verify the integrity of a piece of data.

clock

A signal used to synchronize the circuits in a computer system.

CMOS

Complementary metal-oxide semiconductor. A silicon device formed by a process that combines PMOS and NMOS semiconductor material.

command

A field of the system bus address and command cycle (cycle 1), which encodes the transaction type.

console mode

The state in which the system and the console terminal operate under the control of the console program.

console program

The code that the CPU executes during console mode. The console code is kept in FEPROM on the PB22H-KB system module.

control and status register (CSR)

A device or controller register that resides in the processor's I/O space. The CSR initiates device activity and records its status.

CPU

See central processing unit.

CSR

See control and status register.

cycle

One clock interval.

data alignment

An attribute of a data item that refers to its placement in memory (therefore its address).

data bus

A bus used to carry signals between two or more components of the system.

D-cache

Data cache. A high-speed memory reserved for the storage of data. *Contrast with* I-cache.

deassert

To cause a signal to change to its logical false state.

DECchip 21064 processor

The CMOS-4, single-chip processor used in computers that are based on the Alpha AXP architecture.

DEC OSF/1 operating system

A general-purpose operating system based on the Open Software Foundation OSF/1 1.0 technology. DEC OSF/1 V1.2 runs on the range of Alpha systems, from workstations to servers.

direct-mapping cache

A cache organization in which only one address comparison is needed to locate any data in the cache, because any block of main memory data can be placed in only one possible position in the cache.

direct memory access (DMA)

Access to memory by an I/O device that does not require processor intervention.

dirty

Used in reference to a cache block in the cache of a system bus node. The cache block is valid and has been written so that it differs from the copy in system memory.

dirty victim

Used in reference to a cache block in the cache of a system bus node. The cache block is valid but is about to be replaced because of a resource conflict in a cache block. The data must therefore be written to memory.

DRAM

Dynamic random-access memory. Read/write memory that must be refreshed (read from or written to) periodically to maintain the storage of information.

EBB

The 82352 EISA bus buffer chip.

EBC

The 82358DT EISA bus controller chip.

EDC logic

Error detection and correction logic. Used to detect and correct errors.

EEPROM

Electrically erasable programmable read-only memory. A memory device that can be byte-erased, written to, and read from. *Contrast with* FEPROM.

EISA bus

Extended ISA bus. This is a high-performance 32-bit bus, a superset of the ISA bus which includes all ISA bus features and extensions to enhance capabilities and performance.

EISA bus master

A 16- or 32-bit bus master that uses the EISA superset of signals for accesses to memory or I/O.

EISA slave

An 8-, 16-, or 32-bit I/O or memory slave device that uses the EISA superset of signals to accept accesses from various masters.

extents

The physical locations in a storage device allocated for use by a particular data set.

FEPROM

Flash-erasable programmable read-only memory. FEPROMs can be bank-erased or bulk-erased. *Contrast with* EEPROM.

The console code for the PB22H-KB system module is stored in FEPROM.

firmware

Software code stored in hardware.

floating-point

TBD

FRU

Field-replaceable unit. Any system component that the service engineer is able to replace on-site.

granularity

A characteristic of storage systems that defines the amount of data that can be read, written, or both, with a single instruction, or read, written, or both independently. VAX systems have byte or multibyte granularities, whereas disk systems typically have 512-byte or greater granularities. For a given storage device, a higher granularity generally yields a greater throughput.

hard error

An error that has induced a nonrecoverable failure in a system.

On the PB22H-KB system module, an acknowledgment to the DECchip 21064 CPU when an uncorrectable system error has occurred on a given cycle.

hexaword

Short for hexadecimal word. Thirty two contiguous bytes (256 bits) starting on an addressable byte boundary. Bits are numbered from right to left, 0 through 255.

high-level language

A language for specifying computing procedures or organization of data within a digital computer. High-level languages are distinguished from low-level languages, such as assembly and machine languages, by the omission of machine-specific details required for direct execution on a given computer. *See also* low-level language.

hit

Indicates that a valid copy of a memory location is currently in cache.

Host bus (H_bus)

The bus on which the DECchip 21064 CPU and system memory reside.

Host bus master

A 32-bit bus master that resides on the host bus.

Host bus slave

A 32-bit bus slave residing on the host bus.

I-cache

Instruction cache. A high-speed memory reserved for the storage of instructions.

One of the two areas of primary cache located on the DECchip 21064 CPU used to store instructions. The I-cache on the DECchip 21064 CPU contains 8K bytes of memory space. It is a direct-mapped cache. I-cache blocks, or lines, contain 32 bytes of instruction stream data with associated tag as well as a 6-bit ASM field and an 8-bit branch history field per block. I-cache does not contain hardware for maintaining cache coherency with memory and is unaffected by the invalidate bus. *Contrast with* D-cache.

initialization

The sequence of steps that prepare the system to start. Initialization occurs after a system has been powered up.

internal processor register (IPR)

A register internal to the CPU chip.

ISA bus

Industry Standard Architecture bus. This is a 16-bit bus and was the basis for the 32-bit EISA bus.

ISA master

A 16-bit bus master that uses the ISA subset of EISA bus signals for accesses to memory or I/O.

ISA slave

An 8- or 16-bit slave that uses the ISA subset of EISA bus signals to accept accesses from various masters.

ISP

The Intel 82357 chip integrated system peripheral (ISP) functions.

latency

The amount of time it takes the system to respond to an event.

LED

Light-emitting diode. A semiconductor device that glows when supplied with a voltage.

load and store architecture

A characteristic of a machine architecture where data items are first loaded into a processor register, operated on, and then stored back to memory. No operations on memory other than load and store are provided by the instruction set.

longword

Four contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 to 31.

low-level language

Any language that exposes the details of the hardware implementation to the programmer. Typically this refers to assembly languages that allow direct hardware manipulation. *See also* high-level language.

machine check

An operating system action triggered by certain system hardware-detected errors that can be fatal to system operation. Once triggered, machine check handler software analyzes the error.

masked write

A write cycle that updates only a subset of a nominal data block.

MBO

See must be one.

MBZ

See must be zero.

memory-like

Refers to regions that have predictable behavior. For example, all locations are read/write; a write to a location followed by a read from that location returns precisely the bits written. *See also* nonmemory-like.

MIPS

Millions of instructions per second.

miss

Indicates that a copy of a memory location is not in a cache.

multiplex

To transmit several messages or signals simultaneously on the same circuit or channel.

must be one (MBO)

A field that must be supplied as one.

must be zero (MBZ)

A field that is reserved and must be supplied as zero. If examined, it must be assumed to be undefined.

naturally aligned data

Data stored in memory such that the address of the data is evenly divisible by the size of the data in bytes. For example, an aligned longword is stored so that the address of the longword is evenly divisible by 4.

node

A device that has an address on, is connected to, and is able to communicate with other devices on the bus. In a computer network, a node is an individual computer system connected to the network that can communicate with other systems on the network.

nonmemory-like

Regions that may have arbitrary behavior. For example, there may be unimplemented locations or bits anywhere; some locations or bits may be read-only and others write-only, and so on. *See also* memory-like.

NVRAM

Nonvolatile random-access memory. Memory that retains its information in the absence of power such as magnetic tape, drum, or core memory.

The PB22H-KB system module uses battery-backup RAM, not true NVRAM.

octaword

Sixteen contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 through 127.

OpenVMS

Digital's open version of the VMS operating system, which runs on Alpha machines. *See also* open system.

operand

The data or register upon which an operation is performed.

operating system mode

The state in which the system console terminal is under the control of the operating system software. Also called program mode.

page size

A number of bytes, aligned on an address evenly divisible by that number, which a system's hardware treats as a unit for virtual address mapping, sharing, protection, and movement to and from secondary storage.

PAL

Programmable array logic (hardware), a device that can be programmed by a process that blows individual fuses to create a circuit.

PAL code

Alpha AXP privileged architecture library code, written to support Alpha processors. PAL code implements architecturally defined behavior.

parity

A method for checking the accuracy of data by calculating the sum of the number of ones in a piece of binary data. Even parity requires the correct sum to be an even number. Odd parity requires the correct sum to be an odd number.

pipeline

A CPU design technique whereby multiple instructions are simultaneously overlapped in execution.

power-down

The sequence of steps that stops the flow of electricity to a system or its components.

power-up

The sequence of events that starts the flow of electrical current to a system or its components.

prefetch

Refers to read lookahead activity in which system memory modules fetch DRAM data prior to an actual read request for that data.

primary cache

The cache that is the fastest and closest to the processor.

probe

The act of using the current operation address to perform a cache line lookup to determine if the line is valid and whether it must be invalidated, updated, or returned.

program counter

That portion of the CPU that contains the virtual address of the next instruction to be executed. Most current CPUs implement the program counter (PC) as a register. This register may be visible to the programmer through the instruction set.

program mode

See operating system mode.

quadword

Eight contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 through 63.

read-modify-write operation

A hardware operation that involves the reading, modifying, and writing of a piece of data in main memory as a single, uninterruptable operation.

read-write ordering

Refers to the order in which memory on one CPU becomes visible to an execution agent (a different CPU or device within a tightly coupled system).

register

A temporary storage or control location in hardware logic.

RISC

Reduced instruction set computer. A computer with an instruction set that is reduced in complexity.

ROM

Read-only memory

SBZ

Should be zero.

scalability

The ability to add computing and storage resources to an existing system configuration without making software modifications or application conversions, and without shutting down the system.

scratchpad memory

A small memory for holding instructions and data that can be accessed quickly. Similar to cache memory. Also called scratch memory.

SCSI

Small computer system interface. An ANSI-standard interface for connecting disks and other peripheral devices to computer systems.

self-test

A test that is invoked automatically when the system powers-up.

serial port

An external port for serial line devices such as terminals and printers. On the PB22H-KB system module there are two serial line units: the console serial line and the auxiliary serial line. The EIA 232 serial port on the PB22H-KB system module provides asynchronous communication with a device, such as a modem.

SROM

Serial read-only memory. The PB22H-KB system module SROM contains the power-up code that performs diagnostics and loads the console from FEPROM.

superpipelined

Describes a pipelined machine that has a larger number of pipe stages and more complex scheduling and control. *See also* pipeline.

superscalar

Describes a machine that issues multiple independent instructions per clock cycle.

synchronization

A method of controlling access to some shared resource so that predictable, well-defined results are obtained when operating in a multiprocessing environment.

system fatal error

An error that is fatal to the system operation, because the error occurred in the context of a system process or the context of an error cannot be determined.

system module

The main circuit board on which the EISA bus resides enabling connection of adapter cards. The system module is sometimes called the motherboard.

tristate

Refers to a signal line on a bus that has three states: high, low, and high-impedance.

unmasked write

In memory, a write cycle that updates all locations of a nominal data block. That is, a hexaword update to a cache block.

victim

Used in reference to a cache block in the cache of a system bus node. The cache block is valid but is about to be replaced because of a resource conflict in a cache block.

victim processing

The process of replacing the victim in the cache. *See also* victim.

word

Two contiguous bytes (16 bits) starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 through 15.

write back

A cache management technique in which data from a write operation to cache is written into main memory only when the data in cache must be overwritten. This results in temporary inconsistencies between cache and main memory. *Contrast with* write through.

write-enabled

A device is write-enabled when data can be written to it. *Contrast with* write-protected.

write-protected

A device is write-protected when transfers are prevented from writing information to it. *Contrast with* write-enabled.

write through

A cache management technique in which data from a write operation is copied to both cache and main memory. Cache and main memory data is always consistent. *Contrast with* write back.

Index

C

D

E

F

G

H

I

K

L