DECchip 21064 Evaluation Board

User's Guide

Order Number: EC–N0351–72

Revision/Update Information: This is a revised document.

Digital Equipment Corporation
Maynard, Massachusetts

First edition, May 1993
Second edition, June 1993
Third edition, September 1993

Contents

Preface vii

Document Conventions ix

1 Introduction to the EB64

1.1 Components 1–1
1.2 Uses 1–3
1.3 Features 1–4
1.3.1 Memory Subsystem 1–4
1.3.2 Bcache Subsystem 1–4
1.3.3 ISA Interface 1–4
1.3.4 Slow-Speed Peripheral Device Interface 1–5
1.3.5 Ethernet Port 1–5
1.3.6 Debug ROM 1–5
1.3.7 Serial ROM 1–5
1.3.8 Expansion Interface 1–5

2 System Configuration and Connections

3 Functional Description

3.1 Address Space 3–1
3.1.1 Address Bit Description 3–1
3.1.2 Main Memory Address Space 3–2
3.1.2.1 Main Memory, Bcacheable 3–2
3.1.2.2 Main Memory, Not Bcacheable 3–3
3.1.2.3 Main Memory, Not Bcacheable or Data cacheable 3–4

3.1.3 I/O Address Space . . . 3–5
3.1.3.1 I/O Address Space Mapping . . . 3–5
3.1.3.2 I/O Byte Enable and Transfer Length . . . 3–6
3.1.3.3 I/O Space, Debug ROM, and System Register . . . 3–6
3.1.3.4 I/O Space, SC486 and Interrupt Acknowledge . . . 3–7
3.1.3.5 I/O Space, Expansion Connector . . . 3–7
3.2 Memory Subsystem . . . 3–8
3.2.1 Main Memory Subsystem . . . 3–8
3.2.1.1 Row and Column Addressing . . . 3–11
3.2.1.2 Column 0 . . . 3–11
3.2.1.3 Address Bits 18 and 17 . . . 3–12
3.2.1.4 DRAM Address Multiplexing . . . 3–12
3.2.2 Bcache Subsystem . . . 3–14
3.2.2.1 Tag Address Bits <33:32> . . . 3–16
3.2.3 Memory/Bcache Cycles . . . 3–17
3.2.3.1 Hexaword DRAM Read (READ_BLOCK External Access) . . . 3–17
3.2.3.2 Hexaword DRAM Write (WRITE_BLOCK External Access) . . . 3–18
3.2.3.3 Victim Write . . . 3–18
3.2.3.4 Load Locked (LDxL External Access) . . . 3–18
3.2.3.5 Store Conditional (STxC External Access) . . . 3–19
3.2.3.6 Direct Memory Access . . . 3–19
3.2.3.7 Refresh . . . 3–20
3.3 I/O Subsystem . . . 3–20
3.3.1 I/O Devices . . . 3–20
3.3.1.1 System Register . . . 3–20
3.3.1.2 Debug ROM . . . 3–25
3.3.1.3 ISA Interface and Associated Peripheral Devices . . . 3–26
3.3.1.4 Slow Speed Peripheral Devices . . . 3–26
3.3.1.5 Ethernet Link . . . 3–27
3.3.1.6 Interrupts . . . 3–27
3.4 Power, Reset, and Initialization . . . 3–28
3.4.1 Reset . . . 3–28
3.4.2 Initialization and SROM . . . 3–28
3.5 System Clocks . . . 3–30

4 Expansion Interface

4.1 Expansion Connector Signal Description 4–1
4.2 21064 Expansion Interface 4–15
4.2.1 Data Path Width 4–16
4.2.2 Cache Coherency 4–16
4.2.3 Expansion Connector Clocks 4–18
4.2.4 Programmed I/O Through the Expansion Connector 4–19
4.2.5 DMA Through Expansion Connector 4–24
4.2.6 DRAM Address 0 4–25
4.3 System Bus Acquisition and Release Sequence 4–27
4.4 Expansion Connector Pin Lists 4–28

5 Power Requirements

A Technical Support, Ordering, and Associated Literature

A.1 Information and Technical Support A–2
A.2 Ordering DECchip Products A–2
A.3 Associated DECchip Literature A–2
A.4 Associated Digital Literature A–4
A.5 Associated Third-Party Literature A–5

Index

Examples

3–1 BIU_CTL Initialization Code Segment 3–16

Figures

1–1 Block Diagram 1–2
2–1 EB64 Board Jumpers and Connectors 2–2
3–1 Address Bit Definitions 3–1
3–2 Address Path 3–9
3–3 Data Path 3–10
3–4 DRAM Address Multiplexer Definitions 3–13
3–5 Bcache Tag and Index 3–14
3–6 System Register 3–21
3–7 I/O Address Formation 3–22

3–8 SROM Serial Port Diagram 3–30
3–9 System Clock Timing 3–31
3–10 Clock Distribution 3–32
4–1 Signal Name Changes 4–14
4–2 Expansion Interface Signals 4–16
4–3 Expansion Connector Control Setup 4–18
4–4 Programmed I/O Through the Expansion Connector 4–20
4–5 System Bus Acquisition and Release 4–27

Tables

2–1 EB64 Board Jumpers and Connectors 2–3
3–1 Address Space Partitioning 3–2
3–2 Cached and Noncached Memory Quadrants 3–2
3–3 Transfer Length and Byte Enable Decode 3–6
3–4 SC486 Zone Decode 3–7
3–5 Memory Size Selection 3–8
3–6 DRAM SIMM Row/Column Definition 3–11
3–7 Victim Write Demultiplexed Address Bits 18 and 17 Sources 3–12
3–8 RAS Decode for 2- and 8-MB DRAM SIMMs 3–13
3–9 Bcache Size Selection 3–15
3–10 EB64 System Register Bit Description 3–21
3–11 ISA IRQ Input Definitions 3–27
4–1 Expansion Connector Signals 4–2
4–2 Expansion Connector Pin List 4–28
4–3 Expansion Connector Pin List — buf_data<127:0> 4–33
4–4 Expansion Connector Pin List — io_addr<31:2> 4–34
4–5 Expansion Connector Pin List — Vdd and gnd 4–34
5–1 Voltage/Amp 5–1

Preface

This guide describes the DECchip 21064 Evaluation Board (also called the EB64), an evaluation and development module for computing systems based on the DECchip 21064 microprocessor.

Audience

This guide is written for system designers and others who use the EB64 to design or evaluate computer systems based on the DECchip 21064 microprocessor.

Scope

This guide describes the features, configuration, functional operation, and expansion interface of the EB64. Designing hardware to interface to the expansion connector requires most of the same knowledge as a 21064 design and is not included in this guide. Additional information is available in the EB64 schematics and programmable-logic source files, and also in the literature listed in Appendix A. Additional technical support is available from the DECchip Information Line, also included in Appendix A.

Content

This guide contains the following chapters and appendices:

- Chapter 1 is an overview of the EB64.
- Chapter 2 provides EB64 configuration information.
- Chapter 3 is a functional description of the EB64.
- Chapter 4 describes the EB64 expansion interface.
- Chapter 5 describes the EB64 power requirements.
- Appendix A lists technical support services and associated documentation.

Document Conventions

Bit and Field Abbreviations

RO — Read Only

Bits and fields specified as RO can be read but not written.

RW — Read/Write

Bits and fields specified as RW can be read and written.

WO — Write Only

Bits and fields specified as WO can be written but not read.

Bit Notation

Multiple bit fields are shown as extents (see Ranges and Extents, below).

Caution

Cautions indicate potential damage to equipment or data.

Data Units

The following data unit terminology, common within Digital, is used throughout this guide.

Term	Words	Bytes	Bits	Other
Word	1	2	16	
Longword	2	4	32	
Quadword	4	8	64	
Octaword	8	16	128	Single read fill; that is, the cache space that can be filled in a single read access. It takes two read accesses to fill a backup cache line (see Hexaword).
Hexaword	16	32	256	Cache block, cache line. The space allocated to a single backup cache block.

Note

Notes indicate general information.

Numbering

All numbers are decimal or hexadecimal unless otherwise indicated. In cases of ambiguity, a subscript indicates the radix of nondecimal numbers. For example, 19 is decimal, but 19_{16} and 19A are hexadecimal.

Ranges and Extents

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

Extents are specified by a pair of numbers in angle brackets (<>) separated by a colon (:) and are inclusive. For example, bits <7:3> specifies an extent including bits 7, 6, 5, 4, and 3.

Schematic References

Logic schematics are included in the EB64 design package. In this guide, references to schematic pages are printed in italics. For example:

“ . . . multiplexers (schematic page *addr_mux.1*).”

Signal Names

Signal names in text are printed in boldface lowercase. Mixed-case and uppercase conventions sometimes used for signal names are not used in this document. For example:

Used in this Guide	Not Used in this Guide	
cwmask7	cWMask7	CWMASK7

1 Introduction to the EB64

The DECchip 21064 Evaluation Board (also called EB64) is an evaluation and development module for computing systems based on the DECchip 21064 microprocessor (also called 21064). It gives the user a single-board platform for the design, integration, and analysis of supporting logic, subsystems, and software. The EB64 is the first reference design offered for a Digital microprocessor that implements the Alpha AXP architecture. This chapter is an overview of the EB64, its uses, and its features. Figure 1–1 is a block diagram of the system.

Note: Design Concepts

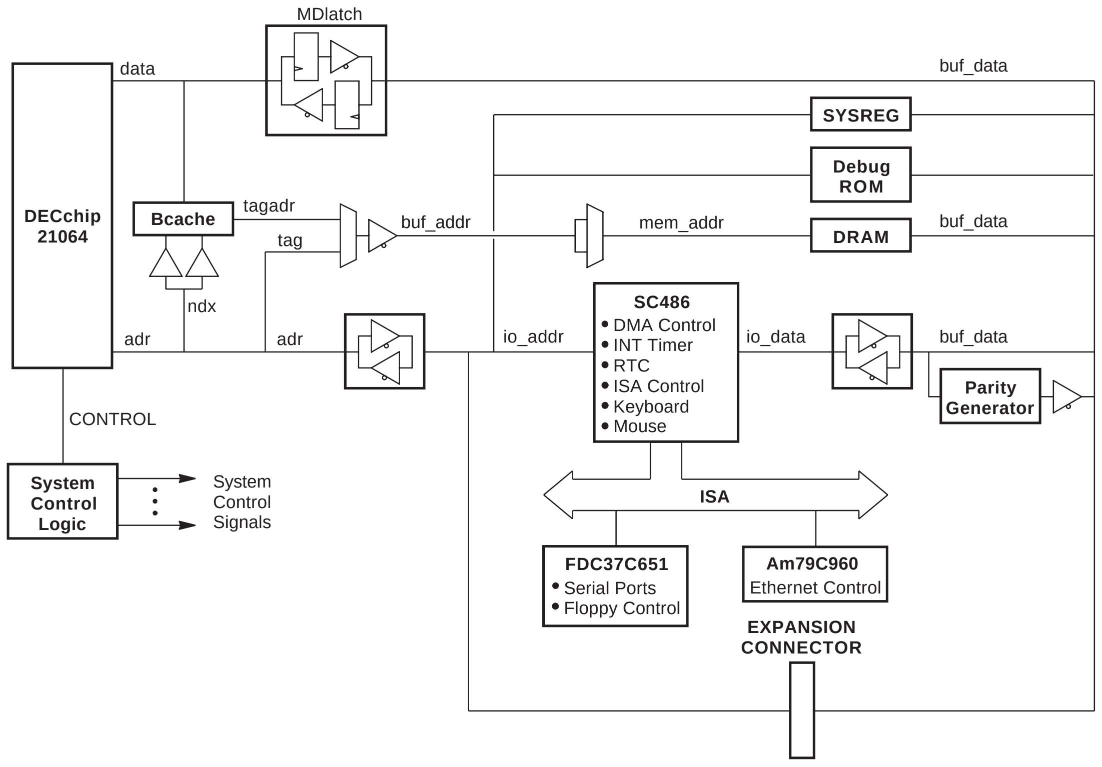
System design concepts are discussed in the application note *Designing a System with the DECchip 21064 Microprocessor.* Memory and backup cache design concepts are discussed in the application note *Designing a Memory/Cache Subsystem for the DECchip 21064 Microprocessor*. If you are not familiar with the 21064, you are encouraged to read both application notes. Appendix A gives ordering information and lists other associated documentation.

1.1 Components

The DECchip 21064 Evaluation Board includes:

- DECchip 21064 running at 150 MHz
- Dynamic RAM (DRAM) main memory subsystem of 4 to 64 Mbytes
- External backup cache (Bcache) subsystem comprising 512 Kbytes of 12 ns static RAMs (SRAMs). The Bcache is read- and write-allocate, write-back, with 32-byte blocks.

Figure 1–1 Block Diagram



WMO_EB64_001

- Serial boot ROM (SROM)
- Two programmable ROMs (PROMs) for debug. One, called the "debug ROM," is preprogrammed by Digital's applications support group; the other is user-programmable.
- ISA (IEEE P996) interface and two ISA connectors
- Embedded serial ports, timers, and Ethernet interface
- Latches, buffers, glue logic, power regulators, oscillators, decoupling capacitors, and so on, as needed to form a complete system
- 21064 cooling-fan sensor in reset/dcok circuitry
- Connector and logic to supply the 21064 signals for user hardware design and debug
- Database and user documentation

The full database, including schematics and source files, are supplied with the EB64. User documentation is also included. The database allows designers with no previous Alpha AXP experience to successfully create a working Alpha AXP system with minimal help.

1.2 Uses

The EB64 has a wide range of uses. The following are a few examples.

System Development

In its released configuration, the EB64 can be used as the basis of a full computing system for which the user must design and integrate the necessary supporting logic. The EB64 works under worst-case voltage, temperature, and process conditions, and can serve as the core of a high-volume product without significant changes.

Software Development

The EB64 has remote debug capability and a software debug monitor for loading code into the system and performing other software debug functions such as memory read, memory write, and instruction breakpoint. When combined with a hardware interface, the debug monitor can be used to write and debug software (for example, device drivers) for workstation and PC-type products, as well as for embedded control products such as laser printers, communication engines (such as bridges and routers), and video products.

Memory and Bcache Subsystem Evaluation

The EB64 is a high-speed design and includes a full 128-bit data path for the DRAM main memory and the Bcache. The 21064 runs at full speed (150 MHz), with power regulation and cooling handled on the board. The EB64 high-performance Bcache SRAM subsystem takes into account signal integrity issues.

The user can select Bcache size and DRAM access time. Such flexibility allows users to change these characteristics and run performance benchmarks to determine the effect on actual programs. The available hardware configurations can also be combined and tested with different coding techniques to determine optimum system performance.

I/O Device Development

The EB64 is a hardware and software platform for developing I/O devices to interface with the 21064. An expansion connector is provided to allow access to a buffered version of the 21064 external interface.[1] The buffered interface

[1] The expansion connector comprises two 80-pin connectors (J10 and J11) and two 100-pin connectors (J26 and J27).

provides sufficient similarity to allow interfaces designed for the EB64 to be easily adapted for direct connection to the 21064.

The expansion interface has a 39.6 ns cycle time, the same speed (25 MHz) as the on-board memory and I/O subsystems. Designers can easily implement appropriate expansion interfaces using commodity programmable devices as needed.

1.3 Features

Sections 1.3.1 through 1.3.8 give a brief overview of the major features of the EB64 system.

1.3.1 Memory Subsystem

The main memory subsystem accommodates 4 to 64 megabytes of DRAM, using four commodity single in-line memory module (SIMM) cards. Each SIMM card is 36 bits wide: 32 data bits, one parity bit, and three unused bits. The possible memory sizes are:

4 MB
8 MB
16 MB
32 MB
64 MB

The default memory subsystem is a high-speed, 128-bit (plus parity) configuration. It can also be configured to emulate a slower (or narrower, 64-bit) memory subsystem. For more information, see Section 3.2.1.

1.3.2 Bcache Subsystem

The 512-Kbyte Bcache uses a combination of commodity 32K×8, 32K×9, and 16K×4 12-ns SRAMs for data, parity, tag address, and tag control. The largest Bcache size is 512 kilobytes, but extra tag address bits are included to configure other Bcache sizes of 256 or 128 kilobytes. The 21064 can be programmed to test SRAMs having lower speeds. Section 3.2.2 gives a detailed description of the Bcache.

1.3.3 ISA Interface

The VLSI Technology SC486 chip (VL82C486 chip with its companion VL82C113A combination I/O chip) provides the following functions:

- ISA bus controller, including bus master
- Direct memory access (DMA) controller
- Interrupt controller

- Programmable and refresh timers
- Real-time clock
- Mouse
- Keyboard

The SC486 also drives the two ISA slots on the EB64 board.

1.3.4 Slow-Speed Peripheral Device Interface

The combo I/O chip is a Standard Microsystems Corporation FDC37C651 Super I/O Floppy Disk Controller. It connects directly to the ISA bus on the EB64 board. It supplies two serial ports and a floppy-disk controller.

1.3.5 Ethernet Port

The AMD Am79C960 PCnet-ISA chip provides an Ethernet link. The chip and its associated glue logic (transformers, level shifters, capacitors, and so on) are connected to the ISA bus on the EB64 board. Both 10BaseT and 10Base2 interfaces are provided. See Section 3.3.1.5 for more information.

1.3.6 Debug ROM

The debug ROM is an industry-standard, 512-kilobyte or 1-megabyte jumper selectable) PROM that contains the debug monitor code. The debug monitor allows the design engineer to develop code on a host system and load the software into the EB64 through the Ethernet port or SROM serial line.

A second PROM socket (jumper selectable) is provided on the board for a user-programmable device. For more information see Section 3.3.1.2.

1.3.7 Serial ROM

The 21064 uses an SROM for its initialization code. When **reset** is deasserted, the contents of the SROM are read into the 21064's instruction cache. The code is then executed from the instruction cache. See Section 3.4.2 for more information.

1.3.8 Expansion Interface

The EB64 board includes a high-speed, 21064 pin-bus expansion interface. This interface gives access to a buffered version of the microprocessor pin bus. It includes the handshake signals needed to perform reads and writes to memory and I/O. This on-board interface ignores the memory zone defined to give system control to an external interface plugged into the connectors. Chapter 4 provides a complete description of the expansion interface.

2 System Configuration and Connections

The EB64 must be configured for the user's environment. The configuration jumpers and connectors (or headers) for user-supplied power and peripheral devices are shown in Figure 2–1 and described in Table 2–1.

After the evaluation board is configured, power can be applied, and the debug monitor can be run. The debug monitor and its commands are described in the *DECchip 21064 Evaluation Board Debug Monitor User's Guide.* For information about other software design tools, see the literature listed in Appendix A.

Figure 2–1 EB64 Board Jumpers and Connectors

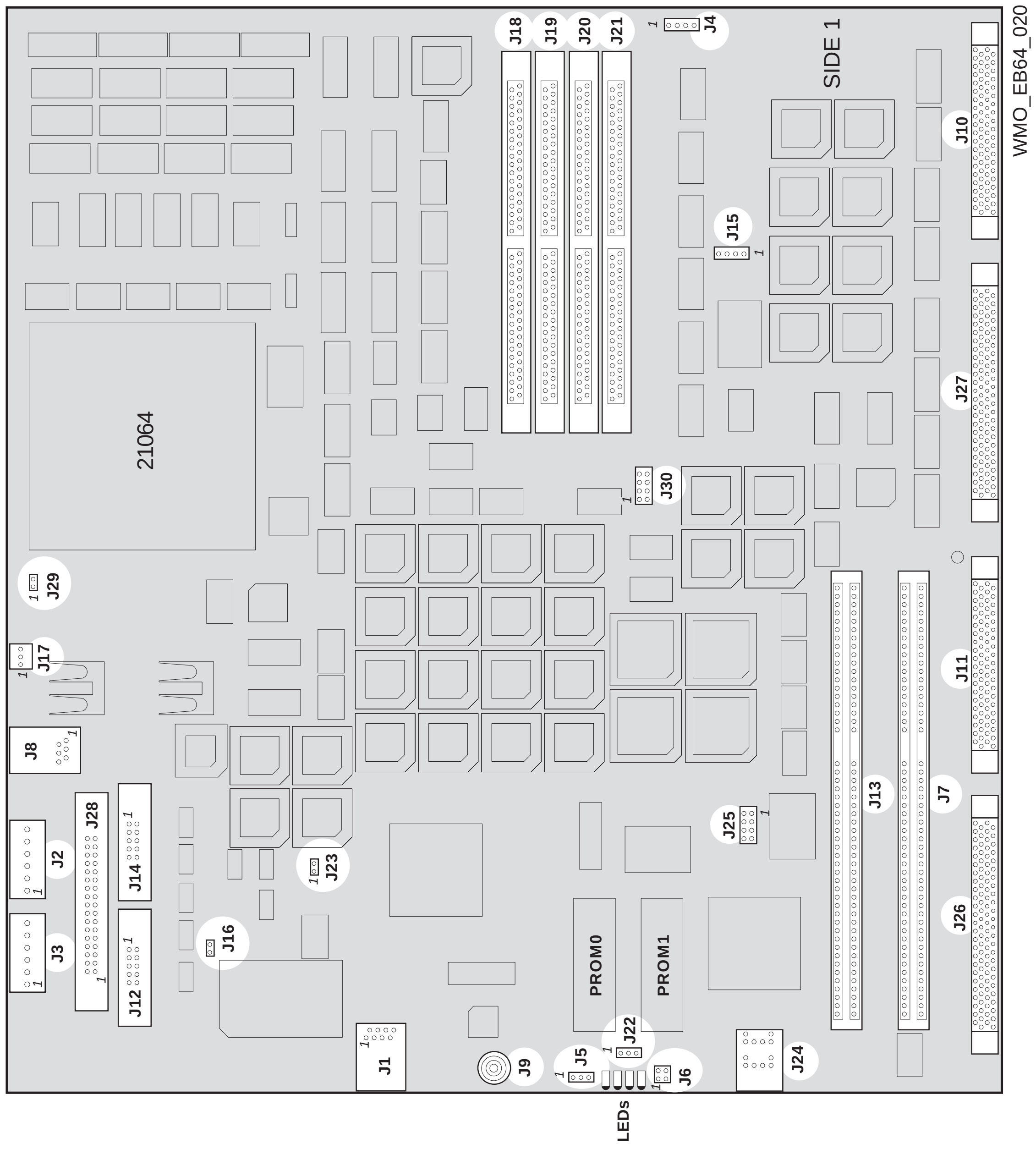


Table 2–1 EB64 Board Jumpers and Connectors

Connector	Pins	Description
J1	8	Ethernet connector, twisted-pair (10BASE-T)
J9	4	Ethernet connector, ThinWire (10BASE-2) **Note: Ethernet LEDs** The four LEDs in Figure 2–1 are driven by the PCnet-ISA chip (Am79C960) and indicate the following: LED0: LINK OK LED1: RCV LED2: RX POL OK LED3: XMIT
J2	6	Board power connector (gnd^{pin1}, gnd, –5 V, Vdd, Vdd, Vdd)
J3	6	Board power connector (p_dcok^{pin1}, Vdd, +12 V, –12 V, gnd, gnd) **Note: Power Supply** Power for the EB64 is provided by a user-supplied, standard, PC power supply. Digital does not provide this power supply. See Section 3.4 for more information.
J4	4	Speaker connector. The speaker should be connected to pins 1 and 4.
J5	3	PROM0 or PROM1 jumper. See Section 3.3.1.2 for more information.
J6	4	Combo chip (FDC37C651) precompensation and drive-type jumpers

J5:

To Select	Set Jumper Pins
PROM0	1 to 2
PROM1	2 to 3

J6:

Signal	Jumper	In	Out
Precompensation	1 to 2	low	high
Drive type	3 to 4	low	high

(continued on next page)

Table 2–1 (Cont.) EB64 Board Jumpers and Connectors

Connector	Pins	Description
J7	98	ISA Slot 1 connector
J13	98	ISA Slot 0 connector
J8	6	SROM serial port connector See Section 1.3.7 for more information.
J9		See J1
J10	80	Expansion interface connector. See Chapter 4 for more information.
J11	80	
J26	100	
J27	100	
J12	10	Combo chip (FDC37C651) serial port 1
J14	10	Combo chip (FDC37C651) serial port 2
J13		See J7
J14		See J12
J15	4	Battery backup connector for time-of-year (TOY) clock
J16*	2	System reset switch connector
J23*	2	dcok reset switch connector See Section 3.4.1 for more information.
J17	3	21064 fan power and sensor connector. **Caution: Fan Sensor Required** The 21064 cooling fan *must* have a built-in sensor that drives a signal if the air flow stops. The sensor is connected to J17. The fan supplied with the EB64 includes an air-flow sensor. See Section 3.4 for more information.
J18	70	DRAM0 SIMM connector
J19	70	DRAM1 SIMM connector
J20	70	DRAM2 SIMM connector
J21	70	DRAM3 SIMM connector All four SIMM connectors must be populated. See Section 3.2.1 for more information.

*The reset switch is connected to J16 or J23.

(continued on next page)

Table 2–1 (Cont.) EB64 Board Jumpers and Connectors

Connector	Pins	Description
J22	3	PROM size jumper. See Section 3.3.1.2 for more information.

To Select	Set Jumper Pins
512-KB PROM	1 to 2
1-MB PROM	2 to 3

Connector	Pins	Description
J23		See J16
J24	12	Keyboard (bottom)/mouse (top) connector
J25	8	System configuration jumpers See Section 3.3.1.1 for more information.

Signal	Jumper:	In	Out	Description
sys_config0	1 to 5	low	high	Boot SROM Mini-Debugger
sys_config1	2 to 6	low	high	1-MB or 16-MB DRAM SIMMs
sys_config2	3 to 7	low	high	Boot an alternate image
sys_config3	4 to 8	low	high	Reserved

Connector	Pins	Description
J26		See J10
J27		See J10
J28	34	Combo chip (FDC37C651) floppy-disk drive 0 and 1 connector
J29	2	Reserved for Digital use

(continued on next page)

Table 2–1 (Cont.) EB64 Board Jumpers and Connectors

Connector	Pins	Description
J30	8	Logic analyzer header See Section 3.3.1.1 for more information.

Signal	Pin	Signal	Pin
sys_output0	1	**gnd**	5
sys_output1	2	**gnd**	6
sys_output2	3	**gnd**	7
sys_output3	4	**gnd**	8

3 Functional Description

This chapter describes the functional operation of the EB64, except for the expansion interface, which is described in Chapter 4.

3.1 Address Space

Sections 3.1.1 through 3.1.3.2 describe EB64 address space partitioning.

3.1.1 Address Bit Description

The 34-bit EB64 address space is partitioned into memory space and I/O space. As shown in Figure 3–1 and Table 3–1, address bit **adr33** determines which space is being addressed. When **adr33** = 0, memory space is addressed, and when **adr33** = 1, I/O space is addressed. Address bits **adr<32:30>** further divide memory and I/O space as shown in Table 3–1.

Figure 3–1 Address Bit Definitions

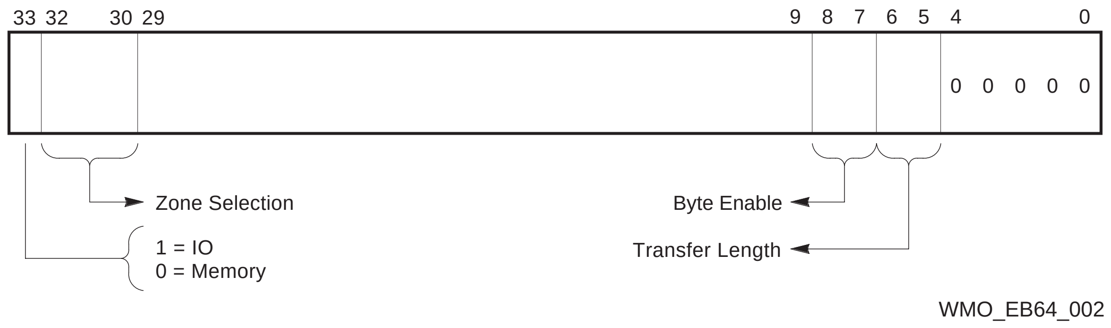


Table 3–1 Address Space Partitioning

Address Bit				
33	32	31	30	Description
0	0	0	0	Main DRAM memory, Bcacheable
0	1	0	0	Main DRAM memory, not Bcacheable, parity not checked*
0	1	1	0	Main DRAM memory, not Bcacheable or data cacheable, parity not checked*
1	0	0	0	I/O space, debug ROM/system register
1	0	0	1	I/O space, interrupt ACK
1	0	1	0	I/O space, SC486 memory
1	0	1	1	I/O space, SC486 I/O
1	1	–	–	I/O Space, expansion connector

*Shadows Bcacheable space.

The address areas are described in Sections 3.1.2 and 3.1.3.5.

3.1.2 Main Memory Address Space

All addresses with **adr33** = 0 address main memory (DRAM) space. Address bits **adr<32:30>** further divide main memory space into the zones described in Sections 3.1.2.1 through 3.1.2.3.

3.1.2.1 Main Memory, Bcacheable

This is the normal local memory area (**adr<33:30>** = 0000), and this address range should be used for most programs. CPU **adr<32:30>** determine whether main memory read or write accesses should go to the Bcache (see Table 3–2). The 21064 probes the Bcache for valid data and starts an external cycle only if there is a cache miss.

Table 3–2 Cached and Noncached Memory Quadrants

Quadrant	adr<33:32>	Cached/Noncached
1	00	Always cached
2	01	Always noncached
3	10	Always noncached
4	11	Cached or noncached

Note: Cached Quadrants Set In BIU_CTL

The BC_PA_DIS field (<35:32>) in the 21064 bus interface unit control (BIU_CTL) internal processor register (IPR) is usually programmed such that only the first memory quadrant (**adr<33:32>** = 00) is Bcached.

Reads

On reads:

1. System logic places return data in the Bcache.
2. Data Bus Read Acknowledge (**drack_h<2:0>**) signals the 21064 to:
   - Store the data in its data cache or instruction cache, according to the type of read.
   - Check the cache fill data for good 32-bit parity.

If the Bcache fill block is valid and dirty, a victim write cycle is performed before the new data is returned. (The victim write cycle is described in Section 3.2.3.3.)

Writes

On writes, data is written into the Bcache under control of the system logic. If the current Bcache block is valid and dirty:

1. It is written back to memory.
2. The DRAM memory is read to fill the background data over which the new data is to be written.
3. After the fill is complete, the new data is written into the Bcache.
4. The Bcache line is updated to be valid and dirty.

3.1.2.2 Main Memory, Not Bcacheable

This memory space (**adr<33:30>** = 0100) shadows the Bcacheable memory space (Section 3.1.2.1) and differs as follows:

- The data cannot be in the Bcache.
- Read data parity is not checked.

In this space, the 21064 does not probe the Bcache on reads and writes (the SROM-resident reset code must set the proper quadrant for this to happen), but goes directly to the expansion interface. Victim writes are never done. Data bus read acknowledge (**drack_h<2:0>**) signals the 21064 to store the data in its data cache or instruction cache, according to the type of read, and fill data is returned to the 21064 without touching the Bcache. Writes bypass the Bcache and go directly to DRAM.

Caution: Do Not Write and Read to Different Memory Space

Do *not* write data to an address in Bcacheable memory space and read that same local DRAM location from space that is not Bcacheable. The Bcached write places the data into the write-back Bcache, where it stays until flushed out for some reason. When you try to read that same location from memory space that is not Bcacheable (**adr32** = 1), the processor bypasses the Bcache and fetches data that may be stale. Because the read appears to be normal, the external control logic does not probe the Bcache for the information.

Caution: Do Not Read and Write to Different Memory Space

Do *not* read data from Bcacheable memory space and write that data to memory space that is not Bcacheable. The read and write will work the first time, but the Bcache will contain stale data. A subsequent read of that data from Bcacheable memory space will hit in the Bcache, and the wrong data will be returned to the CPU.

3.1.2.3 Main Memory, Not Bcacheable or Data cacheable

This address space (**adr<33:30>** = 0110) shadows the Bcacheable address space (Section 3.1.2.1). The 21064 and the system logic bypass the Bcache. Reads and writes are the same as they are for memory space that is not Bcacheable (Section 3.1.2.2), except that on reads, **drack_h<2:0>** signals the 21064 not to load the data into its internal data cache. The system logic can return a single read fill in this address space. A single read fill (128 bytes) usually occurs when the requested data is a byte or longword.

Caution: Not for Instruction Fetch

Instruction stream (Istream) reads to this address range do *not* work, because the 21064 does not load the data into its instruction cache. The processor will malfunction if this address range is used to fetch instructions.

3.1.3 I/O Address Space

All addresses with **adr33** = 1 address I/O space. Address bits **adr<32:30>** further divide I/O space into the zones described in Sections 3.1.3.3 through 3.1.3.5.

3.1.3.1 I/O Address Space Mapping

Address mapping between the 21064 address bus (**adr<33:5>**) and the I/O address (**io_addr<31:2>**[2]) changes, depending on whether the 21064 or a DMA device is bus master. DMA devices on the EB64 are the SC486 and, potentially, circuitry connected to the expansion connector.

Bus Master is 21064

When the 21064 is bus master, the address lines are mapped as follows:

sys_ioadr<31:23>	⇒ **io_addr<31:23>**
adr<29:9>	⇒ **io_addr<22:2>**

In addition to this mapping, byte enables **io_be<3:0>** are decoded from **adr<8:5>** as described in Section 3.1.3.2 to facilitate byte addressing on the I/O bus. The technique of using address lines to decode byte information is known as "sparse address mapping."

A consequence of sparse address mapping is that memory attached to the expansion connector cannot be addressed as a contiguous block (because 21064 **adr<8:5>** signals are not available). To compensate for this, byte enable **ext_be<1:0>** and transfer length **ext_tl<1:0>** signals are available on the expansion connector. They correspond to **buf_adr<8:7>** and **buf_adr<6:5>** as follows:

buf_adr8	⇒ **ext_be1**
buf_adr7	⇒ **ext_be0**
buf_adr6	⇒ **ext_tl1**
buf_adr5	⇒ **ext_tl0**

Bus Master is DMA Device

When a DMA device takes control of the **io_addr** bus and drives an address into the Bcache and DRAM array, the address is not shifted, and the address lines are mapped as follows:

io_addr<25:5>	⇒ **adr<25:5>**

[2] From the system register. See Section 3.3.1.1

Lower address resolution than this is provided by **sel_col2** and **sel_col_a0**, which allow individual 16-byte octawords to be addressed (behaving as **adr4**). Longword resolution is provided on writes by the appropriate Bcache write enable (**we**) and DRAM CAS signals.

3.1.3.2 I/O Byte Enable and Transfer Length

Table 3–3 shows how address bits **adr<8:5>** are encoded. The transfer length for commands directed at I/O space is selected according to **adr<6:5>**. The byte enables asserted for such transfers are determined according to **adr<8:7>**.

Table 3–3 Transfer Length and Byte Enable Decode

Byte Enable adr<8:7>	Transfer Length adr<6:5>	Transfer Length	BE#*	Address Adder†
00	00	Byte access	1110	000_{16}
01	00	Byte access	1101	080_{16}
10	00	Byte access	1011	100_{16}
11	00	Byte access	0111	180_{16}
00	01	Word access	1100	020_{16}
01	01	Word access	1001	0A0
10	01	Word access	0011	120_{16}
11	01	Reserved	–	–
00	10	Tribyte access	1000	040_{16}
01	10	Tribyte access	0001	0C0
1x	10	Reserved	–	–
10	11	Longword access	0000	060_{16}
01	11	Reserved	–	–
1x	11	Reserved	–	–

*Byte enable is asserted low (0).

†Adding this hexadecimal value to the base address asserts the corresponding BE# code.

3.1.3.3 I/O Space, Debug ROM, and System Register

The debug ROM and system register share this address zone (**adr<33:30>** = 1000).

Reads

On read accesses, data bits <7:0> return the contents of the debug ROM location addressed by **io_addr<21:2>**. This address is translated from **adr<28:9>**. The upper data bits come from the system register. The system register returns the same data for any ROM address that is accessed.

Writes

On write accesses to this address zone, the system register is written with the data contained in the upper data bits defined for the system register (see Section 3.3.1.1). The read and write locations of control bits <25:8> remain the same; therefore, a routine can read the current value, change the required bits, and write it back without shifting the data around.

3.1.3.4 I/O Space, SC486 and Interrupt Acknowledge

Address zones **adr<33:30>** = 1001, 1010, and 1011 all access the SC486 device, using the standard host handshake and control signals: **ads#**, **rdy#**, **m/io#**, **d/c#**, and **w/r#**. The interrupt acknowledge cycle is a read cycle. The SC486 memory and I/O cycles can be read or write cycles. Table 3–4 shows the results of each zone access on the SC486 host bus interface.

Table 3–4 SC486 Zone Decode

adr<33:30>	Command Type	m/io#	d/c#	w/r#	SC486 Transfer Type
1001	Read	0	0	0	interrupt acknowledge
1010	Read	1	1	0	SC486 memory read
1010	Write	1	1	1	SC486 memory write
1011	Read	0	1	0	SC486 I/O read
1011	Read	0	1	1	SC486 I/O read

3.1.3.5 I/O Space, Expansion Connector

This address zone (**adr<33:30>** = 11xx) is dedicated to the expansion interface connector. The on-board decode logic does not recognize or act on addresses in this zone except for barrier instructions, which it will terminate with a cycle acknowledge (**cack<2:0>**). User-supplied logic is expected to return the appropriate handshake signals. The expansion connector signals are described in Chapter 4.

Caution: No External Logic

The system will hang if there is no external logic and an address in this zone is accessed. There is no timeout.

3.2 Memory Subsystem

Sections 3.2.1 through 3.2.3.7 describe the DRAM main memory subsystem, Bcache subsystem, and memory cycles. Figure 3–2 shows the memory/Bcache interface address path and Figure 3–3 shows the data path.

3.2.1 Main Memory Subsystem

The main memory subsystem is a high-speed, 128-bit (plus parity) configuration with optional read-data wrapping (see the system register Wrap Read bit in Section 3.3.1.1). The memory subsystem can also be configured to emulate a slower (or narrower, 64-bit) memory subsystem by programming read slip cycles in system register bits <20:19>.

The EB64 supports DRAM main memory sizes from 4 through 64 megabytes. Buffered CPU address bits **adr<25:5>** are delivered directly to the memory subsystem. The data bus width is 128 bits. Each DRAM SIMM is 36 bits wide (32 data bits, one parity bit, and three unused bits), and provides parity protection on each longword (four bytes). The memory is longword writable, so that read/modify/write cycles are *not* required for hexaword (32-byte) transfers that are not fully masked.

The EB64 accommodates several DRAM SIMM sizes in four industry-standard DRAM SIMM sockets (J18, J19, J20, and J21, Figure 2–1 and Table 2–1). To fill the 128-bit (plus parity) DRAM data path width, all four sockets must be populated. Memory size is specified in the system register as shown in Table 3–5.

Table 3–5 Memory Size Selection

System Register <24:22>	DRAM Size	SIMM Type	Main Memory
000	1-MB	256K×36	4 MB
001	2-MB	512K×36	8 MB
010	4-MB	1MB×36	16 MB
011	8-MB	2MB×36	32 MB
100	16-MB	4MB×36	64 MB

Figure 3–2 Address Path

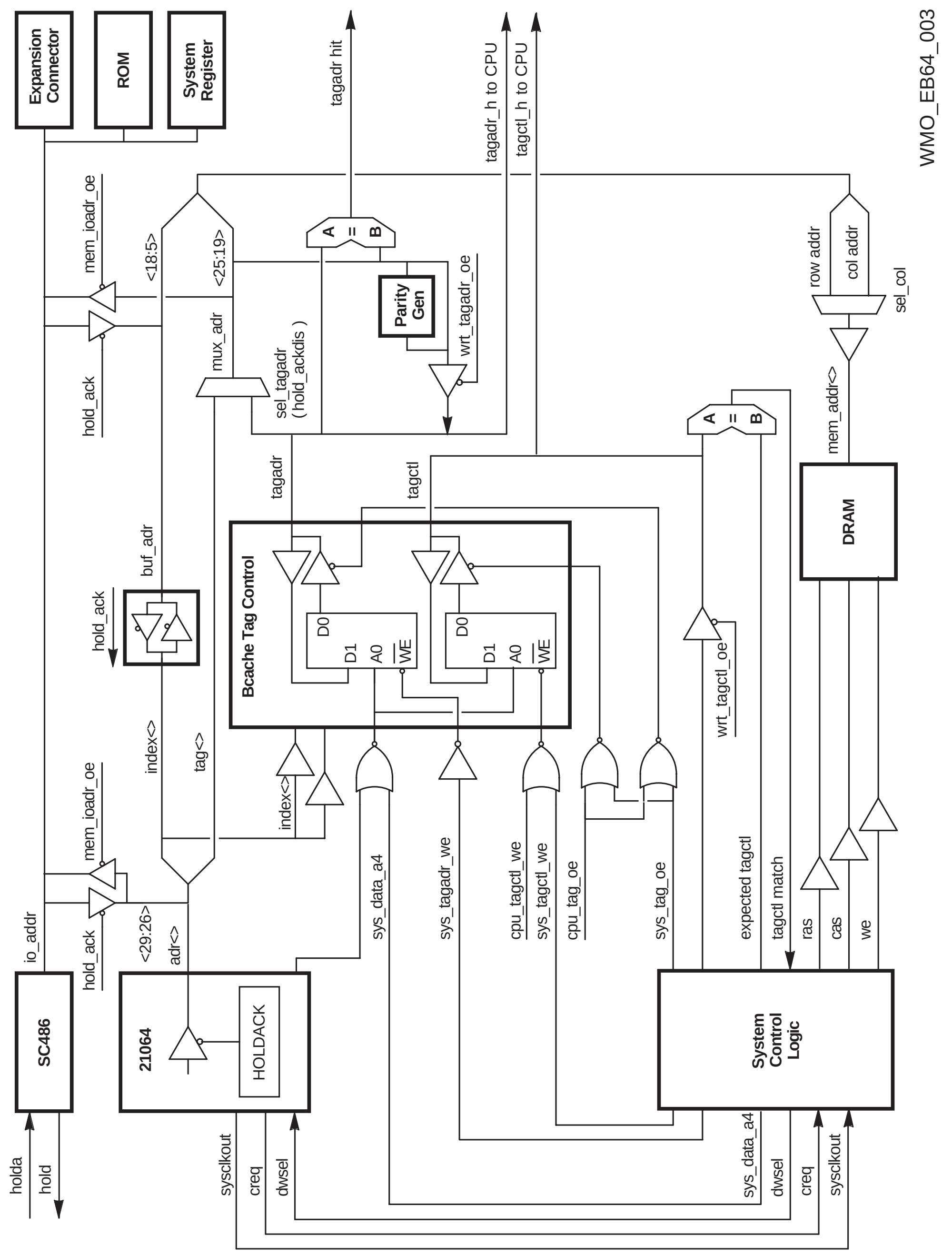
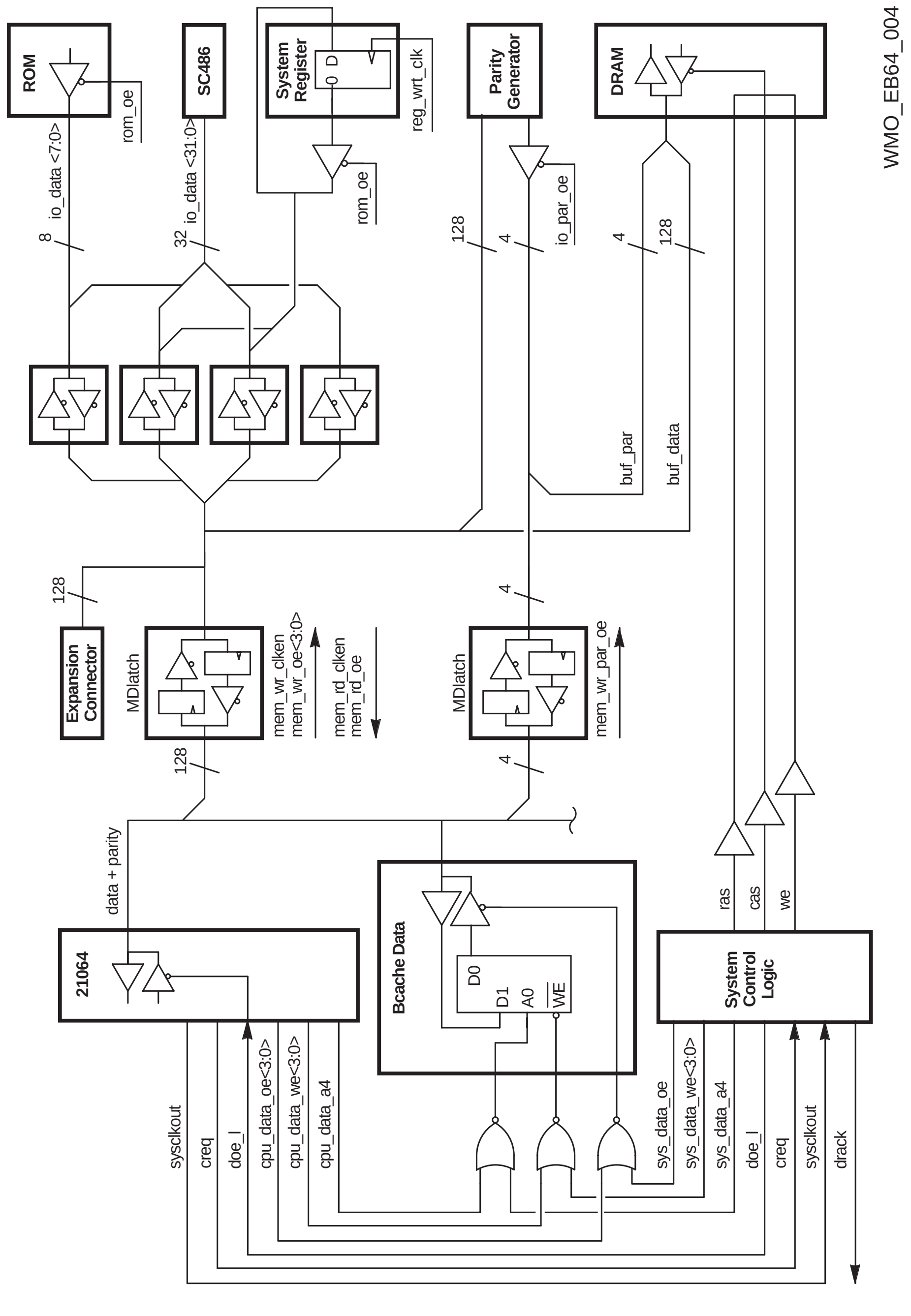


Figure 3–3 Data Path



3.2.1.1 Row and Column Addressing

The system logic multiplexes the DRAM SIMM addresses to present the row and column addresses at the proper time. The address multiplex definitions do not change with SIMM size; SIMMs that do not have a particular input address bit ignore the upper address lines. Table 3–6 shows the row and column definitions.

Table 3–6 DRAM SIMM Row/Column Definition

DRAM Address	Demultiplexed System Address	Used For DRAM SIMM Size
col<8:1>[*]	**adr<12:5>**	All
row<8:0>	**adr<21:13>**[†]	All
col9	**adr22**	4-, 8-, and 16-MB
row9	**adr23**	4-, 8-, and 16-MB
col10	**adr24**	16-MB only
row10	**adr25**	16-MB only

[*]See Section 3.2.1.2, below.

[†]See Section 3.2.1.3, below.

3.2.1.2 Column 0

Because the read and write data width between the 21064 and main memory is 128 bits, two complete access cycles are required to transfer a full hexaword (32 bytes) of data. This transfer is performed using DRAM page-mode access, changing the least significant column address bit between the first and second 128-bit transfer. The column 0 address is formed differently on reads and writes.

- On writes, the column 0 address is always low (0) for the lower octaword (128-bit) transfer, and is then incremented to 1 to write the upper octaword.
- On reads, the column 0 address depends upon the system data wrapping mode. If read data wrapping is:

– Disabled, the column 0 address behaves as it does for a write (low, then high).
– Enabled (system register bit 21 = 1), the column 0 address is set to the value in **cwmask1** for the first octaword (**cwmask1** represents the internal value of **adr4**). The column 0 address is then complemented for the second octaword.

Caution: Setting Data Wrapping Mode

The same wrapping behavior must be set in the 21064 BIU_CTL IPR (SYS_WRAP bit) and in the EB64 system register (Wrap Read bit, Section 3.3.1.1).

3.2.1.3 Address Bits 18 and 17

During victim write cycles, the source of demultiplexed address bits 18 and 17, which become **dram_addr<5:4>**, depends on Bcache size. As Table 3–7 shows, the source is either the 21064 (**adr_h<18:17>**) or the Bcache tag address SRAMs (**tagadr<18:17>**).

Table 3–7 Victim Write Demultiplexed Address Bits 18 and 17 Sources

Bcache Size	Demultiplexed 18	Demultiplexed 17
128 KB	**tagadr18**	**tagadr17**
256 KB	**tagadr18**	**adr_h17**
512 KB	**adr_h18**	**adr_h17**

3.2.1.4 DRAM Address Multiplexing

The EB64 address-decode logic generates two row-address strobe (RAS) signals. With the 1-, 4-, or 16-MB SIMMs, a single RAS signal controls the entire SIMM. With 2-MB or 8-MB SIMMs, a second RAS signal is used to further decode the accessed DRAMs. Figure 3–4 shows DRAM address multiplexing and Table 3–8 shows the decode definitions for each RAS.

Figure 3–4 DRAM Address Multiplexer Definitions

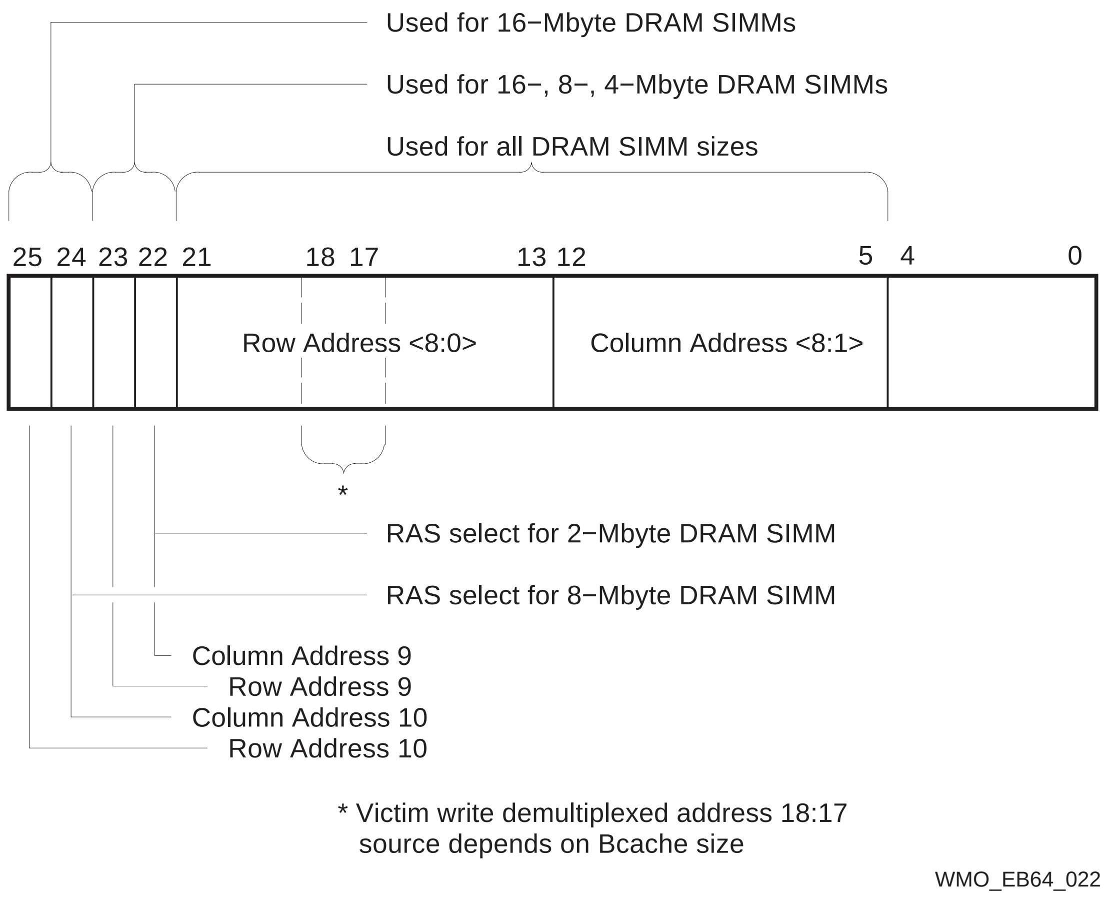

Used for 16−Mbyte DRAM SIMMs
Used for 16−, 8−, 4−Mbyte DRAM SIMMs
Used for all DRAM SIMM sizes
25 24 23 22 21 18 17 13 12 5 4 0
Row Address <8:0>
Column Address <8:1>
*
RAS select for 2−Mbyte DRAM SIMM
RAS select for 8−Mbyte DRAM SIMM
Column Address 9
Row Address 9
Column Address 10
Row Address 10
* Victim write demultiplexed address 18:17 source depends on Bcache size
WMO_EB64_022

Table 3–8 RAS Decode for 2- and 8-MB DRAM SIMMs

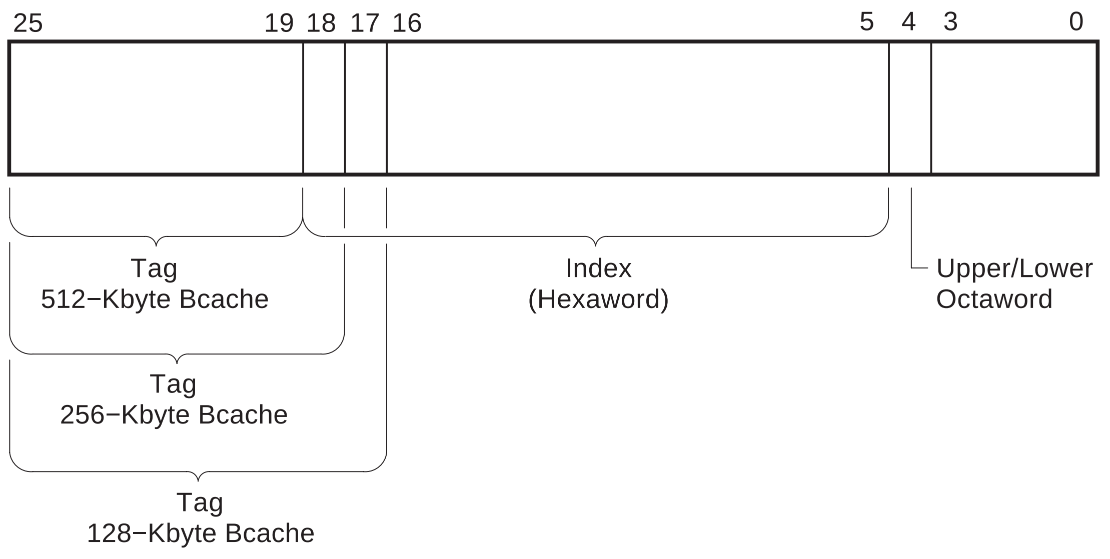
RAS	Memory Type	adr24	adr22
0	2-MB	x	0
	8-MB	0	x
1	2-MB	x	1
	8-MB	1	x

3.2.2 Bcache Subsystem

The 21064 supports Bcache sizes from 128 kilobytes through 16 megabytes, and the EB64 supports Bcache sizes of 128, 256, and 512 kilobytes. The EB64 Bcache uses a combination of commodity 32K×8 and 32K×9 12-ns SRAMs for data. The 32K×9 SRAMs accommodate longword parity (one parity bit per longword). Tag address and tag control SRAMs are commodity 12-ns 16K×4 devices.

The largest Bcache size is 512 kilobytes, and other sizes can be configured through program control (see Figure 3–5 and Table 3–9). The tag address accommodates enough address bits for a 64-megabyte main memory and a 256- or 128-kilobyte Bcache; 128 kilobytes is the smallest Bcache supported by the 21064. The Bcache data path is 128 bits wide.

Figure 3–5 Bcache Tag and Index



WMO_EB64_016

Table 3–9 Bcache Size Selection

System Register <18:17>	Bcache Size	Tag Address Bits
00	128 KB	<25:17>
01	256 KB	<25:18>
10	512 KB	<25:19>
11	Undefined	

The physical EB64 Bcache is always 512 kilobytes. No physical changes are required to test a smaller Bcache. When a 128-kilobyte or 256-kilobyte Bcache size is specified, part of the 512-kilobyte Bcache is ignored. On read-fill operations, the entire tag field is written (as if the 128-kilobyte Bcache size is specified), but the tag address parity includes only the address bits appropriate for the Bcache size selected. For example, when the 512-kilobyte Bcache is selected, tag address bits <18:17> are ignored.

Note: Setting Bcache Size

To configure the EB64 for a smaller Bcache size, the EB64 system register and the 21064 must be programmed for the smaller value. The *DECchip 21064 Microprocessor Hardware Reference Manual* explains how to program the CPU for Bcache size. The EB64 system register (Section 3.3.1.1) specifies a smaller Bcache size in the EB64 as shown in Table 3–9.

The EB64 uses 12 ns data and tag SRAMs, allowing the 21064 BIU_CTL IPR (BC_RD_SPD and BC_WR_SPD fields) to be programmed for 5/5 read/write cycle times. That is, each 128-bit access, including the tag probe, can be programmed for five CPU internal cycles (6.6 ns). It takes ten cycles to read an entire Bcache line — two 16-byte reads with the tag probe hidden under the first octaword (16 bytes) read. A write to the Bcache consists of a five-cycle tag probe and one or two Bcache write accesses, depending on which longwords need to be modified. The write pulse should be centered in each five-cycle write access. The 21064 can be programmed to test SRAMs having slower speeds for performance comparisons.

Example 3–1 is a code segment that can be used as a starting point when programming the 21064 BIU_CTL IPR.

Example 3–1 BIU_CTL Initialization Code Segment

```
# biuCtl = 0x0E20006335
#       what    bits    value
#       ------- ------  -----
#       bc_en   0       1       Enable Bcache
#       ecc     1       0       Disable ECC
#       oe      2       1       output_enable, enabled
#       bc_fhit 3       0       disable Bcache force hits
#       bcRdSpd 7..4    4       5 cycle read
#       bcWrSpd 11..8   4       5 cycle write
#       unused  12      0
#       bcWeCtl 27..13  7       3 cycle write pulse; 2nd, 3rd, and 4th
#                               cycles (000000000000111)
#       bc_size 30..28  010     Bcache size 512KB
#       bad_tp  31      0       Disable, force write bad tag cntl parity
#       bcPaDis 35..32  1110    Cache only 1st quad of physical address
#       bad_dp  36      0       Disable, force write bad ECC/parity
#       unused  63..37  0

lda     r20, 0xE(r31)           # biu_ctl[47:32] = 0xE
sll     r20, 32, r20            # Shift into position
ldah    r20, 0x2000+1(r20)      # biu_ctl[31:16]=0x2001
lda     r20, 0xE445(r20)        # biu_ctl[15:0]=0xE445
mtpr    r20, biuCtl             # Write to biuCtl
```

Caution: Set BIU_CTL OE to Avoid Component Damage

The 21064 BIU_CTL IPR output enable bit (OE, bit 2) *must* be set on the EB64 platform. If this bit is inadvertently cleared, the tag and data SRAMs will be enabled during writes, resulting in damage.

3.2.2.1 Tag Address Bits <33:32>

Tag address bit 33 is stored in the Bcache tag address SRAMs and is driven to the 21064 **tagadr_h<33:32>** inputs for the tag compare. This allows a custom device plugged into the expansion connector to have some of its data in the Bcache (see Chapter 4) . This can be useful for a device such as a video frame buffer, which manipulates data at high speed before writing it out to frame memory on the expansion connector.

Tag address bits 33 and 32 are either both zero or both one, and require only a single bit (**tagadr33**) in the tag address SRAMs. For normal memory traffic, **tagadr33** is zero, and memory operation is not affected. To address a device connected to the expansion connector, **adr<33:32>** = 11, and that value is driven into the Bcache tag address SRAMs when the Bcache is loaded. (The shared [S] bit must be set.) Because parity is the same for **tagadr<33:32>** = 00 or 11, parity generation does not change to support this feature.

The shared bit allows software for devices on the expansion connector to read and manipulate data at high speed, but prevents the data from being written to (and deferred). If the shared bit is not set, the write-back Bcache saves the data (until it is flushed out for some reason), which is not useful for a video frame buffer. Tag address bit 33 also goes to the system-level tag compare logic, so that expansion connector data is not aliased with normal memory data.

Note: Set Cacheable Quadrants In BIU_CTL

When the system is used in this mode, the BC_PA_DIS field in the 21064 BIU_CTL IPR should be programmed to set the lowest and highest memory quadrants (**adr<33:32>** = 00 and 11) as cacheable.

3.2.3 Memory/Bcache Cycles

The EB64 memory control interface contains the state machine and glue logic needed for turning 21064 bus signals into DRAM/Bcache cycles. The supported cycles are described in Sections 3.2.3.1 through 3.2.3.6.

3.2.3.1 Hexaword DRAM Read (READ_BLOCK External Access)

Cached and noncached hexaword DRAM reads are supported. The system logic returns the data to the 21064, and, if the data is in the cached area, simultaneously loads it into the Bcache.

The memory interface supports wrapped reads, where the requested octaword (16 bytes) is returned first. Nonwrapped reads always return the low octaword first, regardless of the address.

Caution: Setting Data Wrapping Mode

The same wrapping behavior must be set in the 21064 BIU_CTL IPR (SYS_WRAP bit) and in the EB64 system register (Wrap Read bit, Section 3.3.1.1).

3.2.3.2 Hexaword DRAM Write (WRITE_BLOCK External Access)

Cached and noncached hexaword DRAM writes are supported. The 21064 write data is accepted and loaded into memory on writes to a noncached area or loaded into the Bcache on writes to the cached area. When the Bcache is the final destination, a victim write is performed, if necessary, and the memory block is loaded into the Bcache, overwriting the modified longwords. (The victim write cycle is described in Section 3.2.3.3.)

3.2.3.3 Victim Write

A victim write cycle is generated by the system logic before a read or write command overwrites a Bcache block that is valid and dirty. The contents of the Bcache block are written to memory before the new data is loaded from memory.

A victim write begins with a probe of the tag control bits, checking the valid (V) and dirty (D) bits on the Bcache block. If both are true, the Bcache block is written back to memory. If either is false, a victim write cycle does not begin and the read or write access proceeds normally.

3.2.3.4 Load Locked (LDxL External Access)

The load locked cycle is similar to a READ_BLOCK access (Section 3.2.3.1). System logic first probes the Bcache for the requested data, because the 21064 does not probe the Bcache on a LDxL command.

- If the data is in the Bcache, it is returned from there; otherwise, the data is returned from memory.
- If the data is returned from cacheable memory space, it is also loaded into the Bcache.

In either case, the lock flag is unconditionally set. If a victim write is necessary, it is performed before the read access.

Because the EB64 is expected to be used as a single-processor evaluation and development vehicle, lock contention is unlikely and the lock flag has no associated address information. The lock is set on the load_locked instruction (LDxL command), and cleared by a store_conditional instruction (STxC command) or a DMA write (unconditional clear, not based on hit).

3.2.3.5 Store Conditional (STxC External Access)

An STxC command on the 21064 external command bus generates this cycle. First, the lock flag is checked. If it is clear, the store_conditional instruction fails and a write is not performed. Command acknowledge (**cack_h<2:0>**) identifies the failed transaction.

If the lock flag is set, then a write command is performed. The Bcache is checked to see if it contains the data, and one of the following occurs:

- If the data is in the Bcache, then the new data overwrites the data in the Bcache and the block is marked dirty.
- If the data is not in the Bcache and the new data is noncacheable, then the new data is written to a noncached memory area.
- If the data is not in the Bcache and the new data is cacheable, then the Bcache is loaded from memory and is then overwritten with the new data. If required, a victim write (Section 3.2.3.3) is performed first.

The lock flag is cleared whether the transaction succeeds or fails.

3.2.3.6 Direct Memory Access

Direct memory access (DMA) cycles transfer data between an external bus and local DRAM or Bcache memory. The EB64 provides the ISA bus as the external bus. Both DMA and master mode ISA transfers are possible. Because the ISA bus is only 16 data bits wide, each data transfer is limited to no more than two bytes (one word) per access. EB64 control logic provides the necessary Bcache probe and write data merge functions.

A DMA access is initiated when the SC486 senses an enabled request from an ISA bus device. When it is idle, the EB64 control logic recognizes the DMA request (**hold**) and responds with an acknowledge (**hold_a**). The acknowledge is returned to the ISA device, and it then takes control of the bus and starts the transfer. When the EB64 senses the transfer start, it probes the Bcache. If the requested data word is currently valid in the Bcache and the transaction is a:

- Write, the EB64 modifies the data in the Bcache and sets the block dirty. It also invalidates the internal data cache.
- Read, the EB64 reads the data from the Bcache.

If the data is not valid in the Bcache, the transfer accesses main memory.

On write cycles, the current Bcache or DRAM data is read and saved, then merged with the new data from the SC486 according to the byte enable signals. The lock flag is cleared for DMA writes to either the Bcache or main memory.

The EB64 can read, write, and merge data from any longword (four bytes) in the address space. It correctly updates the new longword parity on the merged write data in the Bcache and main memory.

3.2.3.7 Refresh

The memory state machine performs memory refresh cycles according to a timer. A CAS-before-RAS refresh is performed on the entire DRAM array.

3.3 I/O Subsystem

The I/O subsystem includes the I/O state machine, glue logic, and the components described in Section 3.3.1.

3.3.1 I/O Devices

There are several input/output devices embedded in the EB64. The SC486, debug ROM, and system register share the same state machine. When the debug ROM or system register are accessed, signal **lba** is asserted low to inhibit the SC486 from responding.[3] This decision is based upon the address zone, as defined in Section 3.1.

3.3.1.1 System Register

The EB64 has one system register. Its bits are defined in Figure 3–6 and described in Table 3–10. The register is addressed as described in Table 3–1. The system register is a combination of a control register and the debug ROM.

Write

When the system register is written, register bits <25:8> are written with **data<25:8>** (data mapping is 1:1), and the register low byte, <7:0>, is ignored. Register bits <29:26> are written to system output registers, where they can be accessed with a logic analyzer connected to J30 (Figure 2–1 and Table 2–1). Bits <29:26> are not readable.

Read

When the system register is read, the bits return the following:

- <7:0> — The low byte returns the data from the debug ROM location selected by the lower address bits, **adr<28:9>** (Figure 3–1). These address bits translate to I/O address bits **io_addr<21:2>**.
- <25:8> — These bits return the control fields, irrespective of lower address bits **adr<28:9>**. The control fields are written into the same data bit positions from which they are read.

[3] For more information about signal **lba**, see SC486 documentation.

- <31:26> — These bits return system configuration information and are not writeable.

Figure 3–6 System Register

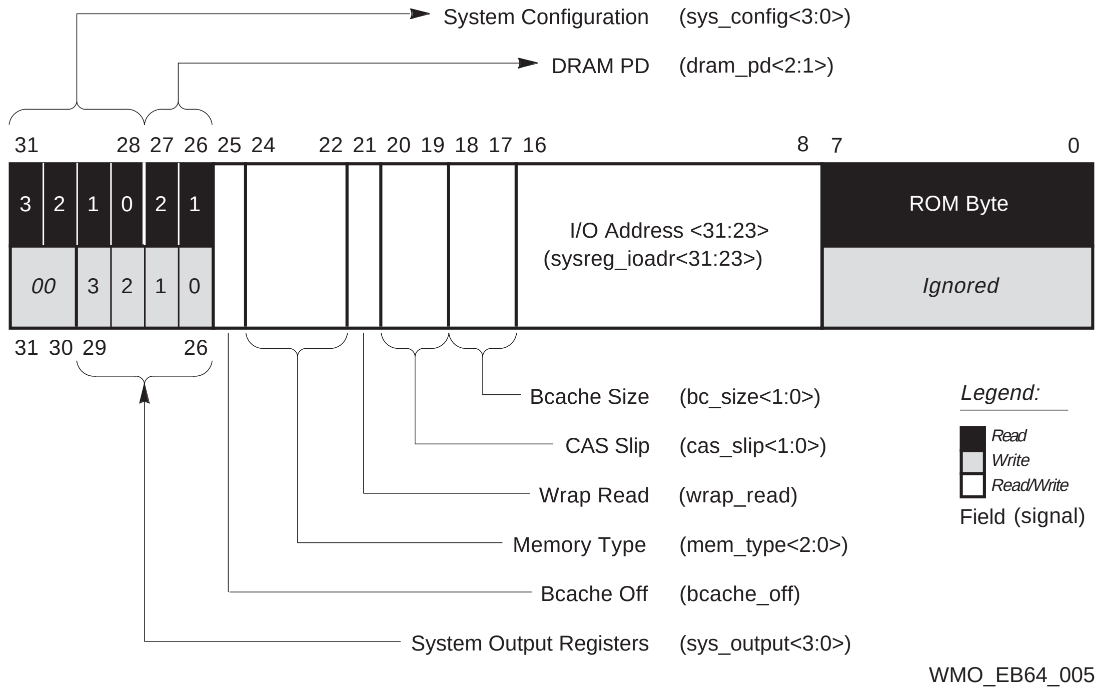


Table 3–10 describes the register fields.

Table 3–10 EB64 System Register Bit Description

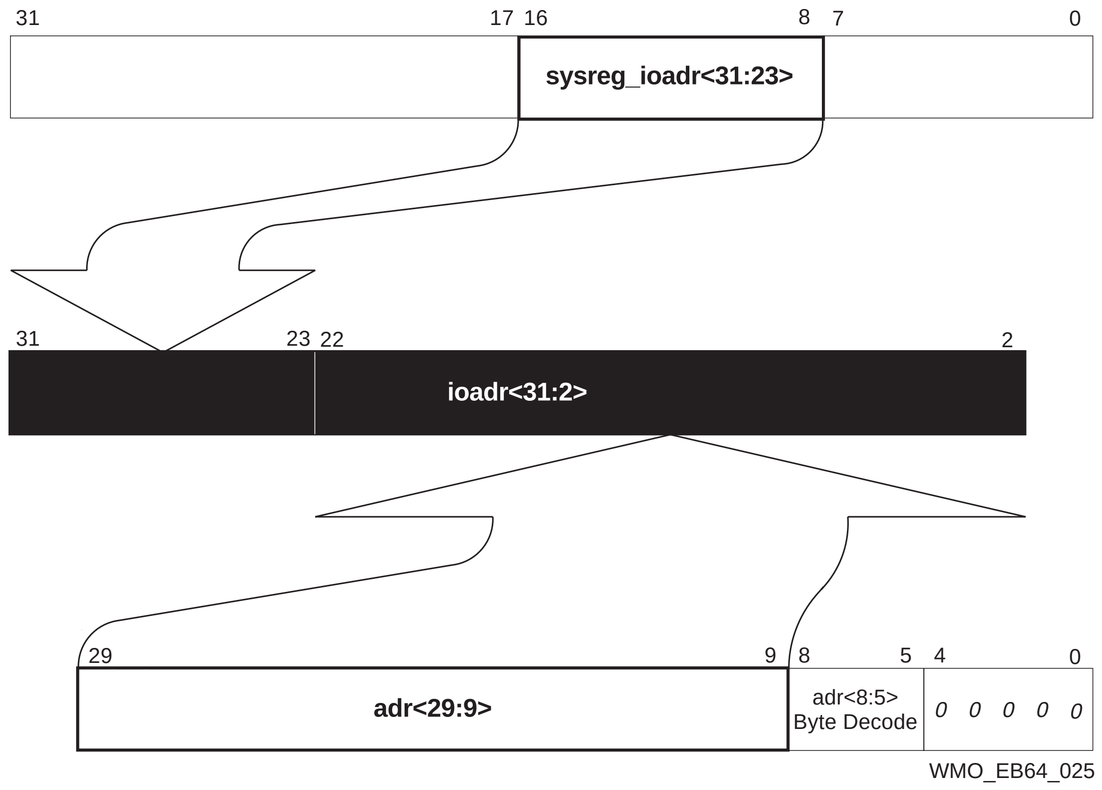
Bits	Access	Description
7:0	RO	ROM Byte — On read cycles, this field returns the data byte residing at the debug ROM location selected by address bits **adr<28:9>**. Address bits **adr29** and **adr<8:0>** should be zero for this access. This field is not used on write cycles.

(continued on next page)

Table 3–10 (Cont.) EB64 System Register Bit Description

Bits	Access	Description
16:8	RW	I/O Address <31:23> — This field defines address bits <31:23> in I/O space. It is concatenated with **io_addr<22:2>** to form the entire I/O address. I/O address bits **io_addr<22:2>** are controlled by address bits **adr<29:9>** from the processor. See Figure 3–7.

Figure 3–7 I/O Address Formation



(continued on next page)

Table 3–10 (Cont.) EB64 System Register Bit Description

Bits	Access	Description
18:17	RW	Bcache Size — This field determines the EB64 Bcache size selection, as follows:

18	17	Bcache Size
0	0	128 KB
0	1	256 KB
1	0	512 KB
1	1	Undefined

Note: Setting Bcache Size

The Bcache size selected by this field must match the Bcache size selected in the BC_SIZE field in 21064 BIU_CTL IPR.

See Section 3.2.2 for more information on the Bcache subsystem.

Bits	Access	Description
20:19	RW	CAS Slip — The memory subsystem is designed to run as fast as possible using 70 ns commodity DRAM SIMMs and a 39.6 ns state machine granularity. To emulate slower devices or narrower data paths (translating to longer latency), slip cycles can be programmed into the read cycles. There are two CAS accesses per read fill, and each CAS is governed by the number of extra cycles in this field. This field programs the number of 39.6 ns slip cycles for each CAS read access, as follows:

20	19	CAS Slip Cycles
0	0	0
0	1	1
1	0	2
1	1	3

(continued on next page)

Table 3–10 (Cont.) EB64 System Register Bit Description

Bits	Access	Description
21	RW	Wrap Read — On read fill operations, the EB64 does two 16-byte DRAM reads to return a 32-byte hexaword. When this bit is clear, read data wrapping is disabled; that is, the low 16 bytes are always returned first, regardless of the request address. When this bit is set, read data wrapping is enabled and the requested 16 bytes are returned first, according to address bit **adr4** (translated to **cwmask1** during read). Write operations are not affected. **Caution: Setting Data Wrapping Mode** The 21064 BIU_CTL IPR (SYS_WRAP bit) must also be programmed for this mode to work correctly.
24:22	RW	Memory Type — The EB64 accommodates several DRAM SIMM sizes, allowing main memory size to be between 4 and 64 megabytes. This field selects the memory size as follows:

24 23 22	DRAM Size	SIMM Type	Main Memory
0 0 0	1-MB	256K×36	4 MB
0 0 1	2-MB	512K×36	8 MB
0 1 0	4-MB	1MB×36	16 MB
0 1 1	8-MB	2MB×36	32 MB
1 0 0	16-MB	4MB×36	64 MB

Bits	Access	Description
		See Section 3.2.1 for more information on memory size selection and control.
25	RW	Bcache Off — When this bit is set, the system logic ignores the Bcache. External system logic does not probe the Bcache, and all DMA transactions assume data is in DRAM memory. **Caution: Setting Bcache Off/On** The setting of this bit should correspond to the setting of the Bcache enable bit (BC_EN, bit 0) in the 21064 BIU_CTL IPR.
27:26	RO	DRAM PD<2:1> — During system register read accesses, these bits reflect the DRAM SIMM configuration information PD<2:1>, which provides information about the size of the DRAMs. Because the 1-megabyte SIMM information aliases with the 16-megabyte information, a system configuration bit should be used with this field (**sys_config1** — see bit 28, below).

(continued on next page)

Table 3–10 (Cont.) EB64 System Register Bit Description

Bits	Access	Description
31:28	RO	System Configuration Jumpers <3:0> — During system register read accesses, these bits reflect the state of the four EB64 board configuration jumpers **sys_config<3:0>** (J25, Figure 2–1 and Table 2–1). The default configuration bit state is high: Each configuration bit is pulled up on the EB64 board, and inserting a jumper pulls them down. The bits are defined as follows:

Signal	J25 pins	Description
sys_config0	1 to 5	Reserved.
sys_config1	2 to 6	DRAM SIMM Size — This bit completes the DRAM SIMM size information. Because the 1-MB and 16-MB SIMMs are aliases, this bit indicates which of the two is installed. If the bit reads high (default), it indicates that 1-, 2-, 4-, or 8-MB DRAM SIMMs are installed. The PD bits (<27:26>, above) identify the specific size. If the jumper is installed (bit reads low), it indicates that 16-MB DRAM SIMMs are installed.
sys_config2	3 to 7	Reserved.
sys_config3	4 to 8	Reserved.

Bits	Access	Description
29:26	WO	System Output Registers — During system register writes, these signals (**sys_output<3:0>**) are latched into registers that drive a set of headers (J30, Figure 2–1 and Table 2–1). The headers are directly usable by a logic analyzer probe (one row of active signals, one row of grounds). They can be written to during testing in order to track progress. They are not readable.

3.3.1.2 Debug ROM

Each of the two debug ROM sockets, PROM0 and PROM1, accommodates a standard 512-kilobyte or 1-megabyte PROM. Jumper J22 (Figure 2–1 and Table 2–1) selects the size, as follows:

- 512-KB — The jumper pulls up the most significant address bit.
- 1-MB — The jumper drives the most significant address bit with **io_addr21**.

Both sockets are addressed as described in Section 3.3.1.1. I/O address bits **io_addr<21:2>** (translated from **adr<28:9>**) go to both sockets, and both sockets drive back the same data byte on **io_data<7:0>**. To avoid output drive contention, only one PROM should be enabled. Jumper header J5 (Figure 2–1 and Table 2–1) selects the active socket. PROM0 is the default.

The debug ROM code is copied into memory and executed as part of the SROM functions described in Section 3.4.2.

The debug monitor supported functions include:

- File load
- Read and write memory and registers
- Memory image dump
- Transfer control to program
- Breakpoints

(For information about software design tools, see the literature listed in Appendix A.)

3.3.1.3 ISA Interface and Associated Peripheral Devices

The VLSI Technology VL82C486 chip (also called SC486) and its companion VL82C113A combination I/O chip are used for the following functions:

- ISA bus controller, including bus master
- Direct memory access (DMA) controller
- Interrupt controller
- Timer

3.3.1.4 Slow Speed Peripheral Devices

The Standard Microsystems Corporation FDC37C651 Super I/O Floppy Disk Controller is used as the combo I/O chip. It connects directly to the ISA bus on the EB64 board. It supplies:

- Two serial ports
- Floppy-disk controller

3.3.1.5 Ethernet Link

The AMD Am79C960 PCnet-ISA chip provides an Ethernet link. This link provides the capability to load program data into main memory at high speed. The chip and its associated glue logic (transformers, level shifters, capacitors, and so on) are connected to the ISA bus on the EB64 board. Both 10BaseT (twisted-pair) and 10Base2 (ThinWire) interfaces are provided (see J1 and J9, Figure 2–1 and Table 2–1). The base address for this Ethernet Controller is selectable and has been set to 360_{16}.

3.3.1.6 Interrupts

The following interrupt sources are accommodated:

- IRQ0 — ISA INTR interrupt from SC486.

  This interrupt supports all devices on the ISA, including:

  - Ethernet chip
  - Serial ports
  - Floppy-disk controller
  - Any device in either of the ISA extension slots.

  Within the ISA interrupt chain, the devices are connected to the ISA IRQ lines as shown in Table 3–11.

Table 3–11 ISA IRQ Input Definitions

ISA IRQ	Source	ISA IRQ	Source
0	Interval timer	9	Ethernet
3	Serial port 2	10	Keyboard
4	Serial port 1	11	Mouse
6	Floppy Disk	–	–

- IRQ1 — ISA NMI interrupt from SC486.
- IRQ2 — Real-time clock interrupt from VL82C113A.
- IRQ<5:3> — These interrupts are available for use by the expansion connector. The interrupt sources are asserted *low* and are pulled up by on-board resistors. Signals **ext_irq<2:0>_l** are the source of these interrupts. (see Table 4–1).

3.4 Power, Reset, and Initialization

The EB64 derives its main system power from a user-supplied, industry-standard, PC power supply. Power is delivered to most of the board's logic on dedicated power planes.

The 21064 requires 3.3 V, and it is provided by a pair of linear regulators driven by the 5 V supply. A power monitor senses the 3.3 V level to ensure that it is stable before the 21064 inputs are driven. Any device that drives the CPU has a tristate output, controlled by power monitor output.

Caution: Fan Sensor Required

The 21064 cooling fan *must* have a built-in sensor that drives a signal if the air flow stops. The sensor is connected to the EB64 board (J17 Figure 2–1 and Table 2–1). When the signal is generated, it places the system into dcok mode. This action protects the CPU under fan-failure conditions, because the 21064 dissipates less heat in dcok mode. The fan supplied with the EB64 includes an air-flow sensor.

3.4.1 Reset

The system has header pins (J16 and J23, Figure 2–1 and Table 2–1) that allow an external switch to control the **reset** or **dcok** signals. Both are individually selectable by attaching the switch to the appropriate header. The reset header (J16) initializes the 21064 and the system control logic, but does not send an initialization pulse to the ISA devices. The dcok header (J23) provides a full system initialization, equivalent to a power off/on cycle.

3.4.2 Initialization and SROM

The 21064 uses an SROM for its initialization code. When **reset** is deasserted, the contents of the SROM are read into the 21064's instruction cache. The code is then executed from the instruction cache.

The SROM provides the following functions:

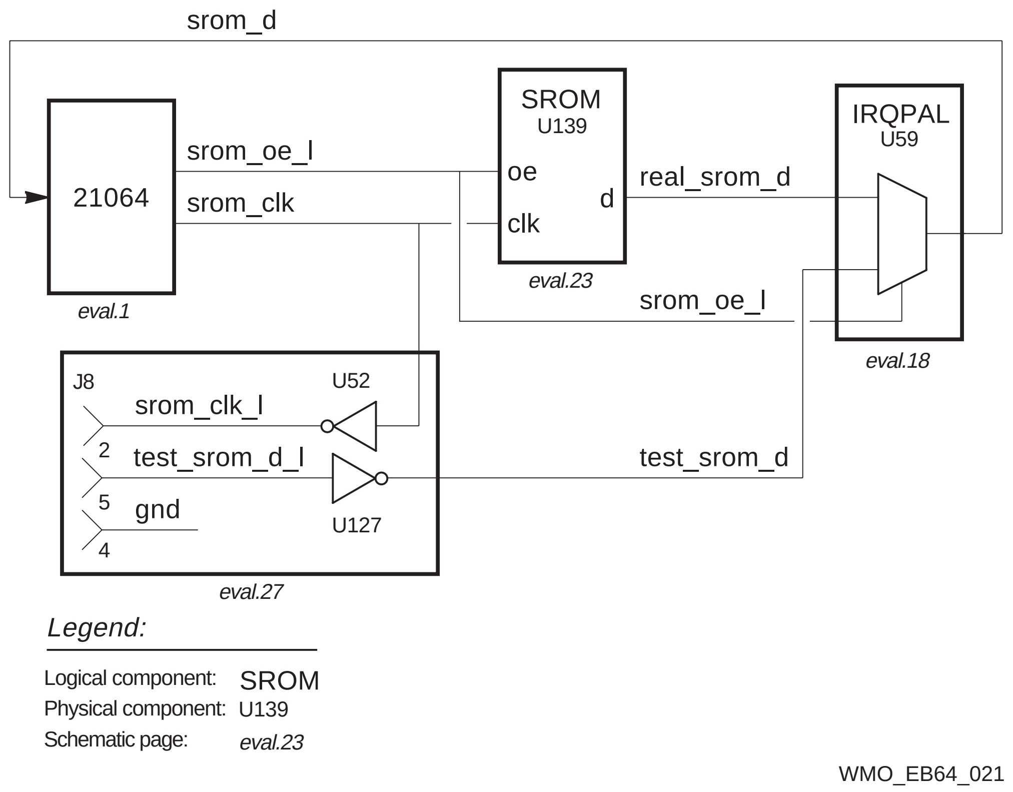
- Initializes the EB64 system register.
- Initializes the 21064 internal processor registers (IPRs).
- Sets up memory refresh and starts the refresh timer.
- Configures the 21064's BIU_CTL IPR for external Bcache accesses.
- Initializes the Bcache.
- Writes good parity by copying the contents of memory to itself.

- Copies the contents of the debug ROM to memory, starting at memory address zero.
- Flushes the instruction cache and jumps to address zero to begin execution.

After the SROM code has been read into the instruction cache, the 21064's SROM port can be used as a software-controlled serial port. This serial port can be used for such things as diagnosing system problems when the only working devices are the 21064 and SROM and the circuits needed for their direct support. Connector J8 (Figure 2–1 and Table 2–1) supports an RS232 terminal connection to this port (additional external logic is not required).

Figure 3–8 is a simplified diagram of the SROM serial port logic. The IRQPAL provides a multiplex function for the SROM (**real_srom_d**) and serial port (**test_srom_d**) data inputs to the 21064. Signal **srom_oe** provides the multiplexer input select function.

Figure 3–8 SROM Serial Port Diagram



3.5 System Clocks

The 21064 is driven by a 300 megahertz oscillator, which provides a 6.6 ns (150 megahertz) processor clock speed. The internal 6.6 ns period is divided by six and sent off chip to form the 39.6 ns (approximately 25 megahertz) external system clock. Two sets of external system clocks are provided: **sysclkout1_h** and **sysclkout2_h** (and their complements **sysclkout1_l** and **sysclkout2_l**) See schematic page *eval.1*. Figure 3–9 shows the external timing relationships between the **sysclkout1_h** and **sysclkout2_h** signals.

Note: General Timing Relationships

The *phase* relationship between the external oscillator input clock (**clkin_h**) and the internal CPU clock (**cpuclk**) shown in Figure 3–9 may not hold true. The figure should only be used to understand the general timing relationships between the various clock signals.

Figure 3–9 System Clock Timing

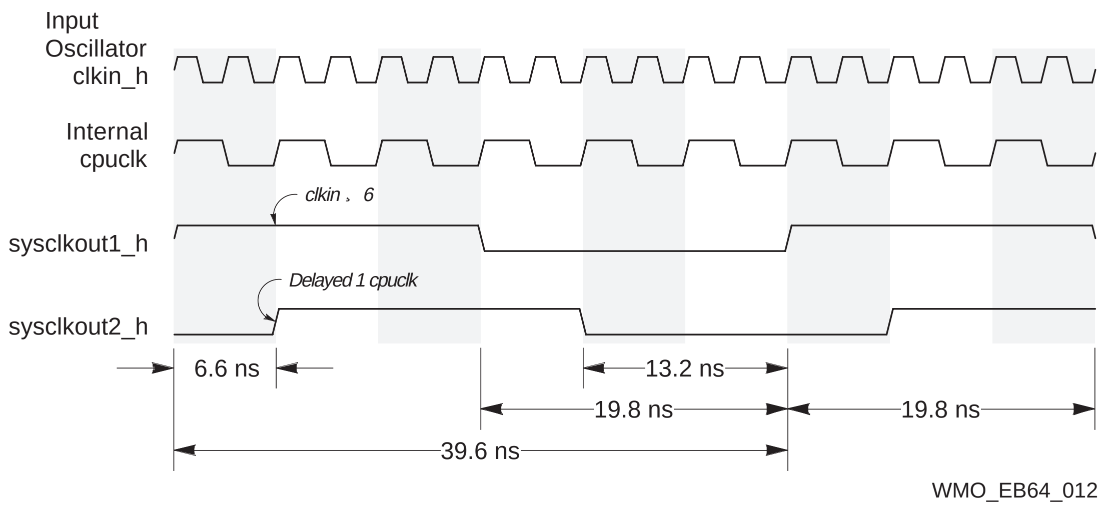
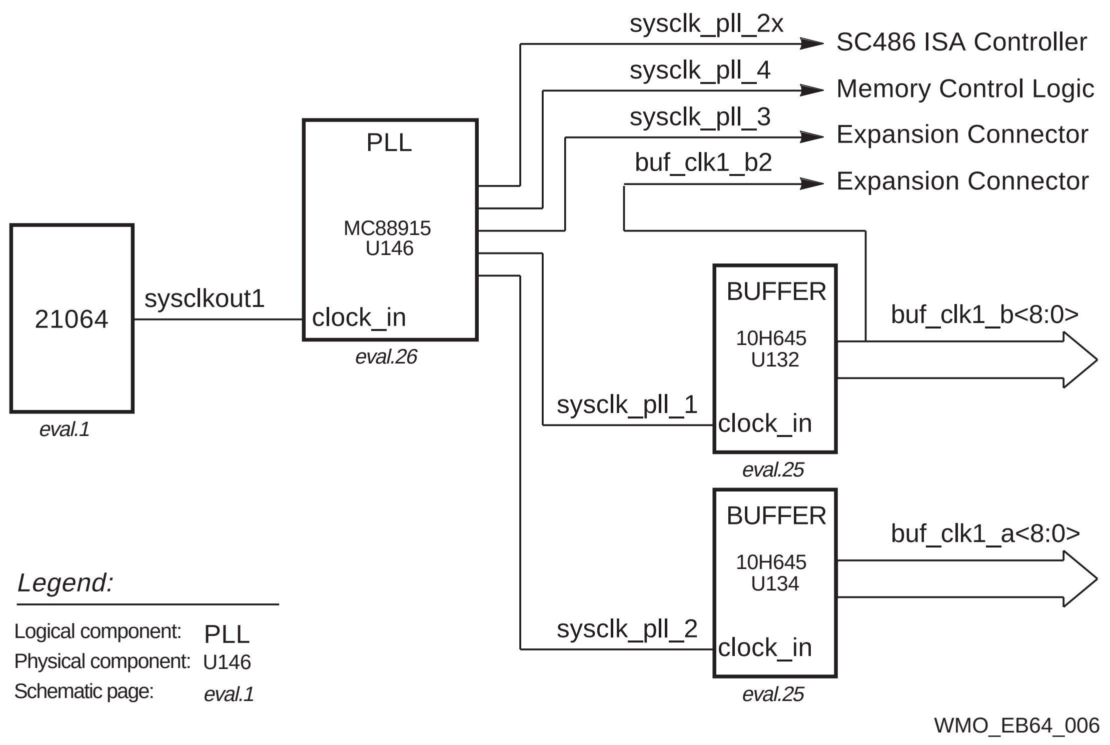


Figure 3–10 shows the clock distribution network for the EB64. The major EB64 module clock is **sysclkout1**. It is initially sent to a phased locked loop (PLL) clock driver. The PLL creates four nearly identical copies of the signal with low latency and skew from the original clock. The copies are distributed as follows:

- Two are sent to buffered clock drivers. Each clock driver generates nine copies of the master system clock, which are sent throughout the EB64 to drive most of the control logic.
- One is sent to the memory control logic to drive one of the Bcache control programmable logic devices (PLDs).
- One is sent to the expansion connector for customer use. See Section 4.2.3 for more information about expansion connector clocks.
- A 2x (twice as fast) version of the system clock is sent to the SC486 to control the ISA bus.

Figure 3–10 Clock Distribution



Note: Design Concepts

System design concepts are discussed in the application note *Designing a System with the DECchip 21064 Microprocessor*. Memory and backup cache design concepts are discussed in the application note *Designing a Memory/Cache Subsystem for the DECchip 21064 Microprocessor*. If you are not familiar with the 21064, you are encouraged to read both application notes. Appendix A gives ordering information and lists other associated documentation.

4 Expansion Interface

This chapter describes the EB64 implementation of the 21064 external interface. It includes a description of the expansion connector signals and design considerations.

4.1 Expansion Connector Signal Description

Table 4–1 describes the signals available at the expansion connector. The *DECchip 21064 Microprocessor Hardware Reference Manual* contains additional information about expansion connector signals that are buffered versions of the 21064 pinbus signals.

Table 4–1 Conventions

- Signal Name column — The **ext_** prefix is ignored, and signals are listed alphabetically.
- I/O column — Signal direction, as follows:

  I = input from expansion connector to EB64 logic (including 21064).
  O = output to expansion connector from EB64 logic (including 21064).
  B = bidirectional.

Table 4–1 Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_adr33_in	J26-B50	I	External address bit 33. This signal is buffered **adr33** from the 21064. It is permanently enabled. Unlike the other signals on this connector, this is active *high* and has an associated pull-down resistor. This signal is needed during DMA from the expansion connector when the DMA engine wants to allocate a Bcache block for data, and the block resides in the upper quadrant of the physical address space. This bit allows the appropriate Bcache tag signal to be asserted. See Section 4.2.5.
ext_be1 **ext_be0**	J27-B1 J27-A1	O	External unencoded transfer length. These signals are used as **adr<8:7>** for logic attached to the expansion connector that requires a contiguous linear address space. See **ext_tl<1:0>** and Section 3.1.3.1.
buf_clk1_b2	J26-B6	O	Buffered system clock (see schematic pages *eval.25* and *eval.26*). This clock is used when the external device uses four or less clock inputs (see Section 4.2.3). Clock **sysclkout1** is a 25 MHz (39.6 ns) system clock that synchronizes all clocked outputs.
buf_clk1_l	J26-B7	O	Buffered system clock. This signal is **sysclk-out1_l** buffered in the expansion connector buffers. Clock **sysclkout1** is a 25 MHz (39.6 ns) system clock that synchronizes all clocked outputs.
buf_clk2	J26-B9	O	Buffered delayed system clock. This signal is **sysclkout2** buffered in the expansion connector buffers.
buf_clk2_l	J26-B10	O	Buffered delayed system clock. This signal is **sysclkout2_l** buffered in the expansion connector buffers.
buf_data<127:0>	–	B	Buffered data. These signals are buffered **data<127:0>** (see schematic page *eval.3*). See Table 4–3 for connector pin numbers.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
buf_par3 **buf_par2** **buf_par1** **buf_par0**	J11-A8 J11-A10 J11-B6 J11-B8	B	Buffered data parity. These signals are combined and buffered to form **check21, check14, check7, check0** to/from DRAM<3:0> and the 21064. See Figure 4–1 **Note: The signal names change.** See Figure 4–1. At the input to U83 *io.6*, **lw0_par<3:0>** becomes **gen_par<3:0>**. The **par<3:0>** output of U133 *mdlatch.5* becomes **check21, check14, check7,** and **check0**.
buf_tagctl_d **buf_tagctl_s** **buf_tagctl_v** **buf_tagctl_p**	J26-A47 J26-A45 J26-A44 J26-A48	O	Buffered tag dirty, shared, and valid bits and their parity bit. These signals are **tagctl_<d,s,v,p>** buffered in the expansion connector buffers. These signals are always enabled. They allow logic on the expansion connector to perform a Bcache probe (see also **ext_hit** and **ext_tagctl_<d,s,v,p>_l>**).
ext_cack2_l **ext_cack1_l** **ext_cack0_l**	J26-A39 J26-A41 J26-A42	I	External cycle acknowledge. These signals are ORed with **io_cack<2:0>** and memory state machine signals in the memory state machine to generate **cack<2:0>** to the CPU. Cycle acknowledge is used to terminate all transaction types.
ext_cas3_l **ext_cas2_l** **ext_cas1_l** **ext_cas0_l**	J11-B20 J11-A25 J11-B21 J11-A26	I	External column address strobe. These signals are combined with **io_cas<3:0>** in the memory state machine to generate **early_cas<3:0>**. In turn, **early_cas<3:0>** is buffered and clocked with **sysclkout1** as **buf1_dram_cas<8:0>** to provide the DRAMs with multiple versions of the signals without excess skew. When asserted, **buf1_dram_cas<8:0>** asserts CAS to the appropriate longword banks of the system DRAM array.*

*For more information, see the *Designing a Memory/Cache Subsystem for the DECchip 21064 Microprocessor* application note listed in Appendix A.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_clr_flag_l	J11-A22	I	External clear lock flag. This signal is ORed with I/O signals in the I/O state machine to generate **io_clr_flag**. One clock later, **io_clr_flag** is combined with memory state machine signals in the RAS PAL to clear **lock_flag**. Signal **lock_flag** clears two clocks after **ext_clr_flag** is asserted. See also Sections 3.2.3.4 and 3.2.3.5.
ext_col_a0_l	J11-B23	I	External column address 0. This signal controls the value of the least-significant address line to the system DRAM. The interactions of this signal are described in Section 4.2.6.
ext_cwmask7 **ext_cwmask6** **ext_cwmask5** **ext_cwmask4** **ext_cwmask3** **ext_cwmask2** **ext_cwmask1** **ext_cwmask0**	J26-B33 J26-B34 J26-B36 J26-B37 J26-B39 J26-B40 J26-B42 J26-B43	O	External cycle write mask. These signals are **cwmask<7:0>** buffered in the expansion connector buffers. During CPU writes, write masks indicate which of the eight 32-bit blocks are to be written. During CPU reads, the CPU provides information on the miss that is being fulfilled. See the *DECchip 21064 Microprocessor Hardware Reference Manual* for more information.
ext_data_a4_l	J11-A15	I	External data address 4. This signal is combined with other signals in the I/O state machine to generate **io_data_a4**. Signal **io_data_a4** is then combined and registered in the memory state machine to generate **sys_data_a4**. This signal is asserted during accesses to the Bcache. Signal **sys_data_a4** is ORed with **cpu_ data_a4** from the 21064. As the least significant address line to the Bcache SRAMs, it selects the octaword of the cache block that is currently addressed. It is used for victim writes and Bcache block fills.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_data_oe_l	J11-A39	I	External data output enable. This signal is combined with **io_data_oe** in the memory state machine to generate **sys_data_oe**. Signal **sys_ data_oe** is ORed with **cpu_data_oe<3:0>** from the 21064. This signal is asserted to drive data and parity from the Bcache SRAMs onto the system data and check lines. The data can be passed directly to the CPU or can be clocked through the memory data transceivers to the expansion connector or the DRAM data bus.
ext_data_we3_l **ext_data_we2_l** **ext_data_we1_l** **ext_data_we0_l**	J11-A40 J11-B36 J11-B37 J11-B38	O	External data write enable. These are longword write strobes to the Bcache data SRAMs.* These signals are combined with **io_data_ we<3:0>** and registered in the memory state machine to generate **early_data_we<3:0>**. Signals **early_data_we<3:0>** become **sys_data_ we<3:0>** through delayed latches clocked by **sysclkout1_l**. Signals **sys_data_we<3:0>** are ORed with **cpu_data_we<3:0>** from the 21064.
dcok	J26-B12	O	This signal is buffered **p_dcok** from power connector J3.
ext_del_creq2 **ext_del_creq1** **ext_del_creq0**	J26-A30 J26-A32 J26-A33	O	External delayed cycle request. These signals are buffered **del_creq<3:0>** generated by **creq<3:0>** from the 21064 via clocked latches. Cycle request is delayed by one CPU clock so that it can be sampled using **sysclkout1**.
ext_dinvreq_l	J11-A17	I	External data cache invalidate request. This signal is ORed with **io_dinvreq** and memory state machine signals in the memory state machine to generate **dinvreq** to the 21064. This signal is used to invalidate a data cache entry when a Bcache block is evicted due to a victim write.

*For more information, see the *Designing a Memory/Cache Subsystem for the DECchip 21064 Microprocessor* application note listed in Appendix A.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_dmapwe	J11-B15	O	External data cache backmap write enable. This signal is **dmapwe** from the 21064 buffered in the expansion connector buffers. It controls the write enable to an optional, external, backmap RAM during external cache reads controlled by the 21064.
ext_doe_l	J26-B30	I	External data output enable. This signal is combined with other signals in the memory state machine to generate **sm_doe_dis**, which in turn generates **doe_dis**. This signal is asserted by external logic to allow the 21064 to drive the data bus during external write transactions. It is deasserted to allow a victim write sequence to proceed during a CPU write (the Bcache data SRAMs can then drive data on the CPU data bus through the memory data transceivers to the DRAM data bus). See also **ext_data_oe_l**. **Note: The signal names change.** See Figure 4–1. The **doe_dis** output of the memory state machine becomes the **doe_l** input to the 21064.
ext_drack2_l **ext_drack1_l** **ext_drack0_l**	J26-A35 J26-A36 J26-A38	I	External data bus read data acknowledge. These signals are combined with **io_drack<2:0>** and memory state machine signals in the memory state machine to generate **drack<2:0>** to the 21064. These signals inform the 21064 whether data on the data bus: • Is valid. • Is cacheable. • Includes good parity.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_dwsel1_l	J26-B31	I	External data bus write select. This signal is combined and registered in the memory state machine to generate **dwsel** to the 21064. During CPU write transactions, external logic uses this signal to tell the 21064 which part of the 32-byte write-data block to drive on the data bus; it switches the CPU data bus from first cache line data to second cache line data. **Note: The signal names change.** See Figure 4–1. Output **dwsel** of U117 *mem.10* becomes **dwsel1**.
gnd	–	–	Ground. See Table 4–5 for pin list.
ext_hit	J26-B45	O	External cache hit. This signal is buffered **hit** from the memory state machine. This signal is asserted on successful Bcache tag probe; that is, **adr<33,25:19>** from 21064 matches **tagadr<33,25:19>**.
ext_hold_ack	J26-B48	O	External hold acknowledge. This signal is buffered **hold_ack** from the 21064. This signal is the handshake response to **ext_hold_req**. See Section 4.3 and Figure 4–5 for bus acquisition and release sequence.
ext_hold_l	J26-B46	I	External hold. This signal is used to get control of the system bus from the memory state machine. Signal **ext_hold** and **hold** from the SC486 are combined in the I/O state machine to generate **io_hold** which becomes **hold** in the memory state machine. When the memory state machine next reaches its idle state, it relinquishes bus control and asserts **ext_start** or **dma_start**. (Expansion bus always has priority over DMA, which represents the SC486). See Section 4.3 and Figure 4–5 for bus acquisition and release sequence. **Note: The signal names change.** See Figure 4–1. Signal **io_hold** changes to **hold** in the memory state machine.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_hold_req_l	J26-A50	I	External hold request. This signal is used in the I/O state machine to generate **hold_req** to the 21064. After **ext_hold** has been asserted to generate **ext_start**, **hold_req** is asserted to get control of the Bcache/external bus from the 21064. See Section 4.3 and Figure 4–5 for bus acquisition and release sequence.
ext_inhibit_tag_oe_l	J11-A13	I	External inhibit tag output enable. In the memory state machine, this signal is combined and registered to generate **inhibit_tag_oe** which is combined with **io_inhibit_tag_oe** to generate **sys_tag_oe**. When deasserted, **inhibit_tag_oe** enables the tag and tag control values. This allows a hit/miss to be determined, along with the current state of the addressed Bcache block (S, V, D, P bits). Signal **inhibit_tag_oe** must be asserted during a Bcache fill sequence, when new values are written into the tag and tag control SRAMs.
ext_ioadr_dis_l	J26-B2	I	External I/O address disable. This signal is combined in the I/O state machine to generate **mem_ioadr_dis** which enables/disables the I/O address bus through the CPU/IO address buffers. When asserted, this signal disables the address buffers that drive I/O addresses (**io_addr<31:2>**) from the EB64 to the expansion connector. This signal is asserted during the bus acquisition sequence by an external DMA device. It must be asserted *before* **hold_ack** can assert, because the assertion of **hold_ack** enables the buffers that drive in the opposite direction. See Section 4.3 for the bus acquisition and release sequence.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
io_addr<31:2>	–	B	I/O address bus. These signals provide the address bus for DMA devices on the expansion connector and for I/O accesses to devices on the expansion connector. Note that the mapping between I/O address and CPU address signals is different for a bus master (DMA device) and a bus slave (I/O device), as described in Section 3.1.3.1. See Table 4–4 for connector pin numbers.
io_be3 **io_be2** **io_be1** **io_be0**	J11-B9 J11-B11 J11-B12 J11-B14	O	I/O byte enables. These signals are combinational decodes of 21064 **adr<8:5>**, as described in Section 3.1.3.2. They can be used by external logic to control byte masks for I/O write cycles.
ext_io_over_l	J11-B18	I	This signal is asserted by a DMA device on the expansion connector to indicate that external use of the bus has completed. It is combined in the I/O state machine to generate **io_over** to the memory state machine. When **io_over** is detected, the memory state machine resumes control of the system bus. See Section 4.3 and Figure 4–5 for bus acquisition and release sequence.
ext_irq2_l **ext_irq1_l** **ext_irq0_l**	J11-A19 J11-B17 J11-A20	I	External interrupt request lines. These signals drive **cpu_irq<5:3>** respectively to 21064 through IRQ PAL delay. Interrupt priority is a function of PALcode. All interrupts to the 21064 may be asynchronous and are level-sensitive.
ext_md_rd_clken_l	J11-B26	I	External memory data read clock enable. This signal is ORed with **io_md_rd_clken** and memory state machine signals in the memory state machine to generate **md_rd_clken**, which enables the read data to be latched.*

*For more information, see the *Designing a Memory/Cache Subsystem for the DECchip 21064 Microprocessor* application note listed in Appendix A.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_md_rd_oe_l	J11-A34	I	External memory data read output enable. This signal is ORed with **io_md_rd_oe** and memory state machine signals in the memory state machine to generate **md_rd_oe**, which enables the MDlatch to drive the Bcache/CPU data bus.*
ext_md_wr_clken_l	J11-A29	I	External memory data write clock enable. This signal is ORed with **io_md_wr_clken** and memory state machine signals in the memory state machine to generate **md_wr_clken**, which latches data in the write direction.*
ext_md_wr_oe3_l **ext_md_wr_oe2_l** **ext_md_wr_oe1_l** **ext_md_wr_oe0_l**	J11-A31 J11-B27 J11-A32 J11-B29	I	External memory data write output enable. These signals are ORed with **io_md_wr_oe<3:0>** and memory state machine signals in the memory state machine to generate **md_wr_oe<3:0>**, which drives the **buf_data<127:0>** bus.*
ext_md_wr_par_oe_l	J11-B30	I	External memory data write parity output enable. This signal is registered in the memory state machine to generate **md_wr_par_oe**, which drives the **buf_par<3:0>** latch with parity.*
ext_ras_l	J11-A23	I	External row address strobe. This signal is ORed with I/O, memory, and refresh signals in RAS PAL to generate **dram_ras<1:0>**. RAS is asserted as a function of **mux_adr<24,22>** and the selected DRAM type, and is buffered to various chunks of the DRAM as **buf<4:1>_dram_ras<1:0>**.
remote_reset_l	J26-B1	I	When asserted, this signal resets the EB64 in the same way as pressing a reset switch on the dcok header (see Section 3.4.1). This allows circuitry attached to the expansion connector to reset the system (and itself) as a result of, for example, some form of fatal error or time out.
ext_reset_l	J36-B13	O	Buffered **rst_l** from reset/dcok circuit. (**rst** is the EB64 master reset.) This signal is asserted as a result of any EB64 reset and should be used to reset circuitry attached to the expansion connector.

*For more information, see the *Designing a Memory/Cache Subsystem for the DECchip 21064 Microprocessor* application note listed in Appendix A.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_sel_col_l	J11-A28	I	External select column. This signal is ORed with **io_sel_col** and memory state machine signals in the memory state machine to generate **sel_col** which controls the DRAM address multiplexers for all DRAM address lines. When this signal is asserted, the system DRAM address multiplexers drive the column address, rather than the row address, into the DRAM array.
ext_sel_col2_l	J11-A16	I	External select column 2. This signal is combined with I/O and memory state machine signals in the memory state machine to generate **dram_addr0**. To avoid an additional external multiplexer delay, the memory state machine logic implements all required **dram_addr0** multiplexing. This signal is asserted during the transfer of the second cache line of a block into the system DRAM. It is used to toggle the least significant CAS address bit during the page-mode second CAS pulse of the DRAM cycle. The interactions of this signal are described in Section 4.2.6.
ext_sel_tagadr_l	J11-A14	I	External select tag address. This signal is combined and registered in the memory state machine to generate **sel_tagadr**, used in the DRAM address multiplexers. When asserted, **sel_tagadr** causes data from the tag SRAMs (that is, the tag address) to be driven into the system DRAM row/column address multiplexer, rather than allowing the system address bus to drive this multiplexer. This is necessary so that a victim write sequence performed by logic attached to the expansion connector can generate the correct address to retire a cache block from the Bcache to the system DRAM.
ext_start	J26-A1	O	This signal is the handshake response to **ext_hold**. Generated by the memory state machine, **ext_start** is a combinational signal and asserts in the same clock tick as **io_hold**. See Section 4.3 and Figure 4–5 for bus acquisition and release sequence.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

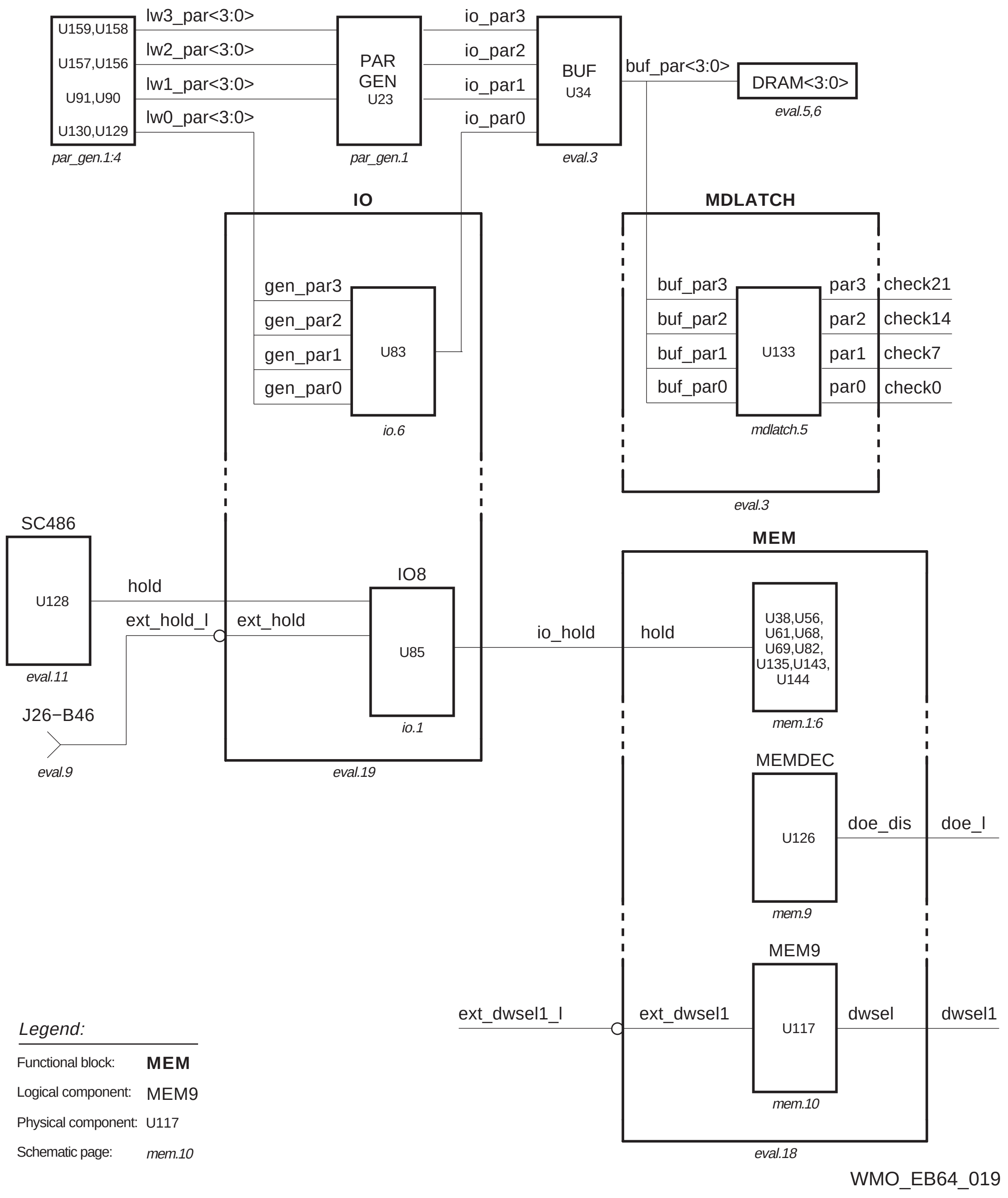
Signal Name	Pin	I/O	Description
sysclk_pll_3	J26-B4	O	This signal is generated by **sysclkout1** in the PLL (schematic page *eval.26*) for use with user-provided devices.
ext_tagadr_we_l	J11-A11	I	External tag address write enable. This signal is combined and registered in the memory state machine to generate **sys_tagadr_we**. This signal is used to control the write strobe of the tag SRAMs during a Bcache block load. See also **ext_wrt_tagadr_oe_l**.
ext_tagctl_d_l **ext_tagctl_s_l** **ext_tagctl_v_l** **ext_tagctl_p_l**	J11-B33 J11-A38 J11-A37 J11-B35	I	External tag dirty, shared, and valid bits and their parity bit. These signals are combined and registered with **io_tagctl_<d,s,v,p>** in the memory state machine to generate **mem_tagctl_<d,s,v,p>**. These are Bcache tag control bits driven by logic attached to the expansion connector when that logic is writing data into the Bcache; for example, a read-fill or write-allocate sequence. Compare with **buf_tagctl_<d,s,v,p>**.
ext_tagctl_we_l	J11-B32	I	External tag control write enable. This signal is combined and registered with **io_tagctl_we** in the memory state machine to generate **sys_tagctl_we**. Signal **sys_tagctl_we** is ORed with **cpu_tagctl_we** to control the write strobe of the tag control SRAMs during a Bcache block load. Circuitry attached to the expansion connector asserts this signal to update the tag control-bit values for the currently-addressed Bcache block to the values driven on **ext_tagctl_<d,s,v,p>_l** when **ext_wrt_tagctl_oe_l** is asserted.
ext_tl1 **ext_tl0**	J27-B50 J27-A50	O	External unencoded transfer length. These signals are used as **adr<6:5>** for logic attached to the expansion connector that requires a contiguous linear address space. See **ext_be<1:0>** and Section 3.1.3.1.
Vdd	–	–	See Table 4–5 for pin list.

(continued on next page)

Table 4–1 (Cont.) Expansion Connector Signals

Signal Name	Pin	I/O	Description
ext_we_l	J11-B24	I	External write enable. This signal is ORed with **io_we** and memory state machine signals in the memory state machine to generate **dram_we**. Signal **dram_we** is buffered as **buf<4:1>_dram_we** to write enable all banks of the DRAM.
wrap_read	J27-B2		System register bit 21 (see Table 3–10). This signal is asserted to enable data wrapping on the 21064. When wrapping is enabled, 21064 data read accesses must use the **cwmask1** signal to determine which of the data portions to access first (**cwmask1** acts as **adr4**).
ext_wrt_tagadr_oe_l	J11-B39	I	External write tag address output enable. This signal is combined and registered in the memory state machine to generate **wrt_tagadr_oe**. It enables the system address onto the data bus of the tag SRAMs. It is used to update the tag information during a Bcache block fill. See also **ext_tagadr_we_l**.
ext_wrt_tagctl_oe_l	J11-A35	I	External write tag control output enable. This signal is combined and registered with **io_wrt_tagctl_oe** in the memory state machine to generate **wrt_tagctl_oe**. When asserted, this signal enables the values of **ext_tagctl_<d,s,v,p>** onto the data bus of the tag control SRAMs. It is used to update the tag control information during a Bcache block fill. See also **ext_tagctl_we_l**.
ext_zone	J26-B15	O	External zone. This signal is asserted during CPU accesses to the upper quadrant of physical memory (**adr<33:32>** = 11). For such cycles, the EB64 memory state machine waits, forever, if necessary, for signal **io_over** or the cycle acknowledge (**cack<2:0>**) signals to indicate the end of the cycle. The I/O state machine generates **ext_zone** when **adr33** and **adr32** are both asserted. When asserted, **ext_zone** indicates that this cycle is a slave cycle to a device in the expansion connector address space.

Figure 4–1 Signal Name Changes



4.2 21064 Expansion Interface

The EB64 supplies a 21064-like protocol to a set of connectors on the board. As shown in Figure 1–1, the signals are buffered versions of the system signals. Because these buffered signals are mostly isolated from the CPU signals, they should not affect operation in normal circumstances. The expansion connector allows users to design their logic following a protocol similar to the 21064 pinbus protocol and test it on real hardware.

The expansion interface includes signals that allow external logic to interact with the CPU, the Bcache, and the DRAM. This allows options to be designed as either simple I/O devices (memory-mapped bus slaves that respond to bus cycles from the 21064) or DMA devices (capable of acquiring control of the bus from the 21064 and performing reads and writes directly to the Bcache and the DRAM). The expansion interface *control* signals (Figure 4–2) are special versions of existing memory control signals with an **ext_** prefix.

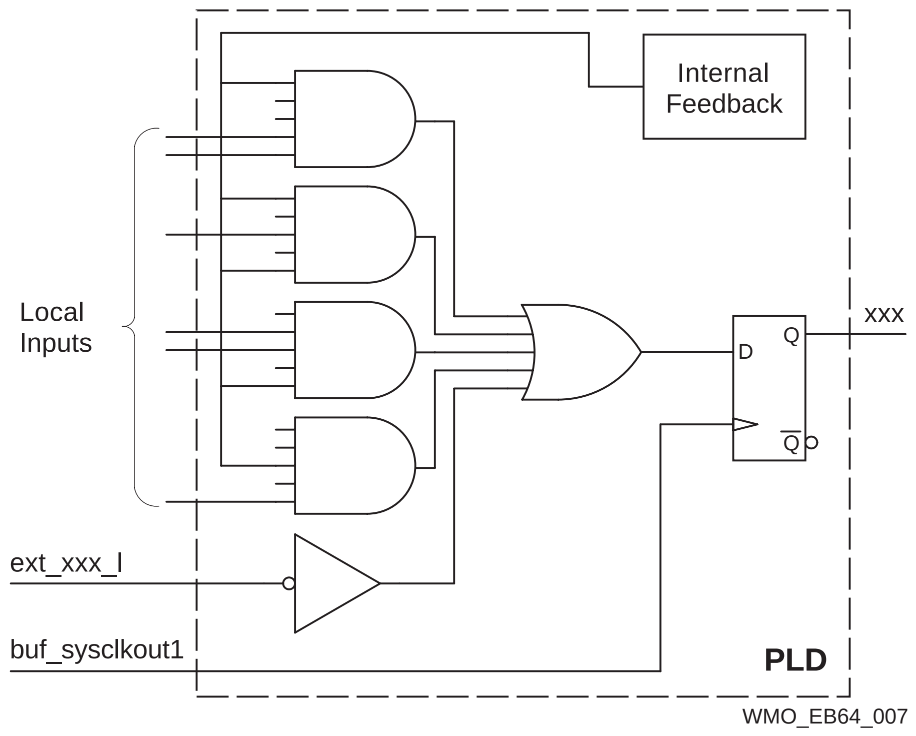
Note: Expansion Control Signals Are Active Low

All *control* inputs *from* the expansion interface are asserted *low* and pulled up by on-board resistors. They are deasserted unless an external module is plugged in and drives them. They also carry an **_l** suffix; for example, **ext_cack2_l**.

Designing hardware to connect to the expansion connector requires most of the same knowledge as a 21064 design and is beyond the scope of this guide. Additional information is available in the EB64 schematics and programmable-logic source files, and also in the literature listed in Appendix A. Additional technical support is available from the DECchip Information Line, also described in Appendix A.

The primary system considerations for external logic concern data path width and cache coherency.

Figure 4–2 Expansion Interface Signals



4.2.1 Data Path Width

The EB64 uses the full, 128-bit width of the 21064 data path. Designing interfaces to peripherals with narrower data path widths is not a problem, but the system implications must be considered: It takes as many clock cycles to transfer 128-bits of data as it does to transfer any smaller number of bits. In addition, for DMA engines, an overhead is associated with bus acquisition and relinquishment; every wasted clock cycle can equate to as many as 12 CPU instructions (six clock cycles, two instructions/cycle).

4.2.2 Cache Coherency

External hardware must maintain cache coherency with the 21064 internal caches (data cache and instruction cache) and the Bcache. The following are some of a variety of hardware and software techniques that can be used to maintain cache coherency.

1. I/O Devices

   The simplest, and usually most desirable approach is to make I/O devices nonBacheable by returning the appropriate values on **ext_cack<2:0>_l**.

2. Additional Memory

   The expansion connector can be used to extend EB64 memory. The additional memory can be conventional DRAM or special-purpose memory such as dual-ported video RAM (VRAM). In these circumstances, a more advanced Bcache scheme may be desirable.

   Because the EB64 DRAM main memory uses the Bcache, the way in which external slave devices can use the Bcache is limited. If circuitry on the expansion interface allocates Bcache blocks for addresses where **adr<33:32>** = 11, it must *never* set the D (dirty) bit for these blocks (Section 3.2.2.1) because the EB64 memory controller would then need to reallocate the Bcache block. If the external circuitry had the capability to mark the block dirty, then the EB64 memory controller would have to be able to perform victim write sequences to the memory on the external board, which it cannot do.

3. DMA Devices

   For DMA devices there are more options. The simplest is to constrain the DMA engine to perform transfers directly to the DRAM and to ignore the Bcache. The 21064 must then access this data solely through the nonBcached aliases of the DRAM in the address space. This guarantees that cache coherency is maintained.

   A more advanced design would need to perform a Bcache probe on DMA reads, and take data from the Bcache if there was a hit. Strictly, the requirement to maintain Bcache coherency is to take data if there was a hit and the Bcache is dirty. Otherwise, the data would be retrieved from the DRAM.

   For DMA writes the design would have to:

   - Probe the Bcache and, if there was a hit, merge the new data back into the Bcache block. (The dirty bit should always be observed clear during such a probe.)
   - Clear the interlock flag.
   - Invalidate the data cache.

   Otherwise, the data would be stored to the DRAM. (A DMA engine does not usually write-allocate.)

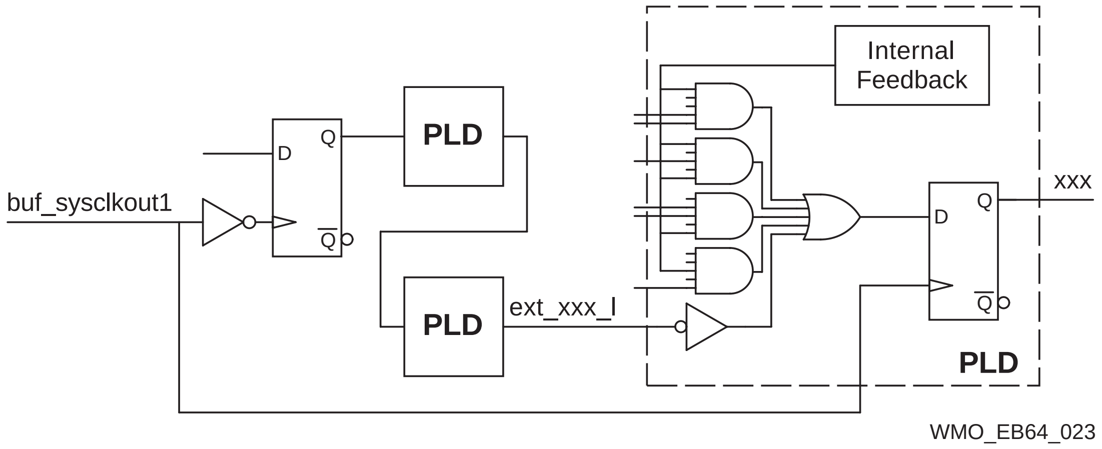
4.2.3 Expansion Connector Clocks

The control signals coming from the expansion connector logic, that is, the control signals generated by user hardware, are clocked by the master system clock. This same clock drives the memory control logic on the EB64 board and is provided to the expansion connector to drive user hardware.

The EB64 system clock setup holds true for signals, such as **ext_cack<2:0>_l**, that eventually control other signals that are driven back to the 21064 with the appropriate setup to the microprocessor system clock. The signals are relatched by the board memory logic, making the setup time easy to achieve.

Figure 4–3 shows this relationship. The clocked device is a PLD, and the AND/OR plane feeds the flip-flop D input. Devices used on the EB64 board require 10 ns setup before the clock. This allows several levels of logic to be placed between the user expansion interface clock and the clock that is used to resynchronize the external signals. This figure also shows the asserted low nature of the **ext_xxx_l** expansion connector control signals.

Figure 4–3 Expansion Connector Control Setup



Buffered versions of all four 21064 system clocks are delivered to the expansion connector (**buf_clk1_h/l, buf_clk2_h/l**). When using these signals, careful attention must be given to their skew with respect to other clock signals. Delayed and nondelayed versions of the master system clock (a version of **sysclkout1_h**) are also sent to the expansion connector. These two signals allow circuitry attached to the expansion connector to be clocked in a way that is closely matched to the clocks on the EB64. To achieve this, signal loading and board layout and characteristics requirements must be followed:

- The clock signals must have a characteristic impedance of 70 Ohms. This can be achieved by using a controlled-impedance printed-circuit board and specifying etch width and board layup (positioning of the internal power and ground planes).
- Four or fewer nodes

  The **buf_clk1_b2** clock, if used, can be loaded by up to four nodes. If fewer than four nodes load the net, dummy loads must be added in the form of 5 pF capacitance for each missing load. The net must be routed so that the four destination nodes are closely grouped and that the etch length from the expansion connector gold finger to each node is *exactly* 49 mm (2 in).
- More than four nodes

  The **sysclk_pll_3 clock**, if used, must be routed through a clock buffer using equivalent circuitry to that shown on schematic page *eval.25*. The etch length from the expansion connector gold finger to the clock buffer input pin must be *exactly* 49 mm (2 in). The series damping resistors on the clocks generated by the buffer must be situated so that the etch connecting them to the buffer is short. Each net must then have the same loading and layout characteristics as described for **buf_clk1_b2** above. The etch length from the series damping resistor to the destination node must be *exactly* 216.56 mm (8.526 in).

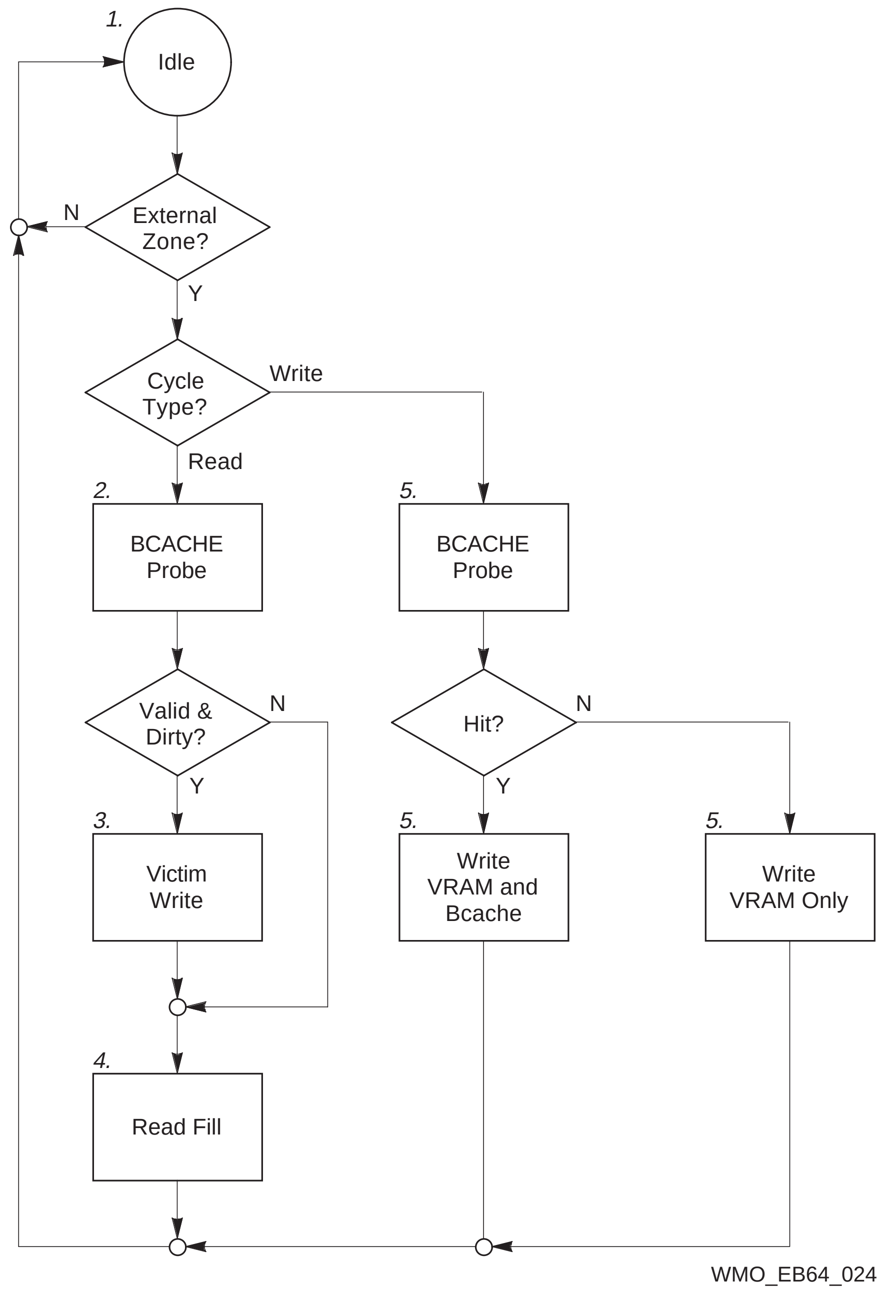
If these layout rules are adhered to, the total skew between any two **buf_clk1_xx** clocks in the system (EB64 and expansion connector) should not exceed 2.0 ns (the specification of the H645 clock buffer devices). Furthermore, the total clock fanout for the external board will be a maximum of 40 nodes.

4.2.4 Programmed I/O Through the Expansion Connector

This section gives an overview of the steps involved in performing reads and writes to a device on the expansion connector. See Figure 4–4.

The external device for this example is a bank of video memory that is Bcached on reads such that the S (shared) bit is always set by external logic on read fills. Therefore, Bcache blocks that get allocated to the video memory always behave as write-through. This means that they *never* become dirty. Consequently, a Bcache block currently allocated to the video memory can be reallocated to the DRAM simply by overwriting the block (a victim write sequence is never needed). Conversely, when the external hardware allocates a

Figure 4–4 Programmed I/O Through the Expansion Connector



Bcache block to the video memory, it must potentially perform a victim write to retire that data to the system DRAM.

1. Idle Loop

   Wait for a cycle addressed to video memory. The **ext_zone** signal will be asserted for accesses to the upper quadrant of physical memory. If necessary, some number of I/O address (**io_addr**) signals can also be decoded. In addition to decoding **ext_zone**, decode **del_creq<2:0>** to determine the cycle type. For read cycles, branch to step 2; for write cycles branch to step 5. For other cycles, some type of fatal error should probably be generated.

   If branching to step 5 (write cycle), prepare to latch data from the 21064 by asserting

   ext_md_wr_oe_l
   ext_md_wr_clken_l
   ext_md_wr_par_oe_l

   Prepare to write to the VRAM as appropriate. Assert **ext_doe_l** so that the 21064 will continue to drive write data.

   The EB64 memory controller remains in a wait loop during the whole of the cycle. It automatically regains control of the system when it detects **ext_cack<2:0>_l** as no longer idle.

2. Read Cycle

   Perform a Bcache probe. The Bcache tag control and tag address (**tagctl** and **tagadr**) lines will be enabled, so this probe simply requires checking the appropriate tag control signals (**buf_tagctl_v**, **buf_tagctl_d**). If the Bcache block is valid and dirty, a victim write sequence (step 3) is necessary. Otherwise, proceed with the read fill (step 4).

   If branching to step 3, assert **ext_sel_tagadr_l** so that the address into the system DRAM is generated in part by the tag address lines; that is, the system DRAM is being addressed at the correct location for the victim write of the Bcache block.

3. Victim Write Sequence

   Retire the valid, dirty Bcache block to the system DRAM so that this Bcache block can be filled with data from VRAM. The following signals need to be manipulated:

   - **ext_ras_l**
     ext_cas<3:0>_l
     ext_we_l

     to actually write to the system DRAM.

- **ext_data_oe_l** so that the Bcache data SRAMs will drive their data into the DRAMs.
- **ext_md_wr_oe<3:0>_l**
  ext_md_wr_par_oe_l
  md_wr_clken_l

  to transfer the data from the CPU data bus (where it is being driven by the Bcache data SRAMs) through the memory data transceivers (MDlatch) to the DRAM data bus.
- **ext_data_a4_l** to select the second location in Bcache to access the second 128-bit octaword of data.
- **ext_sel_col_l** to select the DRAM column address.
- **ext_sel_col2_l** to select the second column address for the adjacent location in DRAM, which is the destination of the second 128-bit octaword of data (this data is written as a page-mode access to the DRAM).

When the victim write sequence is complete, continue with step 4, the read fill sequence.

4. Read Fill Sequence

   Either fall through to this step after a victim write sequence (step 3) or jump to this step as the result of a read where the associated Bcache block is not valid or not dirty (step 2).

   This sequence requires the manipulation of the appropriate user signals to retrieve data from the VRAM on the external board. It requires manipulating the following signals:

   - **ext_inhibit_tag_oe_l**
     ext_md_rd_oe_l
     ext_md_rd_clken_l

     so that the VRAMs on the external board can drive new data through the memory data transceivers (MDlatch) and into the Bcache data SRAMs.
   - **ext_wrt_tagctl_oe_l**
     ext_wrt_tagadr_oe_l
     ext_tagctl_v_l
     ext_tagctl_s_l

     to assert the appropriate tag control bits and drive them into the Bcache tag control SRAMs.

- **ext_data_we<3:0>_l**
  ext_tagctl_we_l
  ext_tagadr_we_l

  to write new data into the Bcache data, tag address, and tag control SRAMs. The tag address itself is still being driven by the 21064 on its address bus. Signal **ext_drack<2:0>_l** needs to be manipulated to simultaneously pass the data to the 21064.
- **ext_data_a4_l** to select the second location in the Bcache data SRAM.
- **ext_dinvreq_l** may have to be manipulated to invalidate a 21064 internal data cache block.
- **ext_cack<2:0>_l** to terminate the 21064 read cycle.

5. Write Block

   Perform a Bcache probe. As for the read cycle, this simply requires checking the appropriate signals. If a Bcache hit is detected (**ext_hit** asserted) the data is written to VRAM (on the external connector logic) and to the appropriate Bcache block. The Bcache behaves as write-through for VRAM data because video memory is always required to be up-to-date. If a Bcache miss is detected, data is written only to VRAM. A write-allocate is never done, and there is no victim sequence for writes. In practice, the same sequence can be used for the Bcache miss and the Bcache hit by conditionally asserting **ext_data_we<3:0>_l** (the write strobes to Bcache). The write sequence involves manipulating the following signals:

   - **ext_doe_l**
     ext_md_wr_oe<3:0>_l
     ext_md_wr_clken_l
     ext_md_wr_par_oe_l
     ext_dwsel1_l

     to transfer the two 128-bit octawords of data from the 21064 to the Bcache and expansion connector data busses.
   - Appropriate user signals on the external connector logic in order to write the data to the VRAM.
   - **ext_data_a4_l** to address the second location in the Bcache.
   - **ext_cack<2:0>_l** to terminate the 21064 write cycle.

   The EB64 memory controller remains in a wait loop during the whole of the cycle. It automatically regains control of the system when it detects **ext_cack<2:0>_l** as no longer idle.

4.2.5 DMA Through Expansion Connector

This section provides an overview of steps involved in performing read and write DMA transfers from a device on the expansion connector.

The external device is assumed to be transferring data to and from the EB64 main memory in response to an input stimuli. The transfers go directly to the DRAM main memory. The procedure for a system that requires Bcache probes can be derived from the programmed I/O example in Section 4.2.4. In addition to the Bcache control signals, the memory data transceiver (MDlatch) controls must be manipulated in order to transfer data between the Bcache data bus and the DRAM data bus, which is directly connected to the expansion connector data bus.

The bus acquisition and release sequences are described in Section 4.3.

1. Bus Acquisition
   a. Assert **ext_hold_l** and wait for the memory controller to complete its current bus cycle, indicated by a pulse of one clock period duration on **ext_start**.
   b. After **ext_start** has pulsed, gain control of the 21064 bus by asserting **ext_hold_req** and waiting for **ext_hold_ack**.
   c. While waiting for **ext_hold_ack** to assert, **ext_ioadr_dis** must be asserted in order to tristate the I/O address bus (**io_addr**) on the expansion connector.

Note: Avoiding Bus Contention

This timing is important. The asserted level of **ext_hold_ack** will drive **io_addr** in the opposite direction; therefore, **ext_ioadr_dis** must have been asserted earlier in order to avoid bus contention.

2. Data Transfers
   a. Data Read Transfer

      During this sequence a number of data units are read from the DRAM main memory and transferred to some data sink on the external connector logic. When a number of 128-bit data chunks are to be transferred, the transfer can be performed in DRAM page-mode. (This imposes some constraints in that a page boundary must not be crossed.) The sequence requires manipulating the **ext_ras_l**, **ext_cas<3:0>_l**, and **ext_sel_col_l** signals, with the I/O address bus (**io_addr**) supplying the row and then all subsequent column addresses.

The least significant column address line is controlled by either **ext_col_a0** or **ext_sel_col2_l**. The DRAM data lines are wired directly to the expansion connector.

Note: DRAM Refresh Requirements

The DMA engine cannot retain control of the bus for extended periods; doing so would compromise the CAS-before-RAS DRAM refresh cycles that must be performed under control of the memory controller.

---

b. Data Write Transfer

During this sequence a number of data units are read from some data source on the external connector logic and written into the DRAM main memory. The procedure is the same as for data read transfers (step 2.a), with the addition that **ext_data_we<3:0>_l** must be appropriately manipulated. Note that, on DMA writes to the DRAM main memory, a parity bit must be generated for each longword (four bytes) written.

Because some external device has had control of the bus, it may be necessary to clear the system lock flag. The lock flag is cleared by pulsing **ext_clr_flag_l** for one clock period.

3. Bus Release

Deassert **io_addr_dis**. Deassert **ext_hold_req** and assert **ext_io_over** for one clock cycle. This action signals the memory controller that it should regain control of the bus.

4.2.6 DRAM Address 0

Sections 3.2.1.1 through 3.2.1.4 give a functional description of DRAM addressing. This section gives an overview of the steps involved in generating DRAM address 0 from the 21064 and from a DMA device attached to the expansion connector.

Most of the DRAM address lines are switched between row and column addresses using a bank of multiplexers (schematic page *addr_mux.1*). For DRAM address 0, the column address must be switched from one value to another in order to page-mode access each 128-bits of the hexaword. This switching, along with the row and column multiplexing, is performed in the memory controller according to the following equation:

```
DRAM_ADDR0 = (!IO_WAIT & ADR13 & !SEL_COL & !SEL_COL2) #
             (!IO_WAIT & FIRST_COL_A0 & SEL_COL & !SEL_COL2) #
             (!IO_WAIT & !FIRST_COL_A0 & SEL_COL2) #
             ( IO_WAIT & ADR13 & !SEL_COL) #
             ( IO_WAIT & IO_COL_A0 & SEL_COL) #
             ( IO_WAIT & EXT_COL_A0 & SEL_COL & !EXT_SEL_COL2) #
             ( IO_WAIT & !EXT_COL_A0 & SEL_COL & EXT_SEL_COL2);
```

The individual lines (product terms) of this equation behave as follows:

- 21064 Access to DRAM (not **io_wait**)
  1. Select the row address, **adr13**.
  2. Do one of the following:
     - If a wrapped cycle is being performed (**first_col_a0** asserted), select the column address. The first column address strobed into the DRAM will be 1 (odd address); the second, when **sel_col2** is asserted, will be 0 (even address).
     - If a nonwrapped cycle is being performed (**first_col_a0** deasserted), select the column address. The first column address strobed into the DRAM will be 0 (even address); the second, when **sel_col2** is asserted, will be 1 (odd address). Assertion of **sel_col2** implies assertion of **sel_col**; therefore, **sel_col** is not required.
- DMA Device Access to the DRAM (**io_wait**)

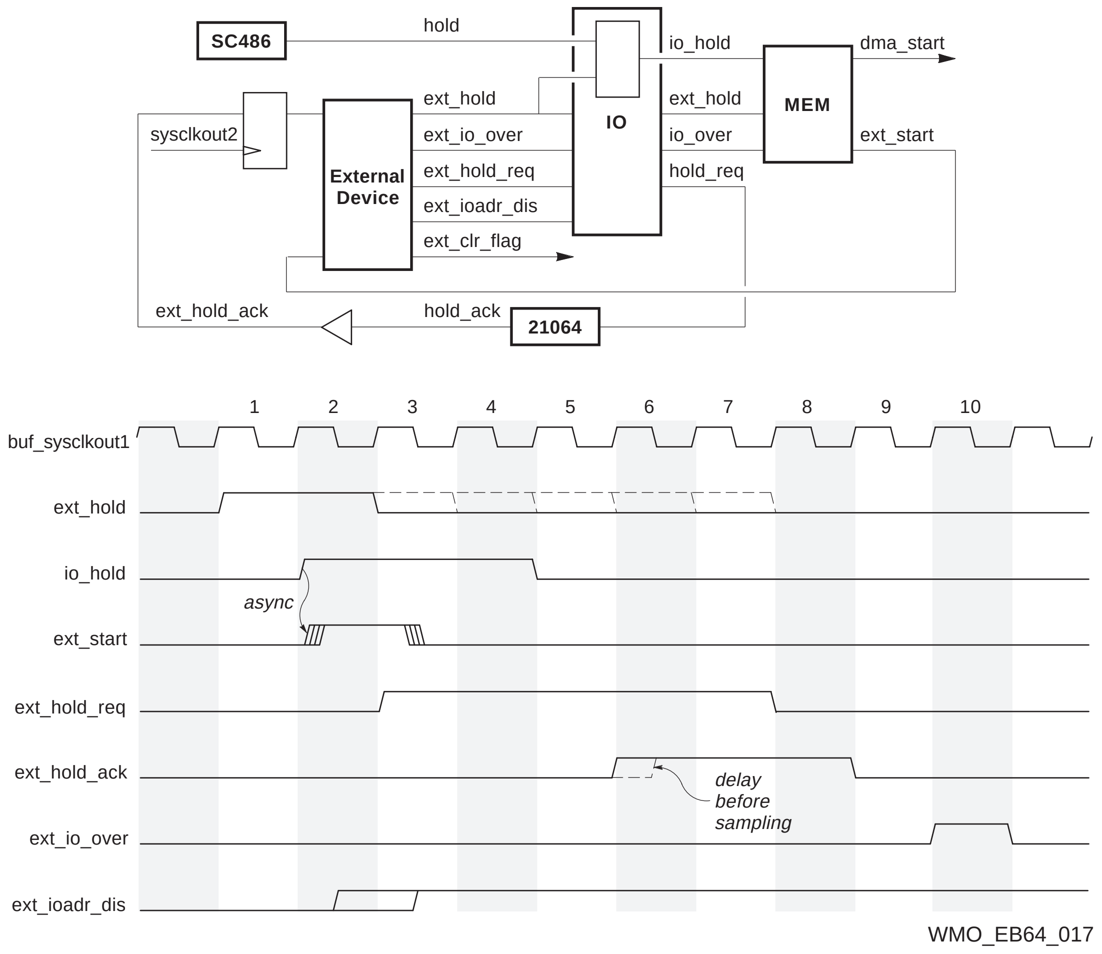
  The following steps are used during DMA accesses to the DRAM or during accesses to the upper quadrant of physical memory; that is, the expansion connector address space.
  1. Select the row address, **adr13**.
  2. Do one of the following:
     - Select the column address. Controlled by the EB64 I/O controller during DMA transfers by the SC486.
     - Select the column address. Controlled by signals from the expansion connector interface. When **sel_col** is asserted (as a result of **ext_sel_col**), **ext_col_a0** directly controls the value of the least significant DRAM column address. Signal **ext_sel_col2** is assumed to be deasserted.
     - Select the column address. Controlled by signals from the expansion connector interface. When **sel_col** is asserted (as a result of **ext_sel_col**), **ext_col_a0** is set to select the first DRAM column address (either 0 or 1); then **ext_sel_col2** toggles to select

the second (complementary) address. This mode of operation is useful when performing wrapped reads.

4.3 System Bus Acquisition and Release Sequence

Figure 4–5 is a simplified block and timing diagram showing the bus acquisition signals and timing.

Figure 4–5 System Bus Acquisition and Release



The acquisition sequence is:

1. Assert **ext_hold**.
2. Wait for **ext_start**.
3. Assert **ext_hold_req**.
4. Wait for **ext_hold_ack**.
5. Perform bus operations.

Note that **ext_start** is asserted for only one clock tick, and that **ext_hold** should be deasserted as soon as **ext_start** is detected.

The release sequence is:

1. Deassert **ext_hold_req** (**ext_hold_ack** deasserts).
2. Assert **ext_io_over** for one clock tick.

4.4 Expansion Connector Pin Lists

Table 4–2 Expansion Connector Pin List

Pin	Signal	Pin	Signal
J10-A1	*not connected*	J10-B1	buf_dat62
J10-A2	buf_dat63	J10-B2	buf_dat60
J10-A3	Vdd	J10-B3	buf_dat58
J10-A4	buf_dat61	J10-B4	gnd
J10-A5	buf_dat59	J10-B5	buf_dat56
J10-A6	gnd	J10-B6	buf_dat54
J10-A7	buf_dat57	J10-B7	Vdd
J10-A8	buf_dat55	J10-B8	buf_dat52
J10-A9	Vdd	J10-B9	buf_dat50
J10-A10	buf_dat53	J10-B10	gnd
J10-A11	buf_dat51	J10-B11	buf_dat48
J10-A12	gnd	J10-B12	buf_dat46
J10-A13	buf_dat49	J10-B13	Vdd
J10-A14	buf_dat47	J10-B14	buf_dat44
J10-A15	buf_dat45	J10-B15	buf_dat42
J10-A16	buf_dat43	J10-B16	gnd
J10-A17	buf_dat41	J10-B17	buf_dat40
J10-A18	gnd	J10-B18	buf_dat38

(continued on next page)

Table 4–2 (Cont.) Expansion Connector Pin List

Pin	Signal	Pin	Signal
J10-A19	buf_dat39	J10-B19	*not connected*
J10-A20	buf_dat37	J10-B20	buf_dat36
J10-A21	Vdd	J10-B21	buf_dat34
J10-A22	buf_dat35	J10-B22	gnd
J10-A23	buf_dat33	J10-B23	buf_dat32
J10-A24	gnd	J10-B24	buf_dat30
J10-A25	buf_dat31	J10-B25	Vdd
J10-A26	buf_dat29	J10-B26	buf_dat28
J10-A27	Vdd	J10-B27	buf_dat26
J10-A28	buf_dat27	J10-B28	gnd
J10-A29	buf_dat25	J10-B29	buf_dat24
J10-A30	gnd	J10-B30	buf_dat22
J10-A31	buf_dat23	J10-B31	Vdd
J10-A32	buf_dat21	J10-B32	buf_dat20
J10-A33	Vdd	J10-B33	buf_dat18
J10-A34	buf_dat19	J10-B34	gnd
J10-A35	buf_dat17	J10-B35	buf_dat16
J10-A36	gnd	J10-B36	buf_dat14
J10-A37	buf_dat15	J10-B37	buf_dat12
J10-A38	buf_dat13	J10-B38	buf_dat10
J10-A39	buf_dat11	J10-B39	buf_dat 8
J10-A40	buf_dat 9	J10-B40	*not connected*
J11-A1	*not connected*	J11-B1	buf_dat 6
J11-A2	buf_dat 7	J11-B2	buf_dat 4
J11-A3	Vdd	J11-B3	buf_dat 2
J11-A4	buf_dat 5	J11-B4	gnd
J11-A5	buf_dat 3	J11-B5	buf_dat 0
J11-A6	gnd	J11-B6	buf_par1
J11-A7	buf_dat 1	J11-B7	Vdd
J11-A8	buf_par3	J11-B8	buf_par0
J11-A9	Vdd	J11-B9	io_be3
J11-A10	buf_par2	J11-B10	gnd
J11-A11	ext_tagadr_we_l	J11-B11	io_be2
J11-A12	gnd	J11-B12	io_be1
J11-A13	ext_inhibit_tag_oe_l	J11-B13	Vdd
J11-A14	ext_sel_tagadr_l	J11-B14	io_be0
J11-A15	ext_data_a4_l	J11-B15	ext_dmapwe
J11-A16	ext_sel_col2_l	J11-B16	gnd

(continued on next page)

Table 4–2 (Cont.) Expansion Connector Pin List

Pin	Signal	Pin	Signal
J11-A17	ext_dinvreq_l	J11-B17	ext_irq1_l
J11-A18	gnd	J11-B18	ext_io_over_l
J11-A19	ext_irq2_l	J11-B19	*not connected*
J11-A20	ext_irq0_l	J11-B20	ext_cas3_l
J11-A21	Vdd	J11-B21	ext_cas1_l
J11-A22	ext_clr_flag_l	J11-B22	gnd
J11-A23	ext_ras_l	J11-B23	ext_col_a0_l
J11-A24	gnd	J11-B24	ext_we_l
J11-A25	ext_cas2_l	J11-B25	Vdd
J11-A26	ext_cas0_l	J11-B26	ext_md_rd_clken_l
J11-A27	Vdd	J11-B27	ext_md_wr_oe2_l
J11-A28	ext_sel_col_l	J11-B28	gnd
J11-A29	ext_md_wr_clken_l	J11-B29	ext_md_wr_oe0_l
J11-A30	gnd	J11-B30	ext_md_wr_par_oe_l
J11-A31	ext_md_wr_oe3_l	J11-B31	Vdd
J11-A32	ext_md_wr_oe1_l	J11-B32	ext_tagctl_we_l
J11-A33	Vdd	J11-B33	ext_tagctl_d_l
J11-A34	ext_md_rd_oe_l	J11-B34	gnd
J11-A35	ext_wrt_tagctl_oe_l	J11-B35	ext_tagctl_p_l
J11-A36	gnd	J11-B36	ext_data_we2_l
J11-A37	ext_tagctl_v_l	J11-B37	ext_data_we1_l
J11-A38	ext_tagctl_s_l	J11-B38	ext_data_we0_l
J11-A39	ext_data_oe_l	J11-B39	ext_wrt_tagadr_oe_l
J11-A40	ext_data_we3_l	J11-B40	*not connected*
J26-A1	ext_start	J26-B1	remote_reset_l
J26-A2	io_addr25	J26-B2	ext_ioadr_dis_l
J26-A3	io_addr24	J26-B3	io_addr2
J26-A4	Vdd	J26-B4	sysclk_pll_3
J26-A5	io_addr23	J26-B5	gnd
J26-A6	io_addr22	J26-B6	buf_clk1_b2
J26-A7	gnd	J26-B7	buf_clk1_l
J26-A8	io_addr21	J26-B8	Vdd
J26-A9	io_addr20	J26-B9	buf_clk2
J26-A10	Vdd	J26-B10	buf_clk2_l
J26-A11	io_addr19	J26-B11	gnd
J26-A12	io_addr18	J26-B12	dcok
J26-A13	gnd	J36-B13	ext_reset_l
J26-A14	io_addr17	J26-B14	Vdd

(continued on next page)

Table 4–2 (Cont.) Expansion Connector Pin List

Pin	Signal	Pin	Signal
J26-A15	io_addr16	J26-B15	ext_zone
J26-A16	Vdd	J26-B16	io_addr3
J26-A17	io_addr15	J26-B17	gnd
J26-A18	io_addr14	J26-B18	io_addr31
J26-A19	gnd	J26-B19	io_addr30
J26-A20	io_addr13	J26-B20	Vdd
J26-A21	io_addr12	J26-B21	io_addr29
J26-A22	Vdd	J26-B22	io_addr28
J26-A23	io_addr11	J26-B23	gnd
J26-A24	io_addr10	J26-B24	io_addr27
J26-A25	gnd	J26-B25	io_addr26
J26-A26	io_addr 9	J26-B26	Vdd
J26-A27	io_addr 8	J26-B27	io_addr7
J26-A28	Vdd	J26-B28	io_addr6
J26-A29	io_addr5	J26-B29	gnd
J26-A30	ext_del_creq2	J26-B30	ext_doe_l
J26-A31	gnd	J26-B31	ext_dwsel1_l
J26-A32	ext_del_creq1	J26-B32	Vdd
J26-A33	ext_del_creq0	J26-B33	ext_cwmask7
J26-A34	Vdd	J26-B34	ext_cwmask6
J26-A35	ext_drack2_l	J26-B35	gnd
J26-A36	ext_drack1_l	J26-B36	ext_cwmask5
J26-A37	gnd	J26-B37	ext_cwmask4
J26-A38	ext_drack0_l	J26-B38	Vdd
J26-A39	ext_cack2_l	J26-B39	ext_cwmask3
J26-A40	Vdd	J26-B40	ext_cwmask2
J26-A41	ext_cack1_l	J26-B41	gnd
J26-A42	ext_cack0_l	J26-B42	ext_cwmask1
J26-A43	gnd	J26-B43	ext_cwmask0
J26-A44	buf_tagctl_v	J26-B44	Vdd
J26-A45	buf_tagctl_s	J26-B45	ext_hit
J26-A46	Vdd	J26-B47	gnd
J26-A47	buf_tagctl_d	J26-B46	ext_hold_l
J26-A48	buf_tagctl_p	J26-B48	ext_hold_ack
J26-A49	gnd	J26-B49	io_addr4
J26-A50	ext_hold_req_l	J26-B50	ext_adr33_in
J27-A1	ext_be0	J27-B1	ext_be1
J27-A2	buf_dat127	J27-B2	wrap_read

(continued on next page)

Table 4–2 (Cont.) Expansion Connector Pin List

Pin	Signal	Pin	Signal
J27-A3	buf_dat125	J27-B3	buf_dat126
J27-A4	Vdd	J27-B4	buf_dat124
J27-A5	buf_dat123	J27-B5	gnd
J27-A6	buf_dat121	J27-B6	buf_dat122
J27-A7	gnd	J27-B7	buf_dat120
J27-A8	buf_dat119	J27-B8	Vdd
J27-A9	buf_dat117	J27-B9	buf_dat118
J27-A10	Vdd	J27-B10	buf_dat116
J27-A11	buf_dat115	J27-B11	gnd
J27-A12	buf_dat113	J27-B12	buf_dat114
J27-A13	gnd	J27-B13	buf_dat112
J27-A14	buf_dat111	J27-B14	Vdd
J27-A15	buf_dat109	J27-B15	buf_dat110
J27-A16	Vdd	J27-B16	buf_dat108
J27-A17	buf_dat107	J27-B17	gnd
J27-A18	buf_dat105	J27-B18	buf_dat106
J27-A19	gnd	J27-B19	buf_dat104
J27-A20	buf_dat103	J27-B20	Vdd
J27-A21	buf_dat101	J27-B21	buf_dat102
J27-A22	Vdd	J27-B22	buf_dat100
J27-A23	buf_dat 99	J27-B23	gnd
J27-A24	buf_dat 97	J27-B24	buf_dat 98
J27-A25	gnd	J27-B25	buf_dat 96
J27-A26	buf_dat95	J27-B26	Vdd
J27-A27	buf_dat93	J27-B27	buf_dat94
J27-A28	Vdd	J27-B28	buf_dat92
J27-A29	buf_dat91	J27-B29	gnd
J27-A30	buf_dat89	J27-B30	buf_dat90
J27-A31	gnd	J27-B31	buf_dat88
J27-A32	buf_dat87	J27-B32	Vdd
J27-A33	buf_dat85	J27-B33	buf_dat86
J27-A34	Vdd	J27-B34	buf_dat84
J27-A35	buf_dat83	J27-B35	gnd
J27-A36	buf_dat81	J27-B36	buf_dat82
J27-A37	gnd	J27-B37	buf_dat80
J27-A38	buf_dat79	J27-B38	Vdd
J27-A39	buf_dat77	J27-B39	buf_dat78
J27-A40	Vdd	J27-B40	buf_dat76
J27-A41	buf_dat75	J27-B41	gnd

(continued on next page)

Table 4–2 (Cont.) Expansion Connector Pin List

Pin	Signal	Pin	Signal
J27-A42	buf_dat73	J27-B42	buf_dat74
J27-A43	gnd	J27-B43	buf_dat72
J27-A44	buf_dat71	J27-B44	Vdd
J27-A45	buf_dat69	J27-B45	buf_dat70
J27-A46	Vdd	J27-B46	buf_dat68
J27-A47	buf_dat67	J27-B47	gnd
J27-A48	buf_dat65	J27-B48	buf_dat66
J27-A49	gnd	J27-B49	buf_dat64
J27-A50	ext_tl0	J27-B50	ext_tl1

Table 4–3 Expansion Connector Pin List — buf_data<127:0>

Bit	Pin	Bit	Pin	Bit	Pin	Bit	Pin
127	J27-A2	95	J27-A26	63	J10-A2	31	J10-A25
126	J27-B3	94	J27-B27	62	J10-B1	30	J10-B24
125	J27-A3	93	J27-A27	61	J10-A4	29	J10-A26
124	J27-B4	92	J27-B28	60	J10-B2	28	J10-B26
123	J27-A5	91	J27-A29	59	J10-A5	27	J10-A28
122	J27-B6	90	J27-B30	58	J10-B3	26	J10-B27
121	J27-A6	89	J27-A30	57	J10-A7	25	J10-A29
120	J27-B7	88	J27-B31	56	J10-B5	24	J10-B29
119	J27-A8	87	J27-A32	55	J10-A8	23	J10-A31
118	J27-B9	86	J27-B33	54	J10-B6	22	J10-B30
117	J27-A9	85	J27-A33	53	J10-A10	21	J10-A32
116	J27-B10	84	J27-B34	52	J10-B8	20	J10-B32
115	J27-A11	83	J27-A35	51	J10-A11	19	J10-A34
114	J27-B12	82	J27-B36	50	J10-B9	18	J10-B33
113	J27-A12	81	J27-A36	49	J10-A13	17	J10-A35
112	J27-B13	80	J27-B37	48	J10-B11	16	J10-B35
111	J27-A14	79	J27-A38	47	J10-A14	15	J10-A37
110	J27-B15	78	J27-B39	46	J10-B12	14	J10-B36
109	J27-A15	77	J27-A39	45	J10-A15	13	J10-A38
108	J27-B16	76	J27-B40	44	J10-B14	12	J10-B37
107	J27-A17	75	J27-A41	43	J10-A16	11	J10-A39
106	J27-B18	74	J27-B42	42	J10-B15	10	J10-B38
105	J27-A18	73	J27-A42	41	J10-A17	9	J10-A40
104	J27-B19	72	J27-B43	40	J10-B17	8	J10-B39
103	J27-A20	71	J27-A44	39	J10-A19	7	J11-A2

(continued on next page)

Table 4–3 (Cont.) Expansion Connector Pin List — buf_data<127:0>

Bit	Pin	Bit	Pin	Bit	Pin	Bit	Pin
102	J27-B21	70	J27-B45	38	J10-B18	6	J11-B1
101	J27-A21	69	J27-A45	37	J10-A20	5	J11-A4
100	J27-B22	68	J27-B46	36	J10-B20	4	J11-B2
99	J27-A23	67	J27-A47	35	J10-A22	3	J11-A5
98	J27-B24	66	J27-B48	34	J10-B21	2	J11-B3
97	J27-A24	65	J27-A48	33	J10-A23	1	J11-A7
96	J27-B25	64	J27-B49	32	J10-B23	0	J11-B5

Table 4–4 Expansion Connector Pin List — io_addr<31:2>

Bit	Pin	Bit	Pin	Bit	Pin	Bit	Pin
31	J26-B18	23	J26-A5	15	J26-A17	7	J26-B27
30	J26-B19	22	J26-A6	14	J26-A18	6	J26-B28
29	J26-B21	21	J26-A8	13	J26-A20	5	J26-A29
28	J26-B22	20	J26-A9	12	J26-A21	4	J26-B49
27	J26-B24	19	J26-A11	11	J26-A23	3	J26-B16
26	J26-B25	18	J26-A12	10	J26-A24	2	J26-B3
25	J26-A2	17	J26-A14	9	J26-A26		
24	J26-A3	16	J26-A15	8	J26-A27		

Table 4–5 Expansion Connector Pin List — Vdd and gnd

Connector	gnd Pins		Vdd Pins	
J26/J27	A7	B5	A4	B8
J26/J27	A13	B11	A10	B14
J26/J27	A19	B17	A16	B20
J26/J27	A25	B23	A22	B26
J26/J27	A31	B29	A28	B32
J26/J27	A37	B35	A34	B38
J26/J27	A43	B41	A40	B44
J26/J27	A49	B47	A46	
J10/J11	A6	B4	A3	B7
J10/J11	A12	B10	A9	B13
J10/J11	A18	B16	A21	B25
J10/J11	A24	B22	A27	B31

(continued on next page)

Table 4–5 (Cont.) **Expansion Connector Pin List — Vdd and gnd**

Connector	gnd Pins		Vdd Pins
J10/J11	A30	B28	A33
J10/J11	A36	B34	

5 Power Requirements

The EB64 derives its main system power from a user-supplied, industry-standard, PC power supply. It consumes 18 A of power. To calculate power supply needs for your system, add the current requirements (in amps) for each voltage required by each device installed in the system to the value of the raw EB64 board (18 A). Table 5–1 shows amps for each voltage.

Table 5–1 Voltage/Amp

Volts	Amps
5 V	18 A
-5 V	0 A
-12 V	.3 A
+12 V	1 A

See Section 3.4 and schematic pages *eval.16*, *eval.24*, and *eval.29* for more information.

Caution: Fan Sensor Required

The 21064 cooling fan *must* have a built-in sensor that will drive a signal if the air flow stops. The sensor is connected to the EB64 board (J17 Figure 2–1 and Table 2–1). When the signal is generated, it places the system into dcok mode. This action protects the CPU under fan-failure conditions, because the 21064 dissipates less heat in dcok mode. The fan supplied with the EB64 includes an air-flow sensor.

A
Technical Support, Ordering, and Associated Literature

This appendix explains how to:

- Call for DECchip information and technical support.
- Order DECchip parts.
- Order associated literature for the DECchip 21064 evaluation board and the DECchip 21064 microprocessor.

A.1 Information and Technical Support

Call the following phone numbers for information and technical support:

United States and Canada	**1–800–332–2717 (1–800–DEC–2717)**
TTY (United States only)	**1–800–332–2515 (1–800–DEC–2515)**
Outside North America	**+1–508–568–6868**

A.2 Ordering DECchip Products

To order the DECchip 21064 and related products, contact your local Digital sales office. Working with your sales representative, you may be able to take advantage of discounts and volume pricing.

You can order the following DECchip products from Digital.

Product	Speed	Order Number
DECchip 21064 Microprocessor	150 MHz	21064–AA
DECchip 21064 Microprocessor	200 MHz	21064–BA
DECchip 21064 Sample Kit	150 MHz	21064–SA
DECchip 21064 Sample Kit	200 MHz	21064–SB
Heat Sink Assembly	–	2106H–AA

A.3 Associated DECchip Literature

The following table lists DECchip literature in the these categories:

- DECchip 21064 Literature
- DECchip 21064 Evaluation Board Literature
- DECchip Marketing Literature

Title	Order Number
DECchip 21064 Literature	
DECchip 21064 Microprocessor Data Sheet	EB–N0136–72
DECchip 21064 Microprocessor Hardware Reference Manual	EC–N0079–72
DECchip 21064 PALcode System Design Guide	EC–N0543–72
Designing a Memory/Cache Subsystem for the DECchip 21064 Microprocessor: An Application Note	EC–N0301–72
Designing a System with the DECchip 21064 Microprocessor: An Application Note	EC–N0107–72
DECchip 21064 Evaluation Board Literature	
Calculating a System I/O Address for the DECchip 21064 Evaluation Board: An Application Note	EC–N0567–72
DECchip 21064 Bus Transactor User’s Guide	EC–N0448–72
DECchip 21064 Evaluation Board Debug Monitor User’s Guide	EC–N0392–72
DECchip 21064 Evaluation Board Design Package Read Me First	EC–N0352–72
DECchip 21064 Evaluation Board Product Brief	EC–N0353–72
DECchip 21064 SROM Mini-Debugger User’s Guide	EC–N0357–72
DECchip 21064 Evaluation Board User’s Guide	EC–N0351–72
DECchip 21064 Software Design Tools User’s Guide	EC–N0441–72
DECchip Marketing Literature	
AMP PGA Burn-In and Product Sockets for DECchip 21064 with Alpha AXP Architecture AXP	EC–X3013–72
DECchip Information Line Brochure	EB–N0109–72
DECchip Preprocessor for Hewlett-Packard Logic Analyzer	EC–X2454–72
DECchip Sample Kit Brochure	EB–N0108–72

Ordering Associated DECchip Literature

To order any of the previously listed DECchip literature, use FAX or mail.

FAX (available 24 hours)

1–508–351–4467

Mail

Digital Equipment Corporation
Order Administration NRO2–2/J6
444 Whitney Street
Northboro, MA 01532 USA

Send the following information by **FAX** or **mail**:

Name
Company name
Street address
City, state or county, and country
Document order number
Quantity

A.4 Associated Digital Literature

You can order and purchase the following literature from Digital.

Title	Order Number
Alpha Architecture Reference Manual	EY–L520E–DP–YCH

The following table explains how to order the previously listed Digital literature.

If you are from . . .	Then use this method to order . . .
The United States or Canada	Call **1–800–DIGITAL**.
Outside the United States	Contact your local Digital office, or technical or reference bookstore where Digital Press books are distributed by Prentice Hall.

A.5 Associated Third-Party Literature

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

Title	Vendor	Order Number
Single-Chip Ethernet Controller for ISA Data Sheet	Advanced Micro Devices P.O Box 3453 Sunnyvale, CA 94088 USA 1–800–222–9323 1–408–749–5703	Am79C960
Super I/O Floppy Disk Controllers Data Sheet	Standard Microsystems Corporation 80 Arkay Drive Hauppauge, NY 11788 USA 1–516–435–6000	FDC37C651
486 PC/AT-Compatible System Controller Data Sheet	VLSI Technology, Inc. 8375 South River Parkway Tempe, AZ 85284 USA 1–602–752–8574	VL82C486
SCAMP Combination I/O Data Sheet	VLSI Technology, Inc. 8375 South River Parkway Tempe, AZ 85284 USA 1–602–752–8574	VL82C113

Index

A

Abbreviations, bit and field, ix
Address
  bit
    definitions, 3–1
    description, 3–1
    33, external, 4–2
  bits 18 and 17, 3–12
  I/O, 4–9
  multiplexing, DRAM, 3–12
  row and column, 3–11
  space, 3–1
    I/O, 3–5
    I/O, debug ROM and system register, 3–6
    I/O, expansion connector, 3–7
    I/O, SC486 and interrupt acknowledge, 3–7
    main memory, 3–2
    main memory, Bcacheable, 3–2
    main memory, not Bcacheable, 3–3
    main memory, not Bcacheable or data cacheable, 3–4
    main memory, not for instruction fetch, 3–4
    mapping, I/O, 3–5
    mapping, sparse, 3–5
    partitioning, 3–2
    reads and writes to different memory space, 3–4
ads#, 3–7
Alpha AXP
  documentation, A–2
Associated literature, A–2

B

10Base2 Ethernet connector, 2–3
10BaseT Ethernet connector, 2–3
Battery backup connector for TOY clock, 2–4
Bcache
  cycles — see Memory/Bcache cycles
  index, 3–14
  size
    field, 3–23
    selection, 3–14, 3–23
  subsystem, 1–4, 3–14
  tag, 3–14
Bcacheable memory space, 3–2
Bit abbreviations, ix
Block diagram, EB64, 1–1
Board
  connectors and headers, 2–1
  power connector, 2–3
Buffered
  data, 4–2
    parity, 4–3
  delayed system clock, 4–2
  system clock, 4–2
  tag control bits, 4–3
buf_clk1_b2 (system clock), 4–2
buf_clk1_l (system clock), 4–2
buf_clk2 (delayed system clock), 4–2

buf_clk2_l (delayed system clock), 4–2
buf_data<127:0>, 4–2
    expansion connector pin list, 4–33
buf_par<3:0> (data parity), 4–3
buf_tagctl_<d,s,v,p> (tag control bits), 4–3
Bus
    acquisition, external DMA device, 4–24
    interface unit control (BIU_CTL) IPR, 3–3
        BC_EN bit, 3–24
        BC_PA_DIS field, 3–3, 3–17
        BC_RD_SPD field, 3–15
        BC_SIZE field, 3–23
        BC_WR_SPD field, 3–15
        OE bit, 3–16
        SYS_WRAP bit, 3–12, 3–17, 3–24
    master
        21064, 3–5
        DMA device, 3–5
    release, external DMA device, 4–25
Byte enable, 3–6
    external, 4–2
    I/O, 4–9

C

Cache
    see also Bcache
    block, x
    coherency design considerations, 4–16
        additional memory, 4–17
        DMA devices, 4–17
        I/O devices, 4–17
    line, x
cack_h<2:0> (command acknowledge), 3–19
CAS slip field, 3–23
Caution
    convention, ix
    do not read and write to different memory space, 3–4
    do not write and read to different memory space, 3–4
    fan sensor required, 2–4, 3–28, 5–1
    not for instruction fetch, 3–4
    set BIU_CTL OE, 3–16
Caution (cont'd)
    setting Bcache off/on, 3–24
    setting data wrapping mode, 3–12, 3–17, 3–24
    to avoid component damage, 3–16
check<21,14,7,0> generation, 4–13
Clear lock flag, external, 4–4
Clocks
    expansion connector, 4–18
    real-time, 1–5
    system, 3–30
        buffered, 4–2
        delayed, buffered, 4–2
        distribution, 3–31
        timing, 3–31
Column
    0, 3–11
    address, 3–11
        0, external, 4–4
        strobe (CAS), external, 4–3
Command acknowledge, 3–19
Components, EB64, 1–1
Connectors and headers, 2–1
Controller
    DMA, 1–4, 3–26
    floppy-disk, 1–5, 3–26
    interrupt, 1–4, 3–26
    ISA bus, 1–4, 3–26
Conventions, ix
    bit abbreviations, ix
    bit notation, ix
    caution, ix
    extents, x
    field abbreviations, ix
    note, x
    numbering, x
    ranges, x
    read only (RO), ix
    read/write (RW), ix
    schematic references, x
    signal names, xi
    write only (WO), ix
Cooling fan
    power and sensor connector, 2–4
    sensor, 2–4, 3–28, 5–1

cwmask1 (cycle write mask1), 3–12
Cycle
  acknowledge, external, 4–3
  request, delayed, external, 4–5
  write mask, 3–12
    external, 4–4

D

Data
  wrapping—see Wrap, wrap_read
  a4, external, 4–4
  buffered, 4–2
  bus read acknowledge, 3–3
  output enable, external
    ext_data_oe_l, 4–5
    ext_doe_l, 4–6
  parity, buffered, 4–3
  path width, design considerations, 4–16
  read acknowledge, external, 4–6
  transfers
    external DMA device, 4–24
    external DMA device, read, 4–24
    external DMA device, write, 4–25
  units, ix
  write
    enable, external, 4–5
    select, external, 4–7
Data cache
  backmap write enable, external, 4–6
  invalidate request, external, 4–5
d/c#, 3–7
dcok, 4–5
  reset, 3–28
    switch connector, 2–4
Debug
  monitor, 1–5
  ROM, 1–2, 1–5, 3–25
    address zone, 3–6
    byte field, 3–21
    selection, 3–25
DECchip
  documentation, A–2
DECchip information, A–2
Design considerations
  cache coherency, 4–16
    additional memory, 4–17
    DMA devices, 4–17
    I/O devices, 4–17
  data path width, 4–16
  expansion connector hardware, 4–15
Direct memory access (DMA)
  controller, 1–4, 3–26
  cycle, 3–19
  device is bus master, 3–5
  devices, cache coherency, 4–17
  through expansion connector, 4–24
    bus acquisition, 4–24
    bus release, 4–25
    data transfers, 4–24
    lock flag, 4–25
Documentation, A–2
doe_1 (data output enable) generation, 4–13
doe_dis signal name change, 4–13
drack_h<2:0> (data bus read acknowledge), 3–3
Drive-type jumper, 2–3
dwsel (data bus write select) signal name change, 4–13
dwsel1 (data bus write select 1) generation, 4–13
Dynamic RAM (DRAM), 1–1
  address multiplexing, 3–12
  SIMM connectors, 2–4

E

EB64
  block diagram, 1–1
  components, 1–1
  features, 1–4
  uses, 1–3
Ethernet
  connector
    10Base2, 2–3
    10BaseT, 2–3
  interrupt, 3–27
  LEDs, 2–3

Ethernet (cont'd)
port, 1–5, 3–27
Expansion
connector
DMA through — see Direct memory access
programmed I/O, see Programmed I/O
address zone, 3–7
clocks, 4–18
control setup, 4–18
design considerations for hardware, 4–15
pin lists, 4–28
buf_data<127:0>, 4–33
gnd, 4–34
io_addr
Vdd, 4–34
signal description, 4–1
signals, 4–1, 4–15
interface, 1–5, 4–1
21064, 4–15
connector, 2–4
21064 expansion interface, 4–15
Extents convention, x
External
address bit 33, 4–2
byte enable, 4–2
clear lock flag, 4–4
column address
0, 4–4
strobe (CAS), 4–3
cycle
acknowledge, 4–3
request, delayed, 4–5
write mask, 4–4
data
a4, 4–4
output enable, 4–5, 4–6
read acknowledge, 4–6
write enable, 4–5
write select, 4–7
data cache
backmap write enable, 4–6
invalidate request, 4–5
hit, 4–7
External (cont'd)
hold, 4–7
acknowledge, 4–7
request, 4–8
I/O
address disable, 4–8
over, 4–9
inhibit tag output enable, 4–8
interrupt request, 4–9
memory data
read clock enable, 4–9
read output enable, 4–10
write clock enable, 4–10
write output enable, 4–10
write parity output enable, 4–10
reset, 4–10
row address strobe (RAS), 4–10
select
column, 4–11
column 2, 4–11
tag address, 4–11
start, 4–11
tag
address write enable, 4–12
control bits, 4–12
control write enable, 4–12
transfer length, 4–12
write
enable, 4–13
tag address output enable, 4–13
tag control output enable, 4–13
zone, 4–13
ext_adr33_in (address bit 33), 4–2
ext_be<1:0> (external byte enable), 4–2
ext_cack<2:0>_l (cycle acknowledge), 4–3
ext_cas<3:0>_l (column address strobe), 4–3
ext_clr_flag_l (clear lock flag), 4–4
ext_col_a0_l (column address 0), 4–4
ext_cwmask<7:0> (cycle write mask), 4–4
ext_data_a4_l (data SRAM address 4), 4–4
ext_data_oe_l (data output enable), 4–5
ext_data_we<3:0>_l (data write enable), 4–5

ext_del_creq<2:0> (delayed cycle request), 4–5
ext_dinvreq_l (data cache invalidate request), 4–5
ext_dmapwe (data cache backmap write enable), 4–6
ext_doe_l (data output enable), 4–6
ext_drack<2:0>_l (data read acknowledge), 4–6
ext_dwsel1_l (data write select), 4–7
ext_hit, 4–7
ext_hold_ack (hold acknowledge), 4–7
ext_hold_l, 4–7
ext_hold_req_l (hold request), 4–8
ext_inhibit_tag_oe_l (inhibit tag output enable), 4–8
ext_ioadr_dis_l (I/O address disable), 4–8
ext_io_over_l, 4–9
ext_irq<2:0>_l (interrupt request), 4–9
ext_md_rd_clken_l (memory read clock enable), 4–9
ext_md_rd_oe_l (memory read output enable), 4–10
ext_md_wr_clken_l (memory write clock enable), 4–10
ext_md_wr_oe<3:0>_l (memory write output enable), 4–10
ext_md_wr_par_oe_l (memory write parity output enable), 4–10
ext_ras_l (row address strobe), 4–10
ext_reset_l, 4–10
ext_sel_col2_l (select column 2), 4–11
ext_sel_col_l (select column 1), 4–11
ext_sel_tagadr_l (select tag address), 4–11
ext_start, 4–11
ext_tagadr_we_l (tag address write enable), 4–12
ext_tagctl_<d,s,v,p>_l (tag control bits), 4–12
ext_tagctl_we_l (tag control write enable), 4–12
ext_tl<1:0> (external transfer length), 4–12
ext_we_l (write enable), 4–13
ext_wrt_tagadr_oe_l (write tag address output enable), 4–13
ext_wrt_tagctl_oe_l (write tag control output enable), 4–13
ext_zone, 4–13

F

Fan
  power and sensor connector, 2–4
  sensor, 2–4, 3–28, 5–1
Features, EB64, 1–4
Field abbreviations, ix
Floppy-disk
  connector, 2–5
  controller, 1–5, 3–26
  interrupt, 3–27

G

gen_par<3:0> generation, 4–13
gnd, 4–7
  expansion connector pin list, 4–34

H

Headers and connectors, 2–1
Hexaword DRAM
  read cycle, 3–17
  write cycle, 3–18
Hit, external, 4–7
Hold
  acknowledge, external, 4–7
  external, 4–7
  generation, 4–13
  request, external, 4–8

I

I/O
  programmed through the expansion connector, see Programmed I/O
  address, 4–9
    disable, external, 4–8
    space mapping, 3–5
  byte enable, 3–6, 4–9

I/O (cont'd)
devices, 3–20
cache coherency considerations, 4–17
system register, 3–20
over, external, 4–9
space
debug ROM and system register, 3–6
expansion connector, 3–7
SC486 and interrupt acknowledge, 3–7
transfer length, 3–6
Idle loop, programmed I/O, 4–21
Initialization, 3–28
Instruction fetch, not Bcacheable or data cacheable memory space, 3–4
Interface
expansion — see Expansion interface
ISA, 1–4, 3–26
Internal processor register (IPR), 3–3
Interrupts, 3–27
acknowledge address zone, 3–7
controller, 1–4, 3–26
Ethernet, 3–27
interrupt request, external, 4–9
IRQ0, 3–27
IRQ1, 3–27
IRQ2, 3–27
IRQ<5:3>, 3–27
ISA
INTR, 3–27
NMI, 3–27
real-time clock, 3–27
SC486, 3–27
io_addr
<31:2>, 4–9
expansion connector pin list, 4–34
field, 3–22
io_be<3:0> (I/O byte enable), 4–9
io_hold signal name change, 4–13
ISA
bus controller, 1–4, 3–26
interface, 1–4, 3–26
INTR interrupt, 3–27
NMI interrupt, 3–27
slot, 1–5

ISA (cont'd)
slot <1:0> connector, 2–4

K

Keyboard, 1–5
and mouse connector, 2–5

L

Light-emitting diode (LED), Ethernet, 2–3
Literature, A–2
Load locked cycle, 3–18
Lock flag, 3–18
external DMA device, clear, 4–25
Logic analyzer header, 2–6
lw0_par<3:0> signal name change, 4–13

M

Master Mode, 3–19
Memory
cache coherency design considerations for additional memory, 4–17
data
read clock enable, external, 4–9
read output enable, external, 4–10
wrapping, 3–24
write clock enable, external, 4–10
write output enable, external, 4–10
write parity output enable, external, 4–10
main memory subsystem, 3–8
read slip cycles, 3–8
size, 1–4
selection, 3–8, 3–24
space
Bcacheable, 3–2
not Bcacheable, 3–3
not Bcacheable or data cacheable, 3–4
not for instruction fetch, 3–4
reads and writes to different memory space, 3–4
subsystem, 1–4, 3–8
type field, 3–24

Memory/Bcache
  cycles, 3–17
    DMA cycle, 3–19
    hexaword DRAM read, 3–17
    hexaword DRAM write, 3–18
    load locked cycle, 3–18
    refresh cycle, 3–20
    store conditional cycle, 3–19
    victim write cycle, 3–18
m/io#, 3–7
Mouse, 1–5
  and keyboard connector, 2–5

N

Note convention, x
Numbering convention, x

O

Ordering information, A–2
Order numbers, A–2

P

par<3:0> signal name change, 4–13
Part numbers, A–2
Peripheral devices, 1–5, 3–26
Ports
  Ethernet, 1–5, 3–27
  serial, 1–5, 3–26
Power, 3–28
  monitor, 3–28
  requirements, 5–1
Precompensation jumper, 2–3
Programmable timer, 1–5
Programmed I/O through the expansion connector, 4–19
  idle loop, 4–21
  read cycle, 4–21
  read fill sequence, 4–22
  victim write sequence, 4–21
  write block, 4–23
PROM
  see also Debug ROM
  jumper
    0 or 1, 2–3
    size, 2–5

R

RAM, see
  Dynamic RAM (DRAM), Memory
  Serial RAM (SRAM), Bcache
  Video RAM (VRAM)
Ranges convention, x
rdy#, 3–7
Read
  data wrapping—see Wrap, wrap_read
  cycle, programmed I/O, 4–21
  fill
    sequence, programmed I/O, 4–22
    single, x, 3–4
  only (RO) convention, ix
  slip cycles, 3–8
  write (RW) convention, ix
Reads and writes to different memory space, 3–4
Real-time clock, 1–5
  interrupts, 3–27
Refresh
  cycle, 3–20
  timer, 1–5
Related documentation, A–2
remote_reset_l, 4–10
Reset, 3–28
  dcok, 3–28
    switch connector, 2–4
  external, 4–10
  remote, 4–10
  (rst), 4–10
  system, 3–28
    switch connector, 2–4
ROM
  see Debug ROM, PROM, Serial ROM

Row address, 3–11
  strobe (RAS), external, 4–10
rst (reset), 4–10

S

SC486
  address zone, 3–7
  definition, 3–26
Schematic references
  convention, x
Select
  column, external, 4–11
  column 2, external, 4–11
  tag address, external, 4–11
Serial
  ports
    1 and 2 connectors, 2–4
    I/O, 1–5, 3–26
    interrupt, 3–27
    SROM, 3–29
  RAM (SRAM) damage caution, 3–16
  ROM (SROM), 1–5, 3–28
    serial port connector, 2–4
    serial port diagram, 3–29
Signal name
  changes, 4–13
    doe_dis, 4–6, 4–13
    dwsel, 4–7, 4–13
    io_hold, 4–7, 4–13
    lw0_par<3:0>, 4–3, 4–13
    par<3:0>, 4–3, 4–13
  convention, xi
Single
  in-line memory module (SIMM)
    connectors, 2–4
  read fill, x, 3–4
Slow-speed peripheral devices, 1–5, 3–26
Sparse address mapping, 3–5
Speaker connector, 2–3
Start, external, 4–11
Store conditional cycle, 3–19
sysclk_pll_3, 4–12
System
  block diagram, 1–1
  bus acquisition and release sequence,
    4–27
  clocks, 3–30
    buffered, 4–2
    delayed, buffered, 4–2
    distribution, 3–31
    PLL 3, 4–12
    timing, 3–31
  components, 1–1
  configuration jumpers, 2–5
  features, 1–4
  power, 3–28
  register, 3–20
    address zone, 3–6
    Bcache size field, 3–23
    bit description, 3–21
    CAS slip field, 3–23
    io_addr field, 3–22
    memory type field, 3–24
    ROM byte field, 3–21
    wrap read bit, 3–24
  reset, 3–28
    switch connector, 2–4
  uses, 1–3

T

Tag
  address
    bits tagadr<33:32>, 3–16
    select, external, 4–11
    write enable, external, 4–12
    write output enable, external, 4–13
  control
    write enable, external, 4–12
    write output enable, external, 4–13
  control bits
    buffered, 4–3
    external, 4–12
  inhibit output enable, external, 4–8
tagctl_<d,s,v,p>
  buffered, 4–3
  external, 4–12

Technical support, A–2
Time-of-year (TOY) clock battery backup
connector, 2–4
Timer
  programmable, 1–5
  refresh, 1–5
Transfer length, 3–6
  external, 4–12

U

Uses, EB64, 1–3

V

Vdd, 4–12
  expansion connector pin list, 4–34
Victim write
  cycle, 3–18
  sequence, programmed I/O, 4–21
Video RAM (VRAM), 4–17, 4–21

W

w/r#, 3–7
Wrap
  data wrapping mode, 3–12, 3–17, 3–24
  read, 3–17
    bit, 3–24
    data wrapping, 3–24
wrap_read, 4–13
Write
  block, programmed I/O, 4–23
  enable, external, 4–13
  only (WO) convention, ix
  tag address output enable, external, 4–13
  tag control output enable, external, 4–13

Z

Zone, external, 4–13