Developer Note

Power Macintosh 5260 Computer

Developer Press

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manual or in the media on which a software product is distributed, APDA will replace the media or manual at no charge to you provided you return the item to be replaced with proof of purchase to APDA.

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Figures, Tables, and Listings

About This Note

This developer note describes the Power Macintosh 5260 computer. It is intended to help experienced Macintosh hardware and software developers design compatible products. If you are unfamiliar with Macintosh computers or would simply like more technical information, you may wish to read the related technical manuals listed in the section "Supplemental Reference Documents."

Contents of This Note

The information is arranged in six chapters, an appendix, and an index.

- Chapter 1, "Introduction to the Power Macintosh 5260 Computer," gives a summary of the features of the computer, describes the physical appearance, and lists available configurations and options.
- Chapter 2, "Architecture," describes the internal organization of the computer. 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. It also describes the built-in monitor configuration and the external video monitors that can be used with the computer.
- Chapter 4, "Expansion Features," describes the expansion slots of the computer. This chapter provides guidelines for designing cards for the I/O expansion slot and brief descriptions of the expansion modules for other slots, such as the PDS slot.
- Chapter 5, "Software Features," summarizes the new features of the ROM software and the system software that accompany the Power Macintosh 5260 computer.
- Chapter 6, "Software for the IDE Hard Disk," describes the program interface for the system software and the driver that support the internal IDE hard disk drive.
- The appendix contains mechanical drawings for the I/O expansion card described in Chapter 4.

Supplemental Reference Documents

To supplement the information in this developer note, you should have copies of the appropriate Motorola/IBM reference books for the PowerPC™ 603e microprocessor:

- Software developers should have a copy of the Motorola/IBM *PowerPC Programmer's Reference Manual*.
- Hardware developers should have copies of the Motorola/IBM *PowerPC 603 RISC Microprocessor User's Manual* and the *MC68030 User's Manual.*

For additional information about the digital data format used in the video input module and the digital video interface refer to:

- *Macintosh DAV Interface for NuBus Expansion Cards*, part of *Macintosh Developer Note Number 8*, ADC (AppleDeveloper Catalog) number R0566LL/A.
- *SAA7194/6 Philips Desktop Video Handbook*.

You may also need copies of the appropriate Apple reference books:

- Relevant books of the *Inside Macintos*h series, particularly *Inside Macintosh: QuickTime Components.*
- *Guide to the Macintosh Family Hardware,* second edition.
- *Designing Cards and Drivers for the Macintosh Family*, third edition. These books are available in technical bookstores and through ADC.

Information About Earlier Models

Many features of the Power Macintosh 5260 computer are similar to those of certain earlier Macintosh models, so you may wish to have the developer notes that describe those earlier machines:

- *Macintosh Developer Note Number 3*, ADC number R0461LL/A
- *Macintosh Developer Note Number 4*, ADC number R0528LL/A
- *Macintosh Developer Note Number 6*, ADC number R0550LL/A
- *Macintosh Developer Note Number 10*, ADC number R0568LL/A
- *Power Macintosh 5200/75 and Power Macintosh 6200/75 Computers*

Macintosh Developer Note Number 3 includes information about the Macintosh LC III and the Macintosh Centris 610 and 650 computers. *Macintosh Developer Note Number 4* includes information about the Macintosh LC 520 computer. *Macintosh Developer Note Number 6* includes information about the Macintosh LC 475 and Macintosh Quadra 605

computers. *Macintosh Developer Note Number 10* includes information about the Macintosh Quadra 630 and the Macintosh LC 630 computers.

The numbered developer notes are available from ADC. Developer notes for individual models are also on the developer CDs.

For More Information

ADC 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 *ADC Tools Catalog* featuring all current versions of Apple development tools and the most popular third-party development tools. ADC offers convenient payment and shipping options, including site licensing.

To order products or to request a complimentary copy of the *Apple Developer Catalog*, contact

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

Telephone	1-800-282-2732 (United States) 1-800-637-0029 (Canada) 716-871-6555 (International)
Fax	716-871-6511
AppleLink	ORDER.ADC
Internet	order.adc@applelink.apple.com

Conventions and Abbreviations

This developer note uses the following typographical conventions and abbreviations.

Typographical Conventions

Terms defined in the glossary appear in **boldface** the first time they are mentioned.

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. ▲

▲ **WARNING**
A note like this directs your attention to something that could cause damage or result in a loss of data. ▲

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	µA	microamperes
dB	decibels	MB	megabytes
GB	gigabytes	Mbit	megabit
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
mA	milliamperes		

Here are other abbreviations used in this developer note:

$*n*	hexadecimal value *n*
AAUI	
AC	alternating current
A/D	analog/digital
ADB	Apple Desktop Bus
AGC	automatic gain control
AV	audio video
CD	compact disk

CD-ROM	compact-disk read-only memory
CMOS	complementary metal-oxide semiconductor
CLUT	color lookup table
DESC	digital video decoder and scaler
DIMM	dual inline memory module
DRAM	dynamic random-access memory
EMI	electromagnetic interference
FPU	floating-point unit
IC	integrated circuit
IDE	integrated device electronics
IIC	inter-integrated circuit (an internal control bus)
I/O	input/output
IR	infrared
L1	level 1 (cache)
L2	level 2 (cache)
LS TTL	low-power Schottky TTL (a standard type of device)
MMU	memory management unit
MOS	metal-oxide semiconductor
NTSC	National Television Standards Committee (the standard system used for broadcast TV in North America and Japan)
NMI	nonmaskable interrupt
PAL	Phase Alternating Line system (the standard for broadcast TV in most of Europe, Africa, South America, and southern Asia)
PDS	processor-direct slot
PWM	pulse-width modulation
RAM	random-access memory
RGB	a video signal format with separate red, green, and blue components
RISC	reduced instruction set computing
RMS	root-mean-square
ROM	read-only memory
SANE	Standard Apple Numerics Environment
SCSI	Small Computer System Interface
SCC	serial communications controller
SECAM	the standard system used for broadcast TV in France and the former Soviet countries
SIMM	single inline memory module
S-video	a type of video connector that keeps luminance and chrominance separate; also called a Y/C connector

SWIM	Super Woz Integrated Machine, a custom IC that controls the floppy disk interface
TTL	transistor-transistor logic (a standard logic device)
VCR	video cassette recorder
VLSI	very large scale integration
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 to the Power Macintosh 5260 Computer

The Power Macintosh 5260 computer is a new Macintosh model that incorporates a PowerPC™ 603e RISC (reduced instruction set computing) microprocessor running at 100 MHz with an L1 (Level 1) cache. The Power Macintosh 5260 has the same AV features (audio and video input and output) as the Macintosh LC 630, the Macintosh Quadra 630, and the Power Macintosh 5200 and 6200 computers. The Power Macintosh 5260 is housed in an all-in-one enclosure featuring a tilt-and-swivel 14-inch monitor with stereo speakers.

Summary of Features

Here is a summary of the hardware features of the Power Macintosh 5260 computer. Each feature is described more fully later in this note.

- PowerPC 603e microprocessor running at 100 MHz, with L1 cache.
- Support for an optional 256 KB L2 (Level 2) cache installed in an L2 DIMM (dual in-line memory module) slot. Apple Computer does not support third-party development of L2 caches.
- Two DRAM SIMM (single in-line memory module) slots provide DRAM capacity from 8 MB to 64 MB, with a 32-bit data bus.
- 1 MB of on-board DRAM for use as a frame buffer.
- 4 MB of 60 ns on-board burst ROM with a 64-bit data bus.
- Internal video support for a built-in 14-inch single-mode monitor.
- Video mode support for Apple 14-inch RGB 640 by 480 monitor, with video-in at 16 bpp (bits per pixel).
- 60-pin connector supports optional video card providing video input for real-time video display, capture, and overlay.
- Video mirror feature allows an external monitor to be connected to the Power Macintosh 5260 using an optional video buffer board connected to the video mirror connector. Apple Computer does not support third-party development of video buffer boards.
- 16 bit/channel stereo sound output and mono sound input, external jack for sound-in, front and rear jacks for stereophonic sound-out, and built-in stereo speakers.
- Optional internal TV tuner expansion board.
- Infrared remote control (included with TV tuner).
- SCSI interface for an internal CD (compact disk) drive.
- External SCSI connector for additional SCSI devices
- Internal 3.5-inch IDE hard disk with 500 MB capacity.
- Internal 1.4 MB Apple Wolfpack floppy disk drive with auto-eject capability (no auto-inject capability).

- Standard Macintosh I/O ports: two serial ports for external modem and printer, sound input and output jacks, and an ADB port for keyboard and mouse.
- 112-pin communications slot connector accepts an optional modem or Ethernet interface. Apple Computer does not support third-party development of boards for the communications slot.
- 114-pin I/O expansion slot connector accepts PDS (processor-direct slot) cards designed for the Macintosh LC series.
- Soft power control from keyboard and remote control.
- Push button control for sound volume and for brightness.
- All-in-one enclosure featuring a tilt-and-swivel monitor and built-in stereo speakers.

Comparison With Power Macintosh 5200 and 6200 Computers

The Power Macintosh 5260 computer is functionally similar to the Power Macintosh 5200 and 6200 computers. Table 1-1 compares the features of these computers.

Table 1-1 Comparing the Power Macintosh 5260 with Power Macintosh 5200 and 6200 computers

Features	Power Macintosh 5260	Power Macintosh 5200 and 6200
Processor type	PowerPC 603e	PowerPC 603
Processor speed	100 MHz	75 MHz
Cache	L1 cache in processor, optional 256 KB L2 cache implemented by a DIMM	256 KB L2 cache
Amount of RAM	8 MB installed in SIMM slot	8 MB installed in SIMM slot
RAM expansion	Up to 64 MB with use of higher capacity SIMMs and the second SIMM slot	Up to 64 MB with use of higher capacity SIMMs and the second SIMM slott
Video RAM	1 MB (DRAM)	1 MB (DRAM)
Video input	Optional card for video input, capture, and overlay	Optional card for video input, capture, and overlay
Video output	Optional mirror connector supports an external monitor operating in mirror mode	Optional mirror connector supports an external monitor operating in mirror mode; Power Macintosh 6200 supports up to 15-inch external monitors at 16 bits per pixel
Sound capabilities	16 bits/channel; mono in, stereo out	8 or 16 bits/channel; mono in, stereo out
Remote control	Built-in IR receiver	Built-in IR receiver

Table 1-1 Comparing the Power Macintosh 5260 with Power Macintosh 5200 and 6200 computers (continued)

Features	Power Macintosh 5260	Power Macintosh 5200 and 6200
Floppy disk drive	One internal	One internal
ADB ports	1	1
Internal hard disk	One (IDE)	One (IDE)
Internal CD-ROM	Optional	Optional
External SCSI ports	One	One
Communications slots	One, for optional modem or Ethernet interface	One, for optional modem or Ethernet interface
Expansion slots	One I/O slot (accepts PDS card for Macintosh LC series)	One I/O slot (accepts PDS card for Macintosh LC series)

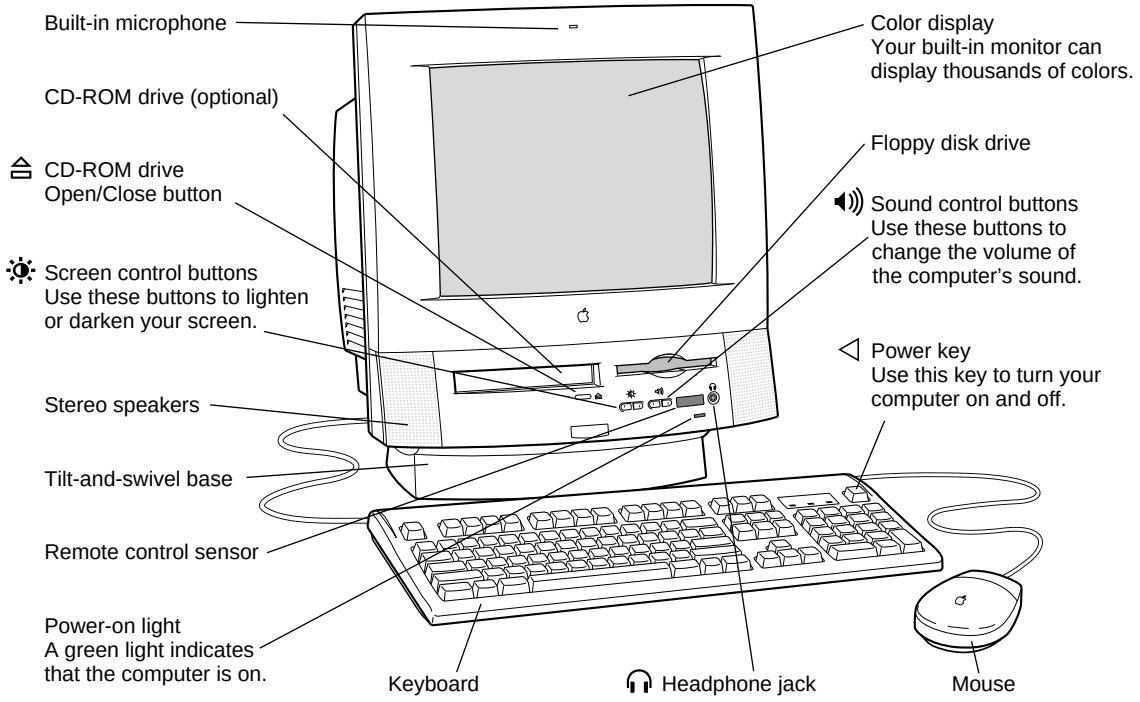
External Features

The Power Macintosh 5260 computer has an integrated design featuring a built-in 14-inch single-mode color display with tilt-and-swivel capabilities, stereo speakers, front panel stereo headphone jack, and front panel push button controls for audio and video.

Front View

Figure 1-1 is a front view of the Power Macintosh 5260 computer. The front view shows the display screen, the built-in microphone and stereo speakers, the openings for the floppy disk and optional CD-ROM drive, the CD-ROM open and close button, the headphone jack, the power-on light, the IR sensor for the remote control, and the push buttons that control the screen intensity (left button) and sound level (right button).

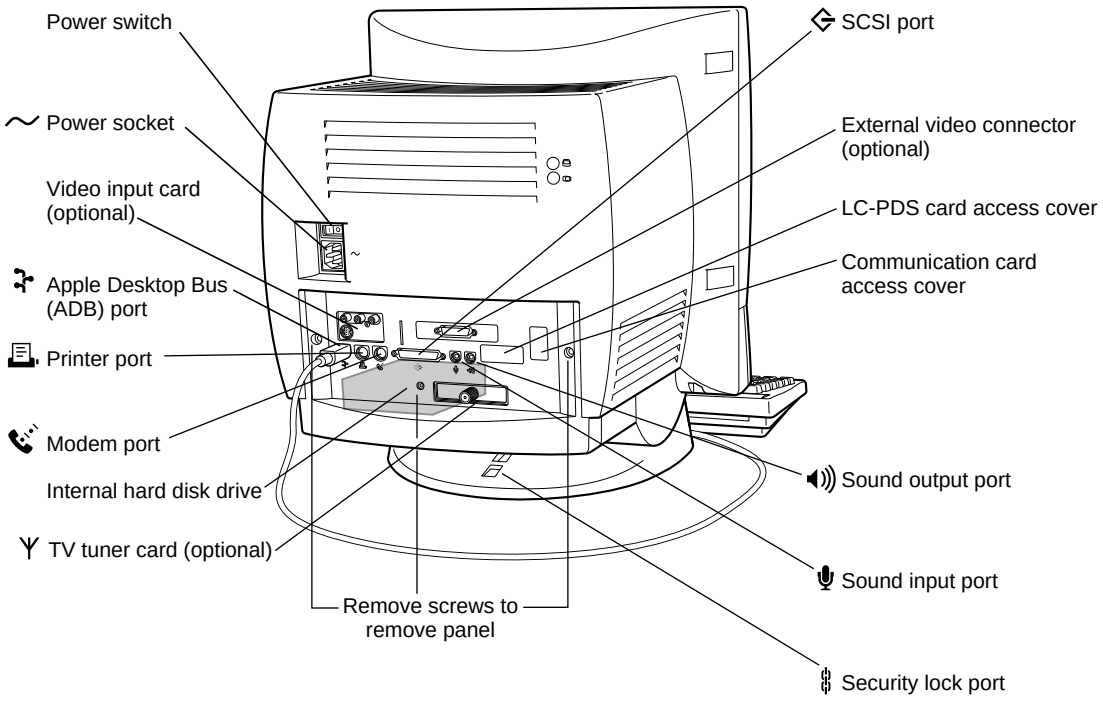
Figure 1-1 Front view of the computer



Back View

Figure 1-2 shows the back panel of the computer. Features on the back panel include the power socket, the reset button, the I/O ports, and openings for I/O access to the expansion modules: the I/O expansion card, the communications card, and the video input card.

Figure 1-2 Back view of the computer



Access to the Logic Board

The logic board can be removed from the case to allow installation of expansion RAM or to plug in an I/O expansion card. To get access to the logic board, you must first remove the back panel of the computer. It is secured by three screws, as shown in Figure 1-2. After removing the screws, you can pull gently on the two latches on the underside of the computer's case and the back panel will slip out. You can then grasp the logic board handle and pull the board straight out of the back of the case.

Power On and Off

You can turn the power on and off by pressing one of two buttons:

- the Power key on the keyboard
- the Power key on the remote control

If files are still open when you attempt to turn off the computer using either one of the Power keys or the Shut Down menu item, the system displays an alert box warning you that files are open and should be closed to avoid loss of data.

Optional Features

Several features of the Power Macintosh 5260 computer are implemented as plug-in modules. These modules are available at the time of purchase or as a later upgrade, and they are designed so that they can be installed by the user.

TV Tuner

The TV tuner module turns the computer into a television receiver, complete with remote control. The features of the TV tuner module are similar to those of the TV tuner in the Macintosh Quadra 630 and LC 630 computers. The TV picture is in its own window on the desktop, and the TV signal is carried in **digital YUV format** for improved picture clarity.

The TV tuner module has the following features:

- the ability to tune 181 broadcast and cable channels (United States version)
- a coaxial connector for TV antenna or cable input (F-type connector in United States and Japanese versions; IEC-type connector in European versions)
- a TV picture in a resizable and movable window
- digital YUV format for improved clarity
- support for closed captioning and teletext
- software password protection
- automatic and manual channel programming
- single remote control for TV and for playback of audio CDs

The TV tuner module is available in versions for NTSC, PAL, and SECAM television systems.

The default size of the window for the TV picture is 320-by-240 pixels. The user can resize the TV window to a maximum of 640-by-480 pixels or to a minimum of 160-by-120 pixels. The resolution of the TV picture does not change as window size changes since the image is changed to fit the area designated by changing the size not the number of the pixels.

The TV tuner module works in conjunction with the video input module, which converts the video data into digital YUV format and stores it in the display buffer.

The TV tuner comes with a remote control device similar to the one used with the Macintosh TV computer. The user can switch channels either by using the remote control or by typing the channel numbers on the keyboard. The user can toggle between the current and previous channel by pressing the Tab key on the keyboard. Each time the channel changes, the computer displays the channel name (assigned by the user) on the picture in the video window.

The user can customize the operation of the TV tuner by adding or removing TV channels that are unused or unwanted. The computer can program the channels automatically, scanning through all available channels and disabling those that do not have a valid signal. When you scan for the next channel using the remote control or the Tab key on the keyboard, the tuner skips the disabled channels.

The software that supports the TV tuner module is an application called Apple Video Player. The application includes password protection for the disabled channels. Parents might use this feature to prevent children from watching undesirable channels.

The software allows you to capture or freeze a single frame of video or record a segment of video as a QuickTime Movie. The TV window cannot be resized while the computer is recording a movie.

Video Input

The video input card accepts video from an external source and displays it in a window on the computer's display. The following are features of the video input card:

- accepts video input in NTSC, PAL, or SECAM format
- connectors for stereo sound, composite video, and S-video (Y/C)
- video display in a 320-by-240-pixel window
- pixel doubling for 640-by-480-pixel maximum display
- video overlay capability
- YUV format for digital video input
- a digital video connector for adding a video processor on an expansion card

The video input card provides AV features similar to those of the Macintosh Quadra 660AV, with one important improvement. Whereas the Macintosh Quadra 660AV digitizes color video using a 16-bit RGB format, the video input card uses a digital YUV format. Because a standard television signal has more information in its **chrominance** channel than in its **luminance** channels, digitizing the video signal as YUV format results in a clearer picture.

The video input card can accept video input from either an external device such as a VCR or camcorder or from the internal TV tuner module. The external device can be connected to the video input card either through the composite video connector or the S-video connector.

The default window size is 320-by-240 pixels; you can resize the window up to 640-by-480 pixels—the full screen on a 14-inch monitor. The large image uses pixel doubling of the 320-by-240 pixel image.

The video input card plugs into a dedicated slot on the main logic board. The slot connector is a 60-pin microchannel connector. The module fits only its proper slot and only in the proper orientation so that you can safely install the video input card.

The video input card has a separate connector called the DVA (digital video application) connector. The DVA connector makes the digitized video data available to a card in the I/O expansion slot. Such a card can contain a hardware video compressor or other video processor. For more information, see the section "DVA Connector" beginning on page 54.

Video Display Mirror Out

The Power Macintosh 5260 computer supports a feature called video display mirror output that allows a second monitor to be connected to the computer through a video buffer card. The video buffer card plugs into a 22-pin connector on the computer's main logic board.

In the video display mirror-out mode, the image on the second monitor's screen is the same as that on the screen of the built-in monitor. This mode of operation is appropriate for presentations, allowing the audience and the presenter to see the same images.

IMPORTANT
Apple Computer does not provide support for third-party developers of video buffer cards. ▲

Communications Slot

The main logic board in the Power Macintosh 5260 computer has a communications slot that is compatible with the communications slot first introduced in the Macintosh LC 575 computer and later used in the Macintosh LC 630 and Macintosh Quadra 630 computers. The slot allows the computer to support a communications module without occupying the expansion slot. A communications module can be installed by either the user or the dealer.

The communications slot in the Power Macintosh 5260 computer supports all communications cards developed for the Macintosh LC 575, Macintosh LC 630, and Macintosh Quadra 630, including

- the 10BaseT (twisted pair) Ethernet card
- the 10Base2 (thin coax) Ethernet card
- the AAUI (Apple standard) Ethernet card
- the 14.4 fax/data modem card

IMPORTANT
Apple Computer does not provide support for third-party developers of communications cards. ▲

Compatibility Issues

The Power Macintosh 5260 computer is different from earlier desktop models in several ways. This section describes key issues you should be aware of to ensure that your hardware and software work properly with this new model. Some of the topics described here are covered in more detail in later parts of this developer note.

Microprocessor Differences

Certain differences between the PowerPC 603e and the PowerPC 601 microprocessors can affect the way code is executed. Because of these differences, programs that execute correctly on the PowerPC 601 microprocessor may cause compatibility and performance problems on the PowerPC 603e microprocessor. Critical differences are described in the next four sections.

POWER-Clean Code

First generation Power Macintosh computers used the PowerPC 601 microprocessor, which bridged the new PowerPC architecture with the POWER architecture from which it descended. The PowerPC 601 implemented most of the old POWER instruction set as well as the newer PowerPC instruction set.

Later versions of the microprocessor, namely the PowerPC 603, 603e, and 604, implement only the PowerPC instruction set, hence the term “power clean.” Because of the differences in instruction set implementation, there may be incompatibility and poor performance, particularly in the area of compilers.

Newer compilers, designed for the PowerPC instruction set, do not generate the old POWER instructions. However, compilers designed for the POWER instruction set are also being used to compile programs for the PowerPC instruction set. Most of those compilers have the option to suppress the generation of offending instructions. For example, the IBM xLc C compiler and the xLCC++ compiler have the option `-garch=ppc`. Developers using these compilers should verify that the options are in effect for all parts of their code. To be on the safe side, you should contact your compiler vendor to make sure that the compiler you are using does not generate POWER instructions.

Completion Serialized Instructions

Several types of instructions can interfere with instruction pipelining and degrade system performance. Most noticeable are completion serialized instructions such as load and store string and load and store multiple. These instructions are referred to as

completion serialized instructions because they cannot be executed until all prior instructions have been completed.

Representatives of Apple Computer, Inc. are working with compiler developers to establish guidelines for the appropriate use of these instructions.

Split Cache

Unlike the PowerPC 601, which has a unified cache, the PowerPC 603e has separate caches for instructions and data. This can lead to cache coherency problems in applications that mix code and data.

In the Mac OS, the Code Fragment Manager loads almost all native code to ensure that the code is suitable for execution. If your code is loaded by the Code Fragment Manager, you do not have to worry about cache coherency.

If, however, your application generates code in memory for execution, problems can arise. Examples include compilers that generate code for immediate execution and interpreters that translate code in memory for execution. For such cases, you can use the `MakeDataExecutable` function to notify the Mac OS that data is subject to execution. This call is defined in OSUtils.h.

Data Alignment

In PowerPC systems, data is normally aligned on 32-bit boundaries, whereas data for the 680x0 is typically aligned on 16-bit boundaries. Even though the PowerPC microprocessor was designed to support the 680x0 type of data alignment, misaligned data can cause some deterioration in performance. Furthermore, performance with misaligned data varies across the different implementations of the PowerPC microprocessor.

While it is essential to use 16-bit alignment for data that is being shared with 680x0 code, you should use PowerPC alignment for all other kinds of data. In particular, you should not use global 680x0 alignment when compiling your PowerPC applications; instead use **alignment pragmas** to turn on 680x0 alignment only when necessary.

Expansion Slot

The I/O expansion slot in the Power Macintosh 5260 computer is compatible with the PDS (processor-direct slot) in the Macintosh LC family of computers, but it is not a true PDS. Like the expansion slot in the Macintosh LC 630 and Macintosh Quadra 630 computers, the I/O expansion slot in the Power Macintosh 5260 computer supports many PDS cards designed to operate with the MC68030 bus, including both bus masters, such as Apple Computer's Ethernet expansion card, and bus slaves, such as display cards.

While the I/O expansion slot accepts PDS cards designed for the Macintosh LC family of computers, some of those cards do not work. Cards that are incompatible with the I/O expansion slot include the following:

- Cards such as accelerators, 68882 FPU cards, and cache cards that are designed to work as coprocessors with an MC68020 or an MC68030, or to replace these

microprocessors. This type of card will not work because the microprocessor is different and because the slot signals are not connected directly to the microprocessor.

- Cards with drivers that include incompatible code. Some drivers that do not follow Apple Computer's programming guidelines will not work on machines that use the PowerPC 603e microprocessor. For example, some of those drivers write directly to the cache control register in an MC68030. Such code will not work on a PowerPC 603e microprocessor.
- Cards with drivers that include code to check the `gestaltMachineType` value and refuse to run on a newer CPU. However, this is an advantage, since such cards refuse to run on machines on which they have not been tested.

IDE Hard Disk

The internal hard disk in the Power Macintosh 5260 computer is an IDE drive, not a SCSI drive. This could cause compatibility problems for hard disk utility programs.

CHAPTER 2

Architecture

This chapter describes the architecture of the Power Macintosh 5260 computer, including the major components of the main logic board, such as the microprocessor, custom ICs, and the display RAM. The chapter also provides a simplified address map.

Block Diagram and Main ICs

The architecture of the Power Macintosh 5260 computer is based on three generations of microprocessors: the MC68020/030, the MC68040, and the new PowerPC 603e. Figure 2-1 shows the system block diagram.

The internal bus structure consists of three internal buses: the 64-bit wide 603e data bus, the 32-bit wide 68040 bus, and the 32-bit wide I/O bus. The 603e bus is connected directly to the main processor and runs at the same clock rate. An optional external 256 KB L2 cache and 4 MB of ROM attach directly to the 603e data bus and help to optimize system performance.

The Capella custom IC provides the bus translation logic that bridges the 603e processor and the 68040-based custom ICs. It translates the 64-bit data from the 603e data bus into 32-bit data required by the 68040 bus and provides the necessary signals to maintain 68040 protocol. Three custom ICs, F108, Valkyrie, and PrimeTime III, connect directly to the 68040 bus. The F108 custom IC provides memory control and bus arbitration logic, the Valkyrie custom IC is the graphics display controller, and the PrimeTime III custom IC controls most of the I/O functions.

The I/O bus is a 32-bit wide buffered bus that runs at 16 MHz and supports 68030 byte steering and dynamic bus sizing. Although it is derived from the 68040 bus, it behaves more like the 68030 interface to support the 68020 and 68030 expansion cards that were designed for use in the Macintosh LC family of computers. The PrimeTime III custom IC buffers the data portion of the I/O bus and provides a compatible interface for I/O devices and software designed for use with the MC68030 miroprocessor.

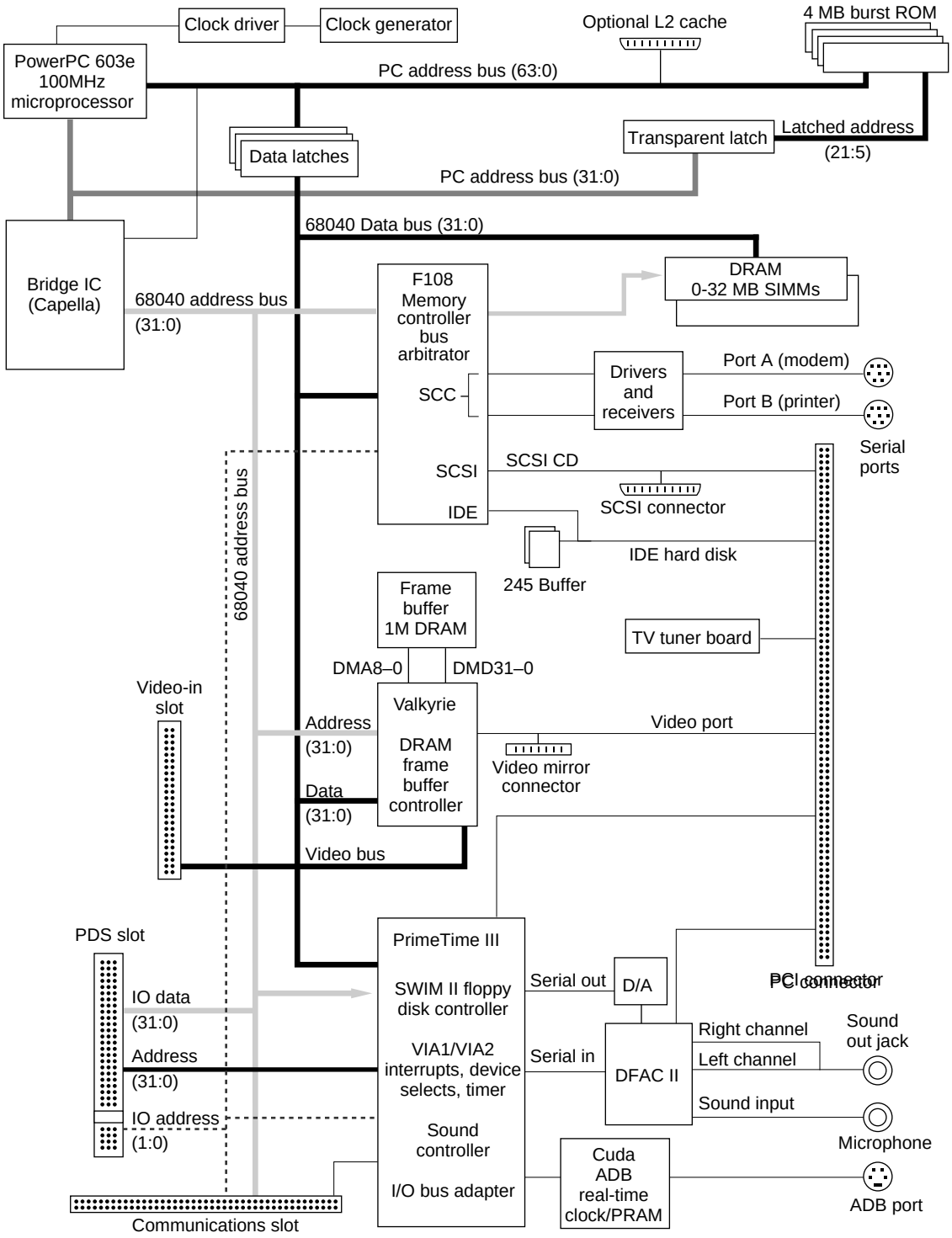
Microprocessor

The Power Macintosh 5260 computer uses a PowerPC 603e microprocessor running at 100 MHz. The principle features of the PowerPC 603e microprocessor include

- full RISC processing architecture
- parallel processing units: two integer and one floating-point
- 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)
- dual 16 KB on-chip caches (16 KB each for data and instructions)

For complete technical details, see the Motorola/IBM *PowerPC 603e RISC Microprocessor User's Manual.*

Figure 2-1 System block diagram



Second-Level Cache and ROM

The memory subsystem of the Power Macintosh 5260 computer consists of a 4 MB ROM and a 256 KB second-level (L2) cache, in addition to the internal cache memory of the PowerPC 603e microprocessor. The L2 cache is contained on the 160-pin cache DIMM card that plugs into the main logic board. The Capella custom IC provides burst mode control to the cache and ROM.

IMPORTANT

Apple Computer does not provide support for third-party developers of L2 cache cards. ▲

System RAM

The Power Macintosh 5260 computer uses two 72-pin SIMM sockets for memory expansion. The computer is shipped with 8 MB installed in one of the SIMM sockets and can be expanded to 64 MB. No soldered on-board memory is provided. The SIMM sockets support both single- and double-sided DRAM modules. Using 4 Mbit devices, each SIMM can be configured as 4 MB or 8 MB providing RAM expansion up to 16 MB. Using 16 Mbit devices, you can configure each SIMM as 16 MB or 32 MB providing RAM expansion up to 64 MB. The F108 custom IC provides memory control for the system RAM. Refer to “DRAM Expansion” on page 40 for further information.

Custom ICs

The architecture of the Power Macintosh 5260 computer is designed around six large custom integrated circuits:

- the Capella bus translation IC
- the F108 memory controller and I/O support IC
- the PrimeTime III I/O subsystem and buffer
- the DFAC II sound input processor
- the Cuda ADB controller
- the Valkyrie video CLUT and DAC

The computer also uses several standard ICs that are used in other Macintosh computers. This section describes only the custom ICs.

Capella IC

The Capella IC functions as the bridge between the PowerPC 603e microprocessor and the 68040 based circuits. It translates 64-bit data from the 603e data bus into 32-bit data for the 68040 bus and synchronizes arbitration between the two buses. The Capella IC supports all combinations of 603e bus translation sizes and data alignment and provides appropriate control signals for the 68040 bus. In addition, the Capella IC includes registers that control the system ROM and the optional L2 cache.

F108 IC

The F108 IC performs the system memory control functions. It also includes circuitry equivalent to SCC and SCSI controller ICs. The functional blocks in the F108 include the following:

- control logic for the system ROM and DRAM
- SCSI controller
- SCC serial I/O controller
- IDE hard disk interface controller

The F108 IC is attached to the system 68040 bus and provides the control and timing signals for the system ROM and RAM. The memory control logic supports byte, word, longword, and line accesses to the system memory. If an access is not aligned to the appropriate address boundary, that access requires multiple data transfers on the bus.

Note
The memory control logic in the F108 IC is the same as that in the F108 IC used in the Power Macintosh, Quadra 630, and LC 630. ◆

The SCSI controller in the F108 IC is just like an NCR 53C96. The PrimeTime III IC provides the interface to the SCSI controller and also provides longword accumulation of SCSI data for better performance. In the Power Macintosh 5260 computer, the clock signal to the SCSI controller is 20 MHz.

The SCC circuitry in the F108 IC is an 8-bit device just like the 8530 SCC. The PCLK signal to the SCC is an 8 MHz clock.

PrimeTime III IC

The PrimeTime III IC supports the I/O bus and functions as the bridge between the 68040 bus and the I/O bus. It combines functions performed by several ICs in previous Macintosh designs. The PrimeTime III IC includes

- data bus buffers for the internal I/O bus
- address decoding for I/O devices
- dynamic bus sizing and data routing for the I/O bus
- interface adapters VIA1 and VIA2
- interrupt controls
- a SWIM II floppy disk controller
- sound control logic and buffers

The PrimeTime III IC provides data bus features such as byte steering**,** which allows 8-bit and 16-bit devices to be connected to a fixed **byte lane**, and dynamic bus sizing**,** which allows software to read and write longwords to 8-bit and 16-bit devices. These features allow the computer to work with existing I/O software designed for the MC68030 and MC68040.

The PrimeTime III IC also contains the sound control logic and the sound input and output buffers. There are three separate buffers—one for sound input and two for stereo sound output. This means the computer can record sound input and process sound output at the same time.

DFAC II IC

The DFAC II custom IC is a digital filter audio chip that contains the sound input processing devices. The DFAC II includes

- input AGC (automatic gain control) comparators
- anti-alias filtering
- an A/D (analog-to-digital) converter for input
- a PWM (pulse-width modulation) converter for output

The DFAC II IC does not include the sound control logic and the input and output buffers; these are part of the PrimeTime III IC.

For sound input, the DFAC II processes the signal from the sound input jack through a sound input amplifier with automatic gain control, an input filter, an A/D converter, and the necessary switching circuits. The DFAC II sends the resulting stream of digital sound data to the PrimeTime III, which stores the data in its input buffer.

For sound output, circuits in the DFAC II take data from the sound output buffers and generate stereo PWM signals. The DFAC II merges the sound playthrough signal with the PWM signals and sends the combined signals to an external stereo PWM converter IC. After low-pass filtering in the PWM converter, the signals go to the sound output jacks and to a separate amplifier for the built-in speaker. See the section “Sound” beginning on page 32 for further information.

Cuda IC

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

- the ADB interface
- parameter RAM
- the real-time clock
- program control of the power supply (soft power)
- the programming interface to devices on the IIC (inter-integrated circuit) bus

The devices on the IIC bus include the DFAC II sound IC, the digital video decoder and scaler (DESC) on the video input module, and the Cyclops IC, which is the controller for the remote control receiver. The computer reads and writes status and control information to these devices by commands to the Cuda IC.

Valkyrie IC

The Valkyrie IC is a custom IC containing the logic for the video display. It includes the following functions:

- display memory controller
- video CLUT (color lookup table)
- video DAC (digital-to-analog converter)

A separate data bus handles data transfers between the Valkyrie IC and the display memory. The display memory data bus is 32 bits wide, and all data is transfered 32 bits at a time. The Valkyrie IC breaks each 32-bit data transfer into several pixels of the appropriate size for the current display mode—4, 8, or 16 bits per pixel. The Valkyrie IC does not support 24 bits per pixel.

To keep up with the large amount of data that must be transferred into and out of the display memory, the Valkyrie IC has several internal buffers. Besides input and output buffers for display data, the Valkyrie IC also has a buffer for both addresses and data being sent from the main processor to the display. That buffer can hold up to four transactions, allowing the main processor to complete a write instruction to the display memory and continue processing without waiting for some other transaction that might be taking place on the display memory bus.

The CLUT in the Valkyrie custom IC provides color palettes for 4-bit and 8-bit display modes. In 16-bit display mode, the CLUT is used to provide **gamma correction** for the stored color values. With a black-and-white or monochrome display mode, all three color components (R, G, and B) are the same.

The Valkyrie IC uses several clocks. Its transactions with the CPU are synchronized to the CPU_BCLK signal. Data transfers from the frame-buffer DRAM are clocked by the MEM_CLK signal, which runs at 60 MHz. Data transfers to the CLUT and the video output are clocked by the dot clock, which has a different rate for different display monitors.

For more information about the interaction between the Valkyrie IC, the display memory, and the main processor, see the section “Display RAM” later in this chapter.

Bus Arbitration

The system bus can support two bus masters, including the main processor and one I/O bus master. The I/O master has higher priority. Either bus master can park on the bus as long as no higher priority master requests the bus.

The bus request from the I/O bus master is initiated by the PrimeTime III IC and comes from one of two sources: the PDS expansion card or the communications card. Devices on these cards are not connected directly to the system bus; they arbitrate the bus by way of the I/O bus and the PrimeTime III IC. See the section “Bus Master on a Card” beginning on page 52.

The Capella IC synchronizes the bus arbitration between the 603e bus and the 68040 bus. The PowerPC 603e microprocessor is the default bus master. The Capella IC gains access to the 68040 bus when the I/O master is granted to the bus. The PowerPC 603e microprocessor continues to execute instructions from the ROM and the L2 cache during bus arbitration, thus improving system performance.

Display RAM

The display memory in the Power Macintosh 5260 computer is separate from the main memory. To reduce the cost of the computer, the display memory is implemented with DRAM devices instead of more expensive VRAM devices. The display memory consists of 1 MB of 60 ns DRAM devices configured to make a 32-bit data bus. The display memory cannot be expanded.

The display memory contains three separate frame buffers. The first frame buffer holds the graphics data—the display that is generated by the computer. The other two frame buffers hold video data from the video input module. The video data frame buffers are used alternately: while one is supplying data to be sent to the video monitor, the other is receiving the next frame of video input.

The display data generated by the computer can have pixel depths of 4, 8, or 16 bits for monitors up to 64-by-480 pixels. Data from the video input module is always stored and transferred at 16 bits per pixel. The video frame buffers support live video in a 320-by-240-pixel frame at 30 frames per second.

The Power Macintosh 5260 computer can display video in a window inside the computer graphics display. The Valkyrie IC has registers that contain the starting location of the video window within the display, the starting address of the video data in the video buffer, and the size of the video window.

Address Map

The Power Macintosh 5260 computer supports only 32-bit addressing. Figure 2-2 on page 22 shows a simplified address map. Address mapping changes after the M7 bit in the Capella bus translation unit has been set.

Note
You should not use actual hardware addresses in applications; you should always communicate with hardware devices by means of system software. ◆

Note
Because the Power Macintosh 5260 computer operates only in 32-bit addressing mode, it does not support the Apple IIe Card for the Macintosh LC. ◆

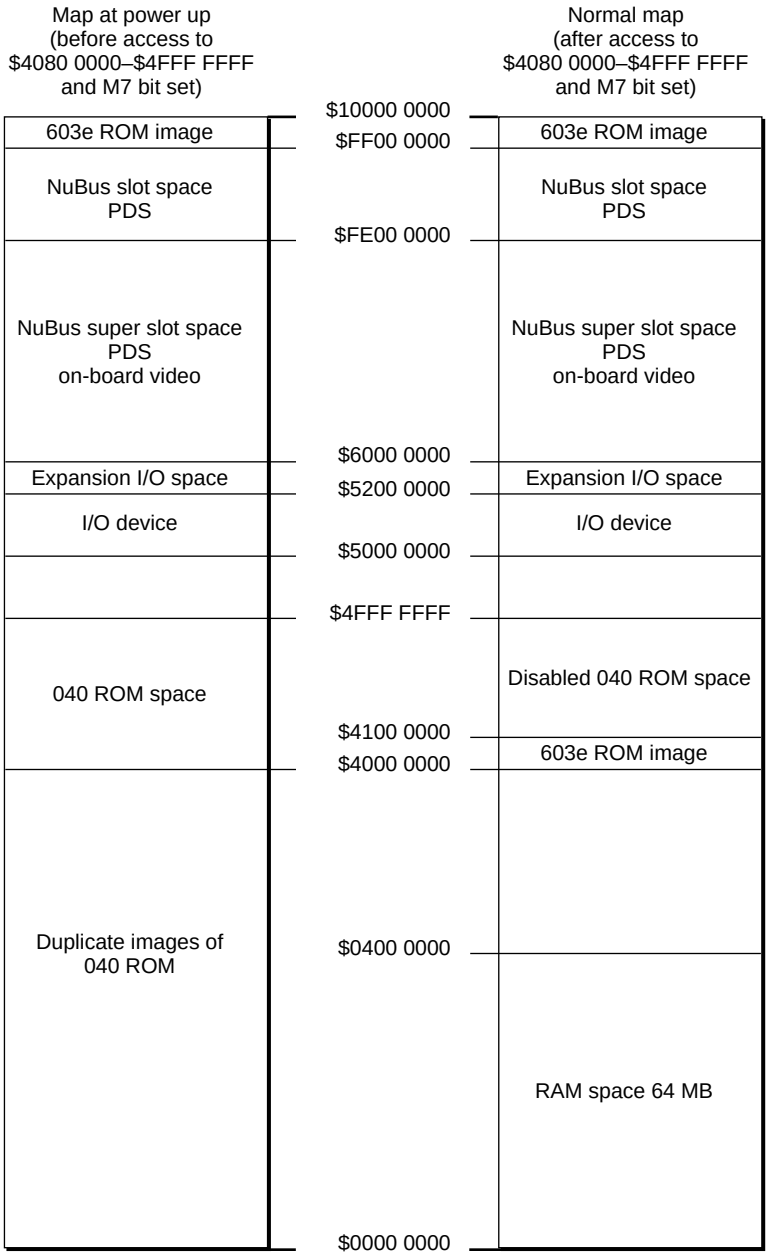
RAM Addresses

The first gigabyte of address space is reserved for RAM. The actual amount of RAM installed can be from 8 MB to 64 MB. At startup time, a routine in the ROM determines the amount of RAM available and stores the size in a low-memory global variable.

Addresses for PDS Expansion Cards

The PDS expansion card uses address space from $FE00 0000 to $FF00 0000, corresponding to NuBus™ slot $E, and from $E000 0000 to $EFFF FFFF, corresponding to NuBus Super Slot $E. For more information, see the section “Card Address Space” on page 53.

Figure 2-2 Simplified address map



CHAPTER 3

I/O Features

This chapter describes both the built-in I/O devices and the interfaces for external I/O devices. It also describes the external video monitor that can be used with the Power Macintosh 5260 computer.

Serial I/O Ports

The Power Macintosh 5260 computer has two serial ports, one for a printer and one for a modem. Both serial ports have 9-pin mini-DIN sockets that accept either 8-pin or 9-pin plugs. Figure 3-1 shows the mechanical arrangement of the pins on the serial port sockets. Table 3-1 shows the signal assignments.

Figure 3-1 Serial port sockets

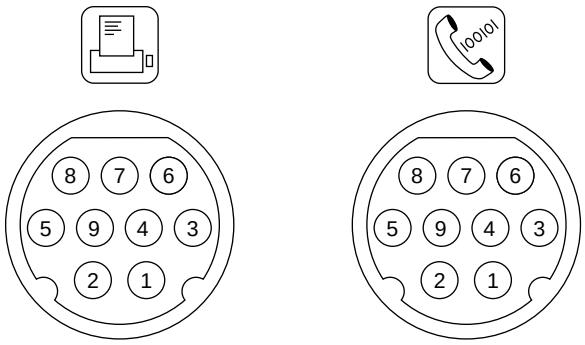


Table 3-1 Serial port signals

Pin number	Signal name	Signal description
1	HSKo	Handshake output
2	HSKi	Handshake input
3	/TXD	Transmit data –
4	GND	Signal ground
5	/RXD	Receive data –
6	TXD	Transmit data +
7	GPi	General-purpose input
8	RXD	Receive data +
9	+5VDC	+5 volts

Pin 9 on each serial connector provides +5 V power from the ADB power supply.

IMPORTANT

An external device should draw no more than 100 mA from pin 9. The total current available for all devices connected to the +5 V supply for the ADB and the serial ports is 300 mA. Excessive current drain will cause a fuse to interrupt the +5 V supply. The fuse 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 F108 custom IC, described in Chapter 2. On serial port A (the modem port), the GPi line can be connected to the receive/transmit clock (RTxCA) signal on the SCC. That connection supports devices that provide separate transmit and receive data clocks, such as synchronous modems. For more information about the serial ports, see *Guide to the Macintosh Family Hardware,* second edition.

ADB Port

The Apple Desktop Bus (ADB) port on the Power Macintosh 5260 computer is functionally the same as on other Macintosh computers.

The ADB is a single-master, multiple-slave, serial communications bus that uses an asynchronous protocol and connects keyboards, graphics tablets, mouse devices, and other devices to the computer. The custom ADB microcontroller drives the bus and reads status from the selected external device. A 4-pin mini-DIN connector connects the ADB to the external devices. Table 3-2 lists the ADB connector pin assignments. For more information about the ADB, see *Guide to the Macintosh Family Hardware,* second edition.

Table 3-2 ADB connector pin assignments

Pin number	Name	Signal description
1	ADB	Bidirectional data bus used for input and output. It is an open-collector signal pulled up to +5 volts through a 470-ohm resistor on the main logic board.
2	PFW	Power-on signal that generates reset and interrupt key combinations.
3	+5 V	+5 volts from the computer.
4	GND	Ground from the computer.

IMPORTANT

The total current available for all devices connected to the +5V pins on the ADB and the modem port is 300 mA. Each device should use no more than 100 mA. ▲

Floppy Disk Drive

The Power Macintosh 5260 computer has one internal high-density floppy disk drive (Apple SuperDrive). The drive is connected to a 20-pin connector on a cable that is connected to the main logic board by the internal chassis connector. Table 3-3 shows the pin assignments on the floppy disk connector. A slash (/) before a signal name indicates an active-low signal.

Table 3-3 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
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

IDE Hard Disk

The Power Macintosh 5260 computer has an internal hard disk that uses the standard IDE interface. This interface, used for IDE drives on IBM AT–compatible computers, is also referred to as the ATA interface. The implementation of the ATA interface on the Power Macintosh 5260 computer is a subset of the ATA interface specification, ANSI proposal X3T9.2/90-143, Revision 3.1.

Hard Disk Specifications

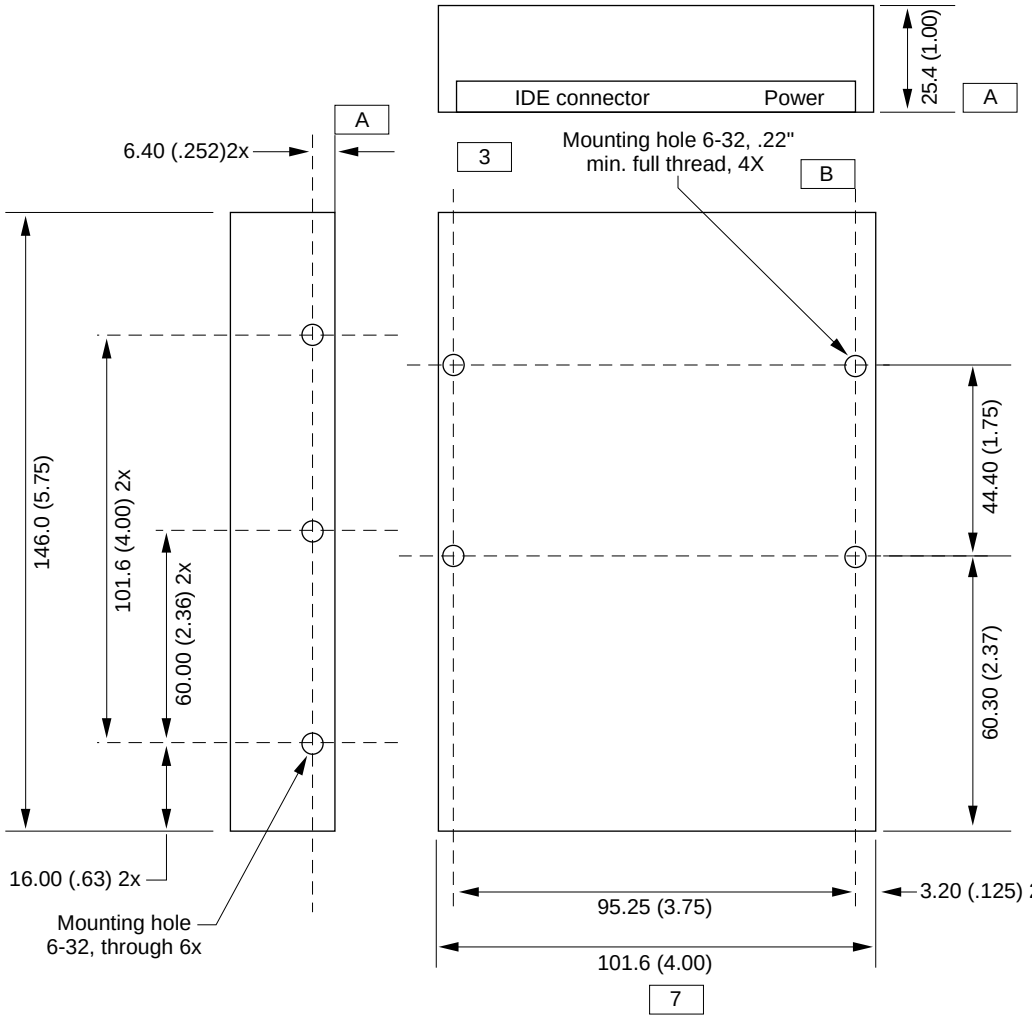
Figure 3-2 on page 28 shows the maximum dimensions of the hard disk and the location of the mounting holes. As the figure shows, the minimum clearance between conductive components and the bottom of the mounting envelope is 0.5 mm.

Hard Disk Connectors

The internal hard disk has a standard 40-pin IDE connector and a separate 4-pin power connector. The 40-pin connector cable is part of the cable harness attached to the main logic board by the internal chassis connector. The power cable is attached directly to the power supply.

The exact locations of the IDE connector and the power connector are not specified, but the relative positions must be as shown in Figure 3-2 on page 28 so that the cables and connectors will fit.

Figure 3-2 Maximum dimensions of the hard disk



Notes:

1 A Defined by plane of bottom mount holes

2 B Defined by center line of bottom mount holes

3 40-pin IDE and 4-pin power connector placement must not be reversed

4 All dimensions in MM (inch)

5 Drawing not to scale

6 Tolerances .X = ±0.50, .XX = ±0.25

7 Dimension to be measured at center line of side-mount holes

8 Minimum 0.5 MM clearance from any conductive PCB components to A

Pin Assignments and Signal Descriptions

Table 3-4 shows the pin assignments on the 40-pin IDE hard disk connector. A slash (/) before a signal name indicates an active-low signal. Table 3-5 on page 30 describes the functions of the signals.

Table 3-4 Pin assignments on the IDE hard disk connector

Pin number	Signal name	Pin number	Signal name
1	/RESET	2	GROUND
3	DD7	4	DD8
5	DD6	6	DD9
7	DD5	8	DD10
9	DD4	10	DD11
11	DD3	12	DD12
13	DD2	14	DD13
15	DD1	16	DD14
17	DD0	18	DD15
19	GROUND	20	KEY
21	Reserved	22	GROUND
23	DIOW	24	GROUND
25	DIOR	26	GROUND
27	/IORDY	28	Reserved
29	Reserved	30	GROUND
31	INTRQ	32	/IOCS16
33	DA1	34	/PDIAG
35	DA0	36	DA2
37	/CS0	38	/CS1
39	/DASP	40	GROUND

Note
The IDE data bus is connected to the I/O bus through bidirectional bus buffers. To match the big-endian format of the MC68030-compatible bus, the bytes are swapped. The lower byte of the IDE data bus, DD(0–7), is connected to the high byte of the upper word of the I/O bus, IOD(24–31). The higher byte of the IDE data bus, DD(8–15), is connected to the low byte of the upper word of the I/O bus, IOD(16–23). ◆

Table 3-5 describes the signals on the IDE hard disk connector. A slash (/) before a signal name indicates an active-low signal.

Table 3-5 Signals on the IDE hard disk connector

Signal name	Signal description
DA(0–2)	IDE device address; used by the computer to select one of the registers in the IDE drive. For more information, see the descriptions of the CS0 and CS1 signals.
DD(0–15)	IDE data bus; buffered from IOD(16–31) of the computer's I/O bus. DD(0–15) are used to transfer 16-bit data to and from the drive buffer. DD(8–15) are used to transfer data to and from the internal registers of the drive, with DD(0–7) driven high when writing.
/CS0	IDE register select signal. It is asserted (driven low) to select the additional control and status registers on the IDE drive.
/CS1	IDE register select signal. It is asserted (driven low) to select the main task file registers. The task file registers indicate the command, the sector address, and the sector count.
/IORDY	IDE I/O ready; when driven low by the drive, signals the CPU to insert wait states into the I/O read or write cycles.
/IOCS16	IDE I/O channel select; asserted low for an access to the data port. The computer uses this signal to indicate a 16-bit data transfer.
/DIOR	IDE I/O data read strobe.
/DIOW	IDE I/O data write strobe.
INTRQ	IDE interrupt request. This active high signal is used to inform the computer that a data transfer is requested or that a command has terminated.
/RESET	Hardware reset to the drive; an active low signal.
Key	This pin is the key for the connector.

SCSI Bus

The Power Macintosh 5260 computer has a SCSI bus for an optional internal CD-ROM device and one or more external SCSI devices. The CD-ROM device receives power directly from the power supply.

SCSI Connectors

The SCSI connector for the internal CD-ROM drive is a 50-pin connector with the standard SCSI pin assignments. It attaches to a cable that is connected to the main logic board by the internal chassis connector. The external SCSI connector is a 25-pin D-type connector with the same pin assignments as other Apple SCSI devices. Table 3-6 shows the pin assignments on the internal and external SCSI connectors. A slash (/) before a signal name indicates an active-low signal.

Table 3-6 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

Table 3-6 Pin assignments for the SCSI connectors (continued)

Pin number (internal 50-pin)	Pin number (external 25-pin)	Signal name	Signal description
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

SCSI Bus Termination

The internal end of the SCSI bus is terminated by a 220/330 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

The sound system supports 16-bit stereo sound output and monaural sound input. Like other Macintosh computers, the Power Macintosh 5260 computer can create sounds digitally and play the sounds through its internal speakers or send the sound signals out through the sound output jacks. It can also record sound from several sources: the built-in microphone, a microphone connected to the sound input jack, the video input module, or a compact disc in the CD-ROM player.

Sound Output

The Power Macintosh 5260 computer has two built-in speakers and two sound output jacks, one on the front and one on the back. Both output jacks are connected to the sound amplifier. The jack on the front provides easy access for headphones. If you insert a plug into either jack, the internal speakers are disconnected.

Sound output is controlled by the PrimeTime III IC. A 16-bit data stream from the PrimeTime III IC is converted to an analog signal and level-shifted before entering the DFAC II IC for further filtering and mixing. The DFAC II IC provides the stereo sound output to both the internal speakers and the sound output jacks.

Sound Input

The Power Macintosh 5260 computer has a sound input jack on the back for connecting an external microphone or other sound source. The sound input jack accepts a standard 1/8-inch phone plug, either monophonic or stereophonic (two signals plus ground).

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 internal circuitry mixes the stereophonic signals into a monophonic signal for digitization.

IMPORTANT
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. ▲

IMPORTANT
The microphone for the Macintosh LC and LC II does not work with the Power Macintosh 5260 computer. It requires the line-level signal provided by the Apple PlainTalk microphone. ▲

Sound Input Specifications

The sound input jack has the following electrical characteristics:

- Input impedance: 100k ohms
- Average line level: 100 mV RMS
- Average microphone level: 70 mV RMS
- Maximum input level: 1.8 V RMS

Routing of the Sound Signals

Sound input signals can be routed in two ways: they can be recorded (digitized) or they can be sent directly to the sound outputs and speakers, bypassing the sound IC. Table 3-7 lists the sound sources and shows how each one can be routed.

Table 3-7 Sound sources and routing

Sound source	Record	Bypass
Sound input jack	√	–
Modem	√	–
CD-ROM player	√	√
Video sound (video input module)	√	√

Digitizing Sound

The Power Macintosh 5260 computer digitizes and records sound as 16-bit samples. It can use either of two sampling rates: 11k samples per second or 22k samples per second. The sound circuits include input and output filters with switchable cutoff (–3 dB) frequencies that correspond to the two sampling rates: 3.5 kHz cutoff for the 11k sampling rate or 7 kHz cutoff for the 22k sampling rate.

The sound system always plays samples at the 22k sampling rate; when playing samples recorded at the 11k sampling rate, the software writes each sample to the sound buffer twice.

Sound Modes

The sound mode is selected by a call to the Sound Manager. The sound circuitry normally operates in one of three modes.

- Sound playback: computer-generated sound is sent to the speaker and the sound output jacks.
- Sound playback with playthrough: computer sound and sound input are mixed and sent to the speaker and the sound output jacks.
- Sound record with playthrough: input sound is recorded and also fed through to the speaker and the sound output jacks.

When recording from a microphone, applications should reduce the playthrough volume to prevent possible feedback from the speaker to the microphone.

The PrimeTime III IC provides separate sound buffers for input and for stereo output, so the computer can record and send digitized sound to the sound outputs simultaneously.

Keyboard

The keyboard has a Power key, identified by the symbol ⏻. When the you choose Shut Down from the Special menu, the computer either shuts down or a dialog appears asking if you really want to shut down. You can also turn off the power by pressing the Power key .

There are no programmer's switches, so you invoke the reset and NMI (nonmaskable interrupt) functions by pressing Command key combinations while holding down the Power key, as shown in Table 3-8. The Command key is identified by the symbols and ⌘.

Note
You must hold down a key combination for at least 1 second to allow the ADB microcontroller enough time to respond to the NMI or hard-reset signal. ◆

Table 3-8 Reset and NMI key combinations

Key combination	Function
Command-Power (⌘-◁)	NMI (always active)
Control-Command-Power (Control-⌘-◁)	Reset

Note
The NMI function can always be activated from the keyboard. This is a change from the Macintosh LC computer, where keyboard activation of the NMI function can be disabled by the software. ◆

Video

The Power Macintosh 5260 computer has a built-in 14-inch multiscan monitor. It supports a pixel display size of 640 by 480, which is set when power is applied.

Optional Video Display Mirror Output Feature

The Power Macintosh 5260 computer uses a feature, called video display mirror output, to make the video information on its built-in monitor available to an external monitor. This means that the information displayed on an external monitor is a mirror image of that displayed on the built-in monitor. This feature is implemented by plugging an optional video buffer board into the 22-pin Video Mirror connector on the main logic board. The Video Mirror connector's pin assignments are shown in Table 3-9.

The optional video buffer board includes a ribbon cable with a DB-15 connector. This connector attaches to a large opening in the upper part of the computer's back panel, identified in Figure 1-2 on page 6, as the Monitor Out port. The cable from an external video monitor plugs into this DB-15 connector to allow the external monitor to display a mirror image of the video on the built-in monitor.

Table 3-9 Video mirror connector pin assignments

Pin	Signal name	Signal description
1	VID GND	Video ground
2	RED	Red signal
3	GREEN	Green signal
4	VID GND	Video ground
5	VID GND	Video ground
6	BLUE	Blue signal
7	CSYNC	C sync

Table 3-9 Video mirror connector pin assignments (continued)

Pin	Signal name	Signal description
8	VSYNC	Vertical sync
9	MLB.SYNC.EN.L	Not used (reserved)
10	HSYNC	Horizontal sync
11	DAC.ISET.1	Not used (reserved)
12	DAC.ISET.2	Not used (reserved)
13	SND GND	Not used (reserved)
14	SND RIGHT	Not used (reserved)
15	SND LEFT	Not used (reserved)
16	+5V	+ 5 volts
17	GND	Ground
18	SDAT	Not used (reserved)
19	SCLK	Not used (reserved)
20	+12V	+12 volts
21	–12V	–12 volts
22	Dot Clock	Scaled dot clock (scaled to 10%)

External Video Monitors

The computer can work with several sizes of external video monitors, however, you can connect an external monitor to the Power Macintosh 5260 only if the optional video display mirror out feature is used on that computer, and then it can only display the same video as the internal monitor. Table 3-10 shows the monitor types supported and the maximum pixel depths available. The pixel depth determines the maximum number of colors that can be displayed.

Table 3-10 Maximum pixel depths for video monitors

Monitor type	Screen size, in pixels	Maximum pixel depth, in bits per pixel	Maximum number of colors displayed
13-inch color	640 by 480	16	32,768
VGA and MultiSync	640 by 480	16	32,768

Video Timing Parameters

Figure 3-3 shows simplified timing diagrams and identifies the horizontal and vertical timing parameters in a video signal. Table 3-11 lists the values of those parameters for the different types of monitors supported by the Power Macintosh 5260 computer.

Figure 3-3 Video timing diagram

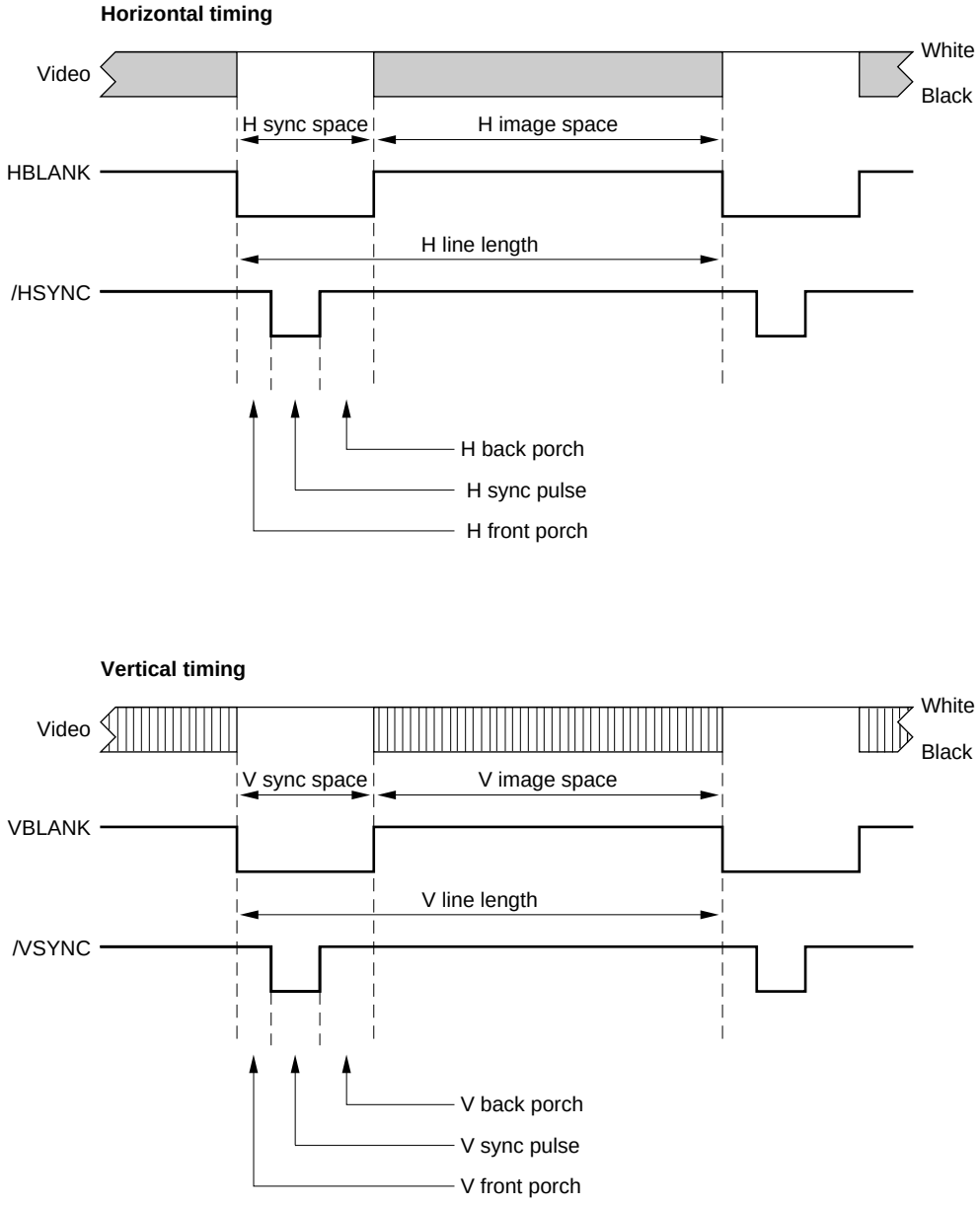


Table 3-11 lists the timing parameters for the 13-inch color monitor.

Table 3-11 Video timing parameters for 13-inch (640 by 480) monitor

Parameter	Specification
Dot clock	30.24 MHz
Dot time	33.07 ns
Line rate	35.00 kHz
Line time	28.57 µs (864 dots)
Horizontal active video	640 dots
Horizontal blanking	224 dots
Horizontal front porch	64 dots
Horizontal sync pulse	64 dots
Horizontal back porch	96 dots
Frame rate	66.72 Hz
Frame time	15.01 ms (525 lines)
Vertical active video	480 lines
Vertical blanking	45 lines
Vertical front porch	3 lines
Vertical sync pulse	3 lines
Vertical back porch	39 lines

CHAPTER 4

Expansion Features

This chapter describes the expansion features of the Power Macintosh 5260 computer: the DRAM expansion slot, the I/O expansion slot, the DVA connector on the video input module, and the communications slot.

Note
Apple does not support development of third-party cards for the video input slot or the communications slot. ◆

DRAM Expansion

The computers come with 8 MB of system DRAM in the form of a SIMM (single inline memory module) installed in one of two 72-pin SIMM slots on the main logic board. You can expand the DRAM up to a maximum of 64 MB by plugging in a second SIMM or by plugging in SIMMs of higher capacity in one or both SIMM slots.

Note
The video display buffer uses separate on-board DRAM. The display DRAM cannot be expanded. ◆

DRAM Configurations

Table 4-1 shows the different possible DRAM configurations. For more information, see the section "RAM Addresses" on page 21.

Table 4-1 DRAM configurations

SIMM 1	SIMM 2	Maximum DRAM
8 MB	0 MB	8 MB
8 MB	4 MB	12 MB
8 MB	8 MB	16 MB
16 MB	4 MB	20 MB
16 MB	8 MB	24 MB
16 MB	16 MB	32 MB
32 MB	4 MB	36 MB
32 MB	8 MB	40 MB
32 MB	16 MB	48 MB
32 MB	32 MB	64 MB

SIMM Slot Signal Assignments

Table 4-2 gives the signal assignments for the pins of the SIMM slot.

IMPORTANT

SIMMs used in Macintosh computers must meet the timing and electrical standards of these machines. SIMMs designed for other computers may not work. ▲

Table 4-2 SIMM slot signal assignments

Pin	Signal name	Signal description
1	GND	Ground
2	DQ0	Data input/output bus, bit 0
3	DQ16	Data input/output bus, bit 16
4	DQ1	Data input/output bus, bit 1
5	DQ17	Data input/output bus, bit 17
6	DQ2	Data input/output bus, bit 2
7	DQ18	Data input/output bus, bit 18
8	DQ3	Data input/output bus, bit 3
9	DQ19	Data input/output bus, bit 19
10	+5 V	+5 volts
11	n.c.	Not connected
12	A0	Address bus, bit 0
13	A1	Address bus, bit 1
14	A2	Address bus, bit 2
15	A3	Address bus, bit 3
16	A4	Address bus, bit 4
17	A5	Address bus, bit 5
18	A6	Address bus, bit 6
19	A10	Address bus, bit 10
20	DQ4	Data input/output bus, bit 4
21	DQ20	Data input/output bus, bit 20
22	DQ5	Data input/output bus, bit 5
23	DQ21	Data input/output bus, bit 21
24	DQ6	Data input/output bus, bit 6

continued

Table 4-2 SIMM slot signal assignments (continued)

Pin	Signal name	Signal description
25	DQ22	Data input/output bus, bit 22
26	DQ7	Data input/output bus, bit 7
27	DQ23	Data input/output bus, bit 23
28	A7	Address bus, bit 7
29	A11	Address bus, bit 11
30	+5 V	+5 volts
31	A8	Address bus, bit 8
32	A9	Address bus, bit 9
33	/RAS3	Row address strobe 3
34	/RAS2	Row address strobe 2
35	—	Reserved
36	—	Reserved
37	—	Reserved
38	—	Reserved
39	GND	Ground
40	/CAS0	Column address strobe 0
41	/CAS2	Column address strobe 2
42	/CAS3	Column address strobe 3
43	/CAS1	Column address strobe 1
44	/RAS0	Row address strobe 0
45	/RAS1	Row address strobe 1
46	n.c.	Not connected
47	/W	Write enable
48	n.c.	Not connected
49	DQ8	Data input/output bus, bit 8
50	DQ24	Data input/output bus, bit 24
51	DQ9	Data input/output bus, bit 9
52	DQ25	Data input/output bus, bit 25
53	DQ10	Data input/output bus, bit 10
54	DQ26	Data input/output bus, bit 26
55	DQ11	Data input/output bus, bit 11

continued

Table 4-2 SIMM slot signal assignments (continued)

Pin	Signal name	Signal description
56	DQ27	Data input/output bus, bit 27
57	DQ12	Data input/output bus, bit 12
58	DQ28	Data input/output bus, bit 28
59	+5 V	+5 volts
60	DQ29	Data input/output bus, bit 29
61	DQ13	Data input/output bus, bit 13
62	DQ30	Data input/output bus, bit 30
63	DQ14	Data input/output bus, bit 14
64	DQ31	Data input/output bus, bit 31
65	DQ15	Data input/output bus, bit 15
66	n.c.	Not connected
67	—	Reserved
68	—	Reserved
69	—	Reserved
70	—	Reserved
71	n.c.	Not connected
72	GND	Ground

DRAM Devices

The DRAM controller in the F108 IC supports 1 Mbit, 4 Mbit, and 16 Mbit devices; it does not support 64 Mbit devices. The DRAM controller supports four banks of DRAM.

Each SIMM can be configured in one or two banks. A single-bank SIMM using 1 Mbit, 4 Mbit, or 16 Mbit devices provides a DRAM capacity of 1 MB, 4 MB, or 16 MB, respectively. A two-bank SIMM using the same devices provides 2 MB, 8 MB, or 32 MB of DRAM.

IMPORTANT
You should not use 1-bit-wide devices in a SIMM slot because the number of devices you would need to provide the required capacity adds excessive capacitive loading to the address and control buses. ▲

The access time of the DRAM devices must be 80 ns or less. The DRAM controller in the F108 IC performs the refresh function, using CAS before RAS refresh and refreshing the DRAM devices within 15.6 µs. DRAM devices that require refreshing within 7.8 µs are not supported.

The DRAM controller in the F108 IC supports line burst transfers but does not support interleaved accesses.

Addressing DRAM

Signals A[11:0] make up a 12-bit multiplexed address bus that can support several different types of DRAM devices.

Depending on their internal design and size, different types of DRAM devices require different row and column address multiplexing. The F108 custom IC provides for two addressing modes, selected individually for each bank of DRAM. The system software initializes the address mode bits as part of the process of determining the amount of DRAM installed in the computer.

Table 4-3 shows how the signals are multiplexed during the row and column address phases for each of the addressing modes.

Table 4-3 Address multiplexing for DRAM devices

Address mode	Individual signals on 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 = 1												
Row address bits	A23	A22	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11
Column address bits	A24	A23	A21	A10	A9	A8	A7	A6	A5	A4	A3	A2
Address mode = 0												
Row address bits	A21	A20	A10	A19	A18	A17	A16	A15	A14	A13	A12	A11
Column address bits	A24	A23	A25	A22	A9	A8	A7	A6	A5	A4	A3	A2

Table 4-4 shows the address 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 256 addresses and stores 4 bits at a time.

Table 4-4 Address modes for various DRAM devices

Device size	Device type	Row bits	Colum bits	Address mode
1 megabit	256K by 4	9	9	1
4 megabits	1 M by 4	10	10	1
4 megabits	512K by 8	10	9	1
4 megabits	256K by 16	10	8	0
16 megabits	4 M by 4	11	11	1
16 megabits	4 M by 4	12	10	1
16 megabits	2 M by 8	11	10	1
16 megabits	2 M by 8	12	9	0
16 megabits	1 M by 16	12	8	0

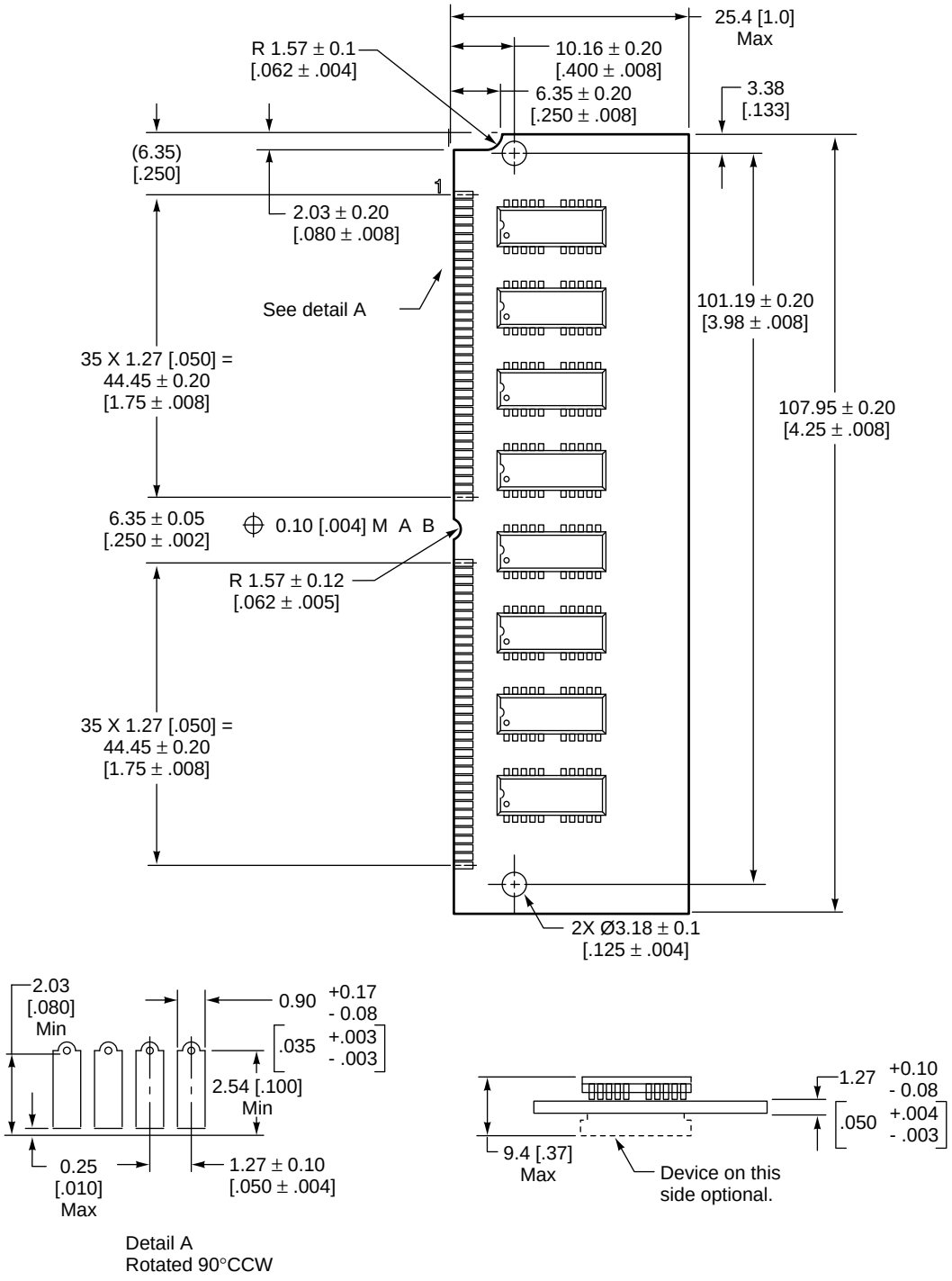
NOTE Refer to Table 4-3 on page 44 for information about addressing modes.

Mechanical Specifications for the SIMM

The SIMMs in the Power Macintosh 5260 computer are mechanically the same as the 72-pin SIMMs used in the Macintosh LC 630, Macintosh Quadra 630, Macintosh LC III, Macintosh LC 475, and Macintosh Quadra 605 and 610 computers. The mechanical design of the 72-pin SIMM is based on the industry standard design defined in the JEDEC Standard Number 21-C.

Figure 4-1 on page 46 shows the mechanical specifications of the SIMM. Pin contacts must be tin, not gold or copper, and the circuit board must dedicate one layer to power and one to ground.

Figure 4-1 SIMM mechanical specifications



Note: Dimensions are in millimeters with inches in brackets.

I/O Expansion Slot

The I/O expansion slot can accept either of two types of expansion cards: a 96-pin card similar to the PDS card used in the Macintosh LC II or a 114-pin card similar to the PDS card used in the Macintosh LC III.

IMPORTANT
Although often referred to as a processor-direct slot (PDS), the I/O expansion slot in the Power Macintosh 5260 computer is not a true PDS because it is not connected directly to the main processor. PDS cards designed to interact with the main processor—to provide, for example, a DRAM cache or an FPU—will not work in the Power Macintosh 5260's I/O expansion slot. ▲

I/O Expansion Connector

The I/O expansion connector is mechanically the same as the PDS connector in the Macintosh Quadra 630 and LC 630. It is essentially a 120-pin Euro-DIN connector with six pins removed to make a notch. The notch divides the connector into two sections: a 96-pin section that accepts the 96-pin connector used on PDS cards for the Macintosh LC II and a separate 18-pin section for additional signals. For more information see the section "Card Connectors" on page 53.

Connector Pin Assignments

Table 4-5 on page 48 gives the pin assignments for the I/O expansion connector. Pins 1 through 32 in all three rows (A, B, and C) correspond to the 96-pin section of the connector. Pins 33 and 34 in all three rows are missing—these pins correspond to the notch in the connector. Pins 35 through 40 in all three rows make up the 18-pin section of the connector. Table 4-6 on page 49 provides descriptions of the signals.

Except for one signal, 16MASTER on pin B31, the pin assignments on the 96-pin section of the extended PDS are the same as those on the PDS in the Macintosh LC II. On the Macintosh LC II, pin B31 is the Apple II video clock input.

▲ **WARNING**
Under no circumstances should you use the Analog GND pin (Pin 1, Row B) for a digital ground on your expansion card. Doing so causes digital noise to be coupled into the audio system, resulting in degraded sound quality. ▲

All the signals on the expansion connector are capable of driving at least one TTL load (1.6 mA sink, 400 µA source). Most of the signals are connected to other MOS devices on the main logic board; for these signals, the DC load on the bus signals is small. All the data lines (D31:0) are connected to the PrimeTime III custom IC so they have CMOS-type loads.

Table 4-5 Pin assignments for the expansion connector

Pin number	Row A	Row B	Row C
1	n.c.	Analog GND	/FPU.SEL
2	/SLOTIRQ	/R/W	/DS
3	/PDS.AS	+5 V	/BERR
4	/PDS.DSACK1	+5 V	/PDS.DSACK0
5	/HALT	SIZ1	SIZ0
6	FC2	GND	FC1
7	FC0	CLK16M	/RESET
8	/RMC	GND	/BG.SLOT
9	D31	D30	D29
10	D28	D27	D26
11	D25	D24	D23
12	D22	D21	D20
13	D19	D18	D17
14	D16	D15	D14
15	D13	D12	D11
16	D10	D9	D8
17	/BGACK	/BR_SLOT	A0
18	A1	A31	A27
19	A26	A25	A24
20	A23	A22	A21
21	A20	/IPL2	/IPL1
22	/IPL0	D3	D4
23	D2	D5	D6
24	D1	D0	D7
25	A4	A2	A3
26	A6	A12	A5
27	A11	A13	A7
28	A9	A8	A10
29	A16	A15	A14
30	A18	A17	A19
31	n.c.	16MASTER	FC3

Table 4-5 Pin assignments for the expansion connector (continued)

Pin number	Row A	Row B	Row C
32	+12 V	GND	–5 V
33	Pin not present	Pin not present	Pin not present
34	Pin not present	Pin not present	Pin not present
35	A28	/BG.SLOT	C16M
36	A29	+5 V	A30
37	/CIOUT	/CPU.AS	/STERM
38	/CBACK	n.c.	/CBREQ
39	n.c.	/CPU.DSACK0	n.c.
40	n.c.	GND	/CPU.DSACK1

Signal Descriptions

The I/O expansion slot is compatible with expansion cards designed for computers that use the MC68030 microprocessor (the Macintosh LC III and Macintosh LC 520 computers). Because the bus protocols on the 68040 I/O bus are not the same as those of the MC68030, the signals on the I/O expansion slot are not connected directly to the main processor. Instead, these signals are connected to the PrimeTime III custom IC, which emulates the MC68030 control and data buses.

The upper 30 address lines (A31:2) are connected directly to the 68040 address bus. The I/O bus adapter logic in the PrimeTime III IC provides the buffered data bus (IOD31:0) and the two lowest address lines (A1:0).

Table 4-6 describes the signals on the I/O expansion connector.

Table 4-6 Descriptions of the signals on the I/O expansion connector

Signal name	Signal description
A0:A31	Address lines.
/BERR	Bus error; bidirectional signal indicating that an error occurred during the current bus cycle; when /HALT is also asserted, /BERR causes the bus cycle to be retried.
/BGACK	Bus grant acknowledge; input signal indicating that a device on the card has become bus master.
/BG.SLOT	Bus grant to the slot; signal indicating that a device on the card can become bus master following completion of the current processor bus cycle (when /PDS.AS, /BGACK, and all the /DSACK signals are inactive).

continued

Table 4-6 Descriptions of the signals on the I/O expansion connector (continued)

Signal name	Signal description
/BR.SLOT	Bus request from the slot; input signal indicating that a device on the card is requesting to become bus master.
/CBACK	CPU burst acknowledge; used with /STERM during a burst transfer to indicate that an individual element of a burst transfer is ready.
/CBREQ	CPU burst request; used to initiate a quadruple longword burst transfer; tied to a 4.7K pull-up resistor.
/CIOUT	Cache inhibit out signal from main processor indicating that a second-level cache is allowed to participate in the current bus transaction; tied to a 300 Ω pull-down resistor.
C16M	Same signal as CLK16M.
CLK16M	Independent clock running at 15.6672 MHz; provided for compatibility with Macintosh LC and LC II PDS cards.
/CPU.AS	Address strobe; three-state signal indicating that an active bus transaction is occurring.
/CPU.DSACK0, /CPU.DSACK1	Data strobe acknowledge signals; asserted by the addressed bus slave to end a bus transaction; also used to inform the master of the size of the slave's data port. These signals are electrically connected to the corresponding /PDS.DSACK signals.
D31:01	Data lines.
/DS	Data strobe. During a read operation, /DS is asserted when a device on the card should place data on the data bus; during a write operation, /DS is asserted when the main processor has put valid data on the data bus.
FC2:0	Function code used to identify address space of current bus cycle; tied to pull-up and pull-down resistors to indicate supervisor data space accesses.
FC3	Additional function code bit used to indicate that the software is running in 32-bit address mode. (As in the Macintosh LC II, the software always runs in 32-bit mode.)
/FPU.SEL	Select signal for an optional MC68881 or MC68882 FPU; tied to a 4.7K pull-up resistor; never asserted by the logic board in a Macintosh LC 475 or Macintosh Quadra 605 computer.
/HALT	Used in conjunction with the /BERR signal to terminate a bus cycle with a retry response; not used to stop processor execution.
/IPL 2:0	Interrupt priority-level lines.

continued

Table 4-6 Descriptions of the signals on the I/O expansion connector (continued)

Signal name	Signal description
/PDS.AS	Address strobe; synchronized to 16 MHz regardless of the actual processor speed; asserted only when a valid slot address is being generated by the bus master. When the card is the active bus master, the card may drive either this signal or /CPU.AS, but not both.
/PDS.DSACK0, /PDS.DSACK1	Data strobe acknowledge signals; asserted by the addressed bus slave to end a bus transaction; also used to inform the master of the size of the slave's data port. These signals are electrically connected to the corresponding /CPU.DSACK signals.
/RESET	Notifies the expansion card that the CPU has reset.
/RMC	Three-state output signal that identifies the current bus cycle as part of an indivisible bus cycle such as a read-modify-write operation.
/R/W	Read/write; three-state output signal that defines the direction of the bus transfer with respect to the current bus master; logical one (1) indicates a bus-master read, zero (0) indicates bus-master write.
16MASTER	Indicates the width of the data port when the card is alternate bus master. A logical one (1) indicates a 16-bit port; logical zero (0) indicates a 32-bit port. The signal is pulled high on the main logic board.
SIZ0, SIZ1	Three-state output signals that work in conjunction with the PrimeTime III IC's dynamic bus sizing capabilities and indicate the number of bytes remaining to be transferred during the current bus cycle.
/SLOTIRQ	Interrupt request line from the card; reported to the system by way of the SLOT.E request; when low, generates a level-2 interrupt if the slot interrupt enable bit is set.
/STERM	Indicates termination of a synchronous transfer by a card using the MC68030 synchronous cycle.

The /BG.SLOT signal appears on two pins; there is no separate CPU.BG signal. The following signals on the expansion slot are permanently connected:

- /PDS.DSACK0 is connected to /CPU.DSACK0
- /PDS.DSACK1 is connected to /CPU.DSACK1

Unlike these signals, the /PDS.AS signal and the /CPU.AS signal are not connected together. The /PDS.AS signal is used only for addresses in the slot $E address range; the /CPU.AS signal is used for addresses in expansion slot and Super Slot spaces $6–$8, $A–$D, and $F (the slot $9 address spaces are used for built-in video circuitry).

IMPORTANT

The I/O expansion slot does not support PowerPC 603e bus transfers since it does not support processors operating at any clock frequency other than 16 MHz. ▲

Bus Master on a Card

The I/O expansion slot support cards with an MC68020 or MC68030 bus master. The PrimeTime III custom IC controls bus arbitration between the card's bus master and the PowerPC 603e microprocessor so that either bus master eventually obtains the bus. The MC68020 or MC68030 obtain the I/O data bus and the address bus. The PowerPC 603e obtains the processor data bus and the address bus. The Capella IC synchronizes the bus arbitration between the PowerPC 603e and the 68040 address bus. Because there is only one address bus, there can be only one bus master at a time.

Asynchronous transfers are the preferred method for transferring data to and from an I/O expansion card. When an I/O expansion card contains an active bus master, the PrimeTime III IC terminates successful data transfers using the DSACK signals. A slave on the expansion card can also terminate a transfer using DSACK signals.

The PrimeTime III IC can never be a synchronous slave on the I/O bus, so PrimeTime III cannot terminate data transfers as a slave using /STERM. On the other hand, a bus slave on an expansion card can terminate a 32-bit wide synchronous transfer using /STERM. PrimeTime III supports /STERM terminations as a master on the I/O bus, and all transfers from PrimeTime III to the expansion slot are based on the 16 MHz clock.

Incompatibility With Older Cards

While the I/O expansion slot physically accepts PDS cards designed for the Macintosh LC II and LC III, some of these cards will not work in the Power Macintosh 5260 computer. Cards that are incompatible include the following:

- Cards such as accelerators, 68882 FPU cards, and cache cards that are designed to work as coprocessors with an MC68020 or an MC68030 or to replace these microprocessors. This type of card will not work because the microprocessor is different and because the slot signals are not connected directly to the microprocessor.
- Cards with drivers that include incompatible code. Some drivers that do not follow Apple Computer's programming guidelines will not work on machines that use the PowerPC 603e microprocessor. For example, some of these drivers write directly to the cache control register in an MC68030 and such code will not work on a PowerPC 603e processor.
- Cards with drivers that include code to check the `gestaltMachineType` value and refuse to run on a newer CPU. Such cards have compatibility problems with all new Macintosh computers. However it provides protection for the user, since it means the cards will not run on a machine on which they have not been tested.

Designing an I/O Expansion Card

The I/O expansion card is approximately 3 inches wide by 5 inches long. It is installed parallel to the main logic board and you can connect to the card through an opening on the back panel of the computer. When an I/O expansion card is not installed, the opening on the back panel is protected by a snap-on cover. The 15-pin D-type connector

on the card can be accessed from the back of the case providing the external I/O connection.

The Appendix, "I/O Expansion Card Mechanical Drawings" beginning on page 97, contains drawings that show the recommended mechanical design guidelines for the expansion card. Drawing 1 shows the maximum dimensions of the expansion card and the location of the expansion connector. Drawing 2 defines component height restrictions. Drawing 3 shows how the card is installed on the main logic board.

Note
The I/O expansion card is the same size and shape as the PDS card used in the Macintosh LC III computer. ◆

Card Connectors

The custom 114-pin PDS connector on the computer's main logic board accepts either a 96-pin or 120-pin standard Euro-DIN connector. You can order connectors meeting Apple specifications from Amp Incorporated, Harrisburg, PA 17105 or from Augat Incorporated, Interconnect Products Division, P. O. Box 779, Attleboro, MA 02703. Refer to *Designing Cards and Drivers for the Macintosh Family*, third edition, for more information about these connectors.

Power for the Card

The maximum current available at each supply voltage is shown in Table 4-7. The card must not dissipate more than 4 W total. For example, if the card uses the maximum current at –5 V and +12 V, it must not use more than 300 mA from the +5 V supply.

Table 4-7 Power available for the expansion card

Voltage	Current
+5	800 mA
–5	20 mA
+12	200 mA

▲ **WARNING**
Cards dissipating more than 4 watts may overheat and damage the computer's circuitry or cause it to become inoperable. ▲

Card Address Space

The address space for the I/O expansion card appears in physical address spaces $E000 0000–$EFFF FFFF and $FE00 0000–$FEFF FFFF. To match the conventions used by the Slot Manager, software should address the card as if it were in slot space $E: either the 16 MB slot space $FE00 0000–$FEFF FFFF or the Super Slot space $E000 0000–$EFFF FFFF.

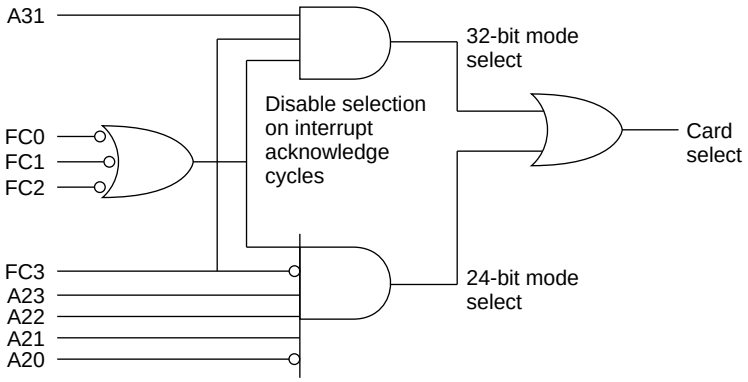
Card Select Signal

The I/O expansion card must generate its own select signal from the address and function code signals on the connector. The card select signal must be disabled when FC0, FC1, and FC2 are all active; that condition corresponds to a function code of 111 (CPU space). Figure 4-2 shows a logic circuit typically used for generating the card select signal.

IMPORTANT

To ensure compatibility with future hardware and software, you should minimize the chance of address conflicts by decoding all the address bits. To ensure that the Slot Manager recognizes your card, the card's declaration ROM must reside at the upper address limit of the 16 MB address space ($FE00 0000–$FEFF FFFF). ▲

Figure 4-2 Generating the card select signal



DVA Connector

The optional video input card has a separate connector called the DVA (digital video application) connector. The DVA connector provides access to the video input card's 4:2:2 unscaled YUV video input data bus and associated control signals. By means of a cable to the DVA connector, an I/O expansion card can gain access to the digital video bus on the video input card and use it to transfer real-time video data to the computer. Such an I/O expansion card can contain a hardware video compressor or other video processor.

The DVA connector is a 34-pin flat ribbon connector located at the top edge of the video input card. Figure 4-3 is a view of the main logic board showing the I/O expansion card and the location of the DVA connector on the video input card.

Figure 4-3 Location of the DVA connector

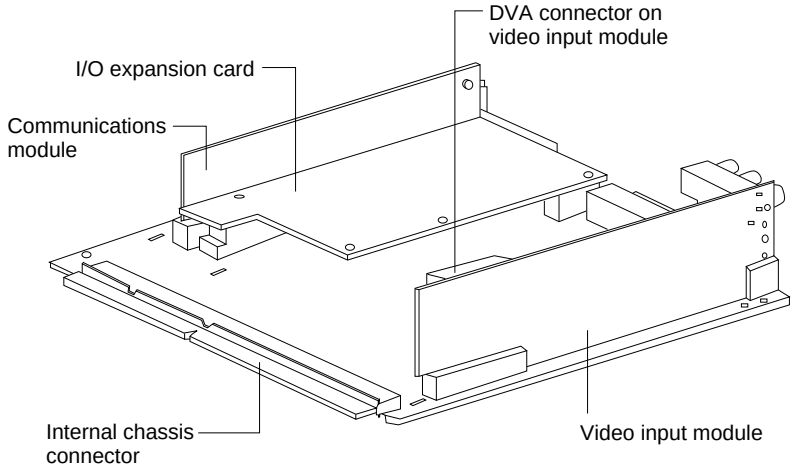
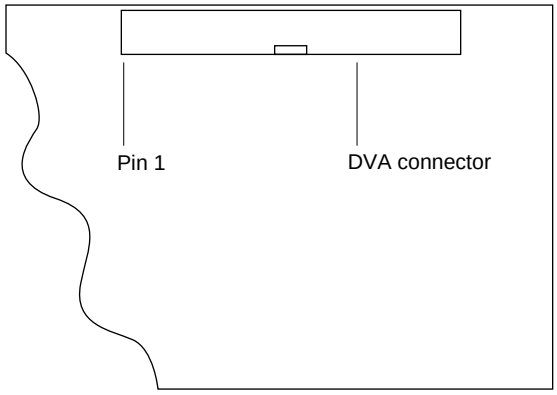


Figure 4-4 shows the orientation of the DVA connector on the video input module.

Figure 4-4 Orientation of the DVA connector



The DVA connector accepts YUV video and analog sound from the expansion card but does not itself generate YUV video output or audio output signals.

IMPORTANT

The DVA connector on the video input card provides some of the functionality of the DAV (digital audio/video) connectors found on the Power Macintosh 7100 and 8100 models and the Macintosh Quadra AV models, but it is not compatible with either of these connectors. Refer to *Macintosh DAV Interface for NuBus Expansion Cards* in *Developer Note Number 8* for more information. ▲

Pin Assignments and Signal Descriptions

Table 4-8 shows the pin assignments on the DVA connector and describes the signal functions. Refer to the glossary for further information about chrominance, luminance, and the YUV format.

IMPORTANT

The YUV video source also has associated stereo sound for left and right channels, YUV_SND_ LEFT and YUV_SND_RIGHT, respectively. In addition, there is the YUV return, YUV_RET, which is the ground reference for YUV_SND_LEFT and YUV_SND_RIGHT. If you are designing a video input module that uses the sound channels, you should incorporate a differential amplifier for each channel to receive the left and right sound signals and to remove audio noise from the system. Typically, analog sound signals are AC coupled. For common mode subtraction to work properly, the AC coupling should have a break frequency around 2 Hz. This enables the common mode rejection at 50 Hz to hit the required target of 40 dB (100:1). If the same technique is used with video signals, the break frequency should be less than 2 Hz to minimize tilt at 25 Hz. ▲

▲ **WARNING**

Under no circumstances should you ground the YUV_RET signal. If you do, your sound will contain a great deal of digital noise, and you will increase the computer's EMI emissions. This is particularly true if you have not used differential amplifiers to receive the YUV_SND_LEFT and YUV_SND_RIGHT signals. ▲

Table 4-8 DVA connector pin assignments and signal descriptions

Pin number	Signal name	Signal description
1	UV(7)	Digital chrominance data bus bit 7
2	UV(6)	Digital chrominance data bus bit 6
3	UV(5)	Digital chrominance data bus bit 5
4	UV(4)	Digital chrominance data bus bit 4
5	UV(3)	Digital chrominance data bus bit 3
6	UV(2)	Digital chrominance data bus bit 2
7	UV(1)	Digital chrominance data bus bit 1
8	UV(0)	Digital chrominance data bus bit 0
9	Y(7)	Digital luminance data bus bit 7
10	Y(6)	Digital luminance data bus bit 6
11	Y(5)	Digital luminance data bus bit 5

continued

Table 4-8 DVA connector pin assignments and signal descriptions (continued)

Pin number	Signal name	Signal description
12	Y(4)	Digital luminance data bus bit 4
13	Y(3)	Digital luminance data bus bit 3
14	Y(2)	Digital luminance data bus bit 2
15	Y(1)	Digital luminance data bus bit 1
16	Y(0)	Digital luminance data bus bit 0
17	Ground	Ground
18	LLCB	Line-locked clock signal
19	Ground	Ground
20	CREFB	Clock reference signal
21	Ground	Ground
22	VS	Vertical sync signal
23	Ground	Ground
24	HS	Horizontal sync signal
25	Ground	Ground
26	HREF	Horizontal reference signal
27	Ground	Ground
28	DIR	YUV directional signal
29	Reserved	Reserved
30	Reserved	Reserved
31	Ground	Ground
32	YUV_SND_LEFT	Sound input left channel; read the information at the beginning of this table for more details
33	YUV_RET	Ground reference for YUV_SND_LEFT and YUV_SND_RIGHT; read the information at the beginning of this table for more details
34	YUV_SND_RIGHT	Sound input right channel; read the information at the beginning of this table for more details

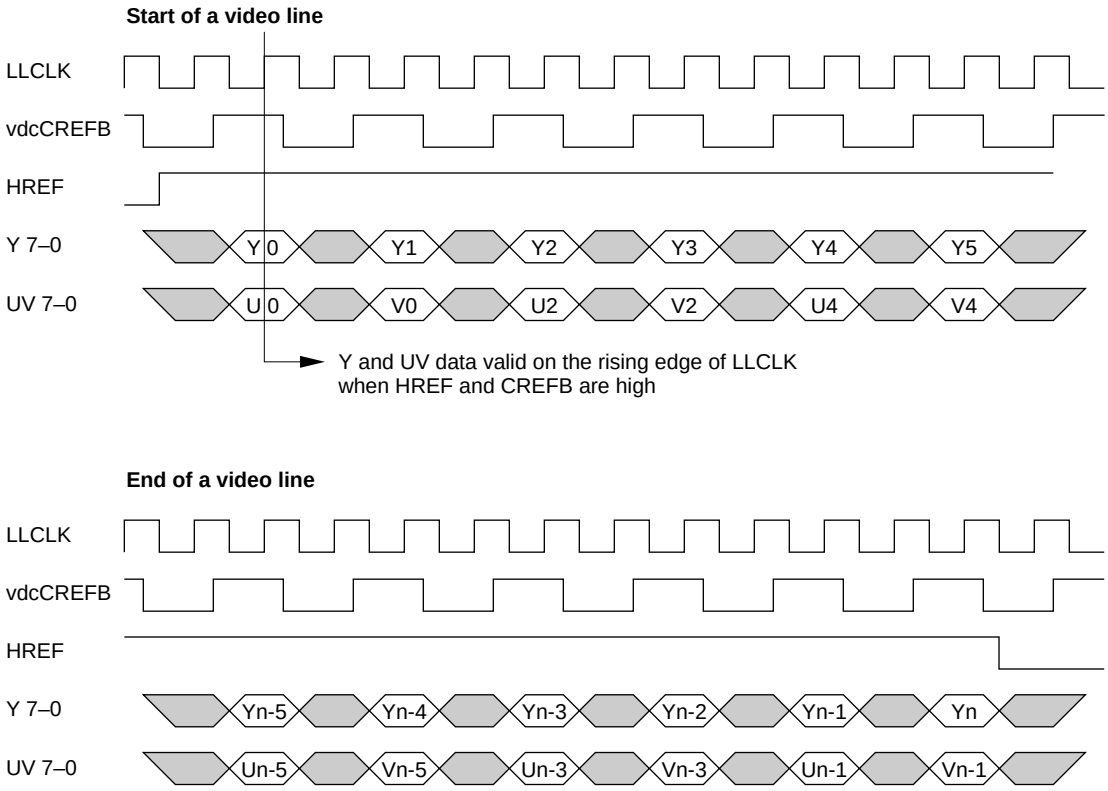
Using the YUV Bus

The video input module contains a digital video decoder and scaler (DESC), the Philips SAA7196 IC. Logic on the video input card uses the CVBS port on the DESC and pulls the DIR signal low, disabling the YUV bus. For an expansion card to use the YUV bus, the software associated with the card must set the DIR signal high so that the DESC will accept data on the YUV bus.

Video Data Format

Digital video data is transmitted as lines and fields. Each line consists of an even number of samples on the Y and UV buses as shown in Figure 4-5. HREF is high during a video line and low during the horizontal blanking interval. The falling edge of the VS signal indicates the beginning of a video field. For more information about digital video data in YUV format, see *Macintosh DAV Interface for NuBus Expansion Cards* in *Developer Note Number 8.*

Figure 4-5 Video data timing



Communications Slot

The main logic board has a separate slot for an optional communications card, 1.57 inches (40 mm) wide by 6 inches (152 mm) long.

The communications slot supports 68030 protocol transfers with a 68030 bus master/slave, SCC port A (modem port) for an Apple designed 2400 baud modem, and a special serial port for an Apple designed 14,400 baud modem. The communications slot is

capable of supporting a 68030 bus master and contains one set of 68030 bus arbitration control signals, but it does not support 68040 bus protocol transfers.

The communications slot connector is a 112-pin half-height microchannel connector. A communications card mounts vertically in the connector and its I/O connector is accessed through the communications port access hole on the right hand side of the back panel. If the card is not in use, a cover is placed over the access hole, as shown in Figure 1-2 on page 6.

A maximum of 2.5 watts of power are allocated to the communications slot. The following shows the maximum possible current ratings for each power line:

Voltage	Current
+5 V	500 mA
+12 V	100 mA
Trickle +5 V	5 mA
–5 V	20 mA

Note
Apple Computer does not support development of third-party cards for the communications slot. This developer note, therefore, does not provide detailed pin assignments for the communications connector. ◆

CHAPTER 5

Software Features

This chapter gives an overview of the software in the Power Macintosh 5260 computer's ROM.

For a description of the system software for the internal IDE hard disk, see Chapter 6, "Software for the IDE Hard Disk."

ROM Software

The ROM in the Power Macintosh 5260 computer is based on the ROM used in the current Power Macintosh models with the changes necessary to support machine-specific hardware.

The following is a list of the most significant ROM changes:

- Hardware Init now includes support for MMU programming and other PowerPC 603e microprocessor functions, addition of new diagnostics, and removal of the 68040 check/support code.
- The nanokernal has been modified to support the PowerPC microprocessor.
- The software no longer supports 1- or 2- bit video modes.
- The software supports both 8-bit and 16-bit sound; the Power Macintosh 5260 computer uses 16-bit sound.

The ROM contains tables and code that identify the computer. Applications can find out which computer they are running on by using the Gestalt Manager routines; see *Inside Macintosh: Overview*. The `gestaltMachineType` value returned by the Power Macintosh 5260 computer is 41 (hexadecimal $29), which is the same as the value returned for the Power Macintosh 5200 computer.

System Software

The Power Macintosh 5260 computer is shipped with a version of System 7.5 software preinstalled. System 7.5 Update 1.0 is included in the preinstalled system software. The disk labeled "Install Me First" includes a system enabler file that contains the resources the system needs to start up and initialize the computer.

As soon as the system software on disk takes over the startup process, it searches for all system enablers that can start up the particular machine. Each system enabler contains a resource that specifies which computers it is able to start up and the time and date of its creation. If the system software finds more than one enabler for the particular computer, it passes control to the one with the most recent time and date.

In general, the system enabler included in each reference release of system software is able to start up all previous computers. The enablers for computers introduced after a reference release may be independent or may use resources from the previous reference release.

The system enabler includes modifications to the video digitizer allowing it to run in native mode to improve video capture performance.

CHAPTER 6

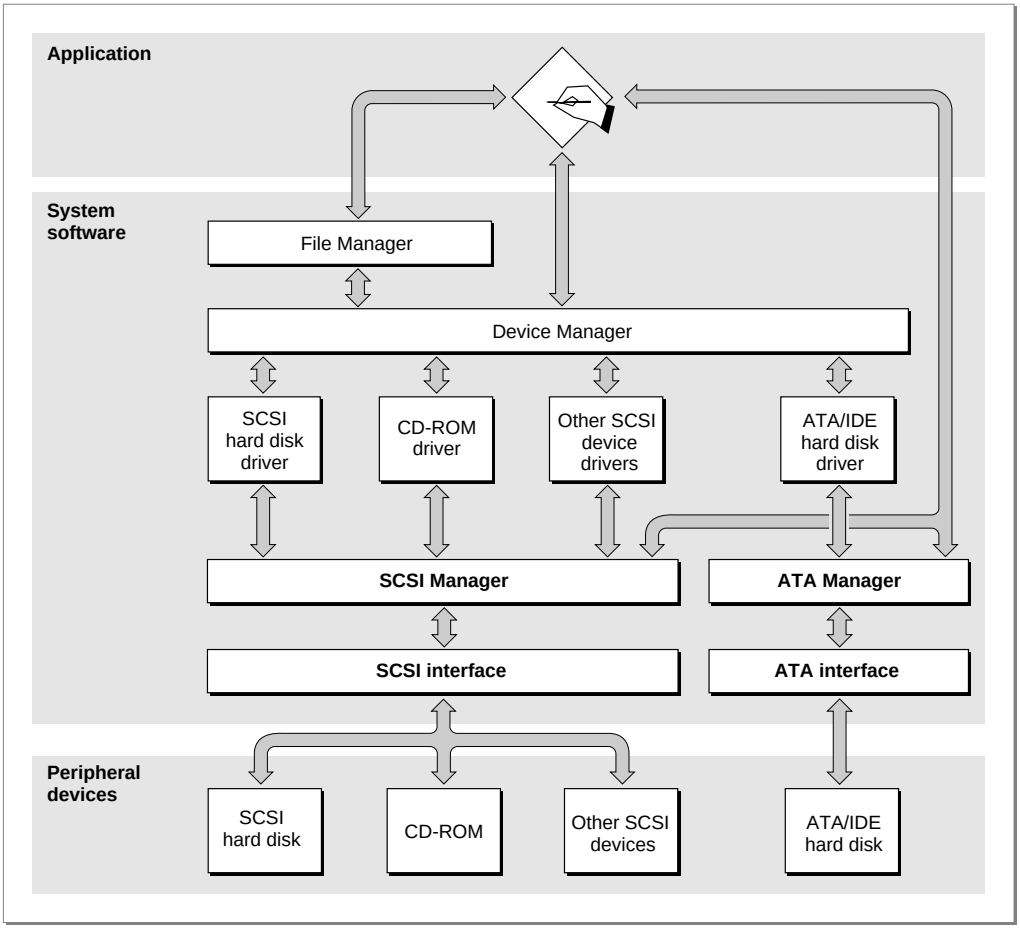
Software for the IDE Hard Disk

This chapter describes the system software that controls an IDE hard disk drive installed in the Power Macintosh 5260 computer. To use the information in this chapter, you should be familiar with writing programs for the Macintosh computer, with device drivers that manipulate devices directly, and with the ATA/IDE specification ANSI AT Attachment Interface with Extensions (ATA-2).

Introduction to IDE Software

Support for IDE (integrated drive electronics) hard disk drives is incorporated into the ROM software. System software for controlling IDE hard drives is included in a new IDE hard disk drive device driver and the ATA Manager. The relationship of the IDE hard disk drive device driver and the ATA Manager to the Macintosh system architecture is shown in Figure 6-1.

Figure 6-1 Relationship of the ATA Manager to the Macintosh system architecture



At the system level, the IDE device driver and ATA Manager work in the same way as the SCSI Manager and the associated SCSI device drivers. The IDE hard disk device driver provides drive partition, data management, and error-handling services for the Mac OS. It also determines device capacity and controls device-specific features. The ATA Manager provides the IDE device driver with an interface to the IDE hard disk drive.

IDE hard disk drives appear on the desktop in the same way as SCSI hard disk drives. Except for applications that perform low-level services, such as formatting and partitioning utilities, applications interact with the IDE hard disk drives in a device-independent manner through the File Manager.

IDE Hard Disk Device Driver

The IDE hard disk device driver provides OS-dependent services through a set of driver routines that interface with the Mac OS. In addition, it provides additional control and status functions that are specific to this implementation of the IDE hard disk device driver. The required driver routines, as specified in *Inside Macintosh: Devices*, are `open`, `close`, `prime`, `control`, and `status`.

In addition to the required functions, the IDE hard disk device driver provides support for device-specific features. IDE hard disk device driver control and status functions are defined in "IDE Hard Disk Driver Reference" beginning on page 66.

At system startup time, if a RAM-based driver is not found on the IDE drive media, the IDE device driver in the ROM is installed as one of the device drivers. This is different from the driver loading sequence for SCSI hard disk drive devices, since these drivers are RAM based and are always loaded from the device media.

The IDE hard disk device driver does not provide request queuing. All driver requests are either completed immediately or are passed to the ATA Manager for further processing. For further information about the control and status functions for the IDE hard disk device driver, see "IDE Hard Disk Driver Reference."

ATA Manager

The Macintosh ATA Manager schedules I/O requests from the IDE hard disk device driver, the operating system, and applications. It is also manages the hardware interface to the IDE controller electronics.

When making calls to the ATA Manager you have to pass and retrieve parameter information through a parameter block. The size and content of the parameter block depends on the function being called. However, all calls to the ATA Manager have a common parameter block header structure. The structure of the `ataPBHdr` parameter block is common to all ATA parameter block data types. Several additional ATA parameter block data types have been defined for the various functions of the ATA Manager. The additional parameter block data types, which are specific to the function being called, are described in "ATA Manager Reference" beginning on page 76.

IDE Hard Disk Driver Reference

This section describes the Macintosh device driver functions provided by the IDE hard disk device driver. The information in this section assumes that you already know how to use device driver services on the Macintosh computer. If you are not familiar with Macintosh device drivers, refer to the chapter "Device Manager" in *Inside Macintosh: Devices* for additional information.

High-Level Device Manager Functions

The IDE hard disk driver supports the required set of high-level Device Manager routines, as defined in the chapter "Device Manager" of *Inside Macintosh: Devices*. Those routines are briefly defined here for convenience. Additional control functions supported in the IDE hard disk driver are defined in "IDE Hard Disk Driver Control Functions" beginning on page 69.

open

The `open` routine opens the IDE hard disk device driver during the startup sequence after the driver code is retrieved from the ROM. The `open` routine returns a reference number to the driver. That number is used in subsequent calls to the driver.

The following operations take place at startup time:

- memory allocation and driver globals and internal variables initialization
- power-on drive diagnostics
- device detection and verification
- device initialization
- device information uploading
- drive queue management and event posting

After startup, the driver responds with `noErr` to subsequent calls to the `open` routine and does not repeat the operations performed at startup time.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`openErr`	–23	Could not open the driver
`DRVRCantAllocate`	–1793	Global memory allocation error
`ATABufFail`	–1796	Device buffer test failed

close

The `close` routine deallocates the driver memory storage, removes the drive queue entry point, and closes the IDE hard disk device driver.

RESULT CODES

`noErr`	0	Successful completion, no error occurred

prime

The `prime` routine performs either a `read` or `write` command as specified by the caller. During this process the following operations take place:

- byte to block translation
- address translation
- update of the `IOParameter` block
- high-level error recovery and retry algorithm
- ATA Manager parameter block management

Refer to "ATA Manager" on page 65 for more information about the parameter block structure for the ATA Manager.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`ioErr`	–36	I/O error
`paramErr`	–50	Invalid parameter specified
`nsDrvErr`	–56	No such drive installed

status

The `status` routine returns status information about the IDE hard disk device driver. The type of information returned is specified in the `csCode` field and the information itself is pointed to by the `csParamPtr` field.

The IDE hard disk device driver implements the same status functions supported by the SCSI hard disk device driver. The status functions supported by the IDE hard disk driver are listed below.

Value of csCode	Definition
8	Return drive status information
43	Return driver Gestalt information
70	Power mode status information

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`statusErr`	–18	Unimplemented status function; could not complete requested operation
`nsDrvErr`	–56	No such drive installed

control

The `control` routine sends control information to the IDE hard disk device driver. The type of control function is specified in `csCode`.

The IDE driver implements the same control functions supported by the SCSI hard disk driver. The control functions are listed below and described in “IDE Hard Disk Driver Control Functions” beginning on page 69.

Value of csCode	Definition
1	Kill I/O
5	Verify media
6	Format media
7	Eject media
21	Return drive icon
22	Return media icon
23	Return drive characteristics
65	Need-time code
70	Power-mode status management control

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`controlErr`	–17	Unimplemented control function; could not complete requested operation
`nsDrvErr`	–56	No such drive installed

IDE Hard Disk Driver Control Functions

The IDE hard disk driver supports a standard set of control functions for IDE hard disk drive devices. The functions are used for control, status, and power management.

killIO

The `killIO` function is a standard function defined in *Inside Macintosh: Devices*.

IMPORTANT

This function is not supported by the IDE hard disk driver. A call to `KillIO` returns a `controlErr` result code. ▲

verify

The `verify` function requests a read verification of the data on the IDE hard drive media. This function performs no operation.

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

→	`csCode`	A value of 5.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	None defined.
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

format

The `format` function initializes the hard drive for use by the operating system. Because IDE hard drives are low-level formatted at the factory, this function does not perform any operation. The driver always returns `noErr` if the logical drive number is valid.

Parameter block

→	`csCode`	A value of 6.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	None defined.
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

ejectMedia

The `ejectMedia` function prepares and initiates an eject operation from the specified drive. This function applies only to drives with removable media.

Note

The `ejectMedia` function is not supported by the IDE hard disk driver; this function returns a `noErr` if the logical drive number is valid. ◆

Parameter block

→	`csCode`	A value of 7.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	None defined.
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

return drive icon

The `return drive icon` function returns a pointer to the device icon and the device name string. The drive icon is the same as the media icon for IDE hard disk drives. The drive icon for IDE hard disk devices is shown in Figure 6-2.

Figure 6-2 IDE hard disk drive icon

Parameter block

→	`csCode`	A value of 21.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	None defined.
←	`csParam[0–1]`	Address of drive icon and name string (information is in `ICN#` format).
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

return media icon

The `return media icon` function returns a pointer to the media icon and the name string. The media icon is the same as the drive icon for IDE hard disk drives, as shown in Figure 6-2.

Parameter block

→	`csCode`	A value of 22.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	None defined.
←	`csParam[0–1]`	Address of drive icon and name string (information is in `ICN#` format).
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

return drive characteristics

The `return drive characteristics` function returns information about the characteristics of the specified drive as defined in *Inside Macintosh,* Volume V.

Parameter block

→	`csCode`	A value of 23.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	None defined.
←	`csParam[0–1]`	Drive information: $0601 = primary, fixed, SCSI, internal $0201 = primary, removable, SCSI, internal
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

needTime code

The `needTime code` function provides time for the driver to perform periodic operations such as checking for media insertion or ejection events related to removable cartridge drives. For additional information about how this function is used, see the description of the driver flag `dNeedTime` in the chapter "Device Manager" of *Inside Macintosh: Devices*. This function performs no operation on the IDE hard disk drive in the Power Macintosh 5260 computer.

Parameter block

→	`csCode`	A value of 65.
→	`csParam[]`	None defined.
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

power management

The `power management` function can be used to reduce the drive's power consumption and decrease system noise levels by putting the hard drive into a standby state.

Note
The `power management` control function is most useful on PowerBook computers, where it can be used to reduce drive power consumption and thereby extend useful battery life. ◆

The `power management` function provides three modes of operation for IDE hard disk drives: idle, standby, and sleep.

In the idle state, the nonessential electronics on the IDE hard drive are disabled, for example, the read and write channels. The spindle motor remains enabled during the idle state, so the drive still responds immediately to any commands requesting media access.

In the standby state, the head is parked and the spindle motor is disabled. The drive interface remains active and is still capable of responding to commands. However, it may take several seconds to respond to media access commands, because the drive's spindle motor must return to full speed before media access can take place.

In the sleep state, the drive interface and spindle motor are disabled. To return the drive to full operation after the sleep state has been enabled, the user must restart or reset the computer.

Parameter block

→	`csCode`	A value of 70.
→	`ioVRefNum`	The logical drive number.
→	`csParam[0]`	The most significant byte contains one of the following codes: 0 = enable the active mode 1 = enable the standby mode 2 = enable the idle mode 3 = enable the sleep mode
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`controlErr`	–17	The power management information could not be returned because of a manager error
`nsDrvErr`	–56	The specified logical drive number does not exist

drive status info

The IDE hard disk device driver provides a function for retrieving status information from the drive. The `drive status info` function returns the same type of information that disk drivers are required to return for the `status` function, as described in the chapter "Device Manager" in *Inside Macintosh: Devices*.

Parameter block

→	`csCode`	A value of 8.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	The `csParam` field contains status information about the internal IDE disk drive.
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	The specified logical drive number does not exist

return driver gestalt

The `return driver gestalt` function provides the application with information about the IDE hard disk driver and the attached device. Several calls are supported under this function. A Gestalt selector is used to specify a particular call.

The `DriverGestaltParam` data type defines the IDE Gestalt parameter block:

```
typedef struct DriverGestaltParam
{
   ataPBHdr            /* ATA Manager parameter block
                          header structure */

   short               ioVRefNum;              /* refNum of device */
   short               csCode;                 /* Driver gestalt code */
   OSType              driverGestaltSelector;  /* Gestalt selector */
   driverGestaltInfo   driverGestaltResponse;  /* Returned result */
} DriverGestaltParam;
```

The fields `driverGestaltSelector` and `driverGestaltResponse` are 32-bit fields.

Parameter block

→	`ioVRefNum`	The logical drive number.
→	`csCode`	A value of 43.
→	`driverGestaltSelector`	Gestalt function selector. This is a 32-bit ASCII field containing one of the following selectors:

`sync`	Indicates a synchronous or asynchronous driver.
`devt`	Specifies the type of device the driver is controlling.
`intf`	Specifies the device interface.
`boot`	Specifies PRAM value to designate this driver or device.
`vers`	Specifies the version number of the driver.

←	`driverGestaltResponse`	Returns result based on the driver gestalt selector. The possible four-character return values are: `TRUE` If the `sync` driver selector is specified, this Boolean value indicates that the driver is synchronous; a value of `FALSE` indicates an asynchronous driver. `disk` If the `devt` driver selector is specified, this value indicates a hard disk driver. `ide` If the `intf` driver selector is specified, this value indicates the interface is IDE. `0` If the `boot` driver selector is specified, this value indicates that this is the boot driver or boot device. *nnnn* If the `vers` selector is specified, the current version number of the driver is returned.
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`statusErr`	–18	Unknown selector was specified
`nsDrvErr`	–56	The specified logical drive number does not exist

power-mode status

The `power-mode status` function returns the current power mode state of the internal hard disk.

Parameter block

→	`csCode`	A value of 70.
→	`ioVRefNum`	The logical drive number.
→	`csParam[]`	None defined.
←	`csParam[]`	The most significant byte of this field contains one of the following values: 1 = drive in standby mode 2 = drive in idle mode 3 = drive in sleep mode
←	`ioResult`	See result codes.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`statusErr`	–18	The power management information could not be returned because of a manager error
`nsDrvErr`	–56	The specified logical drive number does not exist

ATA Manager Reference

This section defines the data structures and functions specific to the ATA Manager. The section "The ATA Parameter Block" shows the data structure of the ATA parameter block. The section "Functions" section describes the functions for managing and performing data transfers through the ATA Manager.

The ATA Parameter Block

This section defines the fields common to all ATA Manager functions that use the ATA parameter block. The fields used for specific functions are defined in the description of the functions to which they apply. You use the ATA parameter block for all calls to the ATA Manager. The `ataPBHdr` data type defines the ATA parameter block.

The parameter block includes a field, `MgrFCode`, in which you specify the function selector for the particular function to be executed; you must specify a value for this field. Each ATA function may use different fields of the ATA parameter block for parameters specific to that function.

An arrow preceding the comment indicates whether the parameter is an input parameter, an output parameter, or both.

Arrow	Meaning
→	Input
←	Output
↔	Both

The ATA parameter block header structure is defined as follows:

```
typedef    struct    ataPBHdr       /* ATA Manager parameter block
                                      header structure */
{
           Ptr       ataLink;       /* Reserved */
           UInt16    ataQType;      /* Type byte */
           UInt8     ataPBVers;     /* → Parameter block
                                       version number */
           Uint8     hdrReserved;   /* Reserved */
           Ptr       hdrReserved2;  /* Reserved */
           ProcPtr   ataCompletion; /* Completion routine */
           OSErr     ataResult;     /* ← Returned result */
```

```
            UInt8     MgrFCode;      /* → Manager function code */
            UInt8     ataIOSpeed;    /* → I/O timing class */
            UInt16    ataFlags;      /* → Control options */
            SInt16    hdrReserved3;  /* Reserved */
            UInt32    deviceID;      /* → Device ID */
            UInt32    TimeOut;       /* → Transaction timeout
                                        value */
            Ptr       ataPtr1;       /* Client storage Ptr 1 */
            Ptr       ataPtr2;       /* Client storage Ptr 2 */
            UInt16    ataState;      /* Reserved, init to 0 */
            UInt16    hdrReserved4;  /* Reserved */
            SInt32    hdrReserved5;  /* Reserved */
} ataPBHdr;
```

Field descriptions

`ataLink` This field is reserved for use by the ATA Manager. It is used internally for queuing I/O requests. It must be initialized to 0 before calling the ATA Manager.

`ataQType` This field is the queue type byte. It should be initialized to 0 before calling the ATA Manager.

`ataPBVers` This field contains the parameter block version number. A value of 1 is the only value currently supported. Any other value results in a `paramErr`.

`hdrReserved` This field is reserved for future use. To ensure future compatibility, all reserved fields should be set to 0.

`hdrReserved2` This field is reserved for future use. To ensure future compatibility, all reserved fields should be set to 0.

`ataCompletion` This field contains the completion routine pointer that is called when the request is completed. When this field is set to 0, it indicates a synchronous I/O request; a nonzero value indicates an asynchronous I/O request. The routine to which this field points is called when the request has finished without error, or when the request has terminated owing to an error. This field is valid for any manager request. The completion routine is called as follows:

```
pascal void (*RoutinePtr) (ataIOPB *)
```

The completion routine is called with the associated manager parameter block in the stack.

`ataResult` Completion status. This field is returned by the ATA Manager after the request is completed. Refer to the section "Result Code Summary" on page 94 for a list of the possible error codes returned in this field.

`MgrFCode` This field is the function selector for the ATA Manager. The functions are summarized in Table 6-2 on page 81. An invalid code in this field results in an `ATAFuncNotSupported` error.

`ataIOSpeed` This field specifies the I/O cycle timing requirement of the specified IDE drive. The field should contain word 51 of the identify drive

data. Currently values 0 through 3 are supported, as defined in the ATA IDE specification. If a timing value higher than one supported is specified, the manager operates in the fastest timing mode supported by the manager. Until the timing value is determined by examining the identify drive data returned by the `ATA_Identify` function, the client should request operations using the slowest mode (mode 0).

`ataFlags` This 16-bit field contains control settings that indicate special handling of the requested function. The control bits are defined in Table 6-1 on page 79.

`hdrReserved3` This field is reserved for future use. To ensure future compatibility, all reserved fields should be set to 0.

`deviceID` A short word that uniquely identifies an IDE device. This field consists of the following structure:

```
typedef struct     /* Device ID structure */
{
ushort Reserved;   /* The upper word is reserved */
ushort deviceNum;  /* Consists of device ID and
                      bus ID */
} deviceIdentification;
```

Bit 15 of the `deviceNum` field indicates master (=0) /slave (=1) selection. Bits 14 through 0 contain the bus ID (for example, 0x0 = master unit of bus 0, 0x80 = slave unit of bus 0). The present implementation allows only one device in the master configuration. This value is always 0.

`TimeOut` This field specifies the transaction timeout value in milliseconds. A value of zero disables the transaction timeout detection.

`ataPtr1` This pointer field is available for application use. It is not modified by the ATA Manager.

`ataPtr2` This pointer field is available for application use. It is not modified by the ATA Manager.

`ataState` This field is used by the ATA Manager to keep track of the current bus state. This field must contain zero when calling the manager. Bus states are defined in Table 6-1 on page 79.

`hdrReserved4` This field is reserved for future use. To ensure future compatibility, all reserved fields should be set to 0.

`hdrReserved5` This field is reserved for future use. To ensure future compatibility, all reserved fields should be set to 0.

Table 6-1 describes the functions of the control bits in the `ataFlags` field.

Table 6-1 Control bits in the `ataFlags` field

Name	Bit	Definition
—	0–2	Reserved.
`RegUpdate`	3	When set to 1 this bit indicates that a set of device registers should be reported back upon completion of the request. This bit is valid for the `ATA_ExecI/O` function only. The following device registers are reported back: Sector count register Sector number register Cylinder register(s) SDH register Refer to the function `ATA_ExecIO` on page 81 for additional information.
—	4–7	Reserved.

`SGType`	8, 9	This 2-bit field specifies the type of scatter gather list passed in. The field is only valid for read/write operations.
		
		The following types are defined:
		
		00 = Scatter gather disabled
		01 = Scatter gather type I enabled
		10 = Reserved
		11 = Reserved
		
		When set to 0, this field indicates that the `ioBuffer` field contains the host buffer address for this transfer, and the `ioReqCount` field contains the byte transfer count.
		
		When set to 1, this field indicates that the `ioBuffer` and the `ioReqCount` fields of the parameter block for this request point to a host scatter gather list and the number of scatter gather entries in the list, respectively.
		
		The format of the scatter gather list is a series of the following structure definitions:
		
		`typedef struct /* SG entry structure */`
		`{`
		`  UInt8* ioBuffer; /* → Data buffer pointer */`
		`  UInt32 ioReqCount; /* → Byte count */`
		`} IOBlock;`

continued

Table 6-1 Control bits in the `ataFlags` field (continued)

Name	Bit	Definition
`QLockOnError`	10	When set to 0, this bit indicates that an error during the transaction should not freeze the I/O queue for the device. When an error occurs on an I/O request with this bit set to 0, the next queued request is processed following this request. When an error occurs on an I/O request with this bit set to 1, the user must issue an `ATA_QRelease` command to continue. A status code of hexadecimal 717 is returned for subsequent asynchronous I/O requests until the `I/O Queue Release` command is issued.
`Immediate`	11	When this bit is set to 1, it indicates that the request must be executed as soon as possible and the status of the request must be returned. It forces the request to the head of the I/O queue for immediate execution. When this bit is set to 0, the request is queued in the order received and is executed in that order.
`ATAioDirection`	12, 13	This bit field specifies the direction of data transfer. Bit values are binary and are defined as follows: 00 = No data transfer 10 = Data direction in (read) 01 = Data direction out (write) 11 = Reserved
`ByteSwap`	14	When set to 1, this bit indicates that every byte of data prior to transmission in write operations and upon reception in read operations is to be swapped. When this bit is set to 0, it forces bytes to go out in the LSB-MSB (least significant-most significant) format compatible with IBM clones. Typically, this bit should be set to 0. Setting this bit has performance implications because the byte swap is performed by the software. Use this bit with caution.
—	15	Reserved. Should be set to 0.

Functions

This section describes the ATA Manager functions that manage and perform data transfers. Each function is requested through a parameter block specific to that service. A request for an IDE function is specified by a function code within the parameter block. The entry point for all the functions is the same.

ATA Manager function names and codes are shown in Table 6-2.

Table 6-2 ATA Manager functions

Function name	Code	Description
`ATA_ExecIO`	$01	Execute ATA I/O
`ATA_MgrInquiry`	$90	ATA Manager inquiry
`ATA_BusInquiry`	$03	Bus inquiry
`ATA_QRelease`	$04	I/O queue release
`ATA_NOP`	$00	No operation
`ATA_Abort`	$10	Terminate command
`ATA_RegAccess`	$12	ATA device register access
`ATA_Identify`	$13	Get the drive identification data
`ATA_ResetBus`	$11	Reset IDE bus
`ATA_DrvrRegister`	$85	Register the driver reference number
`ATA_DrvrDeregister`	$87	Deregister the driver reference number
`ATA_FindRefNum`	$86	Look up the driver reference number

ATA_ExecIO

You can use the `ATA_ExecIO` function to perform all data I/O transfers to or from an IDE device. Your application must provide all the parameters needed to complete the transaction prior to calling the ATA Manager. Upon return, the parameter block contains the result of the request.

The manager function code for the `ATA_ExecIO` function is $01.

The parameter block associated with the `ATA_ExecIO` function is defined below:

```
typedef     struct            /* ATA_ExecIO structure */

{
   ataPBHdr                   /* ATA Manager parameter block
                               header structure */
```

```
   SInt8    ataStatusReg;     /* ← Last device status register
                                  image */
   SInt8    ataErrorReg;      /* ← Last device error register
                                 image (valid if bit 0 of Status
                                 field is set) */
   SInt16   ataReserved;      /* Reserved */
   UInt32   BlindTxSize;      /* → Data transfer size */
   UInt8*   IOBuffer;         /* → Data Buffer pointer */
   UInt32   IORegCount        /* → transfer length*/
   UInt32   ataActualTxCnt;   /* ← Actual number of bytes
                                 transferred */
   UInt32   ataReserved2;     /* Reserved */
   devicePB RegBlock;         /* → Device register images */
   ATAPICmdPacket*
            packetCDBPtr;     /* ATAPI packet command block
                                 pointer */
   SInt16   ataReserved3[6];  /* Reserved */

} ATA_ExecIO;
```

Field descriptions

`ataStatusReg` This field contains the last device status register image. See the ATA IDE specification for status register bit definitions.

`ataErrorReg` This field contains the last device error register image. The field is valid only if the error bit (bit 0) of the `Status` register is set. See the ATA IDE specification for error register bit definitions.

`ataReserved` Reserved. All reserved fields are set to 0 for future compatibility.

`BlindTxSize` This field specifies the maximum number of bytes that can be transferred for each interrupt or detection of a data request. Bytes are transferred in blind mode (no byte level handshake). Once an interrupt or a data request condition is detected, the ATA Manager transfers bytes from or to the selected device up to the number specified in the field. The typical number is 512 bytes.

`ioBuffer` This field contains the host buffer address for the number of bytes specified in the `ioReqCount` field. Upon returning, the `ioBuffer` field is updated to reflect data transfers. When the `SGType` bits of the `ataFlags` field are set, the `ioBuffer` field points to a scatter gather list. The scatter gather list consists of a series of `IOBlk` entries, as defined below:

```
struct IOBlk
{
   UInt8    *ioBuffer;
```

```
   UInt32   ioReqCount;
};
typedef struct IOBlk IOBlk;
```

`ioReqCount` This field contains the number of bytes to be transferred either from or to the buffer specified in `ioBuffer`. Upon returning, the `ioReqCount` field is updated to reflect data transfers (0 if successful; otherwise, the number of bytes that remained to be transferred prior to the error condition). When the `SGType` bits of the `ataFlags` field are set, the `ioReqCount` field contains the number of scatter gather entries in the list pointed to by the `ioBuffer` field.

`ataActualTxCnt` This field contains the total number of bytes transferred for this request. The field is not currently supported.

`ataReserved2` This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.

`RegBlock` This field contains the IDE device register image structure. Values contained in this structure are written out to the device during the command delivery state. The caller must provide the image prior to calling the ATA Manager. The IDE device register image structure is defined as follows:

```
typedef     struct     /* Device register images */
{   UInt8   Features;  /* → Features register
                             image */
    UInt8   Count;     /* ↔ Sector count */
    UInt8   Sector;    /* ↔ Sector start/finish */
    UInt8   Reserved;  /* Reserved */
    UInt16  Cylinder;  /* ↔ Cylinder 68000 format */
    UInt8   SDH;       /* ↔ SDH register image */
    UInt8   Command;   /* → Command register image */
} Device_PB;
```

`packetCDBPtr` This field contains the packet pointer for ATAPI. The current version of the ATA Manager does not support the ATAPI protocol. For ATA commands, the `packetCDBPtr` field should contain 0 in order to ensure compatibility in the future.

`ataReserved3[6]` These fields are reserved. To ensure future compatibility, all reserved fields should be set to 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified logical drive number does not exist
`AT_AbortErr`	–1780	Command aborted bit set in error register
`AT_RecalErr`	–1781	Track 0 not found bit set in error register

AT_WrFltErr	–1782	Write fault bit set in status register
AT_ASeekErr	–1783	Seek complete bit not set upon completion
AT_UncDataErr	–1784	Uncorrected data bit set in error register
AT_CorDataErr	–1785	Data corrected bit set in status register
AT_BadBlkErr	–1786	Bad block bit set in error register
AT_DMarkErr	–1787	Data mark not found bit set in error register
AT_IDNFErr	–1788	ID not found bit set in error register
ATAMgrNotInitialized	–1802	ATA Manager not initialized
ATAPBInvalid	–1803	Invalid device base address detected (= 0)
ATATransTimeOut	–1806	Timeout: transaction timeout detected
ATAReqInProg	–1807	I/O channel in use—cannot proceed
ATAUnknownState	–1808	Device in unknown state
ATAQLocked	–1809	I/O queue locked—cannot proceed
ATAAbortDueToRst	-1812	Request aborted by Manager due to a device reset command

ATA_MgrInquiry

The `ATA_MgrInquiry` function gets information, such as the version number, about the ATA Manager.

The manager function code for the `ATA_MgrInquiry` function is $90.

The parameter block associated with this function is defined below:

```
typedef         struct          /* IDE inquiry structure */
{
   ataPBHdr                     typedefstructataPBHdr/* ATA
                                Manager parameter block
                                 header structure */

   NumVersion   MgrVersion
   UInt8        MGRPBVers;      /* ← Manager PB version number
                                    supported */
   UInt8        Reserved1;      /* Reserved */
   UInt16       ataBusCnt;      /* ← Number of ATA buses in
                                    system */
   UInt16       ataDevCnt;      /* ← Number of ATA devices
                                    detected */
   UInt8        ataMaxMode;     /* ← Maximum I/O speed mode */
   UInt8        Reserved2;      /* Reserved */
```

```
   UInt16       IOClkResolution; /* ← I/O clock in nsec */
   UInt16       Reserved[17];    /* Reserved */
} ATA_MgrInquiry;
```

Field descriptions

`ataPBHdr`	Header structure for the ATA Manager parameter block.
`MgrVersion`	Upon return, this field contains the version number of the ATA Manager.
`MGRPBVers`	This field contains the number corresponding to the latest version of the parameter block supported. A client may use any parameter block definition up to this version.
`Reserved`	Reserved. All reserved fields are set to 0 for future compatibility.
`ataBusCnt`	Upon return, this field contains the total number of ATA buses in the system. The field contains a zero if the ATA Manager has not been initialized.
`ataDevCnt`	Upon return, this field contains the total number of ATA devices detected on all ATA buses. The current architecture allows only one device per bus. This field will contain a zero if the ATA Manager has not been initialized.
`ataMaxMode`	This field specifies the maximum I/O speed mode that the ATA Manager supports. Refer to the ATA IDE specification for information on mode timing.
`IOClkResolution`	This field contains the I/O clock resolution in nanoseconds. The current implementation does not support this field and returns 0.
`Reserved[17]`	This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred

ATA_BusInquiry

The `ATA_BusInquiry` function gets information about a specific ATA bus. This function is provided for possible future expansion of the Macintosh ATA architecture.

The manager function code for the `ATABusInquiry` function is $03.

The parameter block associated with this function is defined below:

```
typedef struct                  /* IDE bus inquiry structure */
{
  ataPBHdr                      /* ATA Manager parameter block
                                   header structure */
```

```
  UInt16     ataEngineCount;     /* ← TBD; zero for now */
  UInt16     ataReserved;        /* Reserved */
  UInt32     ataDataTypes;       /* ← TBD; zero for now */
  UInt16     ataIOpbSize;        /* ← Size of ATA I/O PB */
  UInt16     ataMaxIOpbSize;     /* ← TBD; zero for now */
  UInt32     ataFeatureFlags;    /* ← TBD */
  UInt8      ataVersionNum;      /* ← HBA Version number */
  UInt8      ataHBAInquiry;      /* ← TBD; zero for now */
  UInt16     ataReserved2;       /* Reserved */
  UInt32     ataHBAPrivPtr;      /* ← Ptr to HBA private data */
  UInt32     ataHBAPrivSize;     /* ← Size of HBA private data */
  UInt32     ataAsyncFlags;      /* ← Capability for callback */
  UInt32     ataReserved3[4];    /* Reserved */
  UInt32     ataReserved4;       /* Reserved */
  SInt8      ataReserved5[16];   /* TBD */
  SInt8      ataHBAVendor[16];   /* ← HBA Vendor ID */
  SInt8      ataContrlFamily[16];/* ← Family of ATA controller */
  SInt8      ataContrlType[16];  /* ← Controller model number */
  SInt8      ataXPTversion[4];   /* ← Version number of XPT */
  SInt8      ataReserved6[4];    /* Reserved */
  SInt8      ataHBAversion[4];   /* ← Version number of HBA */
  UInt8      ataHBAslotType;     /* ← Type of slot */
  UInt8      ataHBAslotNum;      /* ← Slot number of the HBA */
  UInt16     ataReserved7;       /* Reserved */
  UInt32     ataReserved8;       /* Reserved */
} ATA_BusInquiry;
```

Field descriptions

`ataPBHdr` Header structure for the ATA Manager parameter block.

`ataEngineCount` This field is currently set to 0.

`ataReserved` Reserved. All reserved fields are set to 0.

`ataDataTypes` Not supported by current ATA architecture. Returns a bit map of data types supported by this HBA. The data types are numbered from 0 to 30; 0 through 15 are reserved for Apple definition and 16 through 30 are available for vendor use. Returns 0.

`ataIOpbSize` This field contains the size of the I/O parameter block supported.

`ataMaxIOpbSize` This field specifies the maximum I/O size for the HBA. This field is not currently supported and returns 0.

`ataFeatureFlags` This field specifies supported features. It is not supported and returns a value of 0.

`ataVersionNum` The version number of the HBA is returned. The current version returns a value of 1.

`ataHBAInquiry` Reserved.

`ataHBAPrivPtr`	This field contains a pointer to the HBA's private data area. It is not supported and returns a value of 0.
`ataHBAPrivSize`	This field contains the byte size of the HBA's private data area. It is not supported and returns a value of 0.
`ataAsyncFlags`	These flags indicate which types of asynchronous events the HBA is capable of generating. It is not supported and returns a value of 0.
`ataHBAVendor`	This field contains the vendor ID of the HBA. This is an ASCII text field. It is not supported.
`ataContrlFamily`	Reserved.
`ataContrlType`	This field identifies the specific type of ATA controller. It is not supported and returns a value of 0.
`ataXPTversion`	Reserved.
`ataHBAversion`	This field specifies the version of the HBA. This field is not supported and returns a value of 0.
`ataHBAslotType`	This field specifies the type of slot. It is not supported and returns a value of 0.
`ataHBAslotNum`	This field specifies the slot number of the HBA. It is not supported and returns a value of 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`ATAMgrNotInitialized`	–1802	ATA Manager not initialized

ATA_QRelease

The `ATA_QRelease` function releases the frozen I/O queue of the selected device.

When the ATA Manager detects an I/O error and the `QLockOnError` bit of the parameter block is set for the request, the ATA Manager freezes the queue for the selected device. No pending or new requests are processed or receive status until the queue is released through the `ATA_QRelease` function. Only those requests with the `Immediate` bit set in the `ATAFlags` field of the `ataPBHdr` parameter block are processed. Consequently, for the ATA I/O queue release command to be processed, it must be issued with the `Immediate` bit set in the parameter block. An ATA I/O queue release command issued while the queue is not frozen returns the `noErr` status.

The manager function code for the `ATA_QRelease` function is $04.

There are no additional function-specific variations on `ataPBHdr` for this function.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist
`ATAMgrNotInitialized`	–1802	ATA Manager not initialized

ATA_NOP

The ATA_NOP function performs no operation across the interface and does not change the state of either the manager or the device. It returns noErr if the drive number is valid.

The manager function code for the ATA_NOP function is $00.

There are no additional function-specific variations on ataPBHdr for this function.

RESULT CODES

noErr	0	Successful completion, no error occurred
nsDrvErr	–56	Specified drive does not exist

ATA_Abort

The ATA_Abort function terminates a specified queued I/O request. This function applies to asynchronous I/O requests only. The ATA_Abort function searches through the I/O queue associated with the selected device and aborts the matching I/O request. The current implementation does not abort if the found request is in progress. If the specified I/O request is not found or has started processing, an ATAUnableToAbort status is returned. If aborted, the ATAReqAborted status is returned.

It is up to the application that called the ATA_Abort function to clean up the aborted request. Clean up includes deallocating the parameter block and reporting to the operating system.

The manager function code for the ATA_Abort function is $10.

The parameter block associated with this function is defined as follows:

```
typedef      struct         /* IDE abort structure */
{
  ataPBHdr                  /* ATA Manager parameter block
                               header structure */

  ATA_PB*    AbortPB        /* Address of the parameter block of
                               the function to be aborted */
  UInt16     Reserved       /* Reserved */
} ATA_Abort;
```

Field descriptions

ataPBHdr	Header structure for the ATA Manager parameter block.
AbortPB	This field contains the address of the I/O parameter block to be aborted.

`Reserved`	This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist
`ATAMgrNotInitialized`	–1802	ATA Manager not initialized
`ATAReqAborted`	–1810	The request was aborted
`ATAUnableToAbort`	–1811	Request to abort could not be honored

ATA_RegAccess

The `ATA_RegAccess` function enables access to a particular device register of a selected device. This function is used for diagnostic and error recovery processes.

The manager function code for the `ATA_RegAccess` function is $12.

The parameter block associated with this function is defined below:

```
typedef        struct            /* Register access structure */
{
   ataPBHdr                      /* ATA Manager parameter block
                                    header structure */

   UInt16      RegSelect;;       /* → Device register selector */
   UInt8       RegValue;         /* ↔ Register value
                                    to read or to be written */
   UInt8       Reserved;         /* ↔ Used for data register
                                    (LSB) only */
   UInt8       Reserved[22]      /* Reserved */
} ATA_RegAccess;
```

Field descriptions

`ataPBHdr`	Header structure for the ATA Manager parameter block.
`RegSelect`	This field specifies which of the device registers to access. The selectors for the registers supported by the `ATA_RegAccess` function are listed in Table 6-3.
`RegValue`	This field represents the value to be written (`IDEioDirection = 01b`) or the value read from the selected register (`ATAioDirection = 10` binary). For `DataReg`, it is assumed this field is a 16-bit field; for other registers, an 8-bit field.

`Reserved[22]` This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.

Table 6-3 IDE register selectors

Selector name	Selector	Register description
`DataReg`	0	Data register (16-bit access only)
`ErrorReg`	1	Error register (R) or features register (W)
`SecCntReg`	2	Sector count register
`SecNumReg`	3	Sector number register
`CylLoReg`	4	Cylinder low register
`CylHiReg`	5	Cylinder high register
`SDHReg`	6	SDH register
`StatusReg` `CmdReg`	7	Status register (R) or command register (W)
`AltStatus` `DevCntr`	14	Alternate status (R) or device control (W)

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist
`ATATransTimeOut`	-1806	Timeout condition detected

ATA_Identify

The `ATA_Identify` function returns the device identification data from the selected device. The identification data contains information necessary to perform I/O accesses to the device. Refer to the ATA IDE specification for the format and the information description provided by the data.

The manager function code for the `ATA_Identify` function is $13.

The parameter block associated with this function is defined below:

```
typedef     struct
{
  ataPBHdr
  UInt16    Reserved1[4];        /* Reserved */
  UInt8     *DataBuf;            /* ↔ Buffer for the data */
  UInt16    Reserved2[18];       /* Reserved */
} ATA_Identify;
```

Field descriptions

`ataPBHdr`	Header structure for the ATA Manager parameter block.
`Reserved1[4],`	Reserved field. To ensure compatibility, all reserved fields should be set to 0.
`DataBuf`	A pointer to the data buffer for the device identify data. The length of the buffer must be at least 512 bytes.
`Reserved2[18]`	Reserved field. To ensure compatibility, all reserved fields should be set to 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist
`ATATransTimeout`	-1806	Timeout condition detected

ATA_ResetBus

The `ATA_ResetBus` function performs a soft reset operation to the selected IDE bus. The IDE interface does not provide a way to reset individual units on the bus. Consequently, all devices on the bus will be reset.

IMPORTANT

This function should be used with caution since it may terminate any active requests to devices on the bus. ▲

The manager function code for the `ATA_ResetBus` function is $11.

The parameter block associated with this function is defined below:

```
typedef     struct         /* IDE reset structure */
{
  ataPBHdr                 /* ATA Manager parameter block
                              header structure */

  UInt8     Status;        /* ← Last ATA status register image */
  UInt8     Reserved;      /* Reserved */
  UInt16    Reserved[23];  /* Reserved */
} ATA_ResetBus;
```

Field descriptions

`ataPBHdr`	Header structure for the ATA Manager parameter block.
`Status`	This field contains the last device status register image following the bus reset. See the ATA IDE specification for definitions of the status register bits.
`Reserved[23]`	This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist
`ATATransTimeout`	-1806	Timeout condition detected

ATA_DrvrRegister

The `ATA_DrvrRegister` function registers the driver reference number passed in for the selected drive. The function does not check for the existence of another driver.

The manager function code for the `ATA_DrvrRegister` function is $85.

The parameter block associated with this function is defined below:

```
typedef       struct             /* Driver register structure */
{
   ataPBHdr                      /* ATA Manager parameter block
                                    header structure */

   SInt16     drvrRefNum;        /* → Driver reference number */
   UInt16     FlagReserved;      /* Reserved. Should be set to 0 */
   UInt16     deviceNextID;      /* Not used */
   SInt16     Reserved[21];      /* Reserved */
} ATA_DrvrRegister;
```

Field descriptions

`ataPBHdr`	Header structure for the ATA Manager parameter block.
`drvrRefNum`	This field specifies the driver reference number to be registered. This value must be less than 0 to be valid.
`FlagReserved`	Reserved.
`deviceNextID`	Not used by this function.
`Reserved[21]`	This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist

ATA_DrvrDeregister

The `ATA_DrvrDeRegister` function deregisters the driver reference number passed in for the selected drive. After successful completion of this function, the driver reference

number for the drive is set to 0, which indicates that there is no driver in control of this device.

The manager function code for the `ATA_DrvrDeRegister` function is $87.

There are no additional function-specific variations on `ataPBHdr` for this function.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist

ATA_FindRefNum

The `ATA_FindRefNum` function allows an application to determine whether a driver has been installed for a given device. You pass in a device ID and the function returns the current driver reference number registered for the given device. A value of 0 indicates that no driver has been registered. The `deviceNextID` field contains a device ID of the next device in the list. The end of the list is indicated with a value of `0xFF`.

To create a list of all drivers for the attached devices, pass in `0xFFFF` for `deviceID`. This causes `deviceNextID` to be filled with the first device in the list. Each successive driver can be found by moving the value returned in `deviceNextID` into `deviceID` until the function returns `0xFF` in `deviceNextID`, which indicates the end of the list.

The manager function code for the `ATA_FindRefNum` function is $86.

The parameter block associated with this function is defined as follows:

```
typedef      struct
{
   ataPBHdr
   SInt16     drvrRefNum;       /* ← Contains the driver refNum */
   UInt16     FlagReserved;     /* Reserved. Should be set to 0 */
   UInt16     deviceNextID;     /* ← Contains the next drive ID */
   SInt16     Reserved[21];     /* Reserved */
} ATA_FindRefNum;
```

Field descriptions

`ataPBHdr`	Header structure for the ATA Manager parameter block.
`drvrRefNum`	Upon return, this field contains the reference number for the device specified in the `deviceID` field of the `ataPBHdr` data.
`FlagReserved`	This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.
`deviceNextID`	Upon return, this field contains the `deviceID` of the next device on the list. A value of 0xFF indicates the end of the driver list.

`Reserved[21]`	This field is reserved. To ensure future compatibility, all reserved fields should be set to 0.

RESULT CODES

`noErr`	0	Successful completion, no error occurred
`nsDrvErr`	–56	Specified drive does not exist

Result Code Summary

This section provides a summary of the IDE hard disk result codes.

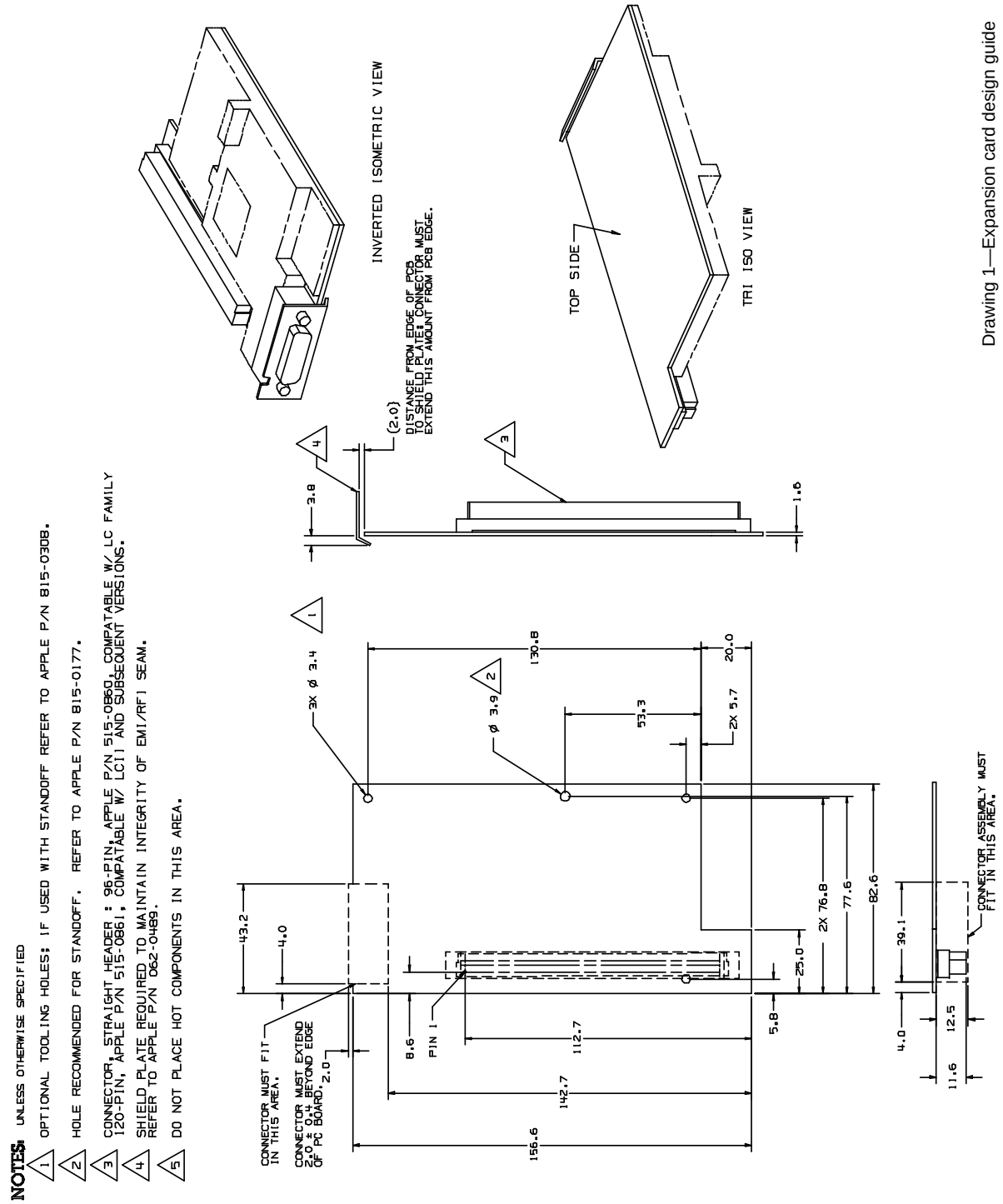
`noErr`	0	No error detected on the request operation.
`controlErr`	–17	Unimplemented control function. Requested control operation could not complete.
`statusErr`	–18	Unimplemented status function. Requested status operation could not complete.
`openErr`	–23	Unimplemented open function. Open operation could not complete.
`ioErr`	–36	An I/O error detected while processing the request.
`paramErr`	–50	Error in parameter block.
`nsDrvErr`	–56	No such drive. No device attached to the specified port.
`AT_AbortErr`	–1780	Command aborted by drive. Unsupported command.
`AT_RecalErr`	–1781	Recalibrate failure detected by device.
`AT_WrFltErr`	–1782	Write fault detected by device.
`AT_SeekErr`	–1783	Seek error detected by device.
`AT_UncDataErr`	–1784	Unable to correct data (possibly bad data).
`AT_CorDataErr`	–1785	Data was corrected (good data)—notification.
`AT_BadBlkErr`	–1786	Bad block detected by device.
`AT_DMarkErr`	–1787	Data mark not found reported by device.
`AT_IDNFErr`	–1788	Sector ID not found; error reported by device.
`AT_NRdyErr`	–1791	Drive ready condition not detected.
`DRVRCantAllocate`	–1793	Driver memory allocation error during driver open.
`ATAInitFail`	–1795	ATA Manager initialization failed.
`ATABufFail`	–1796	Power-on device test failed. Device failure detected. Interface communication error.
`ATAMgrNotInitialized`	–1802	ATA Manager has not been initialized. The request function cannot be performed until initialized.

`ATAPBInvalid`	–1803	Invalid IDE port address detected (ATA Manager initialization problem).
`ATATransTimeOut`	–1806	Timeout condition detected. The operation has not completed within the user specified time limit.
`ATAReqInProg`	–1807	Device busy; the device on the port is busy processing another command.
`ATAUnknownState`	–1808	The device status register reflects an unknown state.
`ATAQLocked`	–1809	The I/O queue for the port is locked owing to a previous I/O error (and must be unlocked prior to continuing).
`ATAReqAborted`	–1810	The I/O queue entry was aborted owing to an abort command.
`ATAUnableToAbort`	–1811	The I/O queue entry could not be aborted. Too late to abort or the entry not found.
`ATAAbortDueToRst`	-1812	Request aborted by Manager due to a device reset command.

I/O Expansion Card Mechanical Drawings

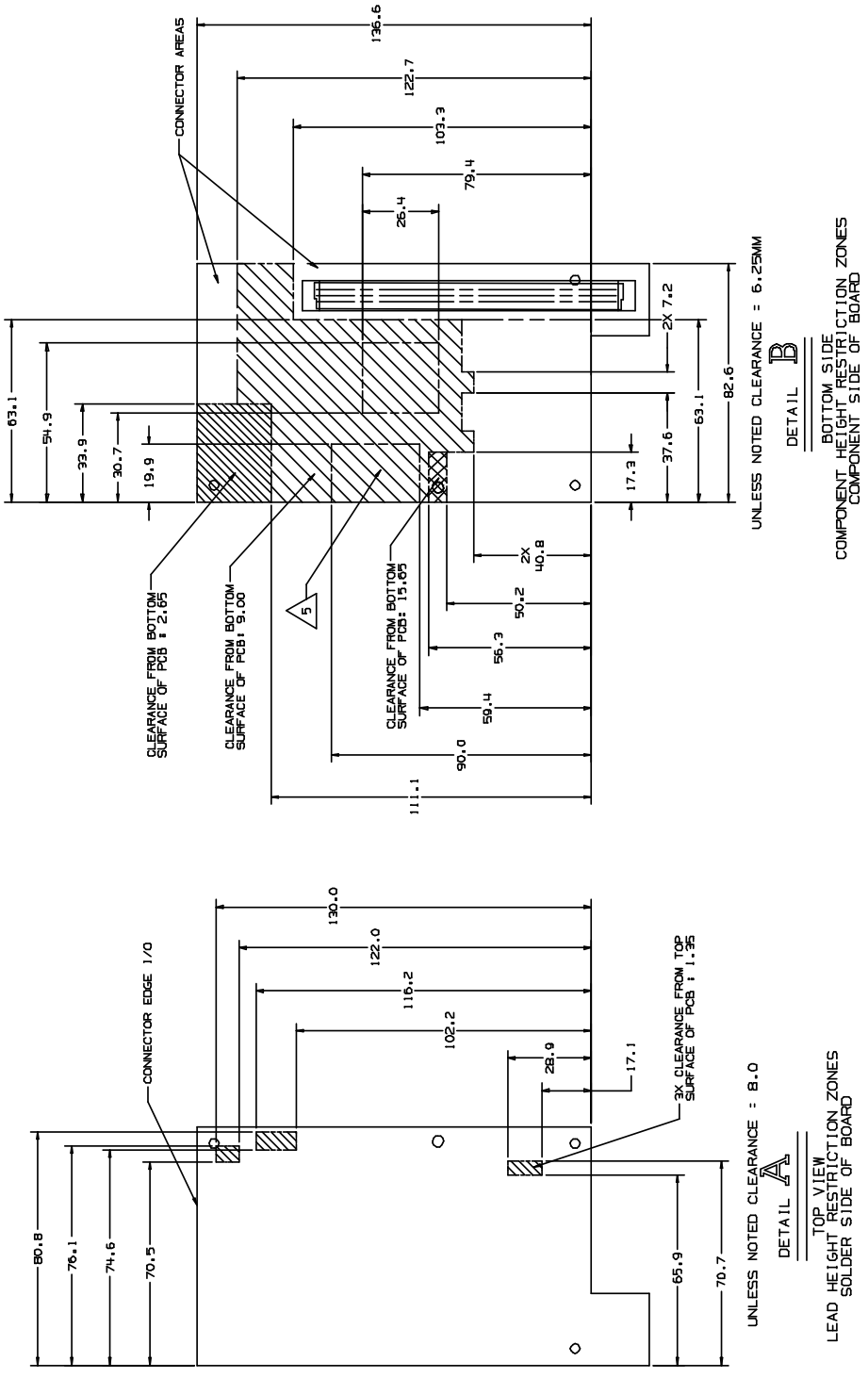
This appendix contains mechanical drawings for the I/O expansion card used in the Power Macintosh 5260 computer. Drawing 1 is a design guide showing the dimensions of the expansion card, part number 062-0487-B. Drawing 2 shows the maximum component height permitted on the card. Drawing 3 is an assembly guide for the I/O expansion card and shows how the I/O expansion card is installed on the main logic board.

I/O Expansion Card Mechanical Drawings



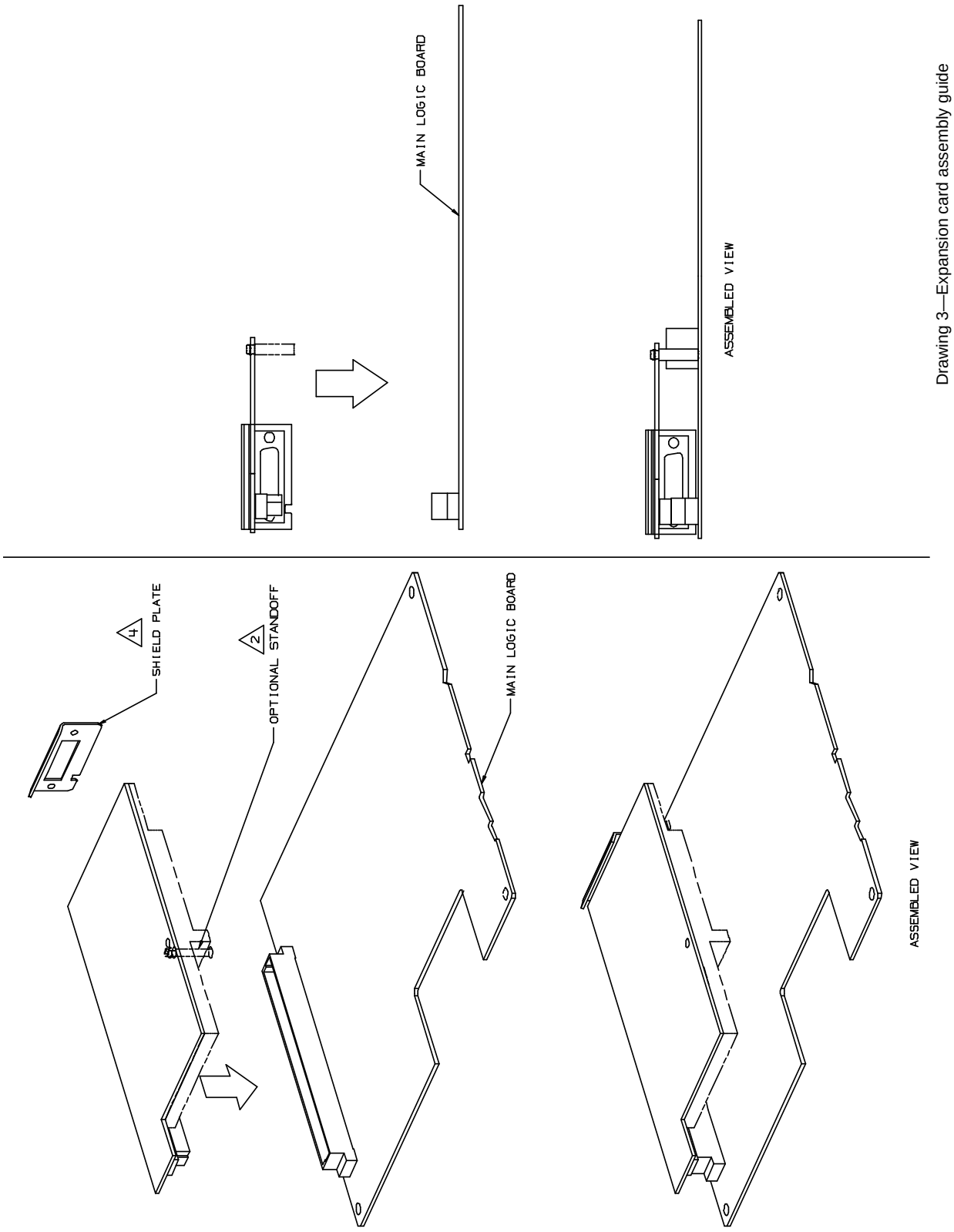
Drawing 1—Expansion card design guide

I/O Expansion Card Mechanical Drawings



Drawing 2—Expansion card component height restrictions

I/O Expansion Card Mechanical Drawings



Drawing 3—Expansion card assembly guide

Glossary

alignment pragma This is an instruction to a C compiler to use a special layout for its data structures. Typically, Power PC C compilers and 68K compilers use different formats. If you want to write PPC code that can call 68K code (for example for an application calling the operating system) you need to get the compiler to use 68K-style data structures. You do this with an alignment pragma.

byte lane An eight-bit channel of a data bridge that passes individual bytes of data.

chrominance This is a component of the picture information you see on your computer monitor. It contains only the color and no other information. The other component is luminance.

digital YUV format YUV is a data format for each pixel on the color monitor in which color is encoded by values calculated from the pixel's native read, green, and blue components. The digital YUV format used in the Power Macintosh 5260 computer provides a clearer TV picture than the RGB format used in the Macintosh Quadra 630 and LC 630 computer. The YUV format accomplishes this in the way bits are allocated. The RGB format generally used is a 16-bit format with 5 bits each for red, green, and blue. The remaining bit is unused. The YUV format used in the Power Macintosh 5260 computer is also a 16-bit format, with 8 bits for the Y (luminance) channel. The U (chrominance) and V channels share the other 8 bits by multiplexing. The YUV format looks clearer because it carries more levels of luminance information.

gamma correction This is a technique that adjusts the gamma curve to compensate for the loss of detail in dark objects displayed on the computer monitor.

gamma curve This is the relationship between color intensity (chrominance) and light (luminance) displayed on the computer monitor. With a low gamma curve, colors are washed out. With a high gamma curve, colors have more contrast.

luminance This is a component of the picture information you see on your computer monitor. It is responsible for detail, shapes, and shading. The other component is chrominance.

Index

A

B

C

D

E

F

G

H

I, J

K

L

M

N

O

P, Q

R

S

T, U

V, W, X

Y, Z

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 Select 360. Line art was created using Adobe™ Illustrator. 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
Joyce D. Mann

COPY EDITOR
Wendy Krafft

ILLUSTRATOR
Sandee Karr

PRODUCTION EDITOR
Alex Solinski

Acknowledgements to

Prabir Sarkar, Dennis Pak, Rich Schnell, Kenneth Lee, Ng Buay Hock, and Jonathan Quinn

Special thanks to Rolly Reed