Developer Note

# Power Macintosh 9500 Computers

Power Macintosh 9500/120 Computer
Power Macintosh 9500/132 Computer

Developer Press
© Apple Computer, Inc. 1995

# Contents

# Figures and Tables

# About This Developer Note

This developer note describes the Power Macintosh 9500/120 and 9500/132 computers and compares them with the earlier Power Macintosh models, emphasizing the features that are new or different.

This developer note is intended to help hardware and software developers design products that are compatible with the Macintosh products described in the note. If you are not already familiar with Macintosh computers or if you would simply like more technical information, you may wish to read the supplementary reference documents described in this preface.

This note is published in two forms: an online version included with the Apple Developer CD and a looseleaf paper version published by APDA. For information about APDA, see “Supplemental Reference Documents.”

## Contents of This Note

This developer note has six chapters.

- Chapter 1, “Introduction,” gives a summary of the features of the Power Macintosh 9500 computers, describes their appearance, and lists the available configurations and options.
- Chapter 2, “Architecture,” describes the internal organization of the computers. It includes a block diagram and descriptions of the main components of the logic board.
- Chapter 3, “I/O Features,” describes the built-in I/O devices and the external I/O ports.
- Chapter 4, “Expansion Features,” describes the memory expansion SIMMS and the PCI expansion slots in the Power Macintosh 9500 computers.
- Chapter 5, “Software Features,” summarizes the new features of the ROM software and system software that accompany the Power Macintosh 9500 computers.
- Chapter 6, “Large Volume Support,” describes the modified software that allows the Power Macintosh 9500 computers to support volume sizes larger than 4 GB.

## Supplemental Reference Documents

The following documents provide information that complements or extends the information in this developer note.

### Apple Publications

For information about the earlier Power Macintosh computers, refer to *Macintosh Developer Note Number 8,* APDA catalog number R0566LL/A. Information about the enhanced versions of those computers is included in *Macintosh Developer Note Number 11,* APDA catalog number R0628LL/A.

For information about PCI expansion cards, refer to *Designing PCI Cards and Drivers for Power Macintosh Computers.*

For more information about the ADB and the serial ports, you may wish to refer to *Guide to the Macintosh Family Hardware,* second edition.

Developers may also need copies of the appropriate Apple reference books. You should have the relevant books of the *Inside Macintos*h series, particularly *Inside Macintosh: Devices* and *Inside Macintosh: Operating System Utilities.*

### Obtaining Information From APDA

The Apple publications listed above are available from APDA. APDA is Apple Computer's worldwide source for hundreds of development tools, technical resources, training products, and information for anyone interested in developing applications on Apple platforms. Customers receive the *APDA Tools Catalog* featuring all current versions of Apple development tools and the most popular third-party development tools. APDA offers convenient payment and shipping options, including site licensing.

To order products or to request a complimentary copy of the *APDA Tools Catalog*, contact

APDA
Apple Computer, Inc.
P.O. Box 319
Buffalo, NY 14207-0319

| | |
|---|---|
| Telephone | 1-800-282-2732 (United States)<br>1-800-637-0029 (Canada)<br>716-871-6555 (International) |
| Fax | 716-871-6511 |
| AppleLink | APDA |
| America Online | APDAorder |
| CompuServe | 76666,2405 |
| Internet | APDA@applelink.apple.com |

### Other Publications

For information about programming the PowerPC™ 601 microprocessor, developers should have copies of Motorola's *PowerPC 601 RISC Microprocessor User's Manual.* Information specific to the PowerPC 604 is published in *PowerPC 604 Microprocessor Implementation Features (Book IV).*

For mechanical specifications of the 8-byte DIMM, refer to the MO-161 specification of the JEDEC JC-11 committee. Electrical specifications are defined by the JEDEC JC-42.5 committee; see JEDEC Standard No. 21-C.

For codec standards, refer to the *ASCO 2300 Audio-Stereo Codec Specification* from IT&T.

For information about the PCI expansion bus, refer to the *PCI Local Bus Specification*, Revision 2.0, and *PCI Bus Binding to IEEE 1275-1994.* You can obtain these documents from

PCI Special Interest Group
Intel Corporation
M/S HF3-15A
5200 NE Elam Young Parkway
Hillsboro, Oregon 97124-6497
Telephone 800-433-5177 (U.S.)
503-797-4207 (International)

For information about the open firmware startup process, see *1275-1994 Standard for Boot (Initialization, Configuration) Firmware,* IEEE part number DS02683. It is referred to in this developer note as IEEE Standard 1275. You can order a copy from

IEEE Standards Department
445 Hoes Lane, P.O. box 1331
Piscataway, NJ 08855-1331
Telephone 800-678-4333

## Conventions and Abbreviations

This developer note uses the following typographical conventions and abbreviations.

### Typographical Conventions

Computer-language text—any text that is literally the same as it appears in computer input or output—appears in `Courier` font.

Hexadecimal numbers are preceded by a dollar sign ($). For example, the hexadecimal equivalent of decimal 16 is written as $10.

**Note**
A note like this contains information that is interesting but not essential for an understanding of the text. ◆

**IMPORTANT**
A note like this contains important information that you should read before proceeding. ▲

## Standard Abbreviations

When unusual abbreviations appear in this book, the corresponding terms are also spelled out. Standard units of measure and other widely used abbreviations are not spelled out. Here are the standard units of measure used in this developer note:

| | | | |
|---|---|---|---|
| A | amperes | mA | milliamperes |
| dB | decibels | µA | microamperes |
| GB | gigabytes | MB | megabytes |
| Hz | hertz | MHz | megahertz |
| in. | inches | mm | millimeters |
| k | 1000 | ms | milliseconds |
| K | 1024 | µs | microseconds |
| KB | kilobytes | ns | nanoseconds |
| kg | kilograms | Ω | ohms |
| kHz | kilohertz | sec. | seconds |
| kΩ | kilohms | V | volts |
| lb. | pounds | W | watts |

Other abbreviations used in this note include:

| | |
|---|---|
| $*n* | hexadecimal value *n* |
| AC | alternating current |
| ADB | Apple Desktop Bus |
| A/D | analog to digital |
| ADC | analog-to-digital converter |
| AGC | automatic gain control |
| AGND | analog ground |
| API | application program interface |
| AUI | auxiliary unit interface |
| AWAC | audio waveform amplifier and converter |
| CAS | column address strobe (a memory control signal) |
| CD-ROM | compact-disc read-only memory |

| | |
|---|---|
| CISC | complex instruction set computing |
| CPU | central processing unit |
| DAC | digital-to-analog converter |
| DBDMA | descriptor-based direct memory access |
| DC | direct current |
| DIMM | dual inline memory module |
| DIN | Deutsche Industrie Normal |
| DMA | direct memory access |
| DRAM | dynamic RAM |
| FIFO | first in, first out |
| FPU | floating-point unit |
| GND | ground |
| GPi | general-purpose input |
| IC | integrated circuit |
| I/O | input and output |
| IR | infrared |
| ISA | Industry Standard Architecture |
| ISDN | Integrated Services Digital Network |
| JEDEC | Joint Electron Device Engineering Council |
| LS TTL | low-power Schottky TTL (a standard type of device) |
| MACE | Media Access Controller for Ethernet |
| MMU | memory management unit |
| NBP | Name-Binding Protocol |
| n.c. | no connection |
| NMI | nonmaskable interrupt |
| NTSC | National Television System Committee (the standard system used for broadcast TV in North America and Japan) |
| NVRAM | non-volatile random-access memory |
| PAL | Phase Alternate Lines (the standard for broadcast TV in most of Europe, Africa, South America, and southern Asia) |
| PB API | parameter-block application program interface |
| PCI | Peripheral Component Interconnect, an industry-standard expansion bus |
| PRAM | parameter random-access memory |
| PROM | programmable read-only memory |
| PWM | pulse width modulation |
| RAM | random-access memory |
| RAMDAC | random-access memory, digital to analog converter |
| RAS | row address strobe |

| | |
|---|---|
| RGB | red-green-blue, a video signal format with separate red, green, and blue color components |
| RISC | reduced instruction set computing |
| rms | root-mean-square |
| ROM | read-only memory |
| SCC | Serial Communications Controller |
| SCSI | Small Computer System Interface |
| SECAM | the standard system used for broadcast TV in France and the former Soviet countries |
| SIMM | single inline memory module |
| SNR | signal to noise ratio |
| SPR | special-purpose register |
| SVGA | super video graphics adapter |
| S-video | a type of video connector that keeps luminance and chrominance separate; also called a Y/C connector |
| SWIM | Super Woz Integrated Machine (custom IC that controls the floppy disk interface) |
| TTL | transistor-transistor logic (a standard type of device) |
| VCC | positive supply voltage (voltage for collectors) |
| VCR | video cassette recorder |
| VGA | video graphics adapter |
| VRAM | video RAM; used for display buffers |
| Y/C | a type of video connector that keeps luminance and chrominance separate; also called an S-video connector |
| YUV | a video signal format with separate luminance and chrominance components |

CHAPTER 1

# Introduction

The Power Macintosh 9500/120 and 9500/132 computers are the new high-performance models in the Power Macintosh family of computers. They have been designed to provide greater performance and flexibility than existing Power Macintosh models. This chapter introduces the appearance and features of the Power Macintosh 9500 computers and compares those computers with earlier Power Macintosh models.

## Appearance and Features

The Power Macintosh 9500 computers have a tower case design with room for two additional internal storage devices and six expansion cards. The case is similar to that of the Power Macintosh 8100 computers, but is about 3 inches taller. Figure 1-1 and Figure 1-2 show front and back views of the computer.

**Figure 1-1** Front view

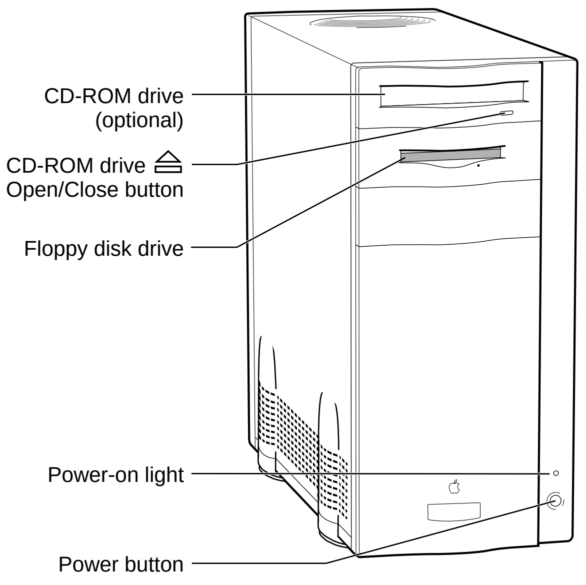
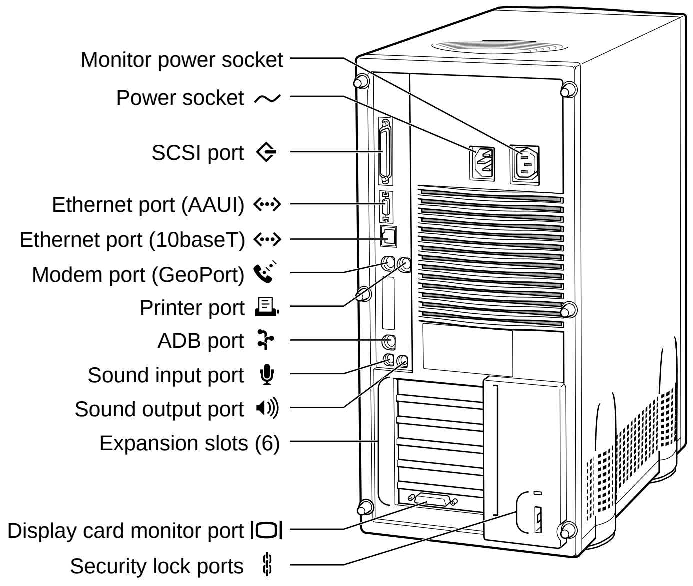


**Figure 1-2** Back view



The following list is a summary of the hardware features of the Power Macintosh 9500 computers. Each of these features is described later in this developer note.

- **Processor.** Both computers have a PowerPC™ 604 microprocessor. The clock frequency is 120 MHz in the Power Macintosh 9500/120 and 132 MHz in the Power Macintosh 9500/132.
- **Processor upgrade.** The processor subsystem is on a replaceable card for easy upgrading to a more advanced microprocessor or coprocessor.
- **Cache SIMM.** Both computers have a second level (L2) cache comprising 512 KB of fast RAM.
- **RAM.** The Power Macintosh 9500/120 comes with 16 MB of main RAM; the Power Macintosh 9500/132 comes with 32 MB.
- **RAM expansion.** Both computers have twelve SIMM slots for RAM expansion up to 1.5 GB.
- **I/O expansion.** Both computers have six expansion slots that conform to PCI V2.0 specifications.
- **Video monitor support.** The computer requires a display card in a PCI slot; the Power Macintosh 9500/120 comes with a video card that has 2 or 4 MB of VRAM and provides up to 24 bpp on a 21-inch monitor.
- **Standard I/O ports.** Both computers have two GeoPort serial ports, an ADB port, stereo sound input and output jacks, a SCSI port, and an Ethernet port (AAUI or 10baseT).

- **Fast internal SCSI.** The internal SCSI bus supports transfer rates up to 10 MB/sec. (The external SCSI bus supports transfer rates up to 5 MB/sec.)
- **Hard disk.** The internal hard disk has a capacity of 1 to 2 GB.
- **CD-ROM drive.** A built-in CD-ROM drive is optional.

## Comparison With Earlier Power Macintosh Computers

This section summarizes the features of the Power Macintosh 9500 computers and compares them with the features of the Power Macintosh 6100, 7100, and 8100.

While the Power Macintosh 9500 computers have many of the same features as the earlier models, they also have important new features. The major new features are

- a processor subsystem on a replaceable card for easy upgrading of the microprocessor
- a memory system using 64-bit SIMMs and a 128-bit data bus for higher performance
- interfaces to I/O devices and expansion cards using the industry-standard PCI bus

Table 1-1 compares the main features of the Power Macintosh 9500 computers with those of the earlier Power Macintosh computers.

**Table 1-1** Comparison with earlier Power Macintosh computers

| Feature | Power Macintosh 6100, 7100, and 8100 computers | Power Macintosh 9500/120 and 9500/132 computers |
|---|---|---|
| Processor type | PowerPC 601 | PowerPC 604 |
| Processor upgrade | None | By replacing processor subsystem card |
| Second-level cache | Up to 256 KB | 512 KB |
| Type of RAM SIMMs | 72-pin (32-bit data bus) | 168-pin (8-byte data bus) |
| Maximum amount of RAM | 72–264 MB | 1.5 GB |
| Maximum amount of VRAM | 2 or 4 MB | 2 or 4 MB (on PCI display card) |
| Support for 21-inch monitors | None, 16 bpp, or 24 bpp | Optional; depends on PCI video card |
| DAV connector | Yes | No |
| Sound | 16-bit, 44.1 MHz, stereo input and output | Same |
| Internal hard disk | 160 MB to 1 GB | 1 to 2 GB |
| Additional internal drives | One 5.25-inch, one or two 3.5-inch | Same |

*continued*

**Table 1-1** Comparison with earlier Power Macintosh computers (continued)

| Feature | Power Macintosh 6100, 7100, and 8100 computers | Power Macintosh 9500/120 and 9500/132 computers |
|---|---|---|
| CD-ROM drive | Built-in on some models | Same except 4X speed |
| SCSI buses | 1 fast internal, 1 external | Same |
| DMA for I/O devices | Yes | Same |
| Network port | Ethernet (AAUI) | Ethernet (AAUI and 10baseT) |
| GeoPort | 2 serial ports | Same |
| Number and types of expansion slots | 1–3 NuBus slots; DAV connector in some models | 6 PCI slots |

**Note**
The maximum amount of main RAM that can be installed in the Power Macintosh 9500 computers is 1.5 GB; the maximum memory size supported by the current Mac OS is 1 GB. ◆

## Configurations and Options

Table 1-2 shows the standard configurations for the Power Macintosh 9500 computers at the time this developer note was written.

The configuration that does not include the Apple video card is intended for users who wish to install a third-party video card. The Power Macintosh 9500/132 computer is compatible with video cards that comply with the *PCI Local Bus Specification,* Revision 2.0. The video card must include startup ROM that conforms to the IEEE Standard 1275. For information about obtaining these standards, see page xi.

**Table 1-2** Configurations

| Model | Installed DRAM | Size of hard disk | Size of L2 cache | Apple video card installed | Internal CD-ROM installed |
|---|---|---|---|---|---|
| Power Macintosh 9500/120 | 16 MB | 1 GB | 512 KB | Yes | Yes |
| Power Macintosh 9500/132 | 32 MB | 2 GB | 512 KB | No | Yes |

## Compatibility

Some features of the Power Macintosh 9500 computers are different from those of earlier Power Macintosh computers. This section highlights key areas you should investigate to ensure that your hardware and software work properly with the new computers.

### POWER-Clean Code

Applications for the Power Macintosh 9500 computers should be free of the POWER-only instructions that were included in the PowerPC 601.

The instruction set of the PowerPC 601 microprocessor includes some of the same instructions found in the instruction set of the POWER processor, and some compilers used to generate native code for earlier Power Macintosh models generated some of those POWER-only instructions. However, the PowerPC 604 microprocessor used in the Power Macintosh 9500 computers does not support the POWER-only instructions. When you compile applications for Power Macintosh computers, you should turn off the option that allows the compiler to generate POWER-only instructions.

### Emulation for Compatibility

Although the term *POWER emulation* is often used, a more appropriate name for this feature is *PowerPC 601 compatibility.* Its goal is to support those features of the POWER architecture that are available to programs running in user mode on PowerPC 601-based Power Macintosh computers. For more information, see “POWER Emulation” beginning on page 47.

Because the emulation of the POWER-only instructions degrades performance, Apple Computer encourages developers to revise any applications that use those instructions to conform with the PowerPC architecture.

### Code Fragments and Cache Coherency

Whereas the PowerPC 601 microprocessor has a single cache for both instructions and data, the PowerPC 604 has separate instruction and data caches. As long as applications deal with executable code by using the Code Fragment Manager, cache coherency is maintained. Applications that bypass the Code Fragment Manager and generate executable code in memory, and that do not use the proper cache synchronization instructions or CFM calls, are likely to encounter problems when running on the PowerPC 604.

The PowerPC 601 compatibility features of the Power Macintosh 9500 computers cannot deal with this situation. Applications that generate executable code in memory must be modified to use the Code Fragment Manager or maintain proper cache synchronization by other means.

C H A P T E R 2

# Architecture

This chapter describes the architecture of the Power Macintosh 9500 computers. It describes the major components on the main logic board: the microprocessor, the main memory, and the custom ICs. It also describes the separate video display card.

## Main ICs and Subsystems

The architecture of the Power Macintosh 9500 computers is based on three buses: the processor bus and two PCI buses. The processor bus connects the microprocessor and the memory; the PCI buses connect the expansion slots and the I/O devices.

### Block Diagram

Figure 2-1 is a simplified block diagram of the Power Macintosh 9500 computers showing the three buses and the major ICs on the main logic board.

### Processor Subsystem Card

The processor subsystem card is a plug-in card that contains the PowerPC 604 microprocessor and its associated clock circuits. The processor subsystem card provides an upgrade path to a more powerful microprocessor.

### Main Processor

The main processor in the Power Macintosh 9500 computers is a PowerPC 604 microprocessor. Its principal features include

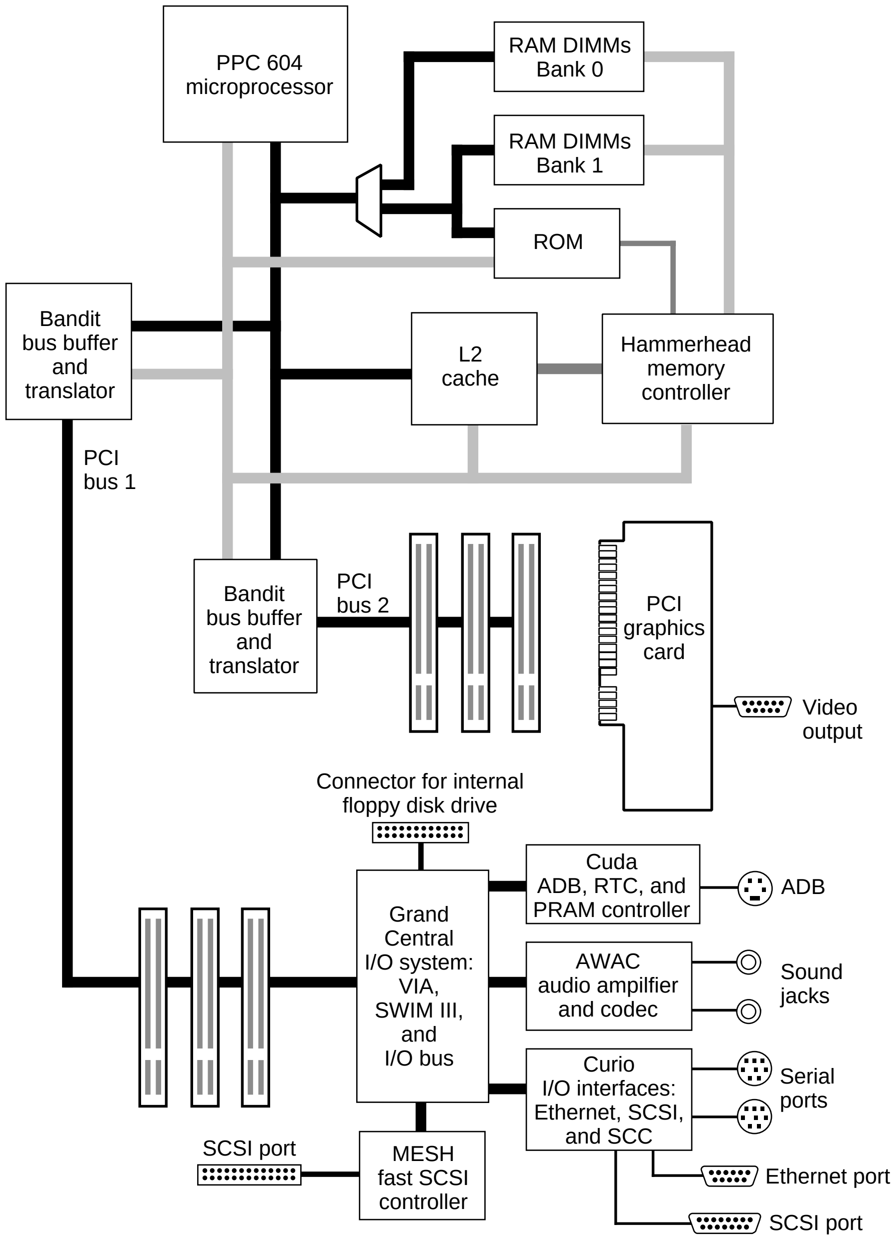
- full RISC processing architecture
- parallel processing units: load-store unit, two integer units, one complex integer unit, and one floating-point unit
- a branch manager that can usually implement branches by reloading the incoming instruction queue without using any processing time
- an internal memory management unit (MMU)
- separate caches for data and instructions, 16 KB each, four-way set associative

For complete technical details, see *PowerPC 604 Microprocessor Implementation Definition Book IV.*

### Read-Only Memory

The Power Macintosh 9500 computers use a ROM SIMM like the one in the earlier Power Macintosh computers. The ROM SIMM contains 4 MB of ROM with 100 ns access time.

**Figure 2-1** Block diagram


PPC 604 microprocessor
RAM DIMMs Bank 0
RAM DIMMs Bank 1
ROM
Bandit bus buffer and translator
L2 cache
Hammerhead memory controller
PCI bus 1
Bandit bus buffer and translator
PCI bus 2
PCI graphics card
Video output
Connector for internal floppy disk drive
Cuda ADB, RTC, and PRAM controller
ADB
Grand Central I/O system: VIA, SWIM III, and I/O bus
AWAC audio ampilfier and codec
Sound jacks
Curio I/O interfaces: Ethernet, SCSI, and SCC
Serial ports
SCSI port
MESH fast SCSI controller
Ethernet port
SCSI port

## Random-Access Memory

The address map for the Power Macintosh 9500 computers has 2 GB allocated for main memory. The memory controller in the Hammerhead IC supports up to 1.5 GB of main memory. Memory addresses are contiguous, starting at address $0000 0000.

All RAM in the Power Macintosh 9500 computers is provided by DRAM devices on 8-byte DIMMs (dual inline memory modules). The computers have twelve DIMM slots that can provide up to 1.5 GB of main memory.

The computers come with either 16 or 32 MB of RAM installed in the form of two or four DIMMs. The user can add more memory by installing one or more additional DIMMs. For more information, see “RAM DIMMs” in Chapter 4, “Expansion Features.”

The Hammerhead custom IC contains bank base registers that are used to make RAM addresses contiguous, starting at address $0000 0000. See “Bank Base Registers” on page 10.

## Second-Level Cache

The Power Macintosh 9500 computers have a built-in second level (L2) cache comprising 512 KB of high-speed memory. The L2 cache is organized as a write-back cache; it is direct mapped (single set) with allocate on read or write.

The cache’s tag store is implemented with standard static RAM (SRAM) devices. The cache’s data store is implemented with synchronous burst SRAM devices that have an access time of 11 ns.

## Hammerhead Memory Controller IC

The Hammerhead IC controls the memory and cache subsystem, which includes the system bus, the main memory, the ROM, and the L2 cache. The main memory provides low-latency memory accesses and improved memory bandwidth. The Hammerhead IC supports main memory sizes up to 2 GB.

### Data Interleaving

Even though the system data bus is only 64 bits wide, the memory controller in the Hammerhead IC can support data read operations 128 bits wide by interleaving the data from two DIMM slots. When the startup software detects two DIMMs that are the same size in adjacent banks (Bank 0–Bank 1), it enables interleaving for that pair of DIMMs.

For an interleaved transfer, the memory controller cycles both banks at the same time into a 128-bit buffer, then transfers data 64 bits at a time to the system data bus. The memory controller determines which bank’s data to transfer first on the basis of address signals that carry the critical word-first information. The controller supports both the critical quad-word first behavior of the PowerPC 601 and the critical double-word first behavior of the PowerPC 603 and PowerPC 604.

### Bank Base Registers

The Hammerhead IC contains a bank base register for each bank of main RAM. Each bank base register has space for a base address and control bits. The control bits set the address multiplexing mode and enable or disable data bus interleaving.

The bank base address is used to make memory banks contiguous. The system software calculates the base addresses based on the amount of memory in each of the SIMMs. The base address for each bank is based on the sum of the sizes of all the lower numbered banks.

## Bandit Bus Bridge ICs

The Bandit IC is a PCI bus bridge controller. In the Power Macintosh 9500 computers, two Bandit ICs provide bus buffering and address translation between the processor bus and the two PCI buses, PCI 1 and PCI 2.

Separate logic devices (gate arrays) provide the priorities for bus arbitration as follows:

1. Grand Central IC (I/O device controller), the highest priority
2. PCI slots and Bandit IC (bus master), in round-robin sequence

### Bus Clock Rates

The two types of buses operate asynchronously: the PCI bus at a clock rate of 33 MHz and the processor bus at 40 MHz or 44 MHz (one-third the processor speed). Each Bandit IC supports the full PCI bandwidth and also supports burst transfers, in both directions, at up to 32 bytes in length, which is the size of a cache block.

### Big-Endian and Little-Endian Bus Addressing

The Power Macintosh 9500 computers support both big-endian and little-endian conventions for addressing bytes in a word. Byte order for addressing on the processor bus is big-endian and byte order on the PCI bus is little-endian. The Bandit ICs perform the appropriate byte swapping and address translations between the two addressing conventions. For more information about the translations between big-endian and little-endian byte order, see Part One, “The PCI Bus,” in *Designing PCI Cards and Drivers for Power Macintosh Computers.*

## Grand Central I/O System IC

The Grand Central custom IC provides an interface between the standard Macintosh I/O devices and the PCI bus. The Grand Central IC performs the following functions:

- support for the Cuda IC (the VIA registers)
- central system interrupt collection
- support for descriptor-based DMA for I/O devices
- floppy disk interface (SWIM III)

The Grand Central IC contains a DMA controller. It provides DBDMA support for all I/O transfers, including transfers through its internal I/O controllers as well as transfers through the Curio IC for other I/O devices.

The SWIM III floppy disk drive controller in the Grand Central IC is an extension of the SWIM II design used in earlier Macintosh models. The SWIM III controller supports DMA data transfers and does not require disabling of interrupts during floppy disk accesses.

The Grand Central IC provides bus interfaces for the following I/O devices:

- Curio multipurpose I/O IC
- Cuda ADB controller IC
- AWAC sound input and output IC
- MESH controller IC for fast internal SCSI devices

The Grand Central IC also provides a 16-bit bus to other devices, including the nonvolatile parameter RAM.

The Grand Central IC is connected to the PCI bus and uses the 33 MHz PCI bus clock.

## Curio I/O Controller IC

The Curio IC is a multipurpose custom IC that contains a Media Access Controller for Ethernet (MACE), a SCSI controller, and a Serial Communications Controller (SCC).

The SCC section of the Curio includes 8-byte FIFO buffers for both transmit and receive data streams.

The functions of the Curio IC are described in the parts of Chapter 3 that deal with the external I/O ports it supports: “Serial Ports” on page 18, “Ethernet Port” on page 20, and “SCSI Port” on page 21.

**Note**
The Curio IC is also used in earlier Power Macintosh models. ◆

## Cuda Microcontroller IC

The Cuda IC is a custom version of the Motorola MC68HC05 microcontroller. It provides several system functions, including

- program control of the power supply (soft power)
- management of system resets
- maintenance of parameter RAM
- control of the Apple Desktop Bus (ADB)
- management of the real-time clock

### AWAC Sound IC

The audio waveform amplifier and converter (AWAC) is a custom IC that combines a waveform amplifier with a 16-bit digital sound encoder and decoder (codec). It conforms to the IT&T *ASCO 2300 Audio-Stereo Codec Specification* and furnishes high-quality sound input and output. For information about the operation of the AWAC IC, see Chapter 3 of *Developer Note: Power Macintosh Computers,* available on the developer CD-ROM and as part of *Macintosh Developer Note Number 8.*

The sound system in the Power Macintosh 9500 computers is similar to that in the earlier Power Macintosh computers. For more information about the sound features, see "Sound Input Jack" on page 22 and "Sound Output Jack" on page 22.

### MESH SCSI Controller IC

The MESH custom IC controls the internal SCSI bus. Because that bus is internal only, its loading characteristics can be more closely controlled than those of an external bus, permitting higher transfer speeds.

## Video Display Card

The Power Macintosh 9500 computers require an external monitor and a video display card installed in a PCI expansion slot. The Power Macintosh 9500/120 comes with a video card installed: the Apple Accelerated PCI Graphics Card.

The Apple Accelerated PCI Graphics Card contains a graphics accelerator with a 64-bit data bus to either 2 MB or 4 MB of fast VRAM. The card provides accelerated graphics on monitors with screen sizes up to 1280-by-1024-pixels.

The Apple Accelerated PCI Graphics Card is a standard PCI card. Its boot ROM conforms to the IEEE Standard 1275. It follows the format and information content defined in the *PCI Local Bus Specification,* Revision 2.0. For information about obtaining these standards, see page xi.

The firmware on the card contains a video device driver that supports the initialization required during system startup. The video driver provides the same function calls and has the same capabilities as the NuBus™ video driver.

### VRAM and Screen Sizes

The Apple Accelerated PCI Graphics Card has 2 MB of VRAM soldered down and an optional expansion card that increases the total VRAM size to 4 MB. The larger amount of VRAM is needed to support the larger pixel depths on large-screen monitors.

Table 2-1 shows the pixel depths supported by 2 and 4 MB of VRAM on monitors with various screen sizes.

**Table 2-1** Screen sizes and pixel depths

| | **Pixel depths** | |
|---|---|---|
| **Monitor screen size** | With 2 MB VRAM | With 4 MB VRAM |
| 640 by 480 | 8, 16, 24 | 8, 16, 24 |
| 832 by 624 | 8, 16, 24 | 8, 16, 24 |
| 1024 by 768 | 8, 16 | 8, 16, 24 |
| 1152 by 870 | 8, 16 | 8, 16, 24 |
| 1280 by 1024 | 8, 16 | 8, 16 |

## Monitor Connector

The video output from the Apple Accelerated PCI Graphics Card consists of standard red, green, and blue video signals with separate sync. The card has a DB-15 video connector for connecting a standard Apple monitor. Table 2-2 shows the pin assignments on the video connector.

**Table 2-2** Pin assignments on the video connector

| **Pin number** | **Signal name** | **Description** |
|---|---|---|
| 1 | RED.GND | Red video ground |
| 2 | RED.VID | Red video signal |
| 3 | /CSYNC | Composite synchronization signal |
| 4 | SENSE0 | Monitor sense signal 0 |
| 5 | GRN.VID | Green video signal |
| 6 | GRN.GND | Green video ground |
| 7 | SENSE1 | Monitor sense signal 1 |
| 8 | +12V | 12-volt power from computer |
| 9 | BLU.VID | Blue video signal |
| 10 | SENSE2 | Monitor sense signal 2 |
| 11 | GND | CSYNC and VSYNC ground |
| 12 | /VSYNC | Vertical synchronization signal |
| 13 | BLU.GND | Blue video ground |

*continued*

Table 2-2 Pin assignments on the video connector (continued)

| Pin number | Signal name | Description |
|---|---|---|
| 14 | HSYNC.GND | HSYNC ground |
| 15 | /HSYNC | Horizontal synchronization signal |
| Shell | SGND | Shield ground |

## Detecting the Monitor Type

The video card uses the sense lines to determine the type of monitor that is connected. Table 2-3 shows the sense line encodings supported by the video card.

Table 2-3 Sense line codes for various monitors

| | Sense lines | | | Extended sense codes | | |
|---|---|---|---|---|---|---|
| **Monitor type** | (2) | (1) | (0) | (1, 2) | (0, 2) | (0, 1) |
| 21-inch color | 0 | 0 | 0 | – | – | – |
| Portrait, grayscale | 0 | 0 | 1 | – | – | – |
| Two-page grayscale | 0 | 1 | 1 | – | – | – |
| Portrait, color | 1 | 0 | 1 | – | – | – |
| 12-inch monochrome, 14-inch color | 1 | 1 | 0 | – | – | – |
| 15-inch multiple scan | 1 | 1 | 0 | 00 | 00 | 11 |
| 17-inch multiple scan | 1 | 1 | 0 | 00 | 10 | 11 |
| 20-inch multiple scan | 1 | 1 | 0 | 10 | 00 | 11 |
| VGA | 1 | 1 | 1 | 00 | 00 | 11 |
| 19-inch color | 1 | 1 | 1 | 11 | 10 | 10 |

The video card also supports a Monitors control panel that allows the user to switch between monitor timings during operation. The switching happens immediately; the user does not need to restart the computer.

C H A P T E R 3

# I/O Features

This chapter describes the I/O features of the Power Macintosh 9500 computers—both the built-in I/O devices and the interfaces for external I/O devices.

## I/O Ports

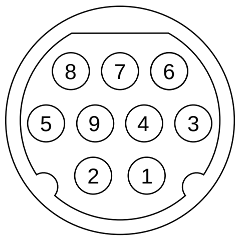
The Power Macintosh 9500 computers have the standard I/O ports found on other Macintosh models.

- two serial ports
- ADB port
- Ethernet ports (AAUI and 10baseT)
- SCSI port
- sound input jack
- sound output jack

### Serial Ports

The Power Macintosh 9500 computers have two serial ports on the back panel. Both ports use 9-pin circular mini-DIN sockets, as shown in Figure 3-1. The serial port sockets accept either 8-pin or 9-pin plugs.

**Figure 3-1** Serial port connector



Either port can be independently programmed for asynchronous or synchronous communication formats up to 9600 baud, including AppleTalk and the full range of Apple GeoPort protocols. With external modules connected to the serial ports, the computer can communicate with a variety of ISDN and other telephone transmission facilities.

Table 3-1 gives the pin assignments for both serial ports.

**Table 3-1** Pin assignments on the serial port connector

| Pin | Name | Function |
|---|---|---|
| 1 | HSKo | Handshake output |
| 2 | HSKi | Handshake input or external clock (up to 920 Kbit/sec.) |
| 3 | TxD– | Transmit data – |
| 4 | Gnd | Ground |
| 5 | RxD– | Receive data – |
| 6 | TxD+ | Transmit data + |
| 7 | GPi | General-purpose input (wake up CPU or perform DMA handshake) |
| 8 | RxD+ | Receive data + |
| 9 | +5V | Power to external device (100 mA maximum) |

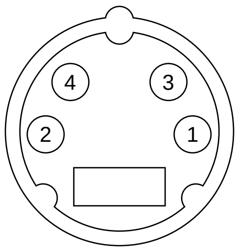
Pin 9 on each serial connector provides +5 V power from the ADB power supply. An external device should draw no more than 100 mA from that pin. The total current available for all devices connected to the +5 V supply for the ADB and the serial ports is 500 mA. Excessive current drain causes a circuit breaker to interrupt the +5 V supply; the breaker automatically resets when the load returns to normal.

Both serial ports include the GPi (general-purpose input) signal on pin 7. The GPi signal for each port connects to the corresponding data carrier detect input on the SCC portion of the Curio custom IC, which is described in Chapter 2. For more information about the serial ports, see *Guide to the Macintosh Family Hardware,* second edition.

## Apple Desktop Bus (ADB)

The Apple Desktop Bus (ADB) is an asynchronous communication bus used for relatively slow user-input devices such as the keyboard and the mouse. The Power Macintosh 9500 computers have a single ADB port on the back panel. The connector is a 4-pin mini-DIN socket, as shown in Figure 3-2.

**Figure 3-2** ADB connector



The ADB is a single-master, multiple-slave serial communication bus that uses an asynchronous protocol. The custom ADB microcontroller (the Cuda IC) drives the bus and reads the status from the selected external device. Table 3-2 lists the ADB connector pin assignments.

**Table 3-2** Pin assignments on the ADB connector

| Pin | Name | Description |
|---|---|---|
| 1 | ADB | Bidirectional data bus |
| 2 | PSW | Power-on signal (generates reset and interrupt key combinations) |
| 3 | +5V | +5 volts |
| 4 | GND | Ground |

**Note**
The total current available for all devices connected to the +5V pins on the ADB and the modem port is 500 mA. Each device should use no more than 100 mA. ◆

For more information about the ADB, see *Guide to the Macintosh Family Hardware,* second edition. The software characteristics of the ADB are described in *Inside Macintosh: Devices.*

## Ethernet Port

The Power Macintosh 9500 computers have a built-in Ethernet port. The Ethernet port has two connectors, one for a 10BaseT cable and the other for the Apple Ethernet adapter for thick net or thin net cables. Both connectors can be occupied, but only one at a time can be used. The electrical and mechanical characteristics of the Ethernet port are the same as on other Macintosh computers.

The Ethernet port pin assignments are shown in Table 3-3.

**Table 3-3** Pin assignments on the Ethernet connector

| Pin | Description | Pin | Description |
|---|---|---|---|
| 1 | +5 V | 8 | +5 V |
| 2 | DI+ | 9 | DO+ |
| 3 | DI– | 10 | DO– |
| 4 | Ground | 11 | Ground |
| 5 | CI+ | 12 | No connection |
| 6 | CI– | 13 | No connection |
| 7 | +5 V | 14 | +5 V |

## SCSI Port

The Power Macintosh 9500 computers use a SCSI bus for external SCSI devices and for the internal CD-ROM drive. The external SCSI connector is a 25-pin D-type connector; the internal CD-ROM drive uses a 50-pin connector.

Table 3-4 shows the pin assignments on the internal and external SCSI connectors.

**Table 3-4** Pin assignments for the SCSI connectors

| Pin number (internal 50-pin) | Pin number (external 25-pin) | Signal name | Signal description |
|---|---|---|---|
| 2 | 8 | /DB0 | Bit 0 of SCSI data bus |
| 4 | 21 | /DB1 | Bit 1 of SCSI data bus |
| 6 | 22 | /DB2 | Bit 2 of SCSI data bus |
| 8 | 10 | /DB3 | Bit 3 of SCSI data bus |
| 10 | 23 | /DB4 | Bit 4 of SCSI data bus |
| 12 | 11 | /DB5 | Bit 5 of SCSI data bus |
| 14 | 12 | /DB6 | Bit 6 of SCSI data bus |
| 16 | 13 | /DB7 | Bit 7 of SCSI data bus |
| 18 | 20 | /DBP | Parity bit of SCSI data bus |
| 25 | – | n.c. | Not connected |
| 26 | 25 | TPWR | +5 V terminator power |
| 32 | 17 | /ATN | Attention |
| 36 | 6 | /BSY | Bus busy |
| 38 | 5 | /ACK | Handshake acknowledge |
| 40 | 4 | /RST | Bus reset |
| 42 | 2 | /MSG | Message phase |
| 44 | 19 | /SEL | Select |
| 46 | 15 | /C/D | Control or data |
| 48 | 1 | /REQ | Handshake request |
| 50 | 3 | /I/O | Input or output |
| 20, 22, 24, 28, 30, 34, and all odd pins except pin 25 | 7, 9, 14, 16, 18, and 24 | GND | Ground |

The external SCSI port has automatic termination like that on the earlier Power Macintosh computers. When no external SCSI device is connected, the automatic termination is active. When one or more external SCSI devices are connected, the automatic termination is removed. As usual, the external SCSI device at the end of the SCSI bus requires termination.

The internal end of the SCSI bus is terminated by a 110 Ω passive terminator. The terminator is located on the main logic board near the portion of the internal chassis connector that contains the signals for the internal CD-ROM drive. The internal CD-ROM drive does not include a terminator.

## Sound Input Jack

The Power Macintosh 9500 computers have a stereo sound input jack for connecting an external microphone or a line-level source. The computer provides sound digitization and recording with 16-bit samples at sample rates of up to 44.1 KHz and supports Apple Computer's speech synthesis and recognition software.

The sound input jack is a stereo mini-phone jack with an additional contact to supply power to an Apple microphone. The sound input jack accepts either the Apple PlainTalk line-level microphone or a pair of line-level signals by way of a separate adapter.

The sound input jack has the following electrical characteristics:

- input impedance: 8000 Ω
- maximum level: 2 V rms
- maximum gain: 22.5 dB
- signal-to-noise ratio: 82 dB

**Note**
The Apple PlainTalk microphone requires power from the main computer, which it obtains by way of an extra-long, 4-conductor plug that makes contact with a 5-volt pin inside the sound input jack. ◆

## Sound Output Jack

The Power Macintosh 9500 computers have a stereo sound output jack for connecting external powered speakers or other line-level devices. The sound output jack is a stereo mini-phone jack. Inserting a plug into the jack disconnects the internal speaker.

The sound output jack is a standard mini-phone jack; it has the following electrical characteristics:

- output impedance: 37 Ω
- maximum level: 0.9 V rms
- maximum attenuation: 22.5 dB

- frequency response: 20 Hz to 20 kHz, plus or minus 2 dB
- harmonic distortion plus noise: less than 0.05 percent at 1 V rms input
- signal to noise ratio: 85 dB; no audible discrete tones

**Note**
Crosstalk degrades from –80 dB to –32 dB when
32 Ω headphones are connected. ◆

## Disk Drives

The Power Macintosh 9500 computers have one internal floppy disk drive and one internal hard disk drive. Some configurations also have an internal CD-ROM drive.

### Floppy Disk Drive

The Power Macintosh 9500 computers have one internal high-density floppy disk drive (Apple SuperDrive). The drive is connected to a 20-pin connector. Table 3-5 shows the pin assignments on the floppy disk connector.

**Table 3-5** Pin assignments on the floppy disk connector

| Pin number | Signal name | Signal description |
|---|---|---|
| 1 | GND | Ground |
| 2 | PH0 | Phase 0: state control line |
| 3 | GND | Ground |
| 4 | PH1 | Phase 1: state control line |
| 5 | GND | Ground |
| 6 | PH2 | Phase 2: state control line |
| 7 | GND | Ground |
| 8 | PH3 | Phase 3: register write strobe |
| 9 | n.c. | Not connected |
| 10 | /WRREQ | Write data request |
| 11 | +5V | +5 volts |
| 12 | SEL | Head select |
| 13 | +12V | +12 volts |
| 14 | /ENBL | Drive enable |

*continued*

**Table 3-5** Pin assignments on the floppy disk connector (continued)

| Pin number | Signal name | Signal description |
|---|---|---|
| 15 | +12V | +12 volts |
| 16 | RD | Read data |
| 17 | +12V | +12 volts |
| 18 | WR | Write data |
| 19 | +12V | +12 volts |
| 20 | n.c. | Not connected |

## Internal Hard Disk Drive

The Power Macintosh 9500 computers have one internal hard disk drive. The drive capacity is either 1 or 2 GB.

The hard disk drive is connected to the internal SCSI bus. For pin assignments on the internal SCSI hard disk connector, see Table 3-4 on page 21.

The internal end of the SCSI bus is terminated by a 110 Ω passive terminator. The terminator is located on the main logic board near the portion of the internal chassis connector that contains the signals for the internal CD-ROM drive. The internal CD-ROM drive does not include a terminator.

## CD-ROM Drive

Some configurations of the Power Macintosh 9500 computers have a built-in CD-ROM drive, an AppleCD 600i. The AppleCD 600i supports the worldwide standards and specifications for CD-ROM and CD-digital audio discs described in the Sony/Phillips Yellow Book and Red Book. The drive can read CD-ROM, CD-ROM XA, CD-I, and PhotoCD discs as well as play standard audio discs.

The AppleCD 600i has a sliding tray to hold the disc. The drive features a double-speed mechanism that supports sustained data transfer rates of 600 KB per second and a data buffer that further enhances performance.

C H A P T E R 4

# Expansion Features

This chapter describes the expansion features of the Power Macintosh 9500 computers: the RAM expansion DIMMs and the PCI expansion slots.

## RAM DIMMs

The Power Macintosh 9500 computers have twelve RAM expansion slots. The RAM expansion slots accept a new type of memory module: the 8-byte DIMM (dual inline memory module). As its name implies, the 8-byte DIMM has a 64-bit-wide data bus.

The 8-byte DIMM is a new industry standard. Its mechanical design is defined by the MO-161 specification published by the JEDEC JC-11 committee; its electrical characteristics are defined by the JEDEC Standard No. 21-C. The 8-byte DIMM connector used in the Power Macintosh 9500 computers is Burndy Corporation's part number ELF168E5GC-3Z50 or equivalent.

**IMPORTANT**
The Power Macintosh 9500 computers do not have any main memory soldered to the logic board. At least one RAM DIMM must be present for the computers to operate. ▲

The 8-byte DIMMs can be installed one or more at a time. When the startup software detects two DIMMs that are the same size and installed in corresponding slots, it configures them as a single 128-bit data bus for increased performance. For 128-bit bus operation, both DIMMs in the pair also have to have the same address mode; see "RAM Address Multiplexing" beginning on page 30.

The minimum bank size supported by the Hammerhead IC is 4 MB and the largest is 64 MB; the largest DIMM supported is a two-bank DIMM holding 128 MB. Table 4-1 shows the DIMM configurations and sizes for a range of DRAM device sizes.

**Table 4-1** Memory sizes and configurations

| DRAM device size | DIMM configuration | DIMM size | Maximum memory with 12 DIMMs installed |
|---|---|---|---|
| 4 Mbit | 512 Kbits by 64 | 4 MB | 48 MB |
| 4 Mbit | 1 Mbit by 64 | 8 MB | 96 MB |
| 16 Mbit | 1 Mbit by 64 | 8 MB | 96 MB |
| 16 Mbit | 2 Mbits by 64 | 16 MB | 192 MB |
| 16 Mbit | 4 Mbits by 64 | 32 MB | 384 MB |

*continued*

**Table 4-1** Memory sizes and configurations (continued)

| DRAM device size | DIMM configuration | DIMM size | Maximum memory with 12 DIMMs installed |
|---|---|---|---|
| 64 Mbit | 4 Mbits by 64 | 32 MB | 384 MB |
| 64 Mbit | 8 Mbits by 64 | 64 MB | 768 MB |
| 64 Mbit | 16 Mbits by 64 | 128 MB | 1.5 GB |

NOTE The maximum memory supported by the current Mac OS is 1 GB.

## RAM DIMM Connectors

Table 4-2 gives the pin assignments for the RAM DIMM connectors.

**Table 4-2** Pin assignments on the RAM DIMM connectors

| Pin number | Signal name | Pin number | Signal name | Pin number | Signal name |
|---|---|---|---|---|---|
| 1 | VSS | 20 | DQ(14) | 39 | A(12) |
| 2 | DQ(0) | 21 | DQ(15) | 40 | VCC |
| 3 | DQ(1) | 22 | Reserved | 41 | Reserved |
| 4 | DQ(2) | 23 | VSS | 42 | Reserved |
| 5 | DQ(3) | 24 | Reserved | 43 | VSS |
| 6 | VCC | 25 | Reserved | 44 | /OE(2) |
| 7 | DQ(4) | 26 | VCC | 45 | /RAS(2) |
| 8 | DQ(5) | 27 | /WE(0) | 46 | /CAS(4) |
| 9 | DQ(6) | 28 | /CAS(0) | 47 | /CAS(6) |
| 10 | DQ(7) | 29 | /CAS(2) | 48 | /WE(2) |
| 11 | Reserved | 30 | /RAS(0) | 49 | VCC |
| 12 | VSS | 31 | /OE(0) | 50 | Reserved |
| 13 | DQ(8) | 32 | VSS | 51 | Reserved |
| 14 | DQ(9) | 33 | A(0) | 52 | DQ(16) |
| 15 | DQ(10) | 34 | A(2) | 53 | DQ(17) |
| 16 | DQ(11) | 35 | A(4) | 54 | VSS |
| 17 | DQ(12) | 36 | A(6) | 55 | DQ(18) |
| 18 | VCC | 37 | A(8) | 56 | DQ(19) |
| 19 | DQ(13) | 38 | A(10) | 57 | DQ(20) |

*continued*

**Table 4-2** Pin assignments on the RAM DIMM connectors (continued)

| Pin number | Signal name | Pin number | Signal name | Pin number | Signal name |
|---|---|---|---|---|---|
| 58 | DQ(21) | 88 | DQ(34) | 118 | A(3) |
| 59 | VCC | 89 | DQ(35) | 119 | A(5) |
| 60 | DQ(22) | 90 | VCC | 120 | A(7) |
| 61 | Reserved | 91 | DQ(36) | 121 | A(9) |
| 62 | Reserved | 92 | DQ(37) | 122 | A(11) |
| 63 | Reserved | 93 | DQ(38) | 123 | A(13) |
| 64 | Reserved | 94 | DQ(39) | 124 | VCC |
| 65 | DQ(23) | 95 | Reserved | 125 | Reserved |
| 66 | Reserved | 96 | VSS | 126 | B(0) |
| 67 | DQ(24) | 97 | DQ(40) | 127 | VSS |
| 68 | VSS | 98 | DQ(41) | 128 | Reserved |
| 69 | DQ(25) | 99 | DQ(42) | 129 | /RAS(3) |
| 70 | DQ(26) | 100 | DQ(43) | 130 | /CAS(5) |
| 71 | DQ(27) | 101 | DQ(44) | 131 | /CAS(7) |
| 72 | DQ(28) | 102 | VCC | 132 | /PDE |
| 73 | VCC | 103 | DQ(45) | 133 | VCC |
| 74 | DQ(29) | 104 | DQ(46) | 134 | Reserved |
| 75 | DQ(30) | 105 | DQ(47) | 135 | Reserved |
| 76 | DQ(31) | 106 | Reserved | 136 | DQ(48) |
| 77 | Reserved | 107 | VSS | 137 | DQ(49) |
| 78 | VSS | 108 | Reserved | 138 | VSS |
| 79 | PD(1) | 109 | Reserved | 139 | DQ(50) |
| 80 | PD(3) | 110 | VCC | 140 | DQ(51) |
| 81 | PD(5) | 111 | Reserved | 141 | DQ(52) |
| 82 | PD(7) | 112 | /CAS(1) | 142 | DQ(53) |
| 83 | ID(0) | 113 | /CAS(3) | 143 | VCC |
| 84 | VCC | 114 | /RAS(1) | 144 | DQ(54) |
| 85 | VSS | 115 | Reserved | 145 | Reserved |
| 86 | DQ(32) | 116 | VSS | 146 | Reserved |
| 87 | DQ(33) | 117 | A(1) | 147 | Reserved |

*continued*

**Table 4-2** Pin assignments on the RAM DIMM connectors (continued)

| Pin number | Signal name | Pin number | Signal name | Pin number | Signal name |
|---|---|---|---|---|---|
| 148 | Reserved | 155 | DQ(59) | 162 | VSS |
| 149 | DQ(55) | 156 | DQ(60) | 163 | PD(2) |
| 150 | Reserved | 157 | VCC | 164 | PD(4) |
| 151 | DQ(56) | 158 | DQ(61) | 165 | PD(6) |
| 152 | VSS | 159 | DQ(62) | 166 | PD(8) |
| 153 | DQ(57) | 160 | DQ(63) | 167 | ID(1) |
| 154 | DQ(58) | 161 | Reserved | 168 | VCC |

Table 4-6 describes the signals on the RAM DIMM connector.

**Table 4-3** RAM DIMM signals

| Signal name | Description |
|---|---|
| A(13:0) | Address inputs, buffered |
| B(0) | Alternate version of A(0), buffered |
| /CAS(7:0) | Column address strobe signals, buffered |
| DQ(63:0) | Data input and output signals |
| ID(1:0) | Memory module identification (not used) |
| /OE(0, 2) | Output enable signals, buffered |
| PD(8:1) | Presence detect signals (not used) |
| /PDE | Presence detect enable signal (not used) |
| /RAS(3:0) | Row address strobe signals |
| Reserved | Reserved, don’t use |
| VCC | +5 V power |
| VSS | Ground |
| /WE(0, 2) | Read/write input signals, buffered |

B(0) is an alternate addressing signal that allows a DIMM to operate with the 64-bit data bus split into two 32-bit halves. In the Power Macintosh 9500 computers, A(0) and B(0) are tied together on the main logic board.

**Note**
Although they are defined in JEDEC Standard No. 21-C, the presence detect signals PD(8:1) and /PDE and the identification signals ID(1:0) are not used in the Power Macintosh 9500 computers. ◆

**Note**
No +3-V power is provided on the DIMM connector. ◆

## RAM Address Multiplexing

Depending on their internal design and size, different types of DRAM devices require different row and column address multiplexing. The memory controller in the Hammerhead IC supports two addressing modes, selected individually for each bank of DRAM. The system software initializes the address mode bits in the bank base registers as part of the process of determining the amount of RAM installed in the computer.

Signals A(11:0) on each RAM DIMM make up a 12-bit multiplexed address bus that can support several different types of DRAM devices. Table 4-4 shows the address multiplexing modes used with several types of DRAM devices. The devices are characterized by their bit dimensions: for example, a 256K by 4-bit device has 256K addresses and stores 4 bits at a time.

**Note**
The memory controller does not support devices that require more than 12 address bits. ◆

**Table 4-4** Address multiplexing modes for various DRAM devices

| Device size | Device type | Row bits | Column bits | Address mode |
|---|---|---|---|---|
| 4 Mbit | 1 M by 4 | 10 | 10 | 1 |
| 4 Mbit | 512K by 8 | 10 | 9 | 1 |
| 4 Mbit | 256K by 16 | 10 | 8 | 0 |
| 16 Mbit | 4 M by 4 | 11 | 11 | 1 |
| 16 Mbit | 4 M by 4 | 12 | 10 | 1 |
| 16 Mbit | 2 M by 8 | 11 | 10 | 1 |
| 16 Mbit | 2 M by 8 | 12 | 9 | 0 |
| 16 Mbit | 1 M by 16 | 12 | 8 | 0 |
| 64 Mbit | 16 M by 4 | 12 | 12 | 0 |
| 64 Mbit | 8 M by 8 | 12 | 11 | 0 |
| 64 Mbit | 4 M by 16 | 11 | 11 | 1 |
| 64 Mbit | 4 M by 16 | 12 | 10 | 1 |

Table 4-5 shows how the address signals to the RAM devices are multiplexed during the row and column address phases.

**Table 4-5** Address multiplexing

| | Individual signals on the DRAM_ADDR bus | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A(11) | A(10) | A(9) | A(8) | A(7) | A(6) | A(5) | A(4) | A(3) | A(2) | A(1) | A(0) |
| **Address mode = 0** | | | | | | | | | | | | |
| Row address bits | A22 | A21 | A20 | A19 | A18 | A17 | A16 | A15 | A14 | A13 | A12 | A11 |
| Column address bits | A26 | A25 | A24 | A23 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 |
| **Address mode = 1** | | | | | | | | | | | | |
| Row address bits | A24 | A23 | A21 | A20 | A19 | A18 | A17 | A16 | A15 | A14 | A13 | A12 |
| Column address bits | A25 | A24 | A22 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 |

## RAM Devices

The memory controller in the Hammerhead IC supports 1 MB, 4 MB, 16 MB, and 64 MB DRAM devices. The access time ($T_{RAS}$) of the DRAM devices must be 70 ns or less.

**IMPORTANT**
The number of DRAM devices in a RAM DIMM for the Power Macintosh 9500 computers is constrained by the load limits of the unbuffered signals. On each DIMM, a maximum of two devices can be connected to each data line and a maximum of eight devices can be connected to each /RAS line. ▲

## RAM Refresh

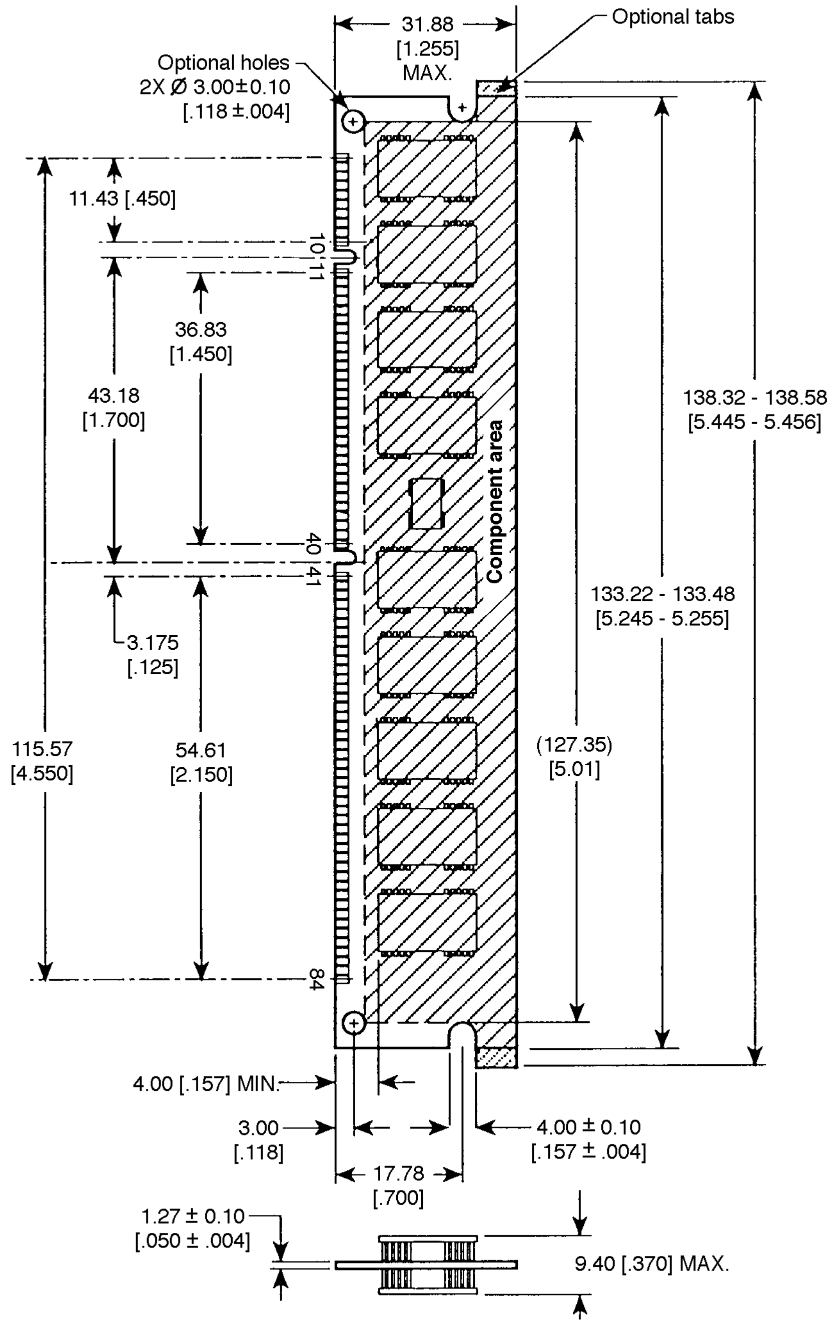
The Hammerhead IC provides a CAS-before-RAS refresh cycle every 15.6 µs. DRAM devices must be compatible; for example, this cycle will refresh 2K-refresh parts within 32 ms.

## RAM DIMM Dimensions

Figure 4-1 on page 32 shows the dimensions of the RAM DIMM.

**IMPORTANT**
The JEDEC MO-161 specification shows three possible heights for the 8-byte DIMM. The Power Macintosh 9500 computers accept only the shorter two of the three specified sizes; the maximum height is 1.255 inches. ▲

**Figure 4-1** Dimensions of the RAM DIMM



## PCI Expansion Slots

The Power Macintosh 9500 computers use the industry standard PCI bus for their I/O expansion bus. The PCI bus is a nonsplit bus with 32-bit multiplexed address and data. The PCI expansion slots in the Power Macintosh 9500 computers have a 33 MHz system clock.

The Power Macintosh 9500 computers have six PCI slots supported by two Bandit custom ICs, one for PCI slots 1–3 and the other for PCI slots 4–6. The computer provides a total of 90 W of power for the six expansion slots. Both 5 V and 3.3 V are supplied; the total power at both voltages must not exceed the 90-W maximum.

The Power Macintosh 9500 computers accept standard PCI cards as defined by the *PCI Local Bus Specification*, Revision 2.0. The computer requires all PCI cards to use the 5-V signaling standard described in the specification. The cards are required to use the standard ISA fence described in the specification.

The PCI slots support all the required PCI signals and certain optional PCI signals. The supported PCI signals are listed in Table 4-6.

**Table 4-6** PCI signals

| Signal name | Description |
|---|---|
| AD [31:0] | Address and data, multiplexed |
| C/BE[3:0] | Bus command and byte enable signals, multiplexed |
| PAR | Parity; used with AD and C/BE signals |
| FRAME# | Cycle frame; asserted to indicate a bus transaction |
| TRDY# | Target ready; selected device is able to complete the current phase |
| IRDY# | Initiator ready; master device is able to complete the current phase |
| STOP# | Stop; indicates the current target device is requesting the master to stop the current transaction |
| DEVSEL# | Device select; indicates that the driving device has decoded its address as the target of the current access |
| IDSEL# | Initialization device select; used during configuration |
| REQ# | Request; indicates to the arbiter that the asserting agent requires use of the bus |
| GNT# | Grant; indicates to the agent that access to the bus has been granted |

*continued*

**Table 4-6** PCI signals (continued)

| Signal name | Description |
|---|---|
| CLK | Clock; rising edge provides timing for all transactions |
| RST# | Reset; used to bring registers and signals to a known state |
| INTA#, INTB#, INTC#, INTD# | Interrupt request pins; wired together on each slot |

The PCI slots in the Power Macintosh 9500 computers do not support the following optional signals:

- 64-bit bus extension signals
- cache support signals
- JTAG (boundary scan) signals
- LOCK#
- error reporting signals PERR# and SERR#

For more information about the PCI expansion slot, refer to *Designing PCI Cards and Drivers for Power Macintosh Computers.*

C H A P T E R 5

# Software Features

# Software Features

The system software for the Power Macintosh 9500 computers is version 7.5.2. It is based on System 7.5 and is augmented by several new features.

**IMPORTANT**
Even though system software 7.5.2 includes significant changes from System 7.5, it is not a reference release: that is, it is not an upgrade for earlier Macintosh models. ▲

The system software includes changes of several kinds, including new features, performance enhancements, and hardware support features.

## New Features

The system software for the Power Macintosh 9500 computers includes the following new features:

- large partition support
- Drive Setup
- transport-independent networking (Open Transport)
- Open Firmware startup

### Large Partition Support

The largest disk partition supported by System 7.5 is 4 GB. The new system software extends that limit to 2 terabytes.

**IMPORTANT**
The largest possible file is still just under 2 GB. ▲

The changes necessary to support the larger partition size affect many parts of the system software. The affected software includes system-level and application-level components.

#### 64-Bit Volume Addresses

The current disk driver API has a 32-bit volume address limitation. This limitation has been circumvented by the addition of a new 64-bit extended volume API (`PBXGetVolInfo`) and 64-bit data types (`UnsignedWide`, `Wide`, `XVolumeParam`, and `XIOParam`).

For the definitions of the new API and data types, please see “The API Modifications” in Chapter 6, “Large Volume Support.”

## System-Level Software

Several system components have been modified to use the 64-bit APIs to correctly calculate true volume sizes and read and write data to and from large disks. The modified system components are

- virtual memory code
- Disk Init
- FSM
- Apple disk drivers
- HFS ROM code

## Application-Level Software

Current applications do not require modification to gain access to disk space beyond the traditional 4 GB limit as long as they do not require the true size of the large partition. Applications that need to obtain the true partition size must be modified to use the new 64-bit APIs and data structures. Typical applications include utilities for disk formatting, partitioning, initialization, and backup.

The following application-level components of the system software have been modified to use the 64-bit APIs:

- Finder
- Finder extensions (AppleScript, AOCE Mailbox, and Catalogs)
- HDSC Setup
- Disk First Aid

In the past, the sum of the sizes of the files and folders selected in the Finder was limited to the largest value that could be stored in a 32-bit number—that is, 4 GB. By using the new 64-bit APIs and data structures, the Finder can now operate on selections whose total size exceeds that limit. Even with very large volumes, the Finder can display accurate information in Folder and Get Info windows and can obtain the true volume size for calculating available space when copying.

The Finder extensions AppleScript, AOCE Mailbox, and Catalogs have been modified in the same way as the Finder because their copy-engine code is similar to that in the Finder.

A later section describes the modified Drive Setup application.

### Limitations

The software modifications that support large partition sizes do not solve all the problems associated with the use of large volumes. In particular, the modifications do not address the following:

- HFS file sizes are still limited to 2 GB or less.
- Large allocation block sizes cause inefficient storage. On a 2 GB volume, the minimum file size is 32 KB; on a 2 terabyte volume, the minimum file size would be a whopping 32 MB.
- Drives with the new large volume driver will not mount on computers running older versions of the Mac OS.

## Drive Setup

The software for the Power Macintosh 9500 computers includes a new disk setup utility named Drive Setup. In addition to the ability to support large volumes, the Drive Setup utility has several other enhancements over the older HDSC Setup utility, including

- an improved user interface
- support for multiple partitions
- the ability to mount volumes from applications
- the ability to start up (boot) from any HFS partition
- support for removable media drives

## Open Transport

Open Transport is the new communications and networking architecture that will become the standard for Macintosh networking and communications. Open Transport provides a mechanism for communications applications to operate independently from underlying networks such as AppleTalk, TCP, or IPX. Open Transport provides a code base and architecture that supports network stacks while eliminating many of the interrupt latency problems associated with AppleTalk.

**Note**
Open Transport runs native on the PowerPC microprocessors. ◆

Open Transport has two major aspects: the client interfaces and the environment for developing protocols and communications modules. The Open Transport client interfaces are a superset of the XTI interface from X/Open, a consortium of UNIX® vendors. XTI is a superset of TLI, a UNIX standard interface. By using the Open Transport interfaces, applications (called *clients*) can operate independently of the transport layer.

The environment for developing protocols and communications modules for Open Transport also uses industry standards. These standards are the UNIX standard Streams, and two other standards, Transport Provider Interface (TPI) and Data Link Provider Interface (DLPI).

Open Transport does not use the conventional `.ENET` style drivers; instead it uses Streams-based DLPI drivers that are more appropriate for use with PCI devices. In addition to being consistent with industry standards, Streams-based DLPI drivers provide higher performance than `.ENET` style drivers.

Apple Computer's Open Transport software includes new stack implementations for AppleTalk and MacTCP. Apple expects that third parties will provide implementations of DECnet™, IPX, and other network protocols.

## New Features of Open Transport

The new features of Open Transport include

- a new API
- dynamic loading and shared code
- multihoming
- an optional static node number (AppleTalk)
- an optional NBP-to-catalog server (AppleTalk)
- IP multicasting (MacTCP)
- dynamic retransmission timers (MacTCP)

The Open Transport implementation of AppleTalk has a significant feature not found in the classic AppleTalk implementation for Macintosh computers. The Open Transport implementation supports multihoming (sometimes called *multiporting*), which makes it possible for AppleTalk to be active on more than one network port on the machine at a time.

The Open Transport implementation of TCP/IP is a replacement for MacTCP. It is designed for use under the Open Transport software interface.

## Compatibility

Open Transport is compatible with existing AppleTalk networks and supports existing `.ENET` clients such as SoftWindows™ and DECnet.

Open Transport provides compatibility with 680x0-based computers by means of the following features:

- environment options
- 680x0-based APIs and stacks
- Open Transport APIs and stacks
- API compatibility glue
- use of parameter-block APIs with Open Transport stacks for 680x0-based applications

Open Transport provides compatibility with Power Macintosh computers by means of the following features:

- environment options
- 680x0-based APIs and stacks run in emulation mode
- Open Transport APIs and stacks run in native mode
- API compatibility glue runs in mixed mode
- 680x0-based applications can use parameter-block APIs with Open Transport stacks
- 680x0-based applications can use Open Transport APIs and stacks
- native applications can use PB APIs with 680x0-based stacks
- native applications can use PB APIs with Open Transport stacks

## Open Firmware Startup

The Open Firmware startup process in PCI-compatible Macintosh computers conforms to the *IEEE Standard 1275 for Boot Firmware* and the *PCI Bus Binding to IEEE 1275-1994* specification. These specifications are listed in "Supplemental Reference Documents" in the preface.

The Open Firmware startup process is driven by startup firmware (also called *boot firmware*) stored in the Macintosh ROM and in PCI card expansion ROMs. While the startup firmware is running, the Macintosh computer starts up and configures its hardware (including peripheral devices) independently of any operating system. The computer then finds an operating system in ROM or on a mass storage device, loads it into RAM, and terminates the Open Firmware startup process by giving the operating system control of the PowerPC main processor. The operating system may be the Macintosh Operating System or a different system, provided it uses the PowerPC instruction set.

The Open Firmware startup process includes these specific features:

- Startup firmware is written in the Forth language, as defined by IEEE Standard 1275. Firmware code is stored in a tokenized representation called FCode, an abbreviated version of Forth in which most Forth words are replaced by single bytes or 2-byte groups. The startup firmware in the Power Macintosh ROM includes an FCode loader that installs FCode in system RAM so that drivers can run on the PowerPC main processor. Expansion card firmware can modify the Open Firmware startup process by supplying FCode that the computer's startup firmware loads and runs before launching an operating system.
- The startup firmware creates a data structure of nodes called a device tree, in which each PCI device is described by a property list. The device tree is stored in system RAM. The operating system that is ultimately installed and launched can search the device tree to determine what devices are available.
- Device drivers required during system startup (called boot drivers) are also stored in the expansion ROM on the PCI card. Plug-in expansion cards must contain all the driver code required during startup. The boot drivers are native drivers and are

embedded in the FCode in the expansion ROM. The startup firmware in the Power Macintosh ROM installs the boot drivers in system RAM and lets them run on the PowerPC main processor.

- The startup firmware in the Power Macintosh ROM contains debugging facilities for both FCode and some kinds of operating-system code. These facilities can help expansion card designers develop the firmware for new peripheral devices compatible with Macintosh computers.

You can write PCI expansion ROM code in standard Forth words and then reduce the result to FCode by using an FCode tokenizer, a program that translates Forth words into FCodes. The Forth vocabulary that you can use is presented in IEEE Standard 1275.

The burden on developers to provide Forth boot drivers need not be heavy. Developers can choose the level of support that they provide. The following are the three possible levels of support:

- **No driver.** The expansion ROM contains minimal FCode. The Open Firmware startup process recognizes the card and installs a node in the device tree, but no driver code is loaded and no device initialization occurs.
- **Run-time driver.** Only a small amount of Forth code is required to install an OS-dependent run-time driver in the device's property list. Sample code is provided in *Designing PCI Cards and Drivers for Power Macintosh Computers.*
- **Boot driver.** Expansion cards that need to be used at startup time must contain a boot driver with the required methods for the type of device (typically Open, Close, Read, and Write). Sample code is provided in *Designing PCI Cards and Drivers for Power Macintosh Computers.*

## Performance Enhancements

The system software for the Power Macintosh 9500 computers includes the following performance enhancements:

- improved file sharing
- a new Dynamic Recompilation Emulator
- a Resource Manager completely in native code
- an improved math library
- new `BlockMove` extensions

### Improved File Sharing

Version 7.6 of the file sharing software incorporates many of the features of AppleShare, including an API for servers.

The user can now set up shared files on ejectable media such as cartridge drives and CD-ROM drives. The software keeps track of the status of the shared files when the media are inserted and removed.

## Dynamic Recompilation Emulator

The Dynamic Recompilation Emulator (or DR Emulator) is an extension to the current interpretive emulator providing on-the-fly translation of 680x0 instructions into PowerPC instructions for increased performance. The DR Emulator operates as an enhancement to a modified version of the existing interpretive emulator.

The design of the DR Emulator mimics a hardware instruction cache and employs a variable size translation cache. Each compiled 680x0 instruction requires on average fewer than four PowerPC instructions. In operation, the DR Emulator depends on locality of execution to make up for the extra cycles used in translating the code.

The DR Emulator provides a high degree of compatibility for 680x0 code. One area where compatibility will be less than that of the current interpretive emulator is for self-modifying code that does not call the cache flushing routines. Such code also has compatibility problems on Macintosh Quadra models with the cache enabled.

## Resource Manager in Native Code

The Resource Manager in the software for the Power Macintosh 9500 computers is similar to the one in the earlier Power Macintosh computers except that it is completely in native PowerPC code. Because the Resource Manager is intensively used both by system software and by applications, the native version provides an improvement in system performance.

The Process Manager has been modified to remove patches it formerly made to the Resource Manager.

## Math Library

The new math library (MathLib) is an enhanced version of the floating-point library included in the ROM in earlier Power Macintosh computers.

The new math library is bit compatible in both results and floating-point exceptions with the math library in the earlier ROM. The only difference is in the speed of computation.

The new math library has been improved to better exploit the floating-point features of the PowerPC microprocessor. The math library now includes enhancements that assist the compiler in carrying out its register allocation, branch prediction, and overlapping of integer and floating-point operations.

Compared with the previous version, the new math library provides much improved performance without compromising its accuracy or robustness. It provides performance gains for often-used functions of up to 15 times.

The application interface and header files for the math library have not been changed.

## New BlockMove Extensions

The system software for the Power Macintosh 9500 computers includes new extensions to the `BlockMove` routine. The extensions provide improved performance for programs running in native mode.

The new `BlockMove` extensions provide several benefits for developers.

- They're optimized for the PowerPC 603 and PowerPC 604 processors, rather than the PowerPC 601.
- They're compatible with the new Dynamic Recompilation Emulator.
- They provide a way to handle cache-inhibited address spaces.
- They include new high-speed routines for setting memory to zero.

**Note**
The new `BlockMove` extensions do not use the string instructions, which are fast on the PowerPC 601 but slow on other PowerPC implementations. ◆

Some of the new `BlockMove` extensions can be called only from native code; see Table 5-1.

Except for `BlockZero` and `BlockZeroUncached`, the new `BlockMove` extensions use the same parameters as `BlockMove`. Calls to `BlockZero` and `BlockZeroUncached` have only two parameters, a pointer and a length; refer to the header file (`Memory.h`).

Table 5-1 summarizes the `BlockMove` routines and according to three criteria: whether the routine can be called from 680x0 code, whether it is okay to use for moving 680x0 code, and whether it is okay to use with buffers or other uncacheable destination locations.

**Table 5-1** Summary of `BlockMove` routines

| `BlockMove` version | Can be called from 680x0 code | Okay to use for moving 680x0 code | Okay to use with buffers |
|---|---|---|---|
| `BlockMove` | Yes | Yes | No |
| `BlockMoveData` | Yes | No | No |
| `BlockMoveDataUncached` | No | No | Yes |
| `BlockMoveUncached` | No | Yes | Yes |
| `BlockZero` | No | — | No |
| `BlockZeroUncached` | No | — | Yes |

The fastest way to move data is to use the `BlockMoveData` routine. It is the recommended method whenever you are certain that the data is cacheable and does not contain executable 680x0 code.

The `BlockMove` routine is slower than the `BlockMoveData` routine only because it has to clear out the software cache used by the DR Emulator. If the DR EMulator is not in use, the `BlockMove` routine and the `BlockMoveData` routine are the same.

**IMPORTANT**
The versions of `BlockMove` for cacheable data use the `dcbz` instruction to avoid unnecessary pre-fetch of destination cache blocks. For uncacheable data, you should avoid using those routines because the `dcbz` instruction faults and must be emulated on uncacheable or write-through locations, making execution extremely slow. ▲

**IMPORTANT**
Driver software cannot call the `BlockMove` routines directly. Instead, drivers must use the `BlockCopy` routine, which is part of the Driver Services Library. The `BlockCopy` routine is an abstraction that allows you to postpone binding the specific type of `BlockMove` operation until implementation time. ▲

The Driver Services Library is a collection of useful routines that Apple Computer provides for developers working with the new Power Macintosh models. For more information, please refer to *Designing PCI Cards and Drivers for Power Macintosh Computers.*

## Hardware Support Features

The system software for the Power Macintosh 9500 computers includes the following features to support the hardware:

- PCI bus support
- POWER-clean native code
- POWER emulation (for PowerPC 601 compatibility)
- QuickDraw acceleration API
- Display Manager
- support of native drivers

### PCI Bus Support

The Power Macintosh 9500 computers are the first Power Macintosh models that do not use NuBus slots for hardware expansion, but instead use the industry standard PCI bus architecture. To support these computers as well as future Macintosh models that do not use the NuBus architecture, the new system software includes a bus-neutral expansion architecture used by system software in place of Slot Manager calls that are specific to NuBus.

## Removal of Slot Manager Dependencies

The system software that controls NuBus cards in current Macintosh models has many explicit dependencies upon the Slot Manager. The system software for models that use PCI bus slots requires changes to each of those dependencies so that PCI cards can operate with the system in the same fashion as NuBus cards.

The system software that formerly called the Slot Manager has been modified to use other services. The new Display Manager provides the means of obtaining video-specific information that was previously obtained by way of the Slot Manager. For example, QuickDraw currently calls the Slot Manager at startup time to check the consistency of the `'scrn'` resource. In the software for the Power Macintosh 9500 computers, QuickDraw calls the new Display Manager to check this consistency.

The following components formerly used the Slot Manager; they have been modified to use the services of the Display Manager:

- Monitors Control Panel
- QuickDraw
- Palette Manager
- Device Manager

## PCI Compatibility

To support a third-party NuBus-to-PCI bridge product for PCI-based computers, it is important to retain Slot Manager capability. Also, several important applications (such as DECnet and SoftWindows) rely on Slot Manager calls to indicate the presence of networking cards. For compatibility, the new expansion architecture will support existing PCI-based cards by way of particular Slot Manager calls.

Ordinarily, calls to the Slot Manager return an error result; the error code depends on the specific Slot Manager routine being called. If a NuBus-to-PCI bridge is present, Slot Manager calls will function normally if a card occupies the specified slot.

To maintain compatibility with some networking software, the Power Macintosh 9500 computers have a declaration ROM for slot 0. This declaration ROM does not contain any drivers or other data. It is intended as an interim solution and is not to be used by new programs.

For more information about PCI expansion cards, please refer to *Designing PCI Cards and Drivers for Power Macintosh Computers.*

## Setting Up a VBL Task

The `SlotVInstall()` and `SlotVRemove()` calls to the Vertical Retrace Manager require a NuBus slot number as a parameter. You make such calls to set up a routine to be executed during the vertical blanking interval of a particular video display (that is, a slot VBL task). The Vertical Retrace Manager in the Power Macintosh 9500 computers still supports those calls, but the slot number no longer corresponds to a NuBus slot.

The preferred method of obtaining a slot number to use with the `SlotVInstall()` and `SlotVRemove()` routines is to look at the `dCtlSlot` field of the DCE entry for the driver controlling the card in question. For a NuBus card, that number corresponds to the card's slot number. In a PCI-based machine, the `dCtlSlot` field contains a pseudo-slot number that is assigned by the Display Manager during system startup. You can use that number as the `theSlot` parameter in calls to the `SlotVInstall()` and `SlotVRemove()` routines. Note that the number does not correspond to the physical location of the slot.

## POWER-Clean Native Code

The instruction set of the PowerPC 601 microprocessor includes some of the same instructions as those found in the instruction set of the POWER processor, and the compiler used to generate native code for the system software in the earlier Power Macintosh models generated some of those POWER-only instructions. However, the PowerPC 604 microprocessor used in the Power Macintosh 9500 computers does not support the POWER-only instructions, so a new POWER-clean version of the compiler is being used to compile the native code fragments.

**Note**
The term *POWER-clean* refers to code that is free of the POWER instructions that would prevent it from running correctly on a PowerPC 603 or PowerPC 604 microprocessor. ◆

Here is a list of the POWER-clean native code fragments in the system software for the Power Macintosh 9500 computers.

- interface library
- private interface library
- native QuickDraw
- MathLib
- Mixed Mode Manager
- Code Fragment Manager
- Font Dispatch
- Memory Manager
- standard text
- the `FMSwapFont` function
- Standard C Library

## POWER Emulation

Earlier Power Macintosh computers included emulation for certain PowerPC 601 instructions that would otherwise cause an exception. The emulation code dealt with memory reference instructions to handle alignment and data storage exceptions. It also handled illegal instruction exceptions caused by some PowerPC instructions that were not implemented in the PowerPC 601. Starting with the Power Macintosh 9500 computers, the emulation code has been enhanced to include the POWER instructions that are implemented on the PowerPC 601 but not on the PowerPC 604.

**Note**
Although the term *POWER emulation* is often used, a more appropriate name for this feature is *PowerPC 601 compatibility.* Rather than supporting the entire POWER architecture, the goal is to support those features of the POWER architecture that are available to programs running in user mode on the PowerPC 601-based Power Macintosh computers. ◆

### POWER-Clean Code

Because the emulation of the POWER-only instructions degrades performance, Apple Computer recommends that developers revise any applications that use those instructions to conform with the PowerPC architecture. POWER emulation works, but at a significant cost in performance; POWER-clean code is preferable.

### Emulation and Exception Handling

When an exception occurs, the emulation code first checks to see whether the instruction encoding is supported by emulation. If it is not, the code passes the original cause of the exception (illegal instruction or privileged instruction) to the application as a native exception.

If the instruction is supported by emulation, the code then checks a flag bit to see whether emulation has been enabled. If emulation is not enabled at the time, the emulator generates an illegal instruction exception.

### Code Fragments and Cache Coherency

Whereas the PowerPC 601 microprocessor has a single cache for both instructions and data, the PowerPC 604 has separate instruction and data caches. As long as applications deal with executable code by using the Code Fragment Manager, cache coherency is maintained. Applications that bypass the Code Fragment Manager and generate executable code in memory, and that do not use the proper cache synchronization instructions or CFM calls, are likely to encounter problems when running on the PowerPC 604.

**IMPORTANT**
The emulation software in the Power Macintosh 9500 computers cannot make the separate caches in the PowerPC 604 behave like the combined cache in the PowerPC 601. Applications that generate executable code in memory must be modified to use the Code Fragment Manager or maintain proper cache synchronization by other means. ▲

### Limitations of PowerPC 601 Compatibility

The emulation code in the Power Macintosh 9500 computers allows programs compiled for the PowerPC 601 to execute without halting on an exception whenever they use a POWER-only feature. For most of those features, the emulation matches the results that are obtained on a Power Macintosh computer with a PowerPC 601. However, there are a few cases where the emulation is not an exact match; those cases are summarized here.

- **MQ register.** Emulation does not match the undefined state of this register after multiply and divide instructions.
- **`div` and `divo` instructions.** Emulation does not match undefined results after an overflow.
- **Real-time clock registers.** Emulation matches the 0.27 percent speed discrepancy of the Power Macintosh models that use the PowerPC 601 microprocessor, but the values of the low-order 7 bits are not 0.
- **POWER version of `dec` register.** Emulation includes the POWER version, but decrementing at a rate determined by the time base clock, not by the real-time clock.
- **Cache line compute size (`clcs`) instruction.** Emulation returns values appropriate for the type of PowerPC microprocessor.
- **Undefined SPR encodings.** Emulation does not ignore SPR encodings higher than 32.
- **Invalid forms.** Invalid combinations of register operands with certain instructions may produce results that do not match those of the PowerPC 601.
- **Floating-point status and control register (FPSCR).** The FPSCR in the PowerPC 601 does not fully conform to the PowerPC architecture, but the newer PowerPC processors do.

## QuickDraw Acceleration API

The Power Macintosh 9500 computers have no built-in display hardware and are dependent on a PCI-bus expansion card to provide their display support. The native QuickDraw acceleration API makes it easier for third-party card vendors and driver writers to produce video accelerator cards for this computer.

The QuickDraw acceleration API is the current accelerator interface for the PowerPC version of native QuickDraw. It allows a patch chaining mechanism for decisions on categories of blits, and also specifies the format and transport of the data to the accelerator.

**IMPORTANT**
This interface and design is intended only for the Power Macintosh 9500 computers and related models; it does not represent a new standard for future Macintosh models. ▲

## Display Manager

Until now, system software has used the NuBus-specific Slot Manager to get and set information about display cards and drivers. Starting with the Power Macintosh 9500 computers, new system software removes this explicit software dependency on the architecture of the expansion bus. The Display Manager provides a uniform API for display devices regardless of the implementation details of the devices.

In a computer that uses PCI expansion cards, the Slot Manager is generally not available to provide information about display cards; instead, the Expansion Manager must be used. Starting with the software for the Power Macintosh 9500 computers, the Display Manager makes the actual calls to either the Slot Manager or the Expansion Manager, as appropriate, thus isolating the bus-specific calls to a single component and avoiding the need to change additional system software in the future. See the section "Removal of Slot Manager Dependencies" on page 45.

## Support of Native Drivers

The Power Macintosh 9500 computers use a new native-driver model for system software and device driver developers. Several components of system software are being modified to support native drivers. The modified components are:

- Device Manager
- interrupt tree services
- driver loader library
- driver support library
- Slot Manager stubs
- Macintosh startup code
- interface libraries
- system registry

For more information, refer to *Designing PCI Cards and Drivers for Power Macintosh Computers.*

C H A P T E R 6

# Large Volume Support

This chapter describes the large volume file system for the Power Macintosh 9500 computers. The large volume file system is a version of the hierarchical file system (HFS) that has been modified to support volume sizes larger than the current 4 GB limit. It incorporates only the changes required to achieve that goal.

## Overview of the Large Volume File System

The large volume file system includes

- modifications to the HFS ROM code, Disk First Aid, and Disk Init
- a new extended API that allows reporting of volume size information beyond the current 4 GB limit
- new device drivers and changes to the Device Manager API to support devices that are greater than 4 GB
- a new version of HDSC Setup that supports large volumes and chainable drivers (Chainable drivers are needed to support booting large volumes on earlier Macintosh models.)

### API Changes

The system software on the Power Macintosh 9500 computers allows all current applications to work without modifications. Unmodified applications that call the file system still receive incorrect values for large volume sizes. The Finder and other utility programs that need to know the actual size of a volume have been modified to use the new extended `PBXGetVolInfo` function to obtain the correct value.

The existing low-level driver interface does not support I/O to a device with a range of addresses greater than 4 GB because the positioning offset (in bytes) for a read or write operation is a 32-bit value. To correct this problem, a new extended I/O parameter block record has been defined. This extended parameter block has a 64-bit positioning offset. The new parameter block and the extended `PBXGetVolInfo` function are described in “The API Modifications” beginning on page 53.

### Allocation Block Size

The format of HFS volumes has not changed. What has changed is the way the HFS software handles the allocation block size. Existing HFS code treats the allocation block as a 16-bit integer. The large volume file system uses the full 32 bits of the allocation block size parameter. In addition, any software that deals directly with the allocation block size from the volume control block must now treat it as a true 32-bit value.

Even for the larger volume sizes, the number of allocation blocks is still defined by a 16-bit integer. As the volume size increases, the size of the allocation block also increases. For a 2 GB volume, the allocation block size is 32 KB and therefore the smallest file on that disk will occupy at least 32 KB of disk space. This inefficient use of disk space is not addressed by the large volume file system.

The maximum number of files will continue to be less than 65,000. This limit is directly related to the fixed number of allocation blocks.

### File Size Limits

The HFS has a maximum file size of 2 GB. The large volume file system does not remove that limit because doing so would require a more extensive change to the current API and would incur more compatibility problems.

### Compatibility Requirements

The large volume file system requires at least a 68020 microprocessor or a Power Macintosh model that emulates it. In addition, the file system requires a Macintosh IIci or more recent model. On a computer that does not meet both those requirements, the large volume file system driver will not load.

The large volume file system requires System 7.5 or higher and a new Finder that supports volumes larger than 4 GB (using the new extended `PBXGetVolInfo` function).

## The API Modifications

The HFS API has been modified to support volume sizes larger than 4 GB. The modifications consist of two extended data structures and a new extended `PBXGetVolInfo` function.

### Data Structures

This section describes the two modified data structures used by the large volume file system:

- the extended volume parameter block
- the extended I/O parameter block

#### Extended Volume Parameter Block

In the current `HVolumeParam` record, volume size information is clipped at 2 GB. Because HFS volumes can now exceed 4 GB, a new extended volume parameter block is needed in order to report the larger size information. The `XVolumeParam` record contains 64-bit integers for reporting the total bytes on the volume and the number of free bytes available (parameter names `ioVTotalBytes` and `ioVFreeBytes`). In addition, several of the fields that were previously signed are now unsigned (parameter names `ioVAtrb`, `ioVBitMap`, `ioAllocPtr`, `ioVAlBlkSiz`, `ioVClpSiz`, `ioAlBlSt`, `ioVNxtCNID`, `ioVWrCnt`, `ioVFilCnt`, and `ioVDirCnt`).

```
struct XVolumeParam {
   ParamBlockHeader
   unsigned long    ioXVersion;      // XVolumeParam version == 0
   short            ioVolIndex;      // volume index
   unsigned long    ioVCrDate;       // date & time of creation
   unsigned long    ioVLsMod;        // date & time of last modification
   unsigned short   ioVAtrb;         // volume attributes
   unsigned short   ioVNmFls;        // number of files in root directory
   unsigned short   ioVBitMap;       // first block of volume bitmap
   unsigned short   ioAllocPtr;      // first block of next new file
   unsigned short   ioVNmAlBlks;     // number of allocation blocks
   unsigned long    ioVAlBlkSiz;     // size of allocation blocks
   unsigned long    ioVClpSiz;       // default clump size
   unsigned short   ioAlBlSt;        // first block in volume map
   unsigned long    ioVNxtCNID;      // next unused node ID
   unsigned short   ioVFrBlk;        // number of free allocation blocks
   unsigned short   ioVSigWord;      // volume signature
   short            ioVDrvInfo;      // drive number
   short            ioVDRefNum;      // driver reference number
   short            ioVFSID;         // file-system identifier
   unsigned long    ioVBkUp;         // date & time of last backup
   unsigned short   ioVSeqNum;       // used internally
   unsigned long    ioVWrCnt;        // volume write count
   unsigned long    ioVFilCnt;       // number of files on volume
   unsigned long    ioVDirCnt;       // number of directories on volume
   long             ioVFndrInfo[8];  // information used by the Finder
   uint64           ioVTotalBytes;   // total number of bytes on volume
   uint64           ioVFreeBytes;    // number of free bytes on volume
};
```

**Field descriptions**

| | |
|---|---|
| `ioVolIndex` | An index for use with the `PBHGetVInfo` function. |
| `ioVCrDate` | The date and time of volume initialization. |
| `ioVLsMod` | The date and time the volume information was last modified. (This field is not changed when information is written to a file and does not necessarily indicate when the volume was flushed.) |
| `ioVAtrb` | The volume attributes. |
| `ioVNmFls` | The number of files in the root directory. |
| `ioVBitMap` | The first block of the volume bitmap. |
| `ioAllocPtr` | The block at which the next new file starts. Used internally. |
| `ioVNmAlBlks` | The number of allocation blocks. |
| `ioVAlBlkSiz` | The size of allocation blocks. |
| `ioVClpSiz` | The clump size. |

| | |
|---|---|
| `ioAlBlSt` | The first block in the volume map. |
| `ioVNxtCNID` | The next unused catalog node ID. |
| `ioVFrBlk` | The number of unused allocation blocks. |
| `ioVSigWord` | A signature word identifying the type of volume; it's $D2D7 for MFS volumes and $4244 for volumes that support HFS calls. |
| `ioVDrvInfo` | The drive number of the drive containing the volume. |
| `ioVDRefNum` | For online volumes, the reference number of the I/O driver for the drive identified by `ioVDrvInfo`. |
| `ioVFSID` | The file-system identifier. It indicates which file system is servicing the volume; it's zero for File Manager volumes and nonzero for volumes handled by an external file system. |
| `ioVBkUp` | The date and time the volume was last backed up (it's 0 if never backed up). |
| `ioVSeqNum` | Used internally. |
| `ioVWrCnt` | The volume write count. |
| `ioVFilCnt` | The total number of files on the volume. |
| `ioVDirCnt` | The total number of directories (not including the root directory) on the volume. |
| `ioVFndrInfo` | Information used by the Finder. |

## Extended I/O Parameter Block

The extended I/O parameter block is needed for low-level access to disk addresses beyond 4 GB. It is used exclusively by `PBRead` and `PBWrite` calls when performing I/O operations at offsets greater than 4 GB. To indicate that you are using an `XIOParam` record, you should set the `kUseWidePositioning` bit in the `ioPosMode` field.

Because file sizes are limited to 2 GB, the regular `IOParam` record should always be used when performing file level I/O operations. The extended parameter block is intended only for Device Manager I/O operations to large block devices at offsets greater than 4 GB.

The only change in the parameter block is the parameter `ioWPosOffset`, which is of type `int64`.

```
struct XIOParam {
   QElemPtr      qLink;       // next queue entry
   short         qType;       // queue type
   short         ioTrap;      // routine trap
   Ptr           ioCmdAddr;   // routine address
   ProcPtr       ioCompletion;// pointer to completion routine
   OSErr         ioResult;    // result code
   StringPtr     ioNamePtr;   // pointer to pathname
   short         ioVRefNum;   // volume specification
   short         ioRefNum;    // file reference number
   char          ioVersNum;   // not used
   char          ioPermssn;   // read/write permission
   Ptr           ioMisc;      // miscellaneous
   Ptr           ioBuffer;    // data buffer
   unsigned long ioReqCount;  // requested number of bytes
   unsigned long ioActCount;  // actual number of bytes
   short         ioPosMode;   // positioning mode (wide mode set)
   int64         ioWPosOffset;// wide positioning offset
};
```

**Field descriptions**

| | |
|---|---|
| `ioRefNum` | The file reference number of an open file. |
| `ioVersNum` | A version number. This field is no longer used and you should always set it to 0. |
| `ioPermssn` | The access mode. |
| `ioMisc` | Depends on the routine called. This field contains either a new logical end-of-file, a new version number, a pointer to an access path buffer, or a pointer to a new pathname. Because `ioMisc` is of type `Ptr`, you'll need to perform type coercion to interpret the value of `ioMisc` correctly when it contains an end-of-file (a `LongInt` value) or version number (a `SignedByte` value). |
| `ioBuffer` | A pointer to a data buffer into which data is written by `_Read` calls and from which data is read by `_Write` calls. |
| `ioReqCount` | The requested number of bytes to be read, written, or allocated. |
| `ioActCount` | The number of bytes actually read, written, or allocated. |
| `ioPosMode` | The positioning mode for setting the mark. Bits 0 and 1 of this field indicate how to position the mark; you can use the following predefined constants to set or test their value: |

```
CONST
fsAtMark = 0;       {at current mark}
fsFromStart = 1;    {from beginning of file}
fsFromLEOF = 2;     {from logical end-of-file}
fsFromMark = 3;     {relative to current mark}
```

You can set bit 4 of the `ioPosMode` field to request that the data be cached, and you can set bit 5 to request that the data not be cached. You can set bit 6 to request that any data written be immediately read; this ensures that the data written to a volume exactly matches the data in memory. To request a read-verify operation, add the following constant to the positioning mode:

```
CONST
rdVerify = 64;      {use read-verify mode}
```

You can set bit 7 to read a continuous stream of bytes, and place the ASCII code of a newline character in the high-order byte to terminate a read operation at the end of a line.

| | |
|---|---|
| `ioPosOffset` | The offset to be used in conjunction with the positioning mode. |

## New Extended Function

This section describes the extended `PBXGetVolInfo` function that provides volume size information for volumes greater than 4 GB.

Before using the new extended call, you should check for availability by calling the `Gestalt` function. Make your call to `Gestalt` with the `gestaltFSAttr` selector to check for new File Manager features. The response parameter has the `gestaltFSSupports2TBVolumes` bit set if the File Manager supports large volumes and the new extended function is available.

### PBXGetVolInfo

You can use the `PBXGetVolInfo` function to get detailed information about a volume. It can report volume size information for volumes up to 2 terabytes.

```
pascal OSErr PBXGetVolInfo (XVolumeParam paramBlock,
Boolean async);
```

| | |
|---|---|
| `paramBlock` | A pointer to an extended volume parameter block. |
| `async` | A Boolean value that specifies asynchronous (true) or synchronous (false) execution. |

An arrow preceding a parameter indicates whether the parameter is an input parameter, an output parameter, or both:

| **Arrow** | **Meaning** |
|---|---|
| → | Input |
| ← | Output |
| ↔ | Both |

**Parameter block**

| | | | |
|---|---|---|---|
| → | `ioCompletion` | `ProcPtr` | Pointer to a completion routine. |
| ← | `ioResult` | `OSErr` | Result code of the function. |
| ↔ | `ioNamePtr` | `StringPtr` | Pointer to the volume's name. |
| ↔ | `ioVRefNum` | `short` | On input, a volume specification; on output, the volume reference number. |
| → | `ioXVersion` | `unsigned long` | Version of `XVolumeParam` (value = 0). |
| → | `ioVolIndex` | `short` | Index used for indexing through all mounted volumes. |
| ← | `ioVCrDate` | `unsigned long` | Date and time of initialization. |
| ← | `ioVLsMod` | `unsigned long` | Date and time of last modification. |

| | | | |
|---|---|---|---|
| ← | `ioVAtrb` | `unsigned short` | Volume attributes. |
| ← | `ioVNmFls` | `unsigned short` | Number of files in the root directory. |
| ← | `ioVBitMap` | `unsigned short` | First block of the volume bitmap. |
| ← | `ioVAllocPtr` | `unsigned short` | Block where the next new file starts. |
| ← | `ioVNmAlBlks` | `unsigned short` | Number of allocation blocks. |
| ← | `ioVAlBlkSiz` | `unsigned long` | Size of allocation blocks. |
| ← | `ioVClpSiz` | `unsigned long` | Default clump size. |
| ← | `ioAlBlSt` | `unsigned short` | First block in the volume block map. |
| ← | `ioVNxtCNID` | `unsigned long` | Next unused catalog node ID. |
| ← | `ioVFrBlk` | `unsigned short` | Number of unused allocation blocks. |
| ← | `ioVSigWord` | `unsigned short` | Volume signature. |
| ← | `ioVDrvInfo` | `short` | Drive number. |
| ← | `ioVDRefNum` | `short` | Driver reference number. |
| ← | `ioVFSID` | `short` | File system handling this volume. |
| ← | `ioVBkUp` | `unsigned long` | Date and time of last backup. |
| ← | `ioVSeqNum` | `unsigned short` | Used internally. |
| ← | `ioVWrCnt` | `unsigned long` | Volume write count. |
| ← | `ioVFilCnt` | `unsigned long` | Number of files on the volume. |
| ← | `ioVDirCnt` | `unsigned long` | Number of directories on the volume. |
| ← | `ioVFndrInfo[8]` | `long` | Used by the Finder. |
| ← | `ioVTotalBytes` | `uint64` | Total number of bytes on the volume. |
| ← | `ioVFreeBytes` | `uint64` | Number of free bytes on the volume. |

**DESCRIPTION**

The `PBXGetVolInfo` function returns information about the specified volume. It is similar to the `PBHGetVInfo` function described in *Inside Macintosh: Files* except that it returns additional volume space information in 64-bit integers.

### ASSEMBLY-LANGUAGE INFORMATION

The trap macro and routine selector for `PBXGetVolInfo` are:

| Trap macro | Selector |
|---|---|
| `_HFSDispatch` | $0012 |

### RESULT CODES

| | | |
|---|---|---|
| `noErr` | 0 | Successful completion, no error occurred |
| `nsvErr` | –35 | No such volume |
| `paramErr` | –50 | No default volume |

# Glossary

**680x0 code** Instructions that can run on a PowerPC processor only by means of an emulator. Compare **native code.**

**ADB** See **Apple Desktop Bus.**

**APDA** Apple Computer's worldwide direct distribution channel for Apple and third-party development tools and documentation products.

**API** See **application programming interface.**

**Apple AV Technologies** A set of advanced I/O features for Macintosh computers that includes video input and output, sophisticated 16-bit stereo sound input and output, and speech recognition and synthesis.

**Apple Desktop Bus (ADB)** An asynchronous bus used to connect relatively slow user-input devices to Apple computers.

**Apple SuperDrive** Apple Computer's disk drive for high-density floppy disks.

**AppleTalk** Apple Computer's local area networking protocol.

**application programming interface (API)** A set of calls, instructions, and data structures in system software or a processor instruction set that application software can use to program the computer.

**arbitration** The process of determining which of several contending subsystems gains control of a bus at any given time.

**ASLM** Apple Shared Library Manager.

**audio waveform amplifier and converter (AWAC)** A custom IC that combines a waveform amplifier with a digital encoder and decoder (codec) for analog sound signals, including speech.

**autoconfiguration** A method of integrating peripheral devices into a computer system that includes a mechanism for configuring devices during system startup and requires that vendors include expansion ROMs on plug-in cards.

**AWAC** See **audio waveform amplifier and converter.**

**Bandit** A custom IC that provides the interface between the system bus and the PCI bus in the Power Macintosh 9200/120 computer.

**big-endian** Data formatting where fields are addressed by pointers to their most significant bytes or bits. See also **little-endian.**

**block transfer** Data transfers of more than one longword at a time.

**boot driver** A device driver that is used during the Open Firmware startup process. It must be written in FCode and is usually loaded from the expansion ROM on a PCI card.

**branch unit** The part of a PowerPC microprocessor that handles branch instructions, usually without adding any time to program execution.

**CISC** See **complex instruction set computing.**

**codec** A digital encoder and decoder.

**color depth** The number of bits required to encode the color of each pixel in a display.

**complex instruction set computing (CISC)** A technology of microprocessor design in which machine instructions have nonuniform formats and are executed through different processes.

**convolution** The process of smoothing alternate lines of a video signal to be shown in succeeding frames for a line-interlaced display.

**CPU bus** The bus connected directly to the main processor.

**Cuda** A microcontroller IC that manages the ADB and real-time clock, maintains parameter RAM, manages power on and reset, and performs other general system functions.

**Curio** A custom IC that provides I/O interfaces for Ethernet, SCSI, SCC, and LocalTalk.

**DAC** See **digital-to-analog converter.**

**data burst** Multiple longwords of data sent over a bus in a single, uninterrupted stream.

**data cache** In a PowerPC microprocessor, the internal registers that hold data being processed.

**descriptor-based direct memory access (DBDMA)** A DMA technique using DMA descriptor lists that are read from memory by the IC performing the DMA transfers.

**device node** In a device tree, a node that serves one hardware device.

**device tree** A software structure, generated during the Open Firmware startup process, that assigns a node to each PCI device available to the system.

**digital-to-analog converter (DAC)** A device that produces an analog electrical signal in response to digital data.

**direct memory access (DMA)** A process for transferring data rapidly into or out of RAM without passing it through a processor or buffer.

**DLPI** Data Link Provider Interface, the standard networking model used in Open Transport.

**DMA** See **direct memory access.**

**DRAM** See **dynamic random-access memory.**

**DR Emulator** The Dynamic Recompilation Emulator, an improved 680x0-code emulator for the PowerPC microprocessors.

**dynamic random-access memory (DRAM)** Random-access memory in which each storage address must be periodically interrogated ("refreshed") to maintain its value.

**Ethernet** A high-speed local area network technology that includes both cable standards and a series of communications protocols.

**exception** An error or other unusual condition detected by a processor during program execution.

**Expansion Manager** A set of toolbox routines that provides bus-neutral support for expansion devices. The Expansion Manager is a superset of the Slot Manager.

**fat binary** An execution module that contains native code and 680x0 code for the same program.

**FCode** A tokenized version of the Forth programming language, used in PCI card expansion ROMs.

**FCode tokenizer** A utility program that translates Forth source code into FCode.

**floating-point format** A data format that stores the magnitude, sign, and significant digits of a number separately.

**fragment** An application, shared library, system extension, or any other block of executable code and its associated data in a Power Macintosh computer.

**GeoPort** A software and hardware solution for digital telecom and wide-area connectivity using the serial port.

**Grand Central** A custom IC that provides core I/O services in the Power Macintosh 9200/120 computer.

**Hammerhead** A custom IC that controls the memory and cache system in the Power Macintosh 9200/120 computer.

**input/output (I/O)** Parts of a computer system that transfer data to or from peripheral devices.

**instruction queue** The part of a PowerPC microprocessor that holds incoming instructions.

**I/O** See **input/output.**

**little-endian** Data formatting where fields are addressed by pointers to their least significant bytes or bits. See also **big-endian.**

**LocalTalk** The cable terminations and other hardware that Apple supplies for local area networking from Macintosh serial ports.

**MACE** See **Media Access Controller for Ethernet.**

**MC68000** The model number of a family of microprocessor ICs manufactured by Motorola.

**Media Access Controller for Ethernet (MACE)** Circuitry within the Curio IC that supports Ethernet I/O.

**MFM** See **Modified Frequency Modulation.**

**mini-DIN** An international standard form of cable connector for peripheral devices.

**Modified Frequency Modulation (MFM)** A recording format for floppy disks used by DOS computers.

**MovieTalk** Apple Computer's set of protocols and APIs that support video conferences over networks.

**multihoming** A feature of Open Transport that makes it possible for AppleTalk to be active on more than one network port on the machine at a time. Also called *multiporting.*

**native code** Instructions that run directly on a PowerPC processor. Compare **680x0 code.**

**nonvolatile RAM** RAM that retains its contents even when the computer is turned off; also known as parameter RAM.

**NuBus** A bus architecture that supports plug-in expansion cards in many Macintosh models.

**Open Firmware startup process** The startup process by which PCI-compatible Power Macintosh computers recognize and configure peripheral devices connected to the PCI bus.

**Open Transport** A networking architecture that allows communications applications to run independently of the underlying network; formerly known as *Transport-Independent Interface (TII).*

**PCI** Peripheral Component Interconnect, an industry-standard expansion bus architecture.

**pipelining** The technique of sending instructions through multiple processing units in such a way that each unit handles one instruction per clock cycle.

**pixel** Contraction of *picture element*; the smallest dot that can be drawn on a display.

**POWER-clean** Refers to PowerPC code free of instructions that are specific to the PowerPC 601 and POWER instruction sets and are not found on the PowerPC 603 and PowerPC 604 microprocessors.

**PowerPC** Tradename for a family of RISC microprocessors. The PowerPC 601, 603, and 604 microprocessors are used in Power Macintosh computers.

**processing unit** The part of a PowerPC microprocessor that executes instructions. A PowerPC microprocessor can have more than one processing unit.

**property list** An element of a device tree that contains information about a device on a PCI bus.

**reduced instruction set computing (RISC)** A technology of microprocessor design in which all machine instructions are uniformly formatted and are processed through the same steps.

**RISC** See **reduced instruction set computing.**

**SCC** See **Serial Communications Controller.**

**SCSI** See **Small Computer System Interface.**

**Serial Communications Controller (SCC)** Circuitry on the Curio IC that provides an interface to the serial data ports.

**SIMM** See **Single Inline Memory Module.**

**Single Inline Memory Module (SIMM)** A plug-in card for memory expansion, containing several RAM ICs and their interconnections.

**Small Computer System Interface (SCSI)** An industry standard parallel bus protocol for connecting computers to peripheral devices such as hard disk drives.

**Streams** The standard UNIX-based networking model used in Open Transport.

**S-video** A video format in which chroma and luminance are transmitted separately; also called Y/C. It provides higher image quality than composite video.

**SWIM III** A custom IC that controls the Apple SuperDrive floppy disk drive.

**Versatile Interface Adapter (VIA)** The interface for system interrupts that is standard on most Apple computers.

**VIA** See **Versatile Interface Adapter.**

**video RAM (VRAM)** Random-access memory used to store both static graphics and video frames.

**VRAM** See **video RAM**.

**VRAM expansion card** A PDS card that adds second monitor support with 2 MB or 4 MB of VRAM to the Power Macintosh 7100/66 or 8100/80.

**Y/C** Same as **S-video.**

**YUV** A data format for each pixel of a color display in which color is encoded by values calculated from its native red, green, and blue components.

# Index

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Proof pages and final pages were created on an Apple LaserWriter Pro 630 printer. Line art was created using Adobe Illustrator™ and Adobe Photoshop™. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

WRITER
Allen Watson III

DEVELOPMENTAL EDITORS
Sanborn Hodgkins, Wendy Krafft

ILLUSTRATORS
Shawn Morningstar, Sandee Karr

Special thanks to Jano Banks, Michael Bowes, Gary Davidian, Paul Freeburn, Brian Heaney, Jim Kelly, Phil Koch, Dave Limp, Tom Llewellyn, Steve Polzin, James Stockdale, and TeriAnn Wakeman.