PowerBook 520 and 520c and PowerBook 540 and 540c Computers
Contents

Figures and Tables vii

Preface

About This Developer Note ix

Contents of This Note ix
Supplementary Documents ix
Conventions and Abbreviations x
Typographical Conventions xi
Standard Abbreviations xi

Chapter 1

Introduction 1

Features 2
Appearance 3
Configurations 5
Peripheral Devices 5
Compatibility Issues 6
RAM Expansion Cards 6
Number of Colors 6
Video Mirror Mode 6
MacsBug Version 6.2.2 7
Sound Sample Rates 7
Power Manager Control Panels 7

Chapter 2

Architecture 9

Processor / Memory Subsystem 10
Microprocessor 10
RAM 10
ROM 12
Pratt Memory Controller IC 12
Input / Output Subsystem 12
Whitney Peripheral Support IC 13
Power Manager IC 13
Display Controller IC 14
Combo IC 14
Singer IC 14
Keystone Video Controller IC 14
Ariel Video Output IC 14
Chapter 3  
I/O Features  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Hard Disk Drive</td>
<td>16</td>
</tr>
<tr>
<td>Hard Disk Power Budget</td>
<td>16</td>
</tr>
<tr>
<td>Hard Disk Carrier Bracket</td>
<td>17</td>
</tr>
<tr>
<td>Hard Disk Mounting Envelope</td>
<td>18</td>
</tr>
<tr>
<td>Hard Disk Connector</td>
<td>18</td>
</tr>
<tr>
<td>Signals on Connector J1</td>
<td>19</td>
</tr>
<tr>
<td>Signals on Connector J2</td>
<td>20</td>
</tr>
<tr>
<td>Internal Floppy Disk Drive</td>
<td>21</td>
</tr>
<tr>
<td>Trackpad</td>
<td>22</td>
</tr>
<tr>
<td>Keyboard</td>
<td>22</td>
</tr>
<tr>
<td>Flat Panel Displays</td>
<td>23</td>
</tr>
<tr>
<td>Flat Panel Display Circuitry</td>
<td>24</td>
</tr>
<tr>
<td>Number of Colors</td>
<td>24</td>
</tr>
<tr>
<td>External Video Port</td>
<td>24</td>
</tr>
<tr>
<td>Video Mirroring</td>
<td>25</td>
</tr>
<tr>
<td>Video Monitors</td>
<td>25</td>
</tr>
<tr>
<td>External Video Connector</td>
<td>26</td>
</tr>
<tr>
<td>Monitor Sense Codes</td>
<td>26</td>
</tr>
<tr>
<td>Serial Port</td>
<td>28</td>
</tr>
<tr>
<td>SCSI Port</td>
<td>28</td>
</tr>
<tr>
<td>SCSI Disk Mode</td>
<td>28</td>
</tr>
<tr>
<td>SCSI Connectors</td>
<td>29</td>
</tr>
<tr>
<td>Ethernet Port</td>
<td>30</td>
</tr>
<tr>
<td>ADB Port</td>
<td>31</td>
</tr>
<tr>
<td>Sound System</td>
<td>31</td>
</tr>
</tbody>
</table>

Chapter 4  
Expansion Modules  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Expansion Card</td>
<td>34</td>
</tr>
<tr>
<td>Electrical Design Guidelines for the RAM Expansion Card</td>
<td>34</td>
</tr>
<tr>
<td>Connector Pin Assignments</td>
<td>35</td>
</tr>
<tr>
<td>Signal Descriptions</td>
<td>36</td>
</tr>
<tr>
<td>Address Multiplexing</td>
<td>36</td>
</tr>
<tr>
<td>Banks of DRAM</td>
<td>38</td>
</tr>
<tr>
<td>DRAM Device Requirements</td>
<td>39</td>
</tr>
<tr>
<td>Expansion Card Electrical Limits</td>
<td>39</td>
</tr>
<tr>
<td>Mechanical Design Guidelines for the RAM Expansion Card</td>
<td>40</td>
</tr>
<tr>
<td>RAM Expansion Connector</td>
<td>40</td>
</tr>
<tr>
<td>RAM Expansion Card Design</td>
<td>40</td>
</tr>
<tr>
<td>PDS Expansion Card</td>
<td>42</td>
</tr>
<tr>
<td>User Installation of the PDS Card</td>
<td>42</td>
</tr>
<tr>
<td>Electrical Description of the PDS</td>
<td>43</td>
</tr>
<tr>
<td>Signals on the PDS Connector</td>
<td>43</td>
</tr>
<tr>
<td>MC68030-Compatible Signals on the PDS Connector</td>
<td>45</td>
</tr>
</tbody>
</table>
Figures and Tables

Chapter 1
Introduction 1

Figure 1-1  Front view of the PowerBook 540 computer 4
Figure 1-2  Back view of the PowerBook 540 computer 4

Table 1-1  Models and configurations 5

Chapter 2
Architecture 9

Figure 2-1  Block diagram of the PowerBook 520 and 540 computers 11

Chapter 3
I/O Features 15

Figure 3-1  Hard disk carrier bracket 17
Figure 3-2  Hard disk drive dimensions 18
Figure 3-3  Pins on the hard disk connector 19
Figure 3-4  PowerBook 520 and 540 keyboard, U.S. layout 22
Figure 3-5  PowerBook 520 and 540 keyboard, ISO layout 23
Figure 3-6  Pin assignments on the video connectors 26

Table 3-1  Hard disk power budget 17
Table 3-2  Signals on connector J1 19
Table 3-3  SCSI ID encoding 19
Table 3-4  Signals on the internal hard disk connector (J2) 20
Table 3-5  Signals on the internal floppy disk connector 21
Table 3-6  Types of flat panel displays 23
Table 3-7  Video monitors and modes 25
Table 3-8  Signals on the video connector 27
Table 3-9  Monitor sense codes 27
Table 3-10  Serial port signals 28
Table 3-11  Internal and external HDI-30 SCSI connector signals 29
Table 3-12  Pin assignments on the Ethernet port 30
Table 3-13  ADB connector pin assignments 31

Chapter 4
Expansion Modules 33

Figure 4-1  Dimensions of the RAM expansion card 40
Figure 4-2  Restricted areas on the top of the RAM expansion card 41
Figure 4-3  Restricted areas on the bottom of the RAM expansion card 41
Figure 4-4  Edge view of the RAM expansion card 42
Figure 4-5  Recommended connection logic on the PDS card 50
Figure 4-6  Recommended inrush current limiting circuit 51
Figure 4-7  Recommended power control circuit 51
Figure 4-8  Generating the card select signal 52
Figure 4-9 The PDS card  53
Figure 4-10 Section through the PDS card connector  53
Figure 4-11 Dimensions of the PDS card  54
Figure 4-12 Restricted areas on the top of the PDS card  55
Figure 4-13 Restricted areas on bottom of the PDS card  56
Figure 4-14 Top view of the modem card  63
Figure 4-15 Restricted areas on the top of the modem card  63
Figure 4-16 Restricted areas on the bottom of the modem card  64
Figure 4-17 A simple DAA card  66
Figure 4-18 Top view of a simple DAA card  67
Figure 4-19 Bottom view of a simple DAA card  68
Figure 4-20 View of the multi-country DAA card  69

Table 4-1 Configurations of RAM banks  34
Table 4-2 Signal assignments on the RAM expansion connector  35
Table 4-3 Descriptions of signals on the RAM expansion connector  37
Table 4-4 Address multiplexing for some typical DRAM devices  38
Table 4-5 Signal assignments on the PDS connector  44
Table 4-6 MC68030-compatible signals on the PDS connector  46
Table 4-7 System support signals on the PDS connector  47
Table 4-8 Power leads on the PDS connector  47
Table 4-9 PDS signal load and drive limits  48
Table 4-10 PDS power budget  49
Table 4-11 Signal assignments on the modem connector  57
Table 4-12 MC68030-compatible signals on the modem connector  59
Table 4-13 System support signals on the modem connector  60
Table 4-14 DAA signals on the modem connector  61
Table 4-15 Signals on the DAA connector  65
Table 4-16 Signals on the DAA connector  65

Chapter 5
Software  71

Figure 5-1 Control strip  75
Figure 5-2 New control panels  77
Figure 5-3 PowerBook control panel in easy mode  78
Figure 5-4 PowerBook control panel in custom mode  78
Figure 5-5 PowerBook Setup control panel  78
Figure 5-6 PowerBook Display control panel  79
Figure 5-7 Trackpad control panel  79
Figure 5-8 Control Strip control panel  80
Figure 5-9 AutoRemounter control panel  80
Figure 5-10 Positioning a bar graph  92
Figure 5-11 Directions of a bar graph  92

Chapter 6
Power Manager Interface  95

Table 6-1 Interface routines and their selector values  98
About This Developer Note

This developer note describes the Macintosh PowerBook 520 and 540 computers, emphasizing the features that are new or different from those of other Macintosh PowerBook computers. 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 “Supplementary Documents.”

Contents of This Note

The information in this note is arranged in six chapters:

- Chapter 1, “Introduction,” introduces the PowerBook 520 and 540 computers and describes their new features.
- Chapter 2, “Architecture,” describes the internal logic of the PowerBook 520 and 540 computers, including the main ICs that appear in the block diagram.
- Chapter 3, “I/O Features,” describes the input/output features, including both the internal I/O devices and the external I/O ports.
- Chapter 4, “Expansion Modules,” describes the expansion features of interest to developers. It includes development guides for the RAM expansion card, the PDS card, and the communications cards.
- Chapter 5, “Software,” describes the new features of the ROM and system software, with the emphasis on software that is specific to this computer.
- Chapter 6, “Power Manager Interface,” describes the application program interface for the Power Manager software.

Supplementary Documents

To supplement the information in this developer note, developers should have copies of the appropriate Motorola reference books for the MC68040 microprocessor. Software developers should have a copy of Motorola’s MC68040 Programmer’s Reference Manual. Hardware developers should have copies of Motorola’s MC68030 User’s Manual, MC68040 User’s Manual, and MC68040 Designer’s Handbook.
Developers should also have copies of the appropriate Apple reference books, including *Inside Macintosh; Guide to the Macintosh Family Hardware*, second edition; and *Designing Cards and Drivers for the Macintosh Family*, third edition. These Apple books are available in technical bookstores and through APDA.

For information about earlier PowerBook models, developers should also have copies of the *Macintosh Classic II, Macintosh PowerBook Family, and Macintosh Quadra Family Developer Notes*; and *Macintosh Developer Notes*, numbers 1 through 5, available on the Developer CD Series, and through APDA.

APDA is Apple’s worldwide source for over three hundred development tools, technical resources, training products, and information for anyone interested in developing applications on Apple platforms. Customers receive the quarterly *APDA Tools Catalog* featuring all current versions of Apple development tools and the most popular third-party development tools. Ordering is easy; there are no membership fees, and application forms are not required for most of the products. APDA offers convenient payment and shipping options, including site licensing.

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

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

Telephone  800-282-2732 (United States)
            800-637-0029 (Canada)
            716-871-6555 (International)

Fax  716-871-6511

AppleLink  APDA
America Online  APDAorder
CompuServe  76666,2405
Internet  APDA@applelink.apple.com

Conventions and Abbreviations

This developer note uses the following typographical conventions and abbreviations.
Typographical Conventions

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

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

Note

A note like this contains information that is of interest but is 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

Standard units of measure used in this note include:

- A  amperes
- dB  decibels
- GB  gigabytes
- Hz  hertz
- K  1024
- KB  kilobytes
- kbps kilobits per second
- kHz kilohertz
- kΩ kilohms
- M  1,048,576
- mA  milliamperes
- MB  megabytes
- MHz megahertz
- mm millimeters
- ms milliseconds
- mV millivolts
- μF microfarads
- ns nanoseconds
- Ω ohms
- pF picofarads
- V volts
- VAC volts alternating current
- VDC volts direct current
- W watts
Other abbreviations used in this note include

$\$n$ hexadecimal value $n$

AC alternating current
ADB Apple Desktop Bus
API application program interface
ASIC application-specific integrated circuit
AUI auxiliary unit interface
BCD binary coded decimal
CAS column address strobe (a memory control signal)
CCFL cold cathode fluorescent lamp
CD compact disc
CLUT color look-up table
CMOS complementary metal oxide semiconductor
CPU central processing unit
CSC color screen controller
DAA data access adapter (a telephone line interface)
DAC digital-to-analog converter
DC direct current
DOS disk operating system
DRAM dynamic RAM
DSP digital signal processor
FIFO first in, first out
FPU floating-point unit
FSTN film supertwist nematic (a type of LCD)
IC integrated circuit
I/O input/output
LCD liquid crystal display
LS TTL low-power Schottky TTL (a standard type of device)
MMU memory management unit
NiCad nickel cadmium
NiMH nickel metal hydride
PCMCIA Personal Computer Memory Card International Association
PDS processor direct slot
PROM programmable read-only memory
PWM pulse width modulation
RAM random-access memory
RAMDAC random-access memory, digital to analog converter
RAS row address strobe
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>red-green-blue (a type of color video system)</td>
</tr>
<tr>
<td>RISC</td>
<td>reduced instruction set computer</td>
</tr>
<tr>
<td>RMS</td>
<td>root-mean-square</td>
</tr>
<tr>
<td>ROM</td>
<td>read-only memory</td>
</tr>
<tr>
<td>SCC</td>
<td>Serial Communications Controller</td>
</tr>
<tr>
<td>SCSI</td>
<td>Small Computer System Interface</td>
</tr>
<tr>
<td>SNR</td>
<td>signal-to-noise ratio</td>
</tr>
<tr>
<td>SVGA</td>
<td>super video graphics adapter</td>
</tr>
<tr>
<td>TFT</td>
<td>thin-film transistor (a type of LCD)</td>
</tr>
<tr>
<td>TTL</td>
<td>transistor-transistor logic (a standard type of device)</td>
</tr>
<tr>
<td>VCC</td>
<td>positive supply voltage (voltage for collectors)</td>
</tr>
<tr>
<td>VGA</td>
<td>video graphics adapter</td>
</tr>
<tr>
<td>VRAM</td>
<td>video RAM</td>
</tr>
</tbody>
</table>
Introduction
CHAPTER 1

Introduction

The Macintosh PowerBook 520 and PowerBook 540 computers are the first of a new
generation of all-in-one notebook computers featuring the powerful Motorola MC68040
microprocessor, color displays, integrated communications architecture, and other
advanced features. Inside the computer’s contoured case are spaces for two rechargeable
batteries, a new high-speed modem, and a processor-direct slot for expansion cards.

The modular design of these new computers offers new opportunities for developers
to expand beyond traditional add-on products by integrating new devices within the
computer’s case. This modular design also makes it easier to upgrade the system with
newer, faster, low-power microprocessors as they become available.

Features

Here is a summary of the major features of the PowerBook 520 and PowerBook 540
computers. Each feature is described more fully later in this note.

- Microprocessor: Motorola MC68LC040 running at 50/25 MHz or 66/33 MHz.

Note

The MC68LC040 uses two processor clocks: one for the system bus
and another, at twice the speed, for the internal circuits. Thus, an
MC68LC040 with a system bus clock of 33 MHz runs its internal
processor at 66 MHz.

- Random-access memory (RAM): built-in memory consists of 4 MB of low-power,
  self-refreshing dynamic RAM (DRAM).

- RAM expansion: support for RAM expansion cards of up to 32 MB, for a total of
  36 MB of RAM.

- Read-only memory (ROM): 2 MB.

- Flat panel display: built-in 640-by-480-pixel LCD with 8-bit capability for 256 colors
  or shades of gray. The PowerBook 540 and 540c have active matrix LCDs; the
  PowerBook 520 and 520c have advanced passive matrix LCDs. Both types of displays
  are backlit by a cold cathode fluorescent lamp (CCFL).

- Video output: 8-bit color video output circuitry supports 256-color display on all
  Apple monitors up to 16 inches in size, 16 shades of gray on the Apple Macintosh
  Portrait Display, and 16-color display on Apple multiscan monitors; with appropriate
  adapter cables, also supports VGA displays and an 800-by-600-pixel SVGA mode.

- Floppy disk: one internal 1.4 MB Apple SuperDrive.

- Hard disk: one internal 2.5-inch SCSI hard disk drive with disk capacity of 240 or 320
  MB. See “Configurations” on page 5.
CHAPTER 1

Introduction

- SCSI disk mode: operates in conjunction with an HDI-30 SCSI Disk Adapter cable to allow users to read and store data on the PowerBook computer’s internal hard disk from another Macintosh computer.
- Networking: built-in Ethernet and LocalTalk network interfaces.
- Expansion: the PowerBook Expansion Bay, a 90-pin processor-direct slot (PDS) inside the left battery compartment.
- Sound: built-in microphone and stereo speakers; microphone jack and headphone jack.
- Modem: internal connector for an optional modem card. The modem card available from Apple is a 14.4kbps fax/data modem with a worldwide DAA interface.
- I/O (input/output): standard Macintosh inputs and outputs, including Ethernet and external video out. The I/O ports are an HDI-30 connector for external SCSI devices, a 14-pin Apple Ethernet AUI port, a 4-pin mini-DIN Apple Desktop Bus (ADB) port, an 8-pin mini-DIN serial port, stereo audio input and output jacks, and a video output connector with adapter for attaching standard Apple video cables.
- Keyboard: full-size keyboard with function keys and power on/off control.
- Trackpad, an integrated flat pad that replaces the trackball used in previous PowerBook computers.
- Batteries: one or two PowerBook Intelligent Batteries. The batteries are 9.6-V, 1.9-ampere-hour NiMH rechargeable batteries with built-in processors that communicate with the computer’s Power Manager to measure battery charge and maximize battery life.
- Power supply: an external wall-mounted recharger/power adapter, included with the computer. The power adapter accepts any worldwide standard voltage from 100–240 VAC at 50–60 Hz. The power adapter can charge two batteries in two hours while the computer is shut down or in sleep mode, or in four hours while it is operating.
- Security connector: a connector on the back panel allows users to attach a security device.
- Weight: 6.3 pounds with one battery installed, 7 pounds with two batteries.
- Size: 11.3 inches wide, 9.5 inches deep, and 2.3 inches high.

Appearance

The PowerBook 520 and 540 models have a new, streamlined case. Figure 1-1 shows a front view of the PowerBook 540 and Figure 1-2 shows a back view.
Introduction

Figure 1-1  Front view of the PowerBook 540 computer

Figure 1-2  Back view of the PowerBook 540 computer
Configurations

The Macintosh PowerBook 520 and 540 computers are available in a total of six configurations, as shown in Table 1-1.

<table>
<thead>
<tr>
<th>Model</th>
<th>Processor speed</th>
<th>Display type</th>
<th>Sizes of RAM and hard disk</th>
<th>Modem included</th>
<th>Number of batteries</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerBook 540c</td>
<td>66/33 MHz</td>
<td>Active matrix color</td>
<td>12 MB/320 MB</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>PowerBook 540c</td>
<td>66/33 MHz</td>
<td>Active matrix color</td>
<td>4 MB/320 MB</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>PowerBook 540</td>
<td>66/33 MHz</td>
<td>Active matrix grayscale</td>
<td>12 MB/240 MB</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>PowerBook 540</td>
<td>66/33 MHz</td>
<td>Active matrix grayscale</td>
<td>4 MB/240 MB</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>PowerBook 520c</td>
<td>50/25 MHz</td>
<td>DualScan color</td>
<td>4 MB/160 MB</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>PowerBook 520</td>
<td>50/25 MHz</td>
<td>Supertwist grayscale</td>
<td>4 MB/160 MB</td>
<td>No</td>
<td>1</td>
</tr>
</tbody>
</table>

Peripheral Devices

In addition to the devices that are included with the computer, several peripheral devices are available separately:

- The PowerBook 8 MB Memory Expansion Kit expands the RAM in the PowerBook 520 and 540 computers to 12 MB.
- The PowerBook Intelligent Battery is available separately as an additional or replacement battery.
- The PowerBook AC Adapter, which comes with the computer, is also available separately. This AC adapter is different from the adapter for the PowerBook 140, 160, and 180 computers. Unlike that earlier model, the new adapter can recharge two internal batteries in just four hours while the computer is running or two hours while the computer is shut down or in sleep mode.
- The PowerBook PCMCIA Expansion Module fits into the PDS connector in the second battery bay and accepts either one Type III PCMCIA card or two stacked Type II cards.
The PowerBook Express Modem II, available in the U.S. and Canada, is a 14.4kbps fax/data modem; it supports the CCITT V.32bis and V.42bis standards as well as industry standard MNP5.

The modem comes with an internal DAA module that is software reconfigurable for use anywhere in the world. In the country where it was purchased, the DAA accepts the standard telephone connector. For travel outside the country of purchase, the user purchases a short cable adapter called the country key for the new country. When in that country, the user plugs the adapter into the DAA module and attaches the local telephone connector.

### Compatibility Issues

The PowerBook 520 and 540 computers incorporate many significant changes from earlier PowerBook designs. This section highlights key areas you should investigate in order to ensure that your hardware and software work properly with the new PowerBook models. These topics are covered in more detail in subsequent sections.

#### RAM Expansion Cards

The RAM expansion card used in the PowerBook 520 and 540 computers has a new design. RAM expansion cards designed for earlier PowerBook models will not work in the new PowerBook models. See the section “RAM Expansion Card” beginning on page 34 for more information.

#### Number of Colors

The controller circuitry for the flat panel display includes a 256-entry color look-up table (CLUT) and is compatible with software that uses QuickDraw and the Palette Manager. The controller supports a palette of 32,768 colors. However, due to the nature of color LCD technology, some colors are dithered or exhibit noticeable flicker. Apple has developed a new gamma table for the color displays that minimizes flicker and optimizes the available colors. For the active matrix color display, the effective range of the CLUT is about 24,000 colors. For the DualScan color display, the effective range of the CLUT is about 4000 colors.

See the section “Flat Panel Displays” beginning on page 23 for more information about the internal display hardware and LCD screen.

#### Video Mirror Mode

The user can select video mirror mode, in which the external monitor mirrors (duplicates) the flat panel display. Applications that write directly to the display buffer may not be compatible with video mirror mode unless they take precautions to ensure that they do not write outside the active portion of the display. That is not a problem for applications that use QuickDraw and never write directly to the display buffer.

See the section “Video Mirroring” on page 25 for more information about video modes.
CHAPTER 1

Introduction

MacsBug Version 6.2.2

MacsBug version 6.2.2 does not work with the PowerBook 520 and 540 computers because it assumes that all MC68040 microprocessors have FPUs. This problem is corrected in newer versions of MacsBug.

Sound Sample Rates

The PowerBook 520 and 540 computers provide sound sample rates of 22.05 kHz, 44.1 kHz, and 48 kHz. The 22.05 kHz sample rate is slower than the sample rate used in some older Macintosh models: 22.254 kHz, which was derived from the 16 MHz system clock. The 22.05 kHz rate was chosen for compatibility with the 44.1 kHz audio CD sample rate.

For sound samples made at the 22.254kHz rate, playback at the 22.05 kHz rate is about 1 percent low in pitch. Furthermore, programs that bypass the Sound Manager and write to the sound FIFOs at the older rate now write too many samples to the FIFOs, causing some samples to be dropped. The result is a degradation in sound quality for those programs. Programs that use the Sound Manager to generate sounds are not affected by the change.

Power Manager Control Panels

Developers have written software that provides expanded Power Manager control for other PowerBook models. That software will not work in the PowerBook 520 and 540 computers.

Until now, third-party software for the Power Manager has worked by reading and writing directly to the Power Manager’s data structures, so it has had to be updated whenever Apple brings out a new model with changes in its Power Manager software. Starting with the PowerBook 520 and 540 computers, the system software includes interface routines for program access to the Power Manager functions, so it is no longer necessary for applications to deal directly with the Power Manager’s data structures. For more information, see Chapter 6, “Power Manager Interface.”.

Developers should not assume that the Power Manager’s data structures are the same on all PowerBook models. In particular, developers should beware of the following assumptions regarding different PowerBook models:

- assuming that timeout values such as the hard disk spindown time reside at the same locations in parameter RAM
- assuming that the power cycling process works the same way or uses the same parameters
- assuming that direct commands to the Power Manager microcontroller are supported on all models
Architecture
The architecture of the PowerBook 520 and 540 computers is partitioned into two subsystems: the processor/memory subsystem and the input/output subsystem. The processor/memory subsystem operates at 25 MHz or 33 MHz on the MC68040 bus. The input/output subsystem operates at 16 MHz on the I/O bus, an MC68030-compatible bus. An Apple custom IC called the Pratt IC acts as the bridge between the two buses, translating MC68040 bus cycles into single or multiple MC68030 bus cycles, as needed.

The block diagram shown in Figure 2-1 shows the two subsystems along with other modules that are attached to them.

**Processor/Memory Subsystem**

The processor/memory subsystem includes the MC68040 microprocessor, main RAM, and ROM. As Figure 2-1 shows, the processor/memory subsystem is a separate module; it occupies a secondary logic board mounted above the main logic board. An optional RAM expansion card attaches to the secondary logic board and becomes part of the processor/memory subsystem.

The modular design of the PowerBook 520 and 540 computers makes it possible to upgrade the system by plugging in a whole new processor/memory subsystem. For example, it will be possible to upgrade to a PowerPC RISC microprocessor when a low-power version becomes available.

**Microprocessor**

The microprocessor used in the PowerBook 520 and 540 computers is the MC68LC040. The MC68LC040 does not contain an FPU (floating-point unit). The MC68LC040 does include a built-in MMU (memory management unit).

The MC68LC040 microprocessor runs at an internal clock rate that is double its external clock rate. With an external clock rate of 25 MHz, the processor’s internal clock runs at 50 MHz; with an external rate of 33 MHz, the internal clock rate is 66 MHz. Table 1-1 on page 5 shows the microprocessor type and speed for each of the new models.

**RAM**

The built-in RAM consists of 4 MB of dynamic RAM (DRAM), supplied by eight 1 M by 4-bit ICs on the secondary logic board. The RAM ICs are low-power, self-refreshing type with an access time of 80 ns.

The RAM array is located in the system memory map between addresses $0000 0000 and $02FF FFFF, except following a system reset or sleep cycle, when it is overlaid by system ROM. As in other Macintosh models, the overlay is removed following the first program access to normal ROM space, making the RAM space accessible.
CHAPTER 2

Architecture

Figure 2-1   Block diagram of the PowerBook 520 and 540 computers

Memory expansion card

Expansion RAM
8–32 MB

Secondary logic board

CPU
MC68040 or
MC68LC040

Pratt
memory controller
and bus adapter

ROM
2 MB

RAM
4 MB

System interconnect

Main logic board

System interconnect

16-bit system bus

Ethernet
ID ROM

Keystone
video controller

Video RAM
512K

Ariel
CLUT/DAC

Video port

PDS
connector

Ethernet

Serial port

External SCSI

Whitney
peripheral
support logic

Power

Manager
(includes
ADB and
KBD
interfaces)

Power

Manager

ADB port

Keyboard

Trackball

Floppy

Sound
out

Sound
in

DAA connector

Modem connector

FPD connector

Internal SCSI connector

US/AWSM DAA card

Modem card

Flat panel display

Internal SCSI hard disk

Processor/Memory Subsystem 11
Architecture

An optional RAM expansion card plugs into a 70-pin connector on the secondary logic board. With the RAM expansion card installed, the processor/memory subsystem supports up to 36 MB of RAM. The RAM card for the PowerBook 520 and 540 computers is not compatible with the RAM card used in earlier PowerBook models. See the section “RAM Expansion Card” beginning on page 34 for details.

ROM

The ROM in the PowerBook 520 and 540 computers is implemented as a 1M by 32-bit array consisting physically of two 1 M by 16-bit ROM ICs with an access time of 150 ns. Although these ICs provide 4 MB of storage, the actual ROM code occupies only the first 2 MB, located in the system memory map between addresses $4000 0000 and $401F FFFF. The ROM data path is 32 bits wide and addressable only as longwords. See Chapter 5, “Software,” for a description of the features of this new ROM.

Like all Macintosh computers, the PowerBook 520 and 540 computers implement an overlay function at startup that maps the system’s ROM address space ($4000 0000–$401F FFFF) to the address space starting at $0000 0000. This overlay allows the MC68040 processor to address a standard default set of exception vectors and trap addresses as well as a starting address at which to begin executing code. As soon as the ROM code makes an access to an address in the normal ROM address space, the hardware turns off the ROM overlay, making RAM accessible starting at address $0000 0000.

Pratt Memory Controller IC

The Pratt IC is a new Apple custom IC that provides RAM and ROM memory control and also acts as the bridge between the MC68040 bus on the secondary logic board and the MC68030 bus on the main logic board. The Pratt IC resides on the secondary logic board and transparently translates MC68040 bus cycles into single or multiple MC68030 dynamically sized bus cycles. Because the Pratt IC seamlessly integrates the two buses, the microprocessor and other bus masters operate as though they were on the same bus.

The Pratt IC supports read, write, and page mode cycles to the RAM. Pratt generates a 2048-byte CAS-before-RAS refresh cycle every 128 ms.

Input/Output Subsystem

The input/output subsystem includes the Whitney IC, the I/O controllers ICs, and the Power Manager IC.
Whitney Peripheral Support IC

The Whitney IC is a new Apple custom IC that provides the interface between the system bus and the I/O bus that supports peripheral devices in the Macintosh PowerBook 520 and 540 computers. The Whitney IC incorporates the following circuitry:

- VIA1 cell
- SWIM II cell
- CPU ID register

The Whitney IC also performs the following functions:

- bus error timing for I/O bus
- bus arbitration for I/O bus
- interrupt prioritization
- VIA2 functions
- sound data buffering
- clock generation
- power control signals

The Whitney IC contains the interface circuitry for the following peripheral ICs:

- Combo, which is a combination of SCC and SCSI ICs
- Singer, the sound codec IC
- the Ethernet ID PROM

The Whitney IC provides the device select signals for the following ICs:

- the Ethernet interface
- the flat panel display controller
- the external video controller

The Whitney IC also provides the power off and reset signals to the peripheral device ICs.

Power Manager IC

The Power Manager IC is a 68HC05 microprocessor that operates with its own RAM and ROM. The Power Manager IC has the following functions:

- controlling sleep, shutdown, and on/off modes
- controlling power to the other ICs
- controlling clock signals to the other ICs
- supporting the ADB
Input/Output Subsystem

- scanning the keyboard
- controlling display brightness
- monitoring battery charge level
- controlling battery charging

Display Controller IC
The flat panel display is controlled by a type 65220 CSC (color support chip) IC manufactured by Chips and Technologies, Inc. The display controller provides the data and control interface to the LCD panel. The CSC IC contains a 256-entry CLUT, RAMDAC, display buffer controller, and flat panel control circuitry. For more information, see “Flat Panel Display Circuitry” on page 24.

Combo IC
The Combo custom IC combines the functions of the SCC (85C30 Serial Communications Controller) and the SCSI controller (53C80) ICs. The SCC portion of the Combo IC supports the serial I/O port. The SCSI controller portion of the Combo IC supports the internal and external SCSI devices.

Singer IC
The Singer custom IC is a 16-bit digital sound codec. It conforms to the IT&T ASCO 2300 Audio-Stereo Code Specification. The Whitney IC maintains sound I/O buffers in main memory for sound samples being send in or out through the Singer IC. For information about the operation of the Singer IC, see the section “Sound System” on page 31.

Keystone Video Controller IC
The Keystone custom IC contains the timing and control circuits for the external video. The Keystone IC has internal registers that the video driver uses to set the horizontal and vertical timing parameters. The Keystone IC also generates the video refresh addresses that are sent to the VRAM.

Ariel Video Output IC
The Ariel custom IC contains the video CLUT (color look-up table) and DAC. The Ariel is pin and software compatible with the AC843 but does not support 24 bits per pixel.
I/O Features
This chapter describes both the built-in I/O devices and the interfaces for external I/O devices. Like the earlier chapters, it emphasizes the similarities and differences between the PowerBook 520 and 540 computers and other PowerBook models.

This chapter describes the following built-in devices and I/O ports:

- internal hard disk drive
- internal floppy disk drive
- built-in trackpad
- built-in keyboard
- built-in flat panel display
- external video port
- serial port
- SCSI port
- Ethernet port
- Apple Desktop Bus (ADB) port
- sound system

### Internal Hard Disk Drive

The PowerBook 520 and 540 computers have an internal 2.5-inch hard disk drive connected to an internal SCSI connector. The following information is provided as a general guideline and is based on the specifications for Apple’s 2.5-inch, 240 MB hard disk drive.

### Hard Disk Power Budget

The hard disk drive operates on 5 VDC ± 5 percent. The maximum ripple voltage the drive must tolerate is 100 mV peak to peak at any frequency from DC to 10 MHz.

Table 3-1 shows the maximum and average current drain and power consumption for various operating modes of the hard disk drive. These limits include 1 kΩ pull-up terminator resistors on all signal lines. All mean specifications are RMS (root-mean-square) values.

**IMPORTANT**

The hard disk drive must not exceed the maximum power specifications given in Table 3-1.
CHAPTER 3

I/O Features

Table 3-1  Hard disk power budget

<table>
<thead>
<tr>
<th>Mode</th>
<th>Current (amps)</th>
<th>Power (watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Maximum</td>
</tr>
<tr>
<td>Startup*</td>
<td>n.a.</td>
<td>1.300</td>
</tr>
<tr>
<td>Operation</td>
<td>0.500</td>
<td>0.600</td>
</tr>
<tr>
<td>Idle</td>
<td>0.300</td>
<td>0.380</td>
</tr>
<tr>
<td>Standby</td>
<td>0.200</td>
<td>0.250</td>
</tr>
<tr>
<td>Shutdown</td>
<td>0.050</td>
<td>0.075</td>
</tr>
</tbody>
</table>

* Maximum values between power on and drive ready (not more than 5 seconds).

Hard Disk Carrier Bracket

The hard disk drive is mounted upside down with a carrier bracket that holds the drive in a well in the computer’s case. Figure 3-1 shows the carrier bracket; the drive is attached beneath the bracket by screws in the four holes in the bracket. When the drive is installed, the three tabs at the rear of the bracket engage cutouts in the case and the two tabs at the front of the bracket are held in place by screws.

Figure 3-1  Hard disk carrier bracket

Apple’s part number for the hard disk carrier bracket is 805-0608. Developers who wish to include a carrier bracket with their hard disk can obtain the bracket from Apple’s vendor, subject to the usual restrictions. For information and authorization for obtaining the bracket, contact Apple’s Developer Support Center.
CHAPTER 3

I/O Features

Hard Disk Mounting Envelope

Figure 3-2 shows the drive and connector envelope requirements for the hard disk drive. The drive and its mating connectors must not extend outside the dimensions shown in the figure.

Figure 3-2  Hard disk drive dimensions

[Diagram showing drive and connector dimensions]

Note: Dimensions are in millimeters and (inches).

Hard Disk Connector

The connector for the internal hard disk is a combination of an 8-pin connector and a 40-pin connector. It is essentially a 50-pin connector with pins 9 and 10 removed. Pins 1–8 constitute the 8-pin connector (J1), and pins 11–50, renumbered as 1–40, constitute the 40-pin connector (J2). Figure 3-3 shows the pin numbering on the combination connector.
I/O Features

The pins on the hard disk connector are 0.5 mm square; pin spacing is 2 mm on centers, horizontal and vertical. Figure 3-2 shows the location of the connector by defining the distance from the edge of the drive to the center line of the nearest pin, which is pin 39.

Figure 3-3  Pins on the hard disk connector

Signals on Connector J1

As shown in Table 3-5, pins 5, 6, and 7 on the 8-pin connector (J1) are used as identification signals to encode the SCSI ID for the internal hard disk. Table 3-5 shows the values of the ID signals corresponding to possible SCSI ID values from 1 to 7.

Table 3-2  Signals on connector J1

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Signal name</th>
<th>Signal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n.c.</td>
<td>No connection</td>
</tr>
<tr>
<td>2</td>
<td>n.c.</td>
<td>No connection</td>
</tr>
<tr>
<td>3</td>
<td>n.c.</td>
<td>No connection</td>
</tr>
<tr>
<td>4</td>
<td>n.c.</td>
<td>No connection</td>
</tr>
<tr>
<td>5</td>
<td>/ID1</td>
<td>Bit 0 of SCSI ID</td>
</tr>
<tr>
<td>6</td>
<td>/ID2</td>
<td>Bit 1 of SCSI ID</td>
</tr>
<tr>
<td>7</td>
<td>/ID4</td>
<td>Bit 2 of SCSI ID</td>
</tr>
<tr>
<td>8</td>
<td>n.c.</td>
<td>No connection</td>
</tr>
</tbody>
</table>

Table 3-3  SCSI ID encoding

<table>
<thead>
<tr>
<th>SCSI ID value</th>
<th>/ID1</th>
<th>/ID2</th>
<th>/ID4</th>
<th>SCSI ID value</th>
<th>/ID1</th>
<th>/ID2</th>
<th>/ID4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>4</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>1</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>5</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>6</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>7</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>
Signals on Connector J2

Table 3-5 shows the signal assignments for the internal hard disk connector (J2). Entries in the table are arranged the same way as the pins on the connector: pin 1 across from pin 2, and so on.

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Signal name</th>
<th>Pin number</th>
<th>Signal name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5 V logic supply</td>
<td>2</td>
<td>+5 V logic supply</td>
</tr>
<tr>
<td>3</td>
<td>Logic return</td>
<td>4</td>
<td>Logic return</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>6</td>
<td>/DB0</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>8</td>
<td>/DB1</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td>10</td>
<td>/DB2</td>
</tr>
<tr>
<td>11</td>
<td>GND</td>
<td>12</td>
<td>/DB3</td>
</tr>
<tr>
<td>13</td>
<td>GND</td>
<td>14</td>
<td>/DB4</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>16</td>
<td>/DB5</td>
</tr>
<tr>
<td>17</td>
<td>Key (n.c.)</td>
<td>18</td>
<td>/DB6</td>
</tr>
<tr>
<td>19</td>
<td>GND</td>
<td>20</td>
<td>/DB7</td>
</tr>
<tr>
<td>21</td>
<td>GND</td>
<td>22</td>
<td>/PARITY</td>
</tr>
<tr>
<td>23</td>
<td>GND</td>
<td>24</td>
<td>TERM_PWR</td>
</tr>
<tr>
<td>25</td>
<td>/ATN</td>
<td>26</td>
<td>/BSY</td>
</tr>
<tr>
<td>27</td>
<td>GND</td>
<td>28</td>
<td>/ACK</td>
</tr>
<tr>
<td>29</td>
<td>/RST</td>
<td>30</td>
<td>/MSG</td>
</tr>
<tr>
<td>31</td>
<td>GND</td>
<td>32</td>
<td>/SEL</td>
</tr>
<tr>
<td>33</td>
<td>/I/O</td>
<td>34</td>
<td>/C/D</td>
</tr>
<tr>
<td>35</td>
<td>GND</td>
<td>36</td>
<td>/REQ</td>
</tr>
<tr>
<td>37</td>
<td>Motor return</td>
<td>38</td>
<td>Motor return</td>
</tr>
<tr>
<td>39</td>
<td>+5 V motor supply</td>
<td>40</td>
<td>+5 V motor supply</td>
</tr>
</tbody>
</table>
Internal Floppy Disk Drive

The floppy disk interface is identical to those in previous PowerBook computers. The internal floppy disk drive is a 1.44 MB Apple SuperDrive. The drive is 15 mm high; it accepts either standard Macintosh disks or DOS floppy disks.

Table 3-5 shows the signal assignments for the internal floppy disk connector.

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Signal name</th>
<th>Signal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>PH0</td>
<td>Phase 0: state control line</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>PH1</td>
<td>Phase 1: state control line</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>6</td>
<td>PH2</td>
<td>Phase 2: state control line</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>8</td>
<td>PH3</td>
<td>Phase 3: register write strobe</td>
</tr>
<tr>
<td>9</td>
<td>/SYS.PWR</td>
<td>System power</td>
</tr>
<tr>
<td>10</td>
<td>/WREQI</td>
<td>Write data request</td>
</tr>
<tr>
<td>11</td>
<td>FD1.+5V/0</td>
<td>+5 V/0 V (awake/sleep)</td>
</tr>
<tr>
<td>12</td>
<td>HDSELI</td>
<td>Head select</td>
</tr>
<tr>
<td>13</td>
<td>FD1.+5V/0</td>
<td>+5 V/0 V (awake/sleep)</td>
</tr>
<tr>
<td>14</td>
<td>/DISK1EN</td>
<td>Drive enable</td>
</tr>
<tr>
<td>15</td>
<td>FD1.+5V/0</td>
<td>+5 V/0 V (awake/sleep)</td>
</tr>
<tr>
<td>16</td>
<td>RD</td>
<td>Read data</td>
</tr>
<tr>
<td>17</td>
<td>FD1.+5V/0</td>
<td>+5 V/0 V (awake/sleep)</td>
</tr>
<tr>
<td>18</td>
<td>WRDATA</td>
<td>Write data</td>
</tr>
<tr>
<td>19</td>
<td>n.c.</td>
<td>Not connected</td>
</tr>
<tr>
<td>20</td>
<td>n.c.</td>
<td>Not connected</td>
</tr>
</tbody>
</table>
CHAPTER 3

I/O Features

Trackpad

For their pointing device, the PowerBook 520 and 540 computers have a trackpad, an integrated flat pad that replaces the trackball used in previous PowerBook computers. The trackpad provides precise cursor positioning in response to motions of the user’s fingertip over the surface of the pad. A single button below the trackpad is used to make selections.

The trackpad is a solid-state device that emulates a mouse by sensing the motions of the user’s finger over its surface and translating those motions into ADB commands. The trackpad is lighter and more durable than the trackball used in earlier PowerBook computers, and it consumes less power.

Also see the section “Trackpad Software” on page 74.

Keyboard

A new keyboard design provides 76 (U.S.) or 77 (ISO) keys, including 12 function keys. Figure 3-4 shows the version of the keyboard used on machines sold in the United States. Figure 3-5 shows the version of the keyboard used on machines sold in countries that require the ISO standard.

Figure 3-4  PowerBook 520 and 540 keyboard, U.S. layout
By removing two screws, the user can lift out the keyboard to obtain access to the internal components and expansion connectors inside the PowerBook 520 and 540 computers.

Flat Panel Displays

The PowerBook 520 and 540 computers have built-in flat panel displays showing 640 by 480 pixels. Four types of flat panel display are used in the different PowerBook models, as shown in Table 3-6. All four types of displays are the same size, 9.5 inches (diagonal); all have a dot pitch of 0.30 mm and are backlit by a cold cathode fluorescent lamp (CCFL). The displays can show up to 8 bits per pixel, which provides 256 colors on color displays or 256 levels of gray on grayscale displays.

<table>
<thead>
<tr>
<th>Model</th>
<th>Display type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerBook 540c</td>
<td>Active matrix color</td>
</tr>
<tr>
<td>PowerBook 540</td>
<td>Active matrix grayscale</td>
</tr>
<tr>
<td>PowerBook 520c</td>
<td>DualScan color</td>
</tr>
<tr>
<td>PowerBook 520</td>
<td>Supertwist grayscale</td>
</tr>
</tbody>
</table>
I/O Features

Types of Displays

Flat-panel displays come in two types: active matrix and passive matrix.
Active matrix displays, also called thin-film transistor (TFT) displays, have a driving transistor for each individual pixel. The driving transistors give active matrix displays high contrast and fast response time.

Passive matrix refers to a display technology that does not have individual transistors. That technology is also called FSTN, for film supertwist nematic, sometimes shortened to just supertwist.
DualScan is Apple’s new type of FSTN color, an improved version of the color display used in the PowerBook 165c.

Flat Panel Display Circuitry

The flat panel display circuitry in the PowerBook 520 and 540 computers emulate a NuBus™ video card installed in slot $0. There is no declaration ROM as such; its functions have been incorporated into the system ROM.

The display circuitry includes the CSC controller IC and a display buffer consisting of 512 KB of VRAM. The LCD display is compatible with software that uses QuickDraw and the Palette Manager. The display supports color table animation.

Note
The display circuitry in the PowerBook 520 and 540 computers is similar to that in the PowerBook 270c.

Number of Colors

The display controller IC contains a 256-entry CLUT. Although the CLUT supports a palette of 32,768 colors, many of the possible colors do not look acceptable on the display.
Due to the nature of color LCD technology, some colors are dithered or exhibit noticeable flicker. Apple has developed new gamma tables for these displays that minimizes flicker and optimizes available colors. With these gamma tables, the effective range of the CLUT for the active matrix color display is about 24,000 colors; for the DualScan color display, the effective range is about 4000 colors.

Note
Unlike the LCD controller on the PowerBook 160 and 180 models, the LCD controller on the PowerBook 520 and 540 computers is compatible with QuickDraw at all bit depths.

External Video Port

The external video port provides video output for all Apple 12-inch, 13-inch, and 16-inch RGB monitors, the Apple Macintosh Portrait Display, and Apple’s new 17-inch multiscan
External Video Port

display. With appropriate adapter cables, the external video port can also support a VGA display or a 800-by-600-pixel SVGA display.

The external video interface is enabled by attaching a monitor and restarting the computer. During the boot process, ROM software tests the monitor sense lines and activates the video output system if a recognized monitor is attached. If no monitor is found, the video output system is deactivated to conserve power.

Video Mirroring

There are two video output modes: dual mode and mirror mode. In dual mode, which is the normal Macintosh mode of operation, the external video monitor is independent of the flat panel display and displays additional information. Alternatively, the user can select mirror mode, in which the external monitor mirrors (duplicates) the flat panel display. In mirror mode, the display on the larger screen uses only the central portion of that screen and matches the horizontal and vertical dimensions of the smaller screen.

**Warning**

Applications that write directly to the display buffer may not be compatible with mirror mode unless they ensure that they do not write outside the active display area. That is not a problem for applications that use QuickDraw and never write directly to the display buffer.

Because the video output circuitry consumes additional power, Apple recommends that customers use the AC adapter when using an external monitor.

Video Monitors

The PowerBook 520 and 540 computers support the external monitors and video modes listed in Table 3-7.

<table>
<thead>
<tr>
<th>Monitor type</th>
<th>Width (pixels)</th>
<th>Height (pixels)</th>
<th>Pixel depth (bits)</th>
<th>Frame rate (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-inch RGB</td>
<td>512</td>
<td>384</td>
<td>8</td>
<td>60.15</td>
</tr>
<tr>
<td>13-inch RGB*</td>
<td>640</td>
<td>480</td>
<td>8</td>
<td>66.67</td>
</tr>
<tr>
<td>Portrait</td>
<td>640</td>
<td>870</td>
<td>4</td>
<td>75.0</td>
</tr>
<tr>
<td>16-inch RGB</td>
<td>832</td>
<td>624</td>
<td>8</td>
<td>66.67</td>
</tr>
<tr>
<td>17-inch multiscan</td>
<td>640</td>
<td>480</td>
<td>8</td>
<td>66.67</td>
</tr>
<tr>
<td>17-inch multiscan</td>
<td>832</td>
<td>624</td>
<td>8</td>
<td>75.0</td>
</tr>
<tr>
<td>VGA or SVGA</td>
<td>640</td>
<td>480</td>
<td>8</td>
<td>59.95</td>
</tr>
<tr>
<td>SVGA</td>
<td>800</td>
<td>600</td>
<td>8</td>
<td>55.98</td>
</tr>
</tbody>
</table>

*Includes Macintosh Color Display and Apple High Resolution Monochrome Monitor.
I/O Features

Note
The largest image the PowerBook 520 and 540 computers can display on a multiscan monitor is 832 by 624 pixels. ♦

With a multiscan monitor, the Monitors control panel allows the user to select from the different modes available. Similarly, with an SVGA monitor, the user can select either the 640-by-480-pixel display or the 800-by-600-pixel display.

External Video Connector

The PowerBook 520 and 540 computers have the same type VID-14 video output connector as earlier models. An adapter cable included with the computer allows the user to attach a standard Apple video cable. Figure 3-6 shows the pin configuration of the VID-14 connector on the back of the computer and the DB-15 connector on the adapter cable. Table 3-8 lists the signal pin assignments.

One source for the VID-14 adapter cable is

Hosiden America Corp.
10090 Pasadena Ave., Suite B2
Cupertino, CA 95014
408-252-0541

Refer to Hosiden part number CMP1220-010100.

Monitor Sense Codes

To identify the type of monitor connected, the external video interface uses the standard Apple monitor sense codes as well as the newer extended sense codes. Table 3-9 shows the sense codes for each of the supported monitors. Refer to the Macintosh Technical Note 30 SenseLines for a description of the sense code system.
External Video Port

Table 3-8  Signals on the video connector

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RED.VID</td>
<td>Red video signal</td>
</tr>
<tr>
<td>2</td>
<td>RED.GND</td>
<td>Red video ground</td>
</tr>
<tr>
<td>3</td>
<td>SENSE0</td>
<td>Monitor sense signal 0</td>
</tr>
<tr>
<td>4</td>
<td>/VSYNC</td>
<td>Vertical synchronization signal</td>
</tr>
<tr>
<td>5</td>
<td>/CSYNC</td>
<td>Composite synchronization signal</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
<td>CSYNC and VSYNC ground</td>
</tr>
<tr>
<td>7</td>
<td>GRN.GND</td>
<td>Green video ground</td>
</tr>
<tr>
<td>8</td>
<td>GRN.VID</td>
<td>Green video signal</td>
</tr>
<tr>
<td>9</td>
<td>SENSE1</td>
<td>Monitor sense signal 1</td>
</tr>
<tr>
<td>10</td>
<td>HSYNC.GND</td>
<td>HSYNC ground</td>
</tr>
<tr>
<td>11</td>
<td>SENSE2</td>
<td>Monitor sense signal 2</td>
</tr>
<tr>
<td>12</td>
<td>/HSYNC</td>
<td>Horizontal synchronization signal</td>
</tr>
<tr>
<td>13</td>
<td>BLU.VID</td>
<td>Blue video signal</td>
</tr>
<tr>
<td>14</td>
<td>BLU.GND</td>
<td>Blue video ground</td>
</tr>
<tr>
<td>15</td>
<td>n.c.</td>
<td>Not connected</td>
</tr>
<tr>
<td>Shell</td>
<td>Shell SGND</td>
<td>Shield ground</td>
</tr>
</tbody>
</table>

Table 3-9  Monitor sense codes

<table>
<thead>
<tr>
<th>Monitor type</th>
<th>Standard sense codes</th>
<th>Extended sense codes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(2–0)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>12-inch RGB</td>
<td>0 1 0</td>
<td>n.a.</td>
</tr>
<tr>
<td>13-inch RGB</td>
<td>1 1 0</td>
<td>n.a.</td>
</tr>
<tr>
<td>Portrait</td>
<td>0 0 1</td>
<td>n.a.</td>
</tr>
<tr>
<td>16-inch RGB</td>
<td>1 1 1</td>
<td>1 0 11 01</td>
</tr>
<tr>
<td>17-inch multiscan</td>
<td>1 1 0</td>
<td>11 01 00</td>
</tr>
<tr>
<td>VGA/SVGA</td>
<td>1 1 1</td>
<td>0 1 01 11</td>
</tr>
<tr>
<td>No monitor</td>
<td>1 1 1</td>
<td>11 11 11</td>
</tr>
</tbody>
</table>
Serial Port

The PowerBook 520 and 540 computers have a standard Macintosh serial port for synchronous, asynchronous, or AppleTalk serial communication. The 8-pin mini-DIN connector on the back panel is the same as those on other Macintosh computers. Table 3-10 shows the signal assignments for the serial port.

Table 3-10 Serial port signals

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Signal name</th>
<th>Signal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HSKo</td>
<td>Handshake output</td>
</tr>
<tr>
<td>2</td>
<td>HSKi</td>
<td>Handshake input</td>
</tr>
<tr>
<td>3</td>
<td>TxD–</td>
<td>Transmit data –</td>
</tr>
<tr>
<td>4</td>
<td>SG</td>
<td>Signal ground</td>
</tr>
<tr>
<td>5</td>
<td>RxD–</td>
<td>Receive data –</td>
</tr>
<tr>
<td>6</td>
<td>TxD+</td>
<td>Transmit data +</td>
</tr>
<tr>
<td>7</td>
<td>GPI</td>
<td>General-purpose input</td>
</tr>
<tr>
<td>8</td>
<td>RxD+</td>
<td>Receive data +</td>
</tr>
</tbody>
</table>

SCSI Port

The SCSI port on the PowerBook 520 and 540 computers supports the SCSI interface as defined by the American National Standards Institute (ANSI) X3T9.2 committee.

SCSI Disk Mode

The PowerBook 520 and 540 computers support SCSI disk mode, which allows the internal hard disk to be mounted and used as an external drive by another Macintosh. To operate a PowerBook computer in SCSI disk mode, the user connects an HDI-30 SCSI Disk Adapter cable between the external SCSI connectors on the PowerBook computer and the desktop computer and then starts both computers. The adapter cable grounds pin 1 of the HDI-30 connector, causing the PowerBook computer’s ROM code to bypass the normal startup procedure and enter SCSI disk mode.
SCSI Connectors

The internal and external HDI-30 connectors are identical to those used in other PowerBook models. The SCSI portion of the Combo IC connects directly to the internal and external SCSI connectors and can sink up to 48 mA through each of the pins connected to the SCSI bus. The data and control signals on the SCSI bus are active low signals that are driven by open drain outputs.

Table 3-11 shows the signal assignments for the internal and external SCSI connectors. Note that pin 1 of the external HDI-30 connector is the /SCSI.DISK.MODE signal.

<table>
<thead>
<tr>
<th>Pin number</th>
<th>HDI-30 (internal)</th>
<th>HDI-30 (external)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DISK.+5</td>
<td>/SCSI.DISK.MODE</td>
</tr>
<tr>
<td>2</td>
<td>DISK.+5</td>
<td>/DB0</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>/DB1</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>TERMPWR (not used; reserved for future use)</td>
</tr>
<tr>
<td>6</td>
<td>/DB0</td>
<td>/DB2</td>
</tr>
<tr>
<td>7</td>
<td>/DB1</td>
<td>/DB3</td>
</tr>
<tr>
<td>8</td>
<td>/DB2</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>/DB3</td>
<td>/ACK</td>
</tr>
<tr>
<td>10</td>
<td>/DB4</td>
<td>GND</td>
</tr>
<tr>
<td>11</td>
<td>/DB5</td>
<td>/DB4</td>
</tr>
<tr>
<td>12</td>
<td>/DB6</td>
<td>GND</td>
</tr>
<tr>
<td>13</td>
<td>/DB7</td>
<td>GND</td>
</tr>
<tr>
<td>14</td>
<td>/DBP</td>
<td>/DB5</td>
</tr>
<tr>
<td>15</td>
<td>DISK.+5</td>
<td>GND</td>
</tr>
<tr>
<td>16</td>
<td>/BSY</td>
<td>/DB6</td>
</tr>
<tr>
<td>17</td>
<td>/ATN</td>
<td>GND</td>
</tr>
<tr>
<td>18</td>
<td>/ACK</td>
<td>/DB7</td>
</tr>
<tr>
<td>19</td>
<td>GND</td>
<td>/DBP</td>
</tr>
<tr>
<td>20</td>
<td>/MSG</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>/RST</td>
<td>/REQ</td>
</tr>
<tr>
<td>22</td>
<td>/SEL</td>
<td>GND</td>
</tr>
</tbody>
</table>

continued
I/O Features

Table 3-11 Internal and external HDI-30 SCSI connector signals (continued)

<table>
<thead>
<tr>
<th>Pin number</th>
<th>HDI-30 (internal)</th>
<th>HDI-30 (external)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>/C/D</td>
<td>/BSY</td>
</tr>
<tr>
<td>24</td>
<td>/I/O</td>
<td>GND</td>
</tr>
<tr>
<td>25</td>
<td>/REQ</td>
<td>/ATN</td>
</tr>
<tr>
<td>26</td>
<td>GND</td>
<td>/C/D</td>
</tr>
<tr>
<td>27</td>
<td>GND</td>
<td>/RST</td>
</tr>
<tr>
<td>28</td>
<td>GND</td>
<td>/MSG</td>
</tr>
<tr>
<td>29</td>
<td>DISK.+5</td>
<td>/SEL</td>
</tr>
<tr>
<td>30</td>
<td>DISK.+5</td>
<td>/I/O</td>
</tr>
</tbody>
</table>

Table 3-11 Internal and external HDI-30 SCSI connector signals (continued)

<table>
<thead>
<tr>
<th>Pin number</th>
<th>HDI-30 (internal)</th>
<th>HDI-30 (external)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>/C/D</td>
<td>/BSY</td>
</tr>
<tr>
<td>24</td>
<td>/I/O</td>
<td>GND</td>
</tr>
<tr>
<td>25</td>
<td>/REQ</td>
<td>/ATN</td>
</tr>
<tr>
<td>26</td>
<td>GND</td>
<td>/C/D</td>
</tr>
<tr>
<td>27</td>
<td>GND</td>
<td>/RST</td>
</tr>
<tr>
<td>28</td>
<td>GND</td>
<td>/MSG</td>
</tr>
<tr>
<td>29</td>
<td>DISK.+5</td>
<td>/SEL</td>
</tr>
<tr>
<td>30</td>
<td>DISK.+5</td>
<td>/I/O</td>
</tr>
</tbody>
</table>

**Ethernet Port**

The Ethernet connector is an Apple AUI connector. It accepts a Friendlynet adapter for either AUI (thick) cable, thin cable, or 10baseT (twisted pair) cable. The Ethernet port pin assignments are shown in Table 3-12.

Table 3-12 Pin assignments on the Ethernet port

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5 V</td>
<td>8</td>
<td>+5 V</td>
</tr>
<tr>
<td>2</td>
<td>DI+</td>
<td>9</td>
<td>DO+</td>
</tr>
<tr>
<td>3</td>
<td>DI−</td>
<td>10</td>
<td>DO−</td>
</tr>
<tr>
<td>4</td>
<td>Ground</td>
<td>11</td>
<td>Ground</td>
</tr>
<tr>
<td>5</td>
<td>CI+</td>
<td>12</td>
<td>NC</td>
</tr>
<tr>
<td>6</td>
<td>CI−</td>
<td>13</td>
<td>NC</td>
</tr>
<tr>
<td>7</td>
<td>+5 V</td>
<td>14</td>
<td>+5 V</td>
</tr>
</tbody>
</table>
ADB Port

The Apple Desktop Bus (ADB) port on the PowerBook 520 and 540 computers 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 PowerBook 520 and 540 computers. The custom ADB microcontroller drives the bus and reads status from the selected external device. A 4-pin mini-DIN connector connects the ADB controller to the outside world. Table 3-13 lists the ADB connector pin assignments. For more information about the ADB, see Guide to the Macintosh Family Hardware, second edition.

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADB</td>
<td>Bidirectional data bus used for input and output; an open collector signal pulled up to +5 volts through a 470-ohm resistor on the main logic board.</td>
</tr>
<tr>
<td>2</td>
<td>PSW</td>
<td>Power on signal; generates reset and interrupt key combinations.</td>
</tr>
<tr>
<td>3</td>
<td>+5V</td>
<td>+5 volts from the computer.</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>Ground from the computer.</td>
</tr>
</tbody>
</table>

**IMPORTANT**
The total current available for all devices connected to the +5V pins on the ADB is 100 mA.

Sound System

The 16-bit stereo audio circuitry provides high-quality sound input and output through the built-in microphone and speakers. The user can also connect external input and output devices by way of the sound input and output jacks.

The sound system is based on the Singer codec IC along with input and output amplifiers. In the PowerBook 520 and 540 computers, the Singer codec supports two channels with sample sizes up to 16 bits and sample rates of 11 kHz, 22.05 kHz, and 44.1 kHz.
I/O Features

The frequency response of the sound circuits, not including the microphone and speakers, is within plus or minus 2 dB from 20 Hz to 20 kHz. Total harmonic distortion and noise is less than 0.05 percent from 20 Hz to 20 kHz with a 1 V RMS sine wave input. The input signal-to-noise ratio (SNR) is 82 dB and the output SNR is 85 dB, with no audible discrete tones.

The sound system includes a built-in microphone and two speakers as well as external audio input and output jacks. The user can also plug in an external microphone and select either the built-in microphone or the external microphone by using the Sound control panel.

The speakers are located in the upper-left and upper-right corners of the display bezel. The speakers are modified versions of the ones used in the PowerBook 140 and 170 computers, with improved low-frequency response.
Expansion Modules
Expansión de módulos

Los ordenadores PowerBook 520 y 540 aceptan los siguientes módulos de expansión, implementados como pequeñas tarjetas de circuito:

- Tarjeta de expansión RAM
- Tarjeta PDS
- Tarjeta módem

Esta capítulo incluye descripciones y guías de diseño para cada uno de estos módulos de expansión.

### Tarjeta de expansión RAM

Un conector en la placa lógica secundaria acepta una tarjeta de expansión RAM que contiene de 8 MB a 32 MB de RAM de auto-refresque. La tarjeta puede contener de uno a cuatro identicos bancos, con 2 MB, 4 MB, o 8 MB en cada banco. La tabla 4-1 muestra cómo los bancos pueden implementarse con dispositivos RAM estándar.

**Tabla 4-1 Configuraciones de bancos RAM**

<table>
<thead>
<tr>
<th>Tamaño de banco</th>
<th>Número de dispositivos por banco</th>
<th>Tamaño de dispositivo (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 MB</td>
<td>4</td>
<td>512K × 8</td>
</tr>
<tr>
<td>4 MB</td>
<td>8</td>
<td>1 M × 4</td>
</tr>
<tr>
<td>4 MB</td>
<td>2</td>
<td>1 M × 16</td>
</tr>
<tr>
<td>8 MB</td>
<td>4</td>
<td>2 M × 8</td>
</tr>
</tbody>
</table>

**IMPORTANTE**

La tarjeta de expansión RAM para los ordenadores PowerBook 520 y 540 es una nueva diseño; las tarjetas diseñadas para otros modelos de ordenadores PowerBook no pueden utilizarse en estos nuevos modelos de PowerBook.

**ADVERTENCIA**

La instalación de una tarjeta de expansión RAM en un ordenador PowerBook 520 o 540 debe ser realizada por un técnico experimentado. Requiere cuidado para evitar dañar los pin del conector de expansión RAM.

### Guía de diseño eléctrico para la tarjeta de expansión RAM

Esta sección proporciona la información eléctrica que necesita para diseñar una tarjeta de expansión RAM para los ordenadores PowerBook 520 y 540.
Expansion Modules

Connector Pin Assignments

Table 4-2 lists the names of the signals on the RAM expansion connector. Entries in the table are arranged the same way as the pins on the connector: pin 1 across from pin 2, and so on. Signal names that end with \_L are active low.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal name</th>
<th>Direction</th>
<th>Pin</th>
<th>Signal name</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>—</td>
<td>2</td>
<td>DRAM_ADDR[0]</td>
<td>O</td>
</tr>
<tr>
<td>5</td>
<td>DRAM_ADDR[3]</td>
<td>O</td>
<td>6</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>11</td>
<td>V_5P_MAIN</td>
<td>—</td>
<td>12</td>
<td>DRAM_ADDR[8]</td>
<td>O</td>
</tr>
<tr>
<td>21</td>
<td>GND</td>
<td>—</td>
<td>22</td>
<td>DRAM_CAS_L[0]</td>
<td>O</td>
</tr>
<tr>
<td>25</td>
<td>DRAM_CAS_L[3]</td>
<td>O</td>
<td>26</td>
<td>V_5P_MAIN</td>
<td>—</td>
</tr>
<tr>
<td>27</td>
<td>DRAM_WE_L</td>
<td>O</td>
<td>28</td>
<td>RESERVED[1]</td>
<td>X</td>
</tr>
<tr>
<td>31</td>
<td>GND</td>
<td>—</td>
<td>32</td>
<td>CPU_DATA[0]</td>
<td>B</td>
</tr>
<tr>
<td>35</td>
<td>CPU_DATA[3]</td>
<td>B</td>
<td>36</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>41</td>
<td>V_5P_MAIN</td>
<td>—</td>
<td>42</td>
<td>CPU_DATA[8]</td>
<td>B</td>
</tr>
<tr>
<td>51</td>
<td>GND</td>
<td>—</td>
<td>52</td>
<td>CPU_DATA[16]</td>
<td>B</td>
</tr>
</tbody>
</table>

continued
Table 4-2  Signal assignments on the RAM expansion connector  (continued)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal name</th>
<th>Direction</th>
<th>Pin</th>
<th>Signal name</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>CPU_DATA[19]</td>
<td>B</td>
<td>56</td>
<td>V_5P_MAIN</td>
<td>—</td>
</tr>
<tr>
<td>59</td>
<td>CPU_DATA[23]</td>
<td>B</td>
<td>60</td>
<td>CPU_DATA[22]</td>
<td>B</td>
</tr>
<tr>
<td>61</td>
<td>GND</td>
<td>—</td>
<td>62</td>
<td>CPU_DATA[24]</td>
<td>B</td>
</tr>
<tr>
<td>65</td>
<td>CPU_DATA[27]</td>
<td>B</td>
<td>66</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>67</td>
<td>CPU_DATA[29]</td>
<td>B</td>
<td>68</td>
<td>CPU_DATA[28]</td>
<td>B</td>
</tr>
</tbody>
</table>

NOTE   Signal directions are defined from the system side of the connector; O = output and B = bidirectional. Signals marked X are reserved.

Signal Descriptions

Table 4-3 describes the signals on the RAM expansion connector. Signal names that end with _L are active low.

Note

In the table, signals are specified as inputs or outputs with respect to the secondary logic board that contains the CPU and memory module; for example, an input is driven by the expansion card into the secondary logic board. ♦

Address Multiplexing

DRAM_ADDR[0–11] is a 12-bit multiplexed address bus and 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 operation of the multiplexing is determined by the way the address pins on the devices are connected to individual signals on the DRAM_ADDR[0–11] bus and depends on the exact type of DRAM used.

Table 4-4 on page 38 shows how the signals on the address bus are connected for several types of DRAM devices. The device types are specified by their size and by the number of row and column address bits they require.
Table 4-4 shows how the signals are multiplexed during the row and column address phases. For each type of DRAM device, the first and second rows show the actual address bits that drive each address pin during row addressing and column addressing, respectively. The third row shows how the device’s address pins are connected to the signals on the DRAM_ADDR bus.

**IMPORTANT**

Some types of DRAM devices don’t use all 12 bits in the row or column address. Address-bit numbers for those unused bits are shown in plain style; bit numbers for the bits that are used are shown in bold. ▲

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM_ADDR[0-11]</td>
<td>Multiplexed row and column address to the DRAM devices. (See the section “Address Multiplexing” on page 36 to determine which bits to use for a given type of DRAM device.)</td>
</tr>
<tr>
<td>DRAM_CAS_L[0-3]</td>
<td>Column address select signals for the individual bytes in a longword. The signals are assigned to the bytes as follows:</td>
</tr>
<tr>
<td></td>
<td>DRAM_CAS_L[0] selects CPU_DATA[0-7]</td>
</tr>
<tr>
<td>DRAM_RAS_L[2-5]</td>
<td>Row address select signals for up to four banks of DRAM. (The first two banks, selected by DRAM_RAS_L[1:0], reside on the CPU and memory module.)</td>
</tr>
<tr>
<td>DRAM_WE_L</td>
<td>Write enable for the banks of DRAM.</td>
</tr>
<tr>
<td>GND</td>
<td>Chassis and logic ground.</td>
</tr>
<tr>
<td>RESERVED[1-3]</td>
<td>Reserved pins; must be left open.</td>
</tr>
<tr>
<td>V_5P_MAIN</td>
<td>5.0 ± 5%; 300 mA maximum</td>
</tr>
</tbody>
</table>
Expansion Modules

Table 4-4 Address multiplexing for some typical DRAM devices

<table>
<thead>
<tr>
<th>Type of DRAM device</th>
<th>Individual signals on DRAM_ADDR bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row address bits</td>
<td>21 20 19 18 17 16 15 14 13 12 11 10</td>
</tr>
<tr>
<td>Column address bits</td>
<td>19 21 18 22 9 8 7 6 5 4 3 2</td>
</tr>
<tr>
<td>Device address pins</td>
<td>11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>Row address bits</td>
<td>21 20 19 18 17 16 15 14 13 12 11 10</td>
</tr>
<tr>
<td>Column address bits</td>
<td>19 21 18 22 9 8 7 6 5 4 3 2</td>
</tr>
<tr>
<td>Device address pins</td>
<td>11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>1 M × 4, 12 row bits, 8 column bits</td>
<td></td>
</tr>
<tr>
<td>Row address bits</td>
<td>21 20 19 18 17 16 15 14 13 12 11 10</td>
</tr>
<tr>
<td>Column address bits</td>
<td>19 21 18 22 9 8 7 6 5 4 3 2</td>
</tr>
<tr>
<td>Device address pins</td>
<td>11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>Row address bits</td>
<td>21 20 19 18 17 16 15 14 13 12 11 10</td>
</tr>
<tr>
<td>Column address bits</td>
<td>19 21 18 22 9 8 7 6 5 4 3 2</td>
</tr>
<tr>
<td>Device address pins</td>
<td>9 10 — 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>1 M × 16, 10 row bits, 10 column bits</td>
<td></td>
</tr>
<tr>
<td>Row address bits</td>
<td>21 20 19 18 17 16 15 14 13 12 11 10</td>
</tr>
<tr>
<td>Column address bits</td>
<td>19 21 18 22 9 8 7 6 5 4 3 2</td>
</tr>
<tr>
<td>Device address pins</td>
<td>— 9 8 — 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>512K × 8, 10 row bits, 9 column bits</td>
<td>[11] [10] [9] [8] [7] [6] [5] [4] [3] [2] [1] [0]</td>
</tr>
<tr>
<td>Row address bits</td>
<td>21 20 19 18 17 16 15 14 13 12 11 10</td>
</tr>
<tr>
<td>Column address bits</td>
<td>19 21 18 22 9 8 7 6 5 4 3 2</td>
</tr>
<tr>
<td>Device address pins</td>
<td>— 9 8 — 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Banks of DRAM

The DRAM expansion card can have up to four banks of RAM. Banks can be 2 MB, 4 MB, or 8 MB in size; on a card with more than one bank, all banks must be the same size.

Because only one bank is active at a time, and because different-sized DRAM devices consume about the same amount of power when active, a card having fewer devices per bank consumes less power than a card having more devices per bank.
Expansion Modules

**DRAM Device Requirements**

The DRAM devices used in a DRAM expansion card must meet the following minimum specifications:

- fast page mode
- self-refreshing
- low-power grade
- row access time ($t_{RAC}$) of 70 ns or less
- column access time ($t_{CAC}$) of 20 ns or less
- page-mode cycle time ($t_{PC}$) of 50 ns or less

**Note**

The DRAM refresh operation depends on the state of the computer. When the computer is operating, the Pratt memory controller IC (described on page 12) provides refresh signals consisting of 2048 CAS before RAS cycles every 128 ms. When the computer goes into sleep mode, it switches the DRAM devices to their self-refresh feature to save power.

**Expansion Card Electrical Limits**

The DRAM expansion card must not exceed the following maximum current limits on the +5 V supply:

- active: 300 mA
- standby: 24 mA
- self-refresh: 6 mA

The capacitative loading on the signal lines must not exceed the following limits:

- DRAM_ADDR[0–11]: 85 pF
- DRAM_RAS_L[2–5]: 25 pF
- DRAM_CAS_L[0–3]: 25 pF
- DRAM_WE_L: 85 pF
- CPU_DATA[0–31]: 25 pF
Expansion Modules

Mechanical Design Guidelines for the RAM Expansion Card

This section contains mechanical drawings showing the recommended design guidelines for RAM expansion cards.

RAM Expansion Connector

The RAM expansion connector is a 70-pin Metrastack connector. The connector is available from AMP, Inc. (part number 535671-6). For a specification sheet or information about obtaining this connector, contact AMP, Inc.

19200 Stevens Creek Blvd.
Cupertino, CA 95014-2578
408-725-4914
AppleLink: AMPCUPERTINO

RAM Expansion Card Design

This section shows the mechanical outline and parts-placement guidelines for the RAM expansion card. Figure 4-1 is a top view of the card showing dimensions. Figure 4-2 and Figure 4-3 are top and bottom views showing the restrictions on placement of components and traces. Figure 4-4 is an edge view of the card; the dotted lines show the maximum component height on either side.

Figure 4-1  Dimensions of the RAM expansion card

Note: Dimensions are in millimeters [inches].
Expansion Modules

Figure 4-2 Restricted areas on the top of the RAM expansion card

![Diagram of RAM expansion card top]

Note: Dimensions are in millimeters [inches].

Figure 4-3 Restricted areas on the bottom of the RAM expansion card

![Diagram of RAM expansion card bottom]

Note: Dimensions are in millimeters [inches].

Note
The drawings in Figure 4-1, Figure 4-2, Figure 4-3, and Figure 4-4 are taken from Apple engineering drawing number 056-0124, revision 02. Developers who are members of Apple’s Developer Program may contact Apple’s Developer Support Center for information about possible later revisions of the drawings.
CHAPTER 4

Expansion Modules

Figure 4-4    Edge view of the RAM expansion card

Note: Dimensions are in millimeters [inches].

*WARNING*

Do not exceed the dimensions shown in the mechanical drawings. Cards that exceed these specifications may damage the computer and may be incompatible with future PowerBook models. ▲

PDS Expansion Card

The processor-direct slot (PDS) in the PowerBook 520 and 540 computers is a 90-pin connector located inside the left battery compartment. Users must remove the left battery to install a PDS card. A battery compartment cover is included with the computer for use when the battery is removed.

The PDS connector provides direct access to the MC68030-compatible bus on the main logic board. The PDS connector also accepts an optional adapter that can accommodate one or two industry-standard PCMCIA cards: two type II cards or one type III card.

Electrically, the PDS connector is not a true PDS because it is connected to the input/output subsystem’s MC68030 bus and not directly to the MC68040 microprocessor bus. Because the PDS is independent of the processor bus, users can upgrade their microprocessor without having to purchase new PDS cards. The fact that the PDS is connected to the MC68030 bus also simplifies card design for developers with existing MC68030 PDS products.

User Installation of the PDS Card

The PDS card can be installed and removed by the user. Even though the instructions warn against inserting or removing the card while the computer is running, the user may not follow instructions. The card must be designed to prevent damage to the card and the computer in that eventuality.
Here are the requirements that minimize the possibility of damage if the user inserts or removes the PDS card with the computer running.

- The PDS connector is designed so that the power and ground pins make contact before the signal lines when the card is inserted.
- The PDS card must buffer all signals so as not to load down the computer’s internal bus. The recommended way to do that is to install CMOS logic buffers on all the logic signals used on the PDS card. The buffers must draw their power directly from the +5 V supply line and must always be powered.
- The PDS card must indicate to the computer that the card is inserted by grounding the PDS insert signal. The line should be tied directly to ground so that it goes low as soon as the PDS card is inserted.
- The PDS card must not pull down the +5 V supply line even when the card is inserted in a discharged state. To prevent a glitch (momentary disturbance) in the +5 V supply line, the card must include a surge suppression circuit that limits the inrush current for the circuitry on the card. See the section “Power Control Circuits” on page 49.
- To prevent the card’s bus drivers from presenting a load to the bus when the card is inserted, the card should have a pull-up resistor (100K) on the address strobe line. This will ensure that the card will have an invalid address and the drivers will be in the off state.

By following the guidelines outlined here, you can design a PDS card that minimizes the possibility of damage even when it is inserted hot, that is, while the computer is running or in sleep mode. Even so, your instructions to the user should include warnings against doing so.

**IMPORTANT**

Even with the design features described here, eventually a circuit may fail; if the user repeatedly inserts and removes the card with power on, damage to the computer or the card may eventually occur. The only time it is completely safe to insert or remove the card is while the computer is shut down.

**Electrical Description of the PDS**

This section describes the electrical characteristics of the PDS. For mechanical specifications, see the section “Mechanical Design Guidelines for the PDS Card” beginning on page 52.

**Signals on the PDS Connector**

Table 4-5 shows the signal assignments on the PDS connector. Entries in the table are arranged the same way as the pins on the connector: pin 1 across from pin 46, pin 2 across from pin 47, and so on. Signal names that end with _L are active low.
## Table 4-5  Signal assignments on the PDS connector

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Signal name</th>
<th>Direction</th>
<th>Pin number</th>
<th>Signal name</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Old</td>
<td>Signal name</td>
<td>New</td>
<td>Old</td>
<td>Signal name</td>
</tr>
<tr>
<td>1</td>
<td>90</td>
<td>GND</td>
<td>31</td>
<td>30</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>86</td>
<td>ADDR[3]</td>
<td>33</td>
<td>26</td>
<td>DS_L</td>
</tr>
<tr>
<td>4</td>
<td>84</td>
<td>ADDR[5]</td>
<td>34</td>
<td>24</td>
<td>BERR_L</td>
</tr>
<tr>
<td>5</td>
<td>82</td>
<td>ADDR[7]</td>
<td>35</td>
<td>22</td>
<td>PDS_BG_L</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>ADDR[9]</td>
<td>36</td>
<td>20</td>
<td>PDS_BUSCLK</td>
</tr>
<tr>
<td>7</td>
<td>78</td>
<td>V_5P_MAIN</td>
<td>37</td>
<td>18</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>74</td>
<td>ADDR[13]</td>
<td>39</td>
<td>14</td>
<td>SLEEP</td>
</tr>
<tr>
<td>10</td>
<td>72</td>
<td>ADDR[15]</td>
<td>40</td>
<td>12</td>
<td>IO_RESET_L</td>
</tr>
<tr>
<td>11</td>
<td>70</td>
<td>ADDR[17]</td>
<td>41</td>
<td>10</td>
<td>PDS_IRQ_L</td>
</tr>
<tr>
<td>12</td>
<td>68</td>
<td>ADDR[19]</td>
<td>42</td>
<td>8</td>
<td>PDS_CPU</td>
</tr>
<tr>
<td>13</td>
<td>66</td>
<td>GND</td>
<td>43</td>
<td>6</td>
<td>MDM_TDM[1]</td>
</tr>
<tr>
<td>15</td>
<td>62</td>
<td>ADDR[23]</td>
<td>45</td>
<td>2</td>
<td>PDS_INSERT_L</td>
</tr>
<tr>
<td>16</td>
<td>60</td>
<td>ADDR[25]</td>
<td>46</td>
<td>89</td>
<td>ADDR[0]</td>
</tr>
<tr>
<td>19</td>
<td>54</td>
<td>GND</td>
<td>49</td>
<td>83</td>
<td>GND</td>
</tr>
<tr>
<td>25</td>
<td>42</td>
<td>V_5P_MAIN</td>
<td>55</td>
<td>71</td>
<td>GND</td>
</tr>
<tr>
<td>26</td>
<td>40</td>
<td>DATA[23]</td>
<td>56</td>
<td>69</td>
<td>ADDR[16]</td>
</tr>
<tr>
<td>28</td>
<td>36</td>
<td>DATA[27]</td>
<td>58</td>
<td>65</td>
<td>ADDR[20]</td>
</tr>
<tr>
<td>29</td>
<td>34</td>
<td>DATA[29]</td>
<td>59</td>
<td>63</td>
<td>ADDR[22]</td>
</tr>
<tr>
<td>30</td>
<td>32</td>
<td>DATA[31]</td>
<td>60</td>
<td>61</td>
<td>ADDR[24]</td>
</tr>
</tbody>
</table>

*continued*
Expansion Modules

Table 4-5  Signal assignments on the PDS connector  (continued)

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Signal name</th>
<th>Direction</th>
<th>Pin number</th>
<th>Signal name</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>New  Old</td>
<td>New  Old</td>
<td></td>
<td>New  Old</td>
<td>New  Old</td>
<td></td>
</tr>
<tr>
<td>61 59</td>
<td>V_5P_MAIN</td>
<td>—</td>
<td>76 29</td>
<td>DATA[30]</td>
<td>B</td>
</tr>
<tr>
<td>62 57</td>
<td>ADDR[26]</td>
<td>B</td>
<td>77 27</td>
<td>DSACK_L[0]</td>
<td>B</td>
</tr>
<tr>
<td>63 55</td>
<td>ADDR[28]</td>
<td>B</td>
<td>78 25</td>
<td>AS_L</td>
<td>B</td>
</tr>
<tr>
<td>64 53</td>
<td>ADDR[30]</td>
<td>B</td>
<td>79 23</td>
<td>V_5P_MAIN</td>
<td>—</td>
</tr>
<tr>
<td>65 51</td>
<td>RW_L</td>
<td>B</td>
<td>80 21</td>
<td>PDS_BR_L</td>
<td>I</td>
</tr>
<tr>
<td>66 49</td>
<td>SIZ[0]</td>
<td>B</td>
<td>81 19</td>
<td>BGACK_L</td>
<td>B</td>
</tr>
<tr>
<td>67 47</td>
<td>GND</td>
<td>—</td>
<td>82 17</td>
<td>IPL_L[0]</td>
<td>O</td>
</tr>
<tr>
<td>69 43</td>
<td>DATA[18]</td>
<td>B</td>
<td>84 13</td>
<td>PDS_RESET_L</td>
<td>O</td>
</tr>
<tr>
<td>70 41</td>
<td>DATA[20]</td>
<td>B</td>
<td>85 11</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>71 39</td>
<td>DATA[22]</td>
<td>B</td>
<td>86 9</td>
<td>SND_SCLK</td>
<td>O</td>
</tr>
<tr>
<td>72 37</td>
<td>DATA[24]</td>
<td>B</td>
<td>87 7</td>
<td>PDS_POWER_SW_L</td>
<td>OC</td>
</tr>
<tr>
<td>73 35</td>
<td>GND</td>
<td>—</td>
<td>88 5</td>
<td>MDM_TDM[0]</td>
<td>X</td>
</tr>
<tr>
<td>75 31</td>
<td>DATA[28]</td>
<td>B</td>
<td>90 1</td>
<td>PDS_DIGITAL_SND</td>
<td>I</td>
</tr>
</tbody>
</table>

The PDS connector on the main logic board is a 90-pin hybrid: half the leads are attached to the board as through-hole pins and half as surface-mount leads. The through-hole pins are numbered from 1 through 45 and the surface-mount pins are numbered from 46 through 90. The pins are divided into two groups by a gap. Pins 1 and 46 are at the end of the connector nearest the gap; pins 45 and 90 are at the end farthest from the gap.

Note
On early schematic diagrams from Apple, the pins on the PDS connector were numbered starting from the other end. That numbering scheme is shown in Table 4-5 in the column labeled Old. ◆

The signals on the PDS connector are divided by function into three groups: MC68030-compatible signals, system support signals, and power leads. The next two sections describe the MC68030 signals and the system signals. The power leads are described in the section “PDS Power Consumption” beginning on page 49.

MC68030-Compatible Signals on the PDS Connector
Table 4-6 lists the MC68030-compatible signals and describes their functions. Signal names that end with _L are active low. Refer to the Motorola MC68030 User’s Manual for a complete description of these signals.
### Table 4-6 MC68030-compatible signals on the PDS connector

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR[31–0]</td>
<td>Bidirectional 32-bit system address bus.</td>
</tr>
<tr>
<td>AS_L</td>
<td>Indicates the occurrence of an active bus transaction: a valid address is on the address bus and the size and read/write signals are valid.</td>
</tr>
<tr>
<td>BERR_L</td>
<td>Open collector signal connected to the system’s bus error watchdog timer and the processor’s /BERR line. This signal is used for two operations: (1) terminating an active bus transaction and (2) causing a bus cycle to be retried when asserted with /HALT.</td>
</tr>
<tr>
<td>BGACK_L</td>
<td>Bidirectional signal connected to the processor’s /BGACK line; indicates that an alternate bus master has control of the system bus. The alternate master on the PDS card drives this line after it has been granted the system bus and has started a bus transaction. A bus master must never assert BR_L and BGACK_L simultaneously; instead, it should deassert PDS_BR_L and assert BGACK_L at the same time (internal PDS clock edge).</td>
</tr>
<tr>
<td>DATA[31–16]</td>
<td>Bidirectional 16-bit system data bus.</td>
</tr>
<tr>
<td>DS_L</td>
<td>Three-state output connected to the processor’s /DS line. This signal is used to manage the transmission of data to and from the processor. During a read cycle, the PDS card places the requested data on the data bus when this signal is asserted. When this signal is asserted during a write, the processor drives the data bus with data to be written to the PDS card. When this signal is released, the data bus is not driven.</td>
</tr>
<tr>
<td>DSACK_L[1–0]</td>
<td>Data strobe acknowledge signals. These signals perform two functions: (1) they indicate the completion of the data transfer portion of a bus transaction by the addressed slave, and (2) they inform the bus master of the size of the data port being accessed during the current transaction.</td>
</tr>
<tr>
<td>PDS_BG_L</td>
<td>PDS bus granted; output signal is driven by the system’s bus arbitrator. When asserted low, the alternate bus master on the PDS card may take ownership of the system bus after any pending bus traffic has completed—that is, /BGACK, /PDS/CPU_AS signals, and PDS/CPU_DSACK[1–0] (as appropriate to the card) have become inactive. /PDS_BG will be asserted only after all pending bus traffic has been completed.</td>
</tr>
<tr>
<td>PDS_BR_L</td>
<td>PDS bus request; logical-ORed with the bus request signals from other alternate bus masters to generate the CPU bus request. This signal is used to request the system’s bus from the processor. A PDS alternate bus master must never assert PDS_BR_L and BGACK_L simultaneously. The alternate bus master should be designed to deassert /BR and assert /BGACK at the same time (internal PDS clock edge). Refer to the Motorola MC68030 User’s Manual for a complete description of this signal. This line is active low.</td>
</tr>
<tr>
<td>RW_L</td>
<td>Indicates the type of bus cycle that is occurring: high indicates a read cycle and low indicates a write cycle.</td>
</tr>
<tr>
<td>SIZ[1–0]</td>
<td>Indicate the number of bytes remaining to be transferred in the current bus cycle.</td>
</tr>
</tbody>
</table>
Expansion Modules

System Support Signals on the PDS Connector

Table 4-7 lists the system support signals and describes their functions. Signal names that end with \_L are active low.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_RESET_L</td>
<td>PDS card reset signal from the main processor.</td>
</tr>
<tr>
<td>IPL_L[2–0]</td>
<td>Interrupt level; driven by the interrupt prioritizer in the system to encode the interrupt level. These signals are needed only by an alternate CPU on the card that must respond to interrupts.</td>
</tr>
<tr>
<td>MDM_TDM[3–0]</td>
<td>These lines are reserved for communication between the PDS slot and the modem slot. Contact Apple Developer Support for information about their use.</td>
</tr>
<tr>
<td>PDS_BUSCLK</td>
<td>PDS bus clock, 15.6672 MHz; supports PDS cards based on the MC68030 synchronous timing specifications.</td>
</tr>
<tr>
<td>PDS_CCU</td>
<td>PDS is CPU. This output, when asserted high by the Power Manager, disables the normal system CPU and enables the PDS card as an alternate system CPU. (This CPU function is not currently supported.)</td>
</tr>
<tr>
<td>PDS_DIGITAL_SND</td>
<td>Digital 1-bit sound signal (PWM or squarewave); gets mixed into both channels of the analog sound output.</td>
</tr>
<tr>
<td>PDS_INSERT_L</td>
<td>PDS card inserted; should be grounded on the PDS card. This signal is used by the system to detect the presence or absence of a PDS card.</td>
</tr>
<tr>
<td>PDS_IRQ_L</td>
<td>PDS interrupt request; when asserted low, causes the PDS card to interrupt the system at IPL level 2.</td>
</tr>
<tr>
<td>PDS_POWER_SW_L</td>
<td>PDS power switch; similar in operation to the ADB power signal. When driven low when the computer is off or in sleep mode, this signal causes the Power Manager to bring up the system.</td>
</tr>
<tr>
<td>PDS_RESET_L</td>
<td>PDS reset signal from the Power Manager IC; when asserted low, resets the PDS card.</td>
</tr>
<tr>
<td>SLEEP</td>
<td>Indicates that the system is in sleep mode (asserted high by the Power Manager).</td>
</tr>
<tr>
<td>SND_SCLK</td>
<td>Sound clock, 22.5792 kHz.</td>
</tr>
</tbody>
</table>

Descriptions of the Power Leads

Table 4-8 lists the electrical characteristics of the power leads on the PDS connector.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_5P_MAIN</td>
<td>+5V power to the PDS card</td>
</tr>
<tr>
<td>GND</td>
<td>PDS card ground</td>
</tr>
</tbody>
</table>
Characteristics of the PDS Signals

Table 4-9 lists the electrical characteristics of the signals on the PDS and indicates the signal directions with respect to the computer. The output drive capabilities are given in terms of standard TTL or LS TTL loads and include a safety factor.

Table 4-9  PDS signal load and drive limits

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Direction</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR[31–0]</td>
<td>Bidirectional</td>
<td>Output can drive one LS TTL load.</td>
</tr>
<tr>
<td>AS_L</td>
<td>Bidirectional</td>
<td>Output can drive one LS TTL load.</td>
</tr>
<tr>
<td>BERR_L</td>
<td>Open collector</td>
<td>Has internal 5K pull-up resistor.</td>
</tr>
<tr>
<td>BGACK_L</td>
<td>Bidirectional</td>
<td>Has internal 5K pull-up resistor.</td>
</tr>
<tr>
<td>DATA[31–16]</td>
<td>Bidirectional</td>
<td>Output can drive one LS TTL load.</td>
</tr>
<tr>
<td>DS_L</td>
<td>Bidirectional</td>
<td>Output can drive one LS TTL load.</td>
</tr>
<tr>
<td>DSACK_L[1–0]</td>
<td>Bidirectional</td>
<td>Output can drive one LS TTL load.</td>
</tr>
<tr>
<td>IO_RESET_L</td>
<td>Output</td>
<td>Has standard TTL output levels.</td>
</tr>
<tr>
<td>PDS_BG_L</td>
<td>Output</td>
<td>Can drive two standard LS TTL loads.</td>
</tr>
<tr>
<td>PDS_BR_L</td>
<td>Input</td>
<td>Has internal 100K pull-up resistor.</td>
</tr>
<tr>
<td>PDS_BUSCLK</td>
<td>Output</td>
<td>TTL voltage levels; source impedance 33 Ω.</td>
</tr>
<tr>
<td>PDS_CPU</td>
<td>Output</td>
<td>Can drive two standard LS TTL loads.</td>
</tr>
<tr>
<td>PDS_DIGITAL_SND</td>
<td>Input</td>
<td>Has internal 10K pull-up resistor.</td>
</tr>
<tr>
<td>PDS_INSERT_L</td>
<td>Input</td>
<td>Has internal 470K pull-up resistor.</td>
</tr>
<tr>
<td>PDS_IRQ_L</td>
<td>Input</td>
<td>Has internal 100K pull-up resistor.</td>
</tr>
<tr>
<td>PDS_POWER_SW_L</td>
<td>Open collector</td>
<td>Has internal 10K pull-up resistor.</td>
</tr>
<tr>
<td>PDS_RESET_L</td>
<td>Output</td>
<td>Can drive two standard LS TTL loads.</td>
</tr>
<tr>
<td>RW_L</td>
<td>Bidirectional</td>
<td>Output can drive one LS TTL load.</td>
</tr>
<tr>
<td>SIZ[1–0]</td>
<td>Bidirectional</td>
<td>Output can drive one LS TTL load.</td>
</tr>
<tr>
<td>SLEEP</td>
<td>Output</td>
<td>Can drive one standard TTL load.</td>
</tr>
<tr>
<td>SND_CLK</td>
<td>Output</td>
<td>Can drive two standard LS TTL loads.</td>
</tr>
</tbody>
</table>
Expansion Modules

PDS Power Consumption
This section specifies the maximum power available and provides guidelines for designing the power control circuitry on a PDS card.

Power Budget
The DC voltages supplied to the PDS connector and the maximum allowable current load for each voltage are listed in Table 4-10. The signal named V_BATT1 is not on the PDS connector but is on a separate battery connector accessible from the PDS card.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Voltage</th>
<th>Maximum current</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_5P_MAIN</td>
<td>5.0 V ± 5%</td>
<td>600 mA in operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 mA in sleep</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00 mA in shutdown</td>
</tr>
<tr>
<td>V_BATT1</td>
<td>8–16 V</td>
<td>300 mA in operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00 mA in shutdown</td>
</tr>
</tbody>
</table>

Note
The V_5P_MAIN signal is on the PDS connector itself. The V_BATT1 signal is only available on the battery terminals, which are accessible from a card installed in the battery well. ♦

Power and thermal restrictions limit the amount of power that can be dissipated by a PDS card to a maximum of 3 watts (average). The entire 3 watts can be from the 5-volt supply or from a combination of the two supply voltages, but the total cannot exceed 3 watts.

⚠️ WARNING
A card dissipating an average power of more than 3 watts may overheat and damage the computer’s circuitry or cause it to become inoperable. ⚠️

Power Control Circuits
To prevent excessive battery drain, the PDS card must include circuitry to put the card into a low-power mode when the computer is in sleep mode. In addition, because the PDS card is not used all the time even when the computer is in run mode, you may wish to include some form of power cycling to reduce current consumption when the PDS card is not being used.

Figure 4-5 shows recommended interface circuitry to isolate the PDS card so that its power can be controlled independently of the computer. Figure 4-6 shows a recommended circuit to limit the inrush current to the PDS. Figure 4-7 is a diagram of the recommended circuit to control the power. The section “User Installation of the PDS Card” beginning on page 42 discusses the reasons for an inrush current limiting circuit.
The bus interface controller on the PDS card should provide a register-controlled reset to the rest of the logic on the card. That arrangement makes it possible for software to reset the devices on the PDS card whenever it restores power to the card.
Expansion Modules

**Figure 4-6** Recommended inrush current limiting circuit

**Figure 4-7** Recommended power control circuit
Expansion Modules

Logic Design Guidelines for the PDS Card

This section provides information about the operation of the logic circuits on a PDS expansion card for the PowerBook 520 and 540 computers.

Addressing Guidelines

The PDS card can only be addressed by 32-bit addressing; the card appears in the address space $8000 0000–$FFFF FFFF. To match the conventions used by the Slot Manager, software should address the card as if it were in slot space $A$: either the 16 MB slot space $FA00 0000–$FAFF FFFF or the super slot space $A000 0000–$AFFF FFFF.

The expansion card must generate its own select signal from the address code signals on the connector. Figure 4-8 shows a typical logic circuit for generating the card select signal.

Figure 4-8 Generating the card select signal

Accessing Memory From the PDS Card

The PDS card in the PowerBook 520 and 540 computers reads and writes to memory in the same way as older PDS cards for the MC68030, as described in Designing Cards and Drivers for the Macintosh Family, third edition.

Timing Considerations

The PDS timing characteristics are identical to those of an MC68030 operating at a clock speed of 15.6677 MHz. See Designing Cards and Drivers for the Macintosh Family, third edition, for more information.

Mechanical Design Guidelines for the PDS Card

This section contains mechanical drawings showing the recommended design guidelines for PDS cards. Figure 4-12 shows the general appearance of the card with its connector.
PDS Card Shell
The PDS card is installed in a plastic shell that resembles a battery and fits into the battery well on the left side of the case. The plastic shell is made up of several components, including two clamshell halves, a bezel or cover, and an internal latch and spring. The shell fits into the computer’s case with close tolerances; the cover fits into the opening of the battery well. Developers who wish to provide a PDS card can obtain shell components from Apple’s vendor, subject to the usual restrictions. For information and authorization for obtaining the shell components, contact Apple’s Developer Support Center.

PDS Card Connector
The PDS card connector is a 90-pin shielded connector. The connector does not sit on one face of the card, but straddles the edge and requires contact pads on both sides of the card. Figure 4-12 shows a section through the connector.

The PDS card connector is available as part number C-93-1817-54 from AMP, Inc. For a specification sheet or information about obtaining this connector, contact AMP at the address shown on page 40.
Expansion Modules

**Note**
The PDS connector is designed so that when the PDS card is inserted, the first connection is the ground by way of the connector shells, then the power pins make contact, and last of all the signal lines. The callout numbered 11 in Figure 4-12 indicates the long pins for the power connections.

**PDS Card Design**
This section shows the mechanical outline and parts-placement guidelines for the PDS expansion card. Figure 4-12 shows the dimensions of the PDS expansion card. Figure 4-12 and Figure 4-13 show the restrictions on placement of components on the top and bottom of the card.

**Figure 4-11** Dimensions of the PDS card

*Note: Dimensions are in millimeters.*
Expansion Modules

**Figure 4-12** Restricted areas on the top of the PDS card

*Note:* Dimensions are in millimeters.

**Note**
The drawings in Figure 4-12, Figure 4-12, and Figure 4-13 are taken from Apple engineering drawing number 613-1706, revision 00. Developers who are members of Apple’s Developer Program may contact Apple’s Developer Support Center for information about possible later revisions of the drawings.

▲ **WARNING**
Do not exceed the dimensions shown in the figures. Cards that exceed these specifications may damage the computer and may be incompatible with future PowerBook models.
CHAPTER 4

Expansion Modules

Figure 4-13  Restricted areas on bottom of the PDS card

Note: Dimensions are in millimeters.

Modem Expansion Card

The modem expansion slot in the PowerBook 520 and 540 computers is designed for a coprocessor card. The modem slot provides a superset of the PDS connector signals. Like the PDS card, a card installed in the modem slot can have a configuration ROM and emulate a NuBus card through the Slot Manager interface. The modem slot interrupt maps to NuBus slot $B.

To monitor sound through the computer’s speaker, a modem card sends digitized sound data to the sound interface by way of the data bus and the Whitney IC. The Sound Manager has been modified to support such transfers of sound data.

The PowerBook Express Modem II, available in the U.S. and Canada, is a 14.4kbps fax/data modem; it supports the CCITT V.32bis and V.42bis standards as well as industry standard MNP5.
Expansion Modules

The PowerBook Express Modem II is designed around an Apple custom IC that includes the data pump, A-to-D and D-to-A converters, a microcontroller, and an interface to the computer’s I/O bus. That custom IC is also available to other modem developers; contact Apple Developer Services for information about licensing.

The design of the PowerBook 520 and 540 computers incorporates a separate DAA (telephone line interface). The DAA provided with the PowerBook Express Modem II works with many different telephone services throughout the world. See the section “DAA Interface Card” on page 64.

Signals on the Modem Connector

The modem connector is a 100-pin connector and is described in the section “Modem Card Connector” on page 62. The signals on the modem connector are divided by function into four groups: MC68030-compatible signals, system support signals, DAA signals, and power. The next section shows the signal assignments on the connector. The following sections describe the first three groups of signals. For information about the power pins, see the section “Modem Power” on page 62.

Signal Assignments on the Modem Connector

Table 4-11 shows the signal assignments on the modem connector. Entries in the table are arranged the same way as the pins on the connector: pin 1 across from pin 2, and so on. Signal names that end with _L are active low.

### Table 4-11  Signal assignments on the modem connector

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADDR[0]</td>
<td>B</td>
<td>2</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>—</td>
<td>8</td>
<td>ADDR[5]</td>
<td>B</td>
</tr>
<tr>
<td>13</td>
<td>ADDR[10]</td>
<td>B</td>
<td>14</td>
<td>V_5P_MAIN</td>
<td>—</td>
</tr>
<tr>
<td>19</td>
<td>GND</td>
<td>—</td>
<td>20</td>
<td>ADDR[15]</td>
<td>B</td>
</tr>
<tr>
<td>25</td>
<td>ADDR[20]</td>
<td>B</td>
<td>26</td>
<td>GND</td>
<td>—</td>
</tr>
</tbody>
</table>

continued
## Table 4-11  Signal assignments on the modem connector  (continued)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Pin</th>
<th>Signal</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>V_5P_MAIN</td>
<td>—</td>
<td>32</td>
<td>ADDR[25]</td>
<td>B</td>
</tr>
<tr>
<td>33</td>
<td>ADDR[26]</td>
<td>B</td>
<td>34</td>
<td>ADDR[27]</td>
<td>B</td>
</tr>
<tr>
<td>35</td>
<td>ADDR[28]</td>
<td>B</td>
<td>36</td>
<td>ADDR[29]</td>
<td>B</td>
</tr>
<tr>
<td>37</td>
<td>ADDR[30]</td>
<td>B</td>
<td>38</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>39</td>
<td>RW_L</td>
<td>B</td>
<td>40</td>
<td>ADDR[31]</td>
<td>B</td>
</tr>
<tr>
<td>41</td>
<td>SIZ[0]</td>
<td>B</td>
<td>42</td>
<td>SIZ[1]</td>
<td>B</td>
</tr>
<tr>
<td>43</td>
<td>GND</td>
<td>—</td>
<td>44</td>
<td>DATA[17]</td>
<td>B</td>
</tr>
<tr>
<td>49</td>
<td>DATA[20]</td>
<td>—</td>
<td>50</td>
<td>V_5P_MAIN</td>
<td>B</td>
</tr>
<tr>
<td>51</td>
<td>DATA[22]</td>
<td>B</td>
<td>52</td>
<td>DATA[23]</td>
<td>B</td>
</tr>
<tr>
<td>55</td>
<td>GND</td>
<td>—</td>
<td>56</td>
<td>DATA[27]</td>
<td>B</td>
</tr>
<tr>
<td>57</td>
<td>DATA[26]</td>
<td>B</td>
<td>58</td>
<td>DATA[29]</td>
<td>B</td>
</tr>
<tr>
<td>59</td>
<td>DATA[28]</td>
<td>B</td>
<td>60</td>
<td>DATA[31]</td>
<td>B</td>
</tr>
<tr>
<td>63</td>
<td>DSACK_L[0]</td>
<td>B</td>
<td>64</td>
<td>DSACK_L[1]</td>
<td>B</td>
</tr>
<tr>
<td>65</td>
<td>AS_L</td>
<td>B</td>
<td>66</td>
<td>DS_L</td>
<td>B</td>
</tr>
<tr>
<td>67</td>
<td>V_5P_MAIN</td>
<td>—</td>
<td>68</td>
<td>BERR_L</td>
<td>OC</td>
</tr>
<tr>
<td>69</td>
<td>MDM_BR_L</td>
<td>I</td>
<td>70</td>
<td>MDM_BG_L</td>
<td>O</td>
</tr>
<tr>
<td>71</td>
<td>BGACK_L</td>
<td>B</td>
<td>72</td>
<td>MDM_BUSCLK</td>
<td>O</td>
</tr>
<tr>
<td>73</td>
<td>IPL_L[0]</td>
<td>O</td>
<td>74</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>77</td>
<td>MDM_RESET_L</td>
<td>O</td>
<td>78</td>
<td>SLEEP</td>
<td>O</td>
</tr>
<tr>
<td>79</td>
<td>GND</td>
<td>—</td>
<td>80</td>
<td>IO_RESET_L</td>
<td>O</td>
</tr>
<tr>
<td>81</td>
<td>SND_SCLK</td>
<td>O</td>
<td>82</td>
<td>MDM_IRQ_L</td>
<td>I</td>
</tr>
<tr>
<td>83</td>
<td>PDS_POWER_SW</td>
<td>OC</td>
<td>84</td>
<td>n.c.</td>
<td>—</td>
</tr>
<tr>
<td>85</td>
<td>MDM_TDM[0]</td>
<td>B</td>
<td>86</td>
<td>MDM_TDM[1]</td>
<td>B</td>
</tr>
</tbody>
</table>

continued
MC68030-Compatible Signals on the Modem Connector

Table 4-12 gives signal descriptions for the MC68030-compatible signals on the modem connector. Signal names that end with _L are active low.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR[31–0]</td>
<td>System address bus; 32 bits, bidirectional.</td>
</tr>
<tr>
<td>AS_L</td>
<td>Indicates the occurrence of an active bus transaction; bidirectional.</td>
</tr>
<tr>
<td>BERR_L</td>
<td>System’s bus error watchdog timer and the processor’s /BERR line; open collector. This line is used for two operations: (1) to terminate an active bus transaction and (2) to cause a bus cycle to be retried when asserted with /HALT.</td>
</tr>
<tr>
<td>BGACK_L</td>
<td>Processor’s /BGACK line; bidirectional. Indicates that an alternate bus master has control of the system bus. The alternate master drives this line after it has been granted the system bus and has started a bus transaction. A bus master must never assert BR_L and BGACK_L simultaneously, but rather should be designed to deassert MDM_BR_L and assert BGACK_L at the same time (an internal modem slot clock edge).</td>
</tr>
<tr>
<td>DATA[31–16]</td>
<td>System data bus; 16 bits, bidirectional.</td>
</tr>
<tr>
<td>DS_L</td>
<td>Processor’s /DS signal; three state, output. This signal is used to manage the transmission of data to and from the processor. During a read cycle, the addressed bus slave places the requested data on the data bus when this signal is asserted. When this signal is asserted during a write, the processor drives the data bus with data to be written to the addressed bus slave. When this signal is released, the data bus is not driven.</td>
</tr>
<tr>
<td>DSACK_L[1–0]</td>
<td>Data strobe acknowledge. These signals perform two functions: (1) to indicate the completion of the data transfer portion of a bus transaction by the addressed slave and (2) to inform the bus master of the size of the data port being accessed during the current transaction.</td>
</tr>
</tbody>
</table>

**NOTE** Signal directions are defined from the main logic board side of the connector; I = input, O = output, B = bidirectional, and OC = open collector.
Expansion Modules

Table 4-12  MC68030-compatible signals on the modem connector  (continued)

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDM_BG_L</td>
<td>Modem slot bus grant; output. This signal is driven by the system’s bus arbitrator. When asserted low, the alternate bus master on the modem slot card may take ownership of the system bus after any pending bus traffic has completed (/BGACK, modem slot /CPU_AS signals, and modem slot /CPU_DSACK[1–0]—as appropriate to the card—have become inactive). MDM_BG_L will only be asserted after all pending bus traffic has been completed.</td>
</tr>
<tr>
<td>MDM_BR_L</td>
<td>Modem slot bus request; input. This signal is logical-ORed with the other alternate bus master’s bus request signals to generate the CPU bus request. This signal is used to request the system’s bus from the processor. An alternate bus master in the modem slot must never assert MDM_BR_L and BGACK_L simultaneously. The alternate bus master should be designed to deassert /BR and assert /BGACK at the same time (an internal modem slot clock edge). Refer to the Motorola MC68030 User’s Manual for a complete description of this signal. This line is active low.</td>
</tr>
<tr>
<td>RW_L</td>
<td>Indicates the data direction for bus transfers; bidirectional.</td>
</tr>
<tr>
<td>SIZ[1–0]</td>
<td>Indicate the transfer size for a given bus transaction; bidirectional.</td>
</tr>
</tbody>
</table>

System Support Signals on the Modem Connector

Table 4-16 gives signal descriptions for the system support signals on the modem connector. Signal names that end with _L are active low.

Table 4-13  System support signals on the modem connector

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPL_L[2–0]</td>
<td>These outputs are driven by the interrupt prioritizer in the system and encode the interrupt level. These signals are not used by a modem; they are needed only when a card in the modem slot has an alternate CPU that must respond to interrupts.</td>
</tr>
<tr>
<td>MDM_BUSCLK</td>
<td>This output pin is connected to the system 15.6672 MHz bus clock and allows for designs based on the 68030 synchronous timing specifications.</td>
</tr>
<tr>
<td>MDM_INSERT_L</td>
<td>This input is grounded by the modem slot card. This allows the system to detect the insertion or removal of a modem slot card and take the appropriate action.</td>
</tr>
<tr>
<td>MDM_IRQ_L</td>
<td>This input, when asserted low, causes the modem slot card to interrupt the system. The modem driver can set the IPL level to either 2 or 3 by a call to the Internal Modem Manager.</td>
</tr>
<tr>
<td>MDM_POWER_SW_L</td>
<td>This input is similar in operation to the ADB power signal. When driven low, this signal causes the Power Manager to bring up the system. It can also be used as a ring detect; it is masked on the main logic board with the system is running.</td>
</tr>
</tbody>
</table>

continued
Developers who write their own modem driver software should make calls to the modem hardware by way of the Internal Modem Manager, which is part of the system software. For information about the Internal Modem Manager, contact Apple’s Developer Support Center.

DAA Signals on the Modem Connector

Table 4-16 lists the signals connecting the modem slot with the DAA slot. These signals are used only by the modem card and the DAA card. For more information, see the section “Descriptions of the DAA Signals” on page 65.

Table 4-16  DAA signals on the modem connector

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDM_DAA[5–0]</td>
<td>These lines are multipurpose I/O digital signal lines operating at CMOS logic levels. They are the control and status signals between the modem card and the DAA card.</td>
</tr>
<tr>
<td>MDM_TX_P, MDM_TX_N</td>
<td>These lines are a differential pair of analog signals for transmitting analog signals from the modem card to the DAA card.</td>
</tr>
<tr>
<td>MDM_RX_P, MDM_RX_N</td>
<td>These lines are a differential pair of analog signals for receiving analog signals sent from the DAA card to the modem card.</td>
</tr>
</tbody>
</table>

Note
Developers who write their own modem driver software should make calls to the modem hardware by way of the Internal Modem Manager, which is part of the system software. For information about the Internal Modem Manager, contact Apple’s Developer Support Center.

DAA Signals on the Modem Connector
Expansion Modules

Modem Power

The power pins on the modem connector are V_5P_MAIN, the 5-volt supply line, and GND, the ground. The voltage on V_5P_MAIN is 5.0V ± 5%. The maximum current available for the modem card is

- 200 mA in normal operation
- 2 mA in sleep mode
- 0.00 mA in shutdown mode

Physical Design Guidelines for Modem Card

This section contains mechanical drawings showing the recommended design guidelines for the modem card.

Modem Card Connector

The modem card is connected to the main logic board by means of a 100-pin connector available from MOLEX, Inc. (part number 53395-1000). For a specification sheet and information about obtaining the connector, contact MOLEX, Inc.

194 S. Hillview Dr.
Milpitas, CA 95035
408-946-4700
AppleLink: MOLEX

Design of the Modem Card

Figure 4-14 is a top view of the modem expansion card showing its dimensions. Figure 4-15 and Figure 4-16 are top and bottom views showing the restrictions on the placement of components and traces.

Note

The drawings in Figure 4-14, Figure 4-15, and Figure 4-16 are taken from Apple engineering drawing number 056-0104, revision 05. Developers who are members of Apple’s Developer Program may contact Apple’s Developer Support Center for information about possible later revisions of the drawings. ♦

▲ WARNING

Do not exceed the dimensions shown in the mechanical drawings. Cards that exceed these specifications may damage the computer and may be incompatible with future PowerBook models. ▲
CHAPTER 4

Expansion Modules

Figure 4-14  Top view of the modem card

Figure 4-15  Restricted areas on the top of the modem card

Note: Dimensions are in millimeters [inches].


**Expansion Modules**

**Figure 4-16  **Restricted areas on the bottom of the modem card

![Restricted areas on the bottom of the modem card](image-url)

**Note:** Dimensions are in millimeters [inches].

---

**DAA Interface Card**

The modem architecture in the PowerBook 520 and 540 computers is unlike most others in that the telephone line interface unit, also called the DAA (data access arrangement), is separate from the rest of the modem’s controller and data pump. This architecture makes it possible to reduce the length of the high-speed buses and still locate the telephone line socket at the rear of the computer’s case.

In addition to describing the signals to the DAA card, this section describes a typical DAA card and then describes the multi-country DAA card that Apple includes with the PowerBook Express Modem II.

**Signals on the DAA Connector**

The DAA card plugs into a connector on the main logic board and the signal lines from that connector are routed to the modem-card connector. Except for its power leads, the DAA card is not connected to any other part of the computer.
Expansion Modules

Signal Assignments on the DAA Connector

Table 4-15 lists the signals on the DAA connector. Entries in the table are arranged the same way as the pins on the connector: pin 1 across from pin 2 and pin 19 across from pin 20. These signals are used only by the modem card and the DAA card.

Table 4-15  Signals on the DAA connector

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal name</th>
<th>Direction</th>
<th>Pin</th>
<th>Signal name</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>—</td>
<td>2</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>3</td>
<td>MDM_TX_P</td>
<td>O</td>
<td>4</td>
<td>MDM_RX_P</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>MDM_TX_N</td>
<td>O</td>
<td>6</td>
<td>MDM_RX_N</td>
<td>I</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>—</td>
<td>8</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>9</td>
<td>MDM_DAA[1]/H_UKNUM</td>
<td>O</td>
<td>10</td>
<td>MDM_DAA[0]/H_AWSMDAT</td>
<td>O</td>
</tr>
<tr>
<td>11</td>
<td>MDM_DAA[3]/H_OHRC</td>
<td>O</td>
<td>12</td>
<td>MDM_DAA[2]/L_RINGREADY</td>
<td>I</td>
</tr>
<tr>
<td>13</td>
<td>MDM_DAA[5]/H_AWSMCLK</td>
<td>O</td>
<td>14</td>
<td>MDM_DAA[4]/L_PHONE</td>
<td>I</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>—</td>
<td>16</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>17</td>
<td>n.c.</td>
<td>—</td>
<td>18</td>
<td>n.c.</td>
<td>—</td>
</tr>
<tr>
<td>19</td>
<td>MAIN_5V</td>
<td>—</td>
<td>20</td>
<td>MAIN_5V</td>
<td>—</td>
</tr>
</tbody>
</table>

Signal directions are defined from the main logic board side of the connector; I = input and O = output.

Descriptions of the DAA Signals

Table 4-16 gives descriptions of the signals on the DAA connector. All these signals are connected to the modem connector.

Table 4-16  Signals on the DAA connector

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDM_DAA[5–0]</td>
<td>These lines are multipurpose I/O digital signal lines operating at CMOS logic levels. They are the control and status signals between the modem card and the DAA card.</td>
</tr>
<tr>
<td>MDM_RX_P, MDM_RX_N</td>
<td>These lines are a differential pair of analog signals for receiving analog signals sent from the DAA card to the modem card. Impedance is 47K in parallel with 1.3 Henries.</td>
</tr>
<tr>
<td>MDM_TX_P, MDM_TX_N</td>
<td>These lines are a differential pair of analog signals for transmitting analog signals from the modem card to the DAA card. Impedance is 47K in parallel with 1.3 Henries.</td>
</tr>
</tbody>
</table>
Expansion Modules

The signals MDM_DAA[5-0] are multipurpose I/O digital control and status lines that operate at CMOS logic levels. The signals MDM_TX_P, MDM_TX_N, MDM_RX_P, and MDM_RX_N are two differential pairs for receiving and transmitting analog signals between the modem and the DAA. The nominal impedance of each differential pair is 47 K in parallel with a transformer winding that has an inductance of 1.3 Henries.

Physical Design of the DAA Card

This section contains information about two versions of the DAA card: a simple DAA such as might be used in a single country, and Apple's multi-country DAA card.

DAA Connector

The DAA card connector is a 20-pin SMT-type dual-row receptacle header manufactured by AMP Incorporated, Harrisburg, PA 17105. The connector on the main logic board is AMP part number 5-174904-0; the mating connector on the DAA board is AMP part number 5-174645-0. For specification sheets or information about obtaining these connectors, contact AMP at the address on page 40.

Design of a Simple DAA Card

Figure 4-17 shows the appearance of a simple DAA expansion card. Figure 4-18 and Figure 4-19 are top and bottom views showing the restrictions on the placement of components and traces.

![Figure 4-17  A simple DAA card](image-url)
**Figure 4-18** Top view of a simple DAA card

![Diagram of DAA card](image)

**Note:** Dimensions are in millimeters [inches].

**Note**
The drawings in Figure 4-18 and Figure 4-19 are taken from Apple engineering drawing number 062-1227, revision 00. Developers who are members of Apple’s Developer Program may contact Apple’s Developer Support Center for information about possible later revisions of the drawings.

**WARNING**
Do not exceed the dimensions shown in the mechanical drawings. Cards that exceed these specifications may damage the computer and may be incompatible with future PowerBook models.
The Multi-Country DAA Card

The multi-country DAA is included with the PowerBook Express Modem II. The multi-country DAA has a special telephone line socket that includes additional contacts. Those contacts are used with different adapter cables, called country keys, that allow the computer to identify the country where it is being used and to reconfigure the DAA interface accordingly.

The multi-country DAA, called Submarine, uses a custom chip set that makes it possible to create a single system that works with many different telephone systems. The multi-country DAA is also available for use with modems from other developers; for information about licensing it, contact Apple Developer Services.

Figure 4-20 shows the appearance of the multi-country DAA card. It fits into the same space as the simple DAA card.
Figure 4-20  View of the multi-country DAA card

Telephone cable socket
Software
This chapter describes the new ROM and system software features of the PowerBook 520 and 540 computers.

**ROM Software**

The ROM software in the PowerBook 520 and 540 computers is based on the ROM used in previous PowerBook computers, with enhancements to support the many new features of these computers. Some of the features this ROM was designed to support include the following:

- MC68040 and MC68LC040 microprocessors
- Pratt memory controller
- new Power Manager software
- new display controller
- new sound features
- Ethernet
- PDS support
- function keys
- smart batteries
- trackpad

The following sections describe each of these features.

**MC68040 and MC68LC040 Microprocessors**

The MC68LC040 microprocessors used in the PowerBook 520 and 540 computers differ from the MC68030 used in earlier PowerBook models in important ways such as the sizes of the caches and the configurations of the control registers. The differences make it necessary to use a different power cycling scheme on the PowerBook 520 and 540 computers.

The PowerBook 520 and 540 computers do not provide the economode reduced speed feature found on the PowerBook 160 and 180 models.

**Memory Controller Software**

The memory control routines have been rewritten to operate with the Pratt IC, which has a control register configuration different from that of the memory controller used in earlier PowerBook models. The memory initialization and size code has been rewritten to deal with

- larger ROM size
- new type of DRAM devices
- new memory configurations
Power Manager Software

Changes to the Power Manager software include

- power cycling and sleep mode for the MC68040 microprocessor
- support for the new smart batteries
- support for turning on and off power to the Ethernet interface

In addition to those changes, the PowerBook 520 and 540 computers include a new public API for power management; it is described in Chapter 6, “Power Manager Interface.”

Display Controller Software

The PowerBook 520 and 540 computers have a new custom IC, the CSC (color support chip), that provides the data and control interface to the flat panel display. The ROM software includes new video drivers for that IC.

The new drivers also support a wider range of external video monitors. See the section “Video Monitors” beginning on page 25 for information about external video.

Sound Features

The ROM software includes new sound driver software to support the new Sound Manager, which is part of the system software. The new driver software also supports the following new features:

- improved sound performance by way of a new interface to the Singer sound IC
- support for 16-bit stereo sound input
- support for automatic gain control in software
- mixing of sound output from the modem

The new ROM software also includes routines to arbitrate the control of the sound hardware between the modem and the Sound Manager.

Ethernet Driver

The driver for the Ethernet interface can now put a sleep task for Ethernet into the Power Manager’s sleep table. This sleep task first makes a control call to the Ethernet driver to prepare the Ethernet interface IC for sleep mode. The sleep task then makes a Power Manager call to turn off power to the IC. The sleep task installs a corresponding wake task that turns the interface power back on and reinitializes the interface IC.
PDS Support Software

As in the desktop Macintosh models, the PDS expansion card is treated like a NuBus expansion slot. The Slot Manager software looks for a configuration ROM on the PDS card and then treats the card like a NuBus expansion card. PDS cards with configuration ROM that conforms to the requirements described in Designing Cards and Drivers for the Macintosh Family, third edition, will work with the existing Slot Manager. The Slot Manager used with the PowerBook 520 and 540 computers is the same as that used in previous PowerBook models.

The interrupt from the PDS slot maps to an interrupt from expansion slot $A.$

Support for Function Keys

The keyboards on the PowerBook 520 and 540 computers have a row of 12 function keys across the top. Except for the function keys, the keyboards are similar to those on previous PowerBook models. The function keys are added to the key matrix in the same way as the function keys on the Apple Extended Keyboard and return the same key codes.

Intelligent Battery Support

The Power Manager IC communicates with the processors in the PowerBook Intelligent Batteries by means of a serial interface. The Power Manager’s command set has been expanded to provide system access to the data from the batteries.

Trackpad Software

The trackpad hardware, the Power Manager IC, and the system software work together to translate the movements of a finger across the surface of the trackpad into cursor movements.

The control registers for the trackpad hardware are part of the Power Manager IC. The Power Manager’s software takes the raw data from the trackpad hardware and converts it to the same format as ADB mouse data before sending it on to the system software.

The ADB software that supports the trackpad includes the Cursor Device Manager, which provides a standard interface for a variety of devices. The ADB software checks to see whether a device connected to the ADB port is able to use the Cursor Device Manager. For more information, see the January 1994 revision of Technical Note HW 01, ADB—The Untold Story: Space Aliens Ate My Mouse.
System Software

The PowerBook 520 and 540 computers are shipped with system software version 7.1. A new system enabler file is required in order to run System 7.1 on the PowerBook 520 and 540 computers.

Identifying the PowerBook 520 and 540 Computers

The correct method for software to use to identify the Macintosh model it is running on is by using the Gestalt Manager routines. The `gestaltMachineType` value returned by the PowerBook 520 and 540 computers is 72 (hexadecimal $48$). *Inside Macintosh: Overview* describes the Gestalt Manager and tells how to use the `gestaltMachineType` value to obtain the machine name string.

New System Enabler

The PowerBook 520 and 540 computers use a new system enabler. Many of the patches that were in the system enabler for earlier PowerBook models have been incorporated into the ROM. The new system enabler includes

- the new Sound Manager
- the sound input driver
- the sound output sifter
- the new Display Manager

Control Strip

The desktop on the PowerBook 520 and 540 computers has a new status and control element called the control strip. It is a strip of graphics with small button controls and indicators in the form of various icons. Figure 5-1 shows the control strip.

The control strip is a control panel that provides the operating environment for control strip modules. It runs on any Macintosh computer with System 7.0 or later.

The control strip is implemented in a private layer that appears in front of the windows in all the application layers so that the windows will not obscure it. The user can move the window for the control strip to any location on the display as long as the right or left edge of the strip is attached to the right or left edge of the display.
Software

The control strip has a tab on its unattached end. The user can drag the tab to adjust the length of the strip or hold down the Option key and drag the tab to move the strip to a new position. The user can hide the control strip, except for the tab, by clicking the tab. Clicking the tab when the control strip is hidden makes the control strip visible again. To make the control strip disappear completely, the user can click the Hide button in the Control Strip control panel, described on page 80.

The different parts of the control strip either display status information or act as buttons. When the user clicks a button, it is highlighted; some buttons also display additional elements such as pop-up menus.

By holding down the Option key and clicking a display area, the user can drag the display area to another position in the control strip. After the user rearranges the parts of the control strip, the new arrangement is saved when the computer is shut down and restarted.

The control strip software provides a standard screen location for a collection of individual modules that provide status and control functions. The control strip functions include:

- **AppleTalk Switch**: shows whether AppleTalk is on or off and lets the user turn AppleTalk on or off without having to open the Chooser.
- **Battery Monitor**: displays the status of the battery or batteries.
- **File Sharing**: displays the state of file sharing (on, off, or users connected), lets the user turn file sharing on or off, and lets the user open the Sharing Setup control panel.
- **HD Spin Down**: shows whether the internal hard disk is on or off; lets the user turn off the hard disk.
- **Power Settings**: lets the user select between maximum conservation or maximum performance without opening the PowerBook control panel; also lets the user open the PowerBook control panel.
- **Sleep Now**: puts the computer into sleep mode.
- **Sound Volume**: lets the user select the sound volume.
- **Video Mirroring**: lets the user turn video mirroring on or off if an external monitor is connected.

**Note**

Several of the functions of the control strip were implemented in separate control panels on earlier PowerBook models. ♦

Developers can add modules to the control strip. For information, see the section “Adding Control Strip Modules” beginning on page 80.

**Selecting SVGA**

An SVGA monitor is considered to be a member of the VGA monitor family. To work with the PowerBook 520 and 540 computers, an SVGA monitor must be capable of operating as a standard VGA monitor. With the monitor operating in VGA mode, the user selects the Monitors control panel and clicks the Options button. The user then selects SVGA in the Monitor types list shown in the Options dialog box.
Similarly, the PowerBook 520 and 540 computers treat the different scanning modes available on multiscan monitors as a family of settings from which the user can select by means of the Options dialog box in the Monitors control panel.

New Control Panels

Several control panels are new or revised for the PowerBook 520 and 540 computers. Figure 5-2 shows the appearance of the new control panel icons.

The new control panels are

- PowerBook control panel
- PowerBook Setup control panel
- PowerBook Display control panel
- Trackpad control panel
- Control Strip control panel
- AutoRemounter control panel

The following sections describe the new control panels.

PowerBook Control Panel

The PowerBook control panel includes several controls that allow the user to balance performance against battery conservation. Figure 5-3 and Figure 5-4 show the PowerBook control panel in two modes: easy, which allows the user to balance performance against battery life with a single slider, and custom, which gives the user control of individual power conservation features.
CHAPTER 5

Software

Figure 5-3  PowerBook control panel in easy mode

In custom mode, the PowerBook control panel lets the user set the time delays in minutes until the computer dims the display, goes into sleep mode, and goes into shutdown mode. The user can also select processor cycling, which slows down the processor and saves power.

PowerBook Setup Control Panel

Figure 5-5 shows the PowerBook Setup control panel.

Figure 5-5  PowerBook Setup control panel

The PowerBook Setup control panel allows the user to set the SCSI ID value for the internal hard disk so that the PowerBook 520 and 540 computers can be used as a hard disk connected to another computer. The control panel also allows the user to program an automatic wake-up time.
Software

PowerBook Display Control Panel

Figure 5-6 shows the PowerBook Display control panel.

![PowerBook Display control panel](image)

The PowerBook Display control panel allows the user with an external video monitor to select video mirroring mode. In normal operation, an external monitor provides additional desktop space, as on any Macintosh computer with multiple displays. When the user selects video mirroring mode, the built-in display is duplicated or mirrored on the external video monitor. For example, mirroring mode is useful for presentations when the user wishes to see the same display that the audience sees on the video monitor.

Trackpad Control Panel

Figure 5-7 shows the Trackpad control panel.

![Trackpad control panel](image)

The Trackpad control panel allows the user to set the characteristics of the trackpad to match the way the user operates it. The user can set the scale and sensitivity of the trackpad and the time interval for a double click.
Software

Control Strip Control Panel

Figure 5-8 shows the Control Strip control panel. The user can hide or show the control strip by clicking the corresponding button in the control panel.

![Figure 5-8 Control Strip control panel](image)

AutoRemounter Control Panel

Figure 5-9 shows the AutoRemounter control panel.

![Figure 5-9 AutoRemounter control panel](image)

When the user has connected a PowerBook 520 or 540 computer to another computer to use its internal hard disk as a shared disk, the AutoRemounter control panel allows the user to determine how the PowerBook computer operates when it resumes operation after a period in sleep mode.

Adding Control Strip Modules

The control strip is implemented in software as a shell with individual control and status modules added. The control strip software draws the strip that acts as the background for the individual modules. Each module is responsible for drawing the icons and other objects that make up its user interface.
Contents of Module Files

The only required resource in a module file is a resource containing the code necessary for the module to interact with the control strip. A module file may contain more than one code resource if it is to provide multifunctional support. In that case, each module in the file is loaded and initialized separately and treated as an independent entity.

If a file contains only a single code resource, the resource may be unnamed, and the module will be referenced by its filename. If more than one module is contained within a module file, each module is required to have a unique name describing its functionality.

All other resources in a module file are optional, but there are several that are recommended in order to support a custom icon and version information. The recommended resources are

- 'BNDL'
- 'FREF'
- 'ICN#', 'icl4', 'icl8', 'ics#', 'ics4', 'ics8'
- signature resource (same type as file's creator)
- 'vers', ID=1

Developers should confine their resources to the range 256–32767.

Module Interface

The module’s interface to the control strip consists of a code resource of type 'sdev'. This code is responsible for performing all of the functions required by the control strip (see below) as well as any functions that are custom to the module itself. The module’s entry point is at the beginning of the resource and is defined as

```pascal
long ControlStripModule(long message,
                        long params,
                        Rect *statusRect,
                        GrafPtr statusPort);
```

Interactions between a module and the control strip are managed by passing messages to the module to tell it what to do or to obtain information about the module and its capabilities. Each module is required to observe Pascal register saving conventions. A module may trash registers D0, D1, D2, A0, and A1, but must preserve all other registers across its call.

Field descriptions

- **message**: A message number, from the list in the section “Control Strip Module Messages”, that tells the module what action to perform.
- **params**: This is the result returned by the initialize call to the module. This would typically be the handle to the module’s private variables since modules can’t have global variables. It will be passed to the module on all subsequent calls.
statusRect  A pointer to a rectangle defining the area that a module may draw within.

statusPort  A pointer to the control strip’s graphics port. This will be either a color or black-and-white graphics port depending on which PowerBook Model the control strip is running on.

The result value returned by the module will vary depending on the message sent to it. Results for each message are described in the sections on the individual messages.

Module Reentrancy

Any module that makes calls to routines such as getNextEvent, ModalDialog or pop-upMenuSelect should assume that it could be called reentrantly; that is, the module could be called again while the initial call is still in progress. Situations to avoid are such things as reusing a single parameter block for multiple calls, or indiscriminately locking and unlocking your global variables around the module’s invocation.

Instead of using a single parameter block, it’s better, if possible, to allocate the parameter block on the stack. In the case of asynchronous calls, using the stack could cause problems; in that case, preventing the block’s reuse should be sufficient.

If you need to lock and unlock your global variables, it’s better to use HGetState and HLock at the beginning of the call, and HSetState at the end, so that the state is restored to what it was on entry.

Control Strip Module Reference

Control strip modules interact with the control strip software in three ways: by accepting messages, by calling utility routines, and by calling Gestalt selectors. The next three sections describe each of those interactions.

Control Strip Module Messages

All control strip modules must respond to messages from the control strip. The following messages have been defined:

<table>
<thead>
<tr>
<th>Message name</th>
<th>Message number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdevInitModule</td>
<td>0</td>
<td>Initialize the module</td>
</tr>
<tr>
<td>sdevCloseModule</td>
<td>1</td>
<td>Clean up before being closed</td>
</tr>
<tr>
<td>sdevFeatures</td>
<td>2</td>
<td>Return the feature bits</td>
</tr>
<tr>
<td>sdevGetWidth</td>
<td>3</td>
<td>Return the width of the module’s display</td>
</tr>
<tr>
<td>sdevPeriodicTickle</td>
<td>4</td>
<td>Periodic tickle when nothing else is happening</td>
</tr>
<tr>
<td>sdevDrawStatus</td>
<td>5</td>
<td>Update the interface in the control strip</td>
</tr>
</tbody>
</table>
sdevInitModule

The sdevInitModule message is the first message sent to a module after the module has been loaded from its file. Initialization allows the module to initialize its variables and to determine whether it can run on a particular machine: for example, if the module’s function is to display battery information it can run only on a PowerBook.

The module needs to load and detach any resources in the module’s resource file that will be used, because the resource file will not be kept permanently open. What that means is that your code can’t use GetResource() or the like to retrieve the handle to one of the module’s resources on a subsequent call. Typically you would allocate space in your global variables for handles to those detached resources.

The sdevInitModule message returns a result depending on its success at installing itself. A positive result (≥0) indicates successful installation. This result value will be passed to the module on all subsequent calls. A negative result indicates an error condition, and installation of the module is aborted by the control strip software. The module will not receive a close message when installation has been aborted.

sdevCloseModule

The sDevCloseModule message is sent to a module when it should be closed. Typically the module itself will decide when this ought to happen. When the module receives this message, it should dispose of all the detached resources it loaded as well as its global storage. No result is expected.

sdevFeatures

The sdevFeatures message queries the module for the features it supports. It returns as its result a bitmap consisting of 1 bits for supported features and 0 bits for unsupported features. All undefined bits are reserved by Apple for future features, and must be set to 0. The bits are defined as

<table>
<thead>
<tr>
<th>Message name</th>
<th>Message number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdevMouseClick</td>
<td>6</td>
<td>User has clicked on the module’s display area</td>
</tr>
<tr>
<td>sdevSaveSettings</td>
<td>7</td>
<td>Save any changed settings in the module’s preferences file</td>
</tr>
<tr>
<td>sdevShowBalloonHelp</td>
<td>8</td>
<td>Display a help balloon, if the module has one</td>
</tr>
</tbody>
</table>
Software

The control strip assumes that the module only displays status information with no user interaction.

`sdevDontAutoTrack`  1 If this bit is set, the control strip highlights the module’s display and then calls the module to perform mouse tracking; this bit is usually set when, for example, a module has a pop-up menu associated with it. If this bit is cleared, the control strip tracks the cursor until the mouse button is released, then sends an `sdevMouseClicked` message to the module to notify it that there was a mouse-down event.

`sdevHasCustomHelp`  2 If this bit is set, the module is responsible for displaying its own help messages, which can be customized depending on its current state. If the bit is cleared, the control strip will display a generic help message when the cursor passes over the module’s display area and Balloon Help is on.

`sdevKeepModuleLocked`  3 If this bit is set, the module’s code will be kept locked in the heap. This bit should be set only if the module is passing the address of one of its routines to the outside world (for example, installing itself in a queue).

`sdevGetWidth`  5 The `sdevGetWidth` message is sent to a module to determine how much horizontal space (in pixels) its display currently requires on the control strip. The module should return the number of pixels as its result. The returned width should not be the maximum width it requires for any configuration, but should reflect how much space it currently requires, because it’s possible for a module to request that its display be resized.

**IMPORTANT**
You should be conservative in your use of control strip display space, which is limited. Because several modules could be requesting space, it’s possible that your module could be shoved off the end.

`sdevPeriodicTickle`  5 The `sdevPeriodicTickle` message is passed to the module periodically to allow the module to update its display due to changes in its state. You should not assume any minimum or maximum interval between tickles. The module should return, as its result, some bits that signal requests for actions from the control strip software. All undefined
bits in the result are reserved for future use by Apple and must be set to 0. The bits are defined as:

- **sdevResizeDisplay** (0): If this bit is set, the module needs to resize its display. The control strip sends a `sdevGetDisplayWidth` message to the module and then update the control strip on the display.

- **sdevNeedToSave** (1): If this bit is set, the module needs to save changed settings to disk. The control strip software will mark the request but may defer the actual save operation to a better time (for example, when the hard disk is spinning).

- **sdevHelpStateChange** (2): If this bit is set, the module's state has changed so it needs to update its help message. If a help balloon is being displayed for this module, the control strip software will remove it and put up a new help balloon for the current state.

- **sdevCloseNow** (3): If this bit is set, the module is requesting to be closed. The control strip software will call the module to save its settings, then call it again to close itself.

### sdevDrawStatus

The `sdevDrawStatus` message indicates that the module has to redraw its display to reflect the most recent state. This message is typically sent when the user clicks on the module's display area, when any of the module's displays is resized, or when the control strip itself needs to be updated, perhaps in response to a screen saver deactivation.

The `statusRect` parameter points to a rectangle bounding the module’s display area, in local coordinates. All drawing done by a module within the bounds of the control strip must be limited to the module’s display rectangle. The graphics port’s `clipRgn` will be set to the visible portion of this rectangle so you can draw all the elements in the display. If you need to change the `clipRgn`, you should observe the initial `clipRgn` to avoid drawing over other items in the control strip.

### sdevMouseClick

When the user clicks in a module’s display area, the control strip software calls the module with the `sdevMouseClick` message if the `sdevWantMouseClicks` bit is set in the module’s features.
If the sdevDontAutoTrack bit is also set, the control strip draws the module’s display in its highlighted state and then sends the sdevMouseClick message to the module. If the sdevDontAutoTrack bit is not set, the control strip software tracks the cursor until the mouse button is released. If the cursor is still within the module’s display area, the control strip software sends the sdevMouseClick message to notify the module that a click occurred. In either case, the module can then perform the appropriate function in response to a mouse-down event.

This message returns the same result as the sdevPeriodicTickle message.

**sdevSaveSettings**

The sdevSaveSettings message is passed to the module when the control strip software has determined that it’s a good time to save configuration information to the disk. This message will be sent only if the module had previously set the sdevNeedToSave bit in the result of a sdevPeriodicTickle or sdevMouseClick message. The call returns an error code (File Manager, Resource Manager, or the like) indicating the success of the save operation. The control strip software will continue to send this message to the module until the module returns a result of 0, indicating a successful save.

**sdevShowBalloonHelp**

The control strip software calls the module with the sdevShowBalloonHelp message if Balloon Help is turned on, the module has previously set the sdevHasCustomHelp bit in its features, and the cursor is over the module’s display area. The module should then call the Help Manager to display a help balloon describing the current state of the module. The module should return a value of 0 if it’s successful or an appropriate error result if not.

**Utility Routines**

The control strip software provides a set of utility routines that are available to control strip modules. They are provided to promote a consistent user interface within the control strip and to reduce the amount of duplicated code that each module would have to include to support common functions.

The utility routines are called through a selector-based trap, _ControlStripDispatch ($AAF2). If an unimplemented routine is called, it will return paramErr as the result.

**IMPORTANT**

These routines should not be called at interrupt time because they all move memory.
SBIsControlStripVisible

You can use the SBIsControlStripVisible routine to find out whether the control strip is visible.

```pascal
Boolean SBIsControlStripVisible();
```

The SBIsControlStripVisible routine returns a Boolean value indicating whether or not the control strip is currently visible. It returns a value of `true` if the control strip is visible, or a value of `false` if it's hidden.

It is possible for this call to return a value of `true` even when the control strip is not visible. That happens whenever the control strip is not accessible in the current environment. As soon as that condition changes, the control strip becomes visible again and the returned value correctly reflects the actual state.

SBShowHideControlStrip

You can use the SBShowHideControlStrip routine to show or hide the control strip.

```pascal
void SBShowHideControlStrip(Boolean showIt);
```

The SBShowHideControlStrip routine determines the visibility state for the control strip based on the value of the `showIt` parameter. Passing a value of `true` makes the control strip visible, and passing a value of `false` hides it. Modules shouldn’t typically need to call this routine, but it’s provided as a means for other software to hide the control strip when it might get in the way.

Calling SBShowHideControlStrip with a `showIt` value of `true` may or may not show the control strip, depending on the current environment: if the control strip is not accessible, it does not become visible. If a `showIt` value of `true` is passed to this routine, then when the environment changes, the control strip will become visible.

SBSafeToAccessStartupDisk

You can use the SBSafeToAccessStartupDisk routine to find out whether the internal hard disk is spinning so that your software can determine whether to make a disk access or postpone it until a time when the disk is already spinnings.

```pascal
Boolean SBSafeToAccessStartupDisk();
```

The SBSafeToAccessStartupDisk routine returns a Boolean value of `true` if the disk is spinning and `false` if it is not.
SBOpenModuleResourceFile

You can use the SBOpenModuleResourceFile routine to open a module resource file.

```pascal
short SBOpenModuleResourceFile(OSType fileCreator);
```

The SBOpenModuleResourceFile routine opens the resource fork of the module file whose creator is `fileCreator`, and return the file’s reference number as its result. If the file cannot be found or opened, SBOpenModuleResourceFile returns a result of -1.

SBOpenModuleResourceFile provides a means for a module to load in large or infrequently used resources that it doesn’t usually need, but that it requires for a particular operation.

SBLoadPreferences

You can use the SBLoadPreferences routine to load a resource from a preferences file.

```pascal
OSErr SBLoadPreferences(ConstStr255Param prefsResourceName, Handle *preferences);
```

The SBLoadPreferences routine loads a resource containing a module’s configuration information from the control strip’s preferences file. The `PrefsResourceName` parameter points to a Pascal string containing the name of the resource. The `Preferences` parameter points to a variable that will hold a handle to the resource read from the file. The handle does not need to be preallocated.

If either `prefsResourceName` or `preferences` contains a nil pointer, SBLoadPreferences does nothing and returns a result of `paramErr`. If the resource is successfully loaded, it returns a result of 0. SBLoadPreferences can also return other Memory Manager and Resource Manager errors if it fails during some part of the process.

SBSavePreferences

You can use the SBSavePreferences routine to save a resource to a preferences file.

```pascal
OSErr SBSavePreferences(ConstStr255Param prefsResourceName, Handle preferences);
```

The SBSavePreferences routine saves a resource containing a module’s configuration information to the control strip’s preferences file. The `PrefsResourceName` parameter
points to a Pascal string containing the name of the resource. The `preferences` parameter contains a handle to a block of data which will be written to the file.

If either `prefsResourceName` or `preferences` has a nil value, `SBSavePreferences` does nothing and returns a result of `paramErr`. If the resource is successfully saved, `SBSavePreferences` returns a result of 0. `SBSavePreferences` can also return other Memory Manager and Resource Manager errors if it fails during some part of the process.

### SBGetDetachedIndString

You can use the `SBGetDetachedIndString` routine to get a string from a detached resource.

```
pascal void SBGetDetachedIndString(StringPtr theString,
                                   Handle stringList,
                                   short whichString);
```

The `SBGetDetachedIndString` routine is the detached resource version of `GetIndString`. The parameter `theString` points to a Pascal string; `stringList` is a handle to a detached `STR#` resource; and `whichString` is the index (1–n) into the array of Pascal strings contained in the detached resource. `SBGetDetachedIndString` will copy the string whose index is `whichString` into the space pointed to by `theString`. If `whichString` is out of range, `SBGetDetachedIndString` will return a zero-length string.

### SBGetDetachIconSuite

You can use the `SBGetDetachIconSuite` routine to set up a detached icon suite.

```
pascal OSErr SBGetDetachIconSuite(Handle *theIconSuite,
                                   short theResID,
                                   unsigned long selector);
```

The `SBGetDetachIconSuite` routine creates a new icon suite, loads all of the requested icons, and then detaches the icons. The parameter `theIconSuite` points to the location where the handle to the icon suite will be stored; the parameter `theResID` is the resource ID of the icons that make up the icon suite; and the parameter `selector` tells which icons should be loaded into the suite. The `selector` parameter should typically contain one (or a combination of) the following values:

- `svAllLargeData` 0x000000FF load large 32-by-32-pixel icons ('ICN#', 'icl4', 'icl8')
Software

svAllSmallData 0x0000FF00 load small 16-by-16-pixel icons ('ics#', 'ics4', 'ics8')
svAllMiniData 0x00FF0000 load mini 12-by-12-pixel icons ('icm#', 'icm4', 'icm8')

These values may be OR-ed together to load combinations of icon sizes. SBGetDetachIconSuite returns an appropriate error code if it's unsuccessful, or 0 if if was able to load the icon suite. Note that if none of the icons comprising the icon suite could be found, the call returns the error resNotFound.

IMPORTANT
You should call SBGetDetachIconSuite only when the module's resource file is open, which is typically the case during a module's initialization call.

SBTrackpopupMenu

You can use the SBTrackpopupMenu routine to manage a pop-up menu.

pascal short SBTrackpopupMenu(const Rect *moduleRect,
                             MenuHandle theMenu);

The SBTrackpopupMenu routine handles setting up and displaying a pop-up menu associated with a module. The module should pass a pointer to its display rectangle and a handle to the menu to use. The menu will be displayed just above the module's display rectangle, allowing the user to view the current configuration or to change the settings. SBTrackpopupMenu returns which menu item was selected, or 0 if no item was selected because the user moved the cursor outside the menu's bounds.

IMPORTANT
Menus are displayed in the control strip's font, so don't use the CheckItem() routine to mark menu items, because a checkmark is supported only in the system font. Use the SetItemMark() routine instead and pass it a bullet (•).

SBTrackSlider

You can use the SBTrackSlider routine to display and set an arbitrary parameter.

pascal short SBTrackSlider(const Rect *moduleRect,
                          short ticksOnSlider,
                          short initialValue);
The SBTrackSlider routine displays an unlabeled slider above the module’s display rectangle. You can use the slider for displaying and setting the state of an arbitrary parameter. The parameter ModuleRect contains a pointer to the module’s display rectangle; ticksOnSlider is the upper bounds of the value returned by the slider; and initialValue is the starting position (0 to ticksOnSlider-1). When the user releases the mouse button, SBTrackSlider returns the final position.

**SBShowHelpString**

You can use the SBShowHelpString routine to display a help balloon.

```pascal
OSErr SBShowHelpString(const Rect *moduleRect, StringPtr helpString);
```

The SBShowHelpString routine displays a module’s help balloon. The module passes a pointer to its display rectangle and a pointer to a Pascal string, and the routine displays the balloon if possible. If the help string has a length of 0 or the Help Manager is unable to display a balloon, an error result is returned. If SBShowHelpString successfully displays the help balloon, it returns a result of 0.

**SBGetBarGraphWidth**

You can use the SBGetBarGraphWidth routine to find out the how wide a bar graph drawn by SBDrawBarGraph (described next) will be so that a module can calculate its display width.

```pascal
short SBGetBarGraphWidth(short barCount);
```

The SBGetBarGraphWidth routine returns the width of a bar graph containing barCount segments. If barCount has a value less than 0, the SBGetBarGraphWidth routine returns a width of 0.

**SBDrawBarGraph**

You can use the SBDrawBarGraph routine to draw a bar graph.

```pascal
void SBDrawBarGraph(short level, short barCount, short direction, Point barGraphTopLeft);
```
The **SBDrawBarGraph** routine draws a bar graph containing the number of segments specified by the `barCount` parameter in a module’s display area. If the value of `barCount` is less than or equal to 0, **SBDrawBarGraph** does nothing.

The bar graph is drawn relative to the location specified by `barGraphTopLeft`. Figure 5-10 shows the way the point `barGraphTopLeft` determines the position of the bar graph.

**Figure 5-10** Positioning a bar graph

```
#define BarGraphSlopeLeft   -1   // max end of sloping graph is on the left
#define BarGraphFlatRight   0    // max end of flat graph is on the right
#define BarGraphSlopeRight  1    // max end of sloping graph is on the right
```

The `level` parameter determines how many segments are highlighted. The value of `level` should be in the range of 0 to `barCount-1`. If the value of `level` is less than 0, no segments in the bar graph are highlighted; if `level` is greater than or equal to `barCount`, all segments in the bar graph are highlighted.

The direction parameter specifies which way the bar graph will be drawn to show a larger level. It should be one of the following values:

```
#define BarGraphSlopeLeft   -1   // max end of sloping graph is on the left
#define BarGraphFlatRight   0    // max end of flat graph is on the right
#define BarGraphSlopeRight  1    // max end of sloping graph is on the right
```

Figure 5-11 shows the resulting bar graph for each direction value. The arrows indicate which way an increasing level value is displayed. For sloped versions of the bar graph, the number of segments specified by the `barCount` value may not be larger than 8. If a larger `barCount` value is passed, **SBDrawBarGraph** draws nothing.

**Figure 5-11** Directions of a bar graph
SBModalDialogInContext

You can use the SBModalDialogInContext in place of ModalDialog routine to keep background applications from getting run while your modal dialog window is visible.

```pascal
void SBModalDialogInContext(ModalFilterProcPtr filterProc, short *itemHit);
```

The SBModalDialogInContext routine is a special version of ModalDialog that doesn’t allow background applications to get time while a modal dialog window is visible. You should use SBModalDialogInContext when you don’t want any context switching to occur.

Gestalt Selectors

The control strip software installs two Gestalt selectors to return information to the outside world. One selector returns software attributes, and the other returns the software version.

gestaltControlStripAttr

The selector `gestaltControlStripAttr` returns 32 bits describing the software attributes of this version of the control strip. Currently only the following bit is defined:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><code>gestaltControlStripExists</code></td>
</tr>
<tr>
<td>1</td>
<td>control strip is installed</td>
</tr>
</tbody>
</table>

gestaltControlStripVersion

The selector `gestaltControlStripVersion` returns the version of control strip software that is installed. The format of the returned version is the same as that of the numeric part of a ‘vers’ resource, that is:

- Bits 31-24: Major part of the version, in BCD
- Bits 23-20: Minor part of the version, in BCD
- Bits 19-16: Bug release version, in BCD
- Bits 15-8: Release stage:
  - $80$ = final
  - $60$ = beta
  - $40$ = alpha
  - $20$ = development
- Bits 7-0: Revision level of nonreleased version, in binary

Thus, if the software version were 1.5.3b25, the `gestaltControlStripVersion` selector would return $01536019$. 
Power Manager Interface
This chapter describes the new application programming interface (API) to the Power Manager control software. Developers who provide expanded control panel software for the PowerBook Duo 280 and 280c computers will no longer need access to the Power Manager’s internal data structures.

About the Power Manager Interface

Developers have written control panel software for previous PowerBook models that gives the user more control over the power management settings than is provided in the PowerBook control panel. Because that software reads and writes directly to the Power Manager’s private data structure and parameter RAM, the software needs to be updated any time Apple Computer makes a change to the internal operation of the Power Manager.

System software for the PowerBook 520 and 540 computers and for future PowerBook models includes interface routines for program access to the Power Manager functions, so it is no longer necessary for applications to deal directly with the Power Manager’s data structures. The new routines provide access to most of the Power Manager’s parameters. Some functions will be reserved because of their overall effect on the system. The interface is extensible; it will probably grow over time to accommodate new kinds of functions.

Things That May Change

By using the Power Manager interface, developers can isolate themselves from future changes to the internal operation of the Power Manager software.

IMPORTANT
Apple Computer reserves the right to change the internal operation of the Power Manager software. Developers should not make their applications depend on the Power Manager’s internal data structures or parameter RAM.

Starting with the PowerBook 520 and 540 models, developers should not depend on the Power Manager’s internal data structures staying the same. In particular, developers should beware of the following assumptions regarding different PowerBook models:

- assuming that timeout values such as the hard disk spindown time reside at the same locations in parameter RAM
- assuming that the power cycling process works the same way or uses the same parameters
- assuming that direct commands to the Power Manager microcontroller are supported on all models
Checking for Routines

Before calling any of the Power Manager interface routines, it’s always a good idea to call the Gestalt Manager to see if they’re present on the computer. The Gestalt Manager is described in *Inside Macintosh: Overview.*

A new bit has been added to the `gestaltPowerMgrAttr` selector:

```
#define gestaltPMgrDispatchExists 4
```

If that bit is set to 1, then the routines are present.

Because more routines may be added in the future, one of the new routines simply returns the number of routines that are implemented. The following code fragment determines both that the routines in general exist and that at least the hard disk spindown routine exists.

```c
long   pmgrAttributes;
Boolean routinesExist;

routinesExist = false;
if (! Gestalt(gestaltPowerMgrAttr, &pmgrAttributes))
    if (pmgrAttributes & (1<<gestaltPMgrDispatchExists))
        if (PMSelectorCount() >= 7)
            routinesExist = true;
```

**Warning**

If you call a routine that does not exist, the call to the public Power Manager trap (if the trap exists) will return an error code, which your program could misinterpret as data.

Power Manager Interface Routines

This section tells you how to call the interface routines for the Power Manager software. The interface routines are listed here in the order of their routine selector values, as shown in Table 6-1.

**Assembly-language note:**

All the routines share a single trap, `_PowerMgrDispatch ($A09E)`. The trap is register based: parameters are passed in register D0 and sometimes also in A0. A routine selector value passed in the low word of register D0 determines which routine is executed.

---

**C H A P T E R  6**

Power Manager Interface
## Table 6-1 Interface routines and their selector values

<table>
<thead>
<tr>
<th>Routine name</th>
<th>Selector value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMSelectorCount</td>
<td>0 $00</td>
</tr>
<tr>
<td>PMFeatures</td>
<td>1 $01</td>
</tr>
<tr>
<td>GetSleepTimeout</td>
<td>2 $02</td>
</tr>
<tr>
<td>SetSleepTimeout</td>
<td>3 $03</td>
</tr>
<tr>
<td>GetHardDiskTimeout</td>
<td>4 $04</td>
</tr>
<tr>
<td>SetHardDiskTimeout</td>
<td>5 $05</td>
</tr>
<tr>
<td>HardDiskPowered</td>
<td>6 $06</td>
</tr>
<tr>
<td>SpinDownHardDisk</td>
<td>7 $07</td>
</tr>
<tr>
<td>IsSpindownDisabled</td>
<td>8 $08</td>
</tr>
<tr>
<td>SetSpindownDisable</td>
<td>9 $09</td>
</tr>
<tr>
<td>HardDiskQInstall</td>
<td>10 $0A</td>
</tr>
<tr>
<td>HardDiskQRemove</td>
<td>11 $0B</td>
</tr>
<tr>
<td>GetScaledBatteryInfo</td>
<td>12 $0C</td>
</tr>
<tr>
<td>AutoSleepControl</td>
<td>13 $0D</td>
</tr>
<tr>
<td>GetIntModemInfo</td>
<td>14 $0E</td>
</tr>
<tr>
<td>SetIntModemState</td>
<td>15 $0F</td>
</tr>
<tr>
<td>MaximumProcessorSpeed</td>
<td>16 $10</td>
</tr>
<tr>
<td>CurrentProcessorSpeed</td>
<td>17 $11</td>
</tr>
<tr>
<td>FullProcessorSpeed</td>
<td>18 $12</td>
</tr>
<tr>
<td>SetProcessorSpeed</td>
<td>19 $13</td>
</tr>
<tr>
<td>GetSCSIDiskModeAddress</td>
<td>20 $14</td>
</tr>
<tr>
<td>SetSCSIDiskModeAddress</td>
<td>21 $15</td>
</tr>
<tr>
<td>GetWakeupTimer</td>
<td>22 $16</td>
</tr>
<tr>
<td>SetWakeupTimer</td>
<td>23 $17</td>
</tr>
<tr>
<td>IsProcessorCyclingEnabled</td>
<td>24 $18</td>
</tr>
<tr>
<td>EnableProcessorCycling</td>
<td>25 $19</td>
</tr>
<tr>
<td>BatteryCount</td>
<td>26 $1A</td>
</tr>
<tr>
<td>GetBatteryVoltage</td>
<td>27 $1B</td>
</tr>
<tr>
<td>GetBatteryTimes</td>
<td>28 $1C</td>
</tr>
</tbody>
</table>
PMSelectorCount

You can use the PMSelectorCount routine to determine which routines are implemented.

short PMSelectorCount();

DESCRIPTION

The PMSelectorCount routine returns the number of routine selectors present. Any routine whose selector value is greater than the returned value is not implemented.

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for PMSelectorCount is 0 ($00) in the low word of register D0. The number of selectors is returned in the low word of register D0.

PMFeatures

You can use the PMFeatures routine to find out which features of the Power Manager are implemented.

unsigned long PMFeatures();

DESCRIPTION

The PMFeatures routine returns a 32-bit field describing hardware and software features associated with the Power Manager on a particular machine. If a bit value is 1, that feature is supported or available; if the bit value is 0, that feature is not available. Unused bits are reserved by Apple for future expansion.

Field descriptions

<table>
<thead>
<tr>
<th>Bit name</th>
<th>Bit number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasWakeupTimer</td>
<td>0</td>
<td>The wakeup timer is supported.</td>
</tr>
<tr>
<td>hasSharedModemPort</td>
<td>1</td>
<td>The hardware forces exclusive access to either SCC port A or the internal modem. (If this bit is not set, then typically port A and the internal modem may be used simultaneously by means of the Communications Toolbox.)</td>
</tr>
<tr>
<td>hasProcessorCycling</td>
<td>2</td>
<td>Processor cycling is supported; that is, when the computer is idle, the processor power will be cycled to reduce the power usage.</td>
</tr>
</tbody>
</table>
### Power Manager Interface

**mustProcessorCycle** 3  The processor cycling feature must be left on (turn it off at your own risk).

**hasReducedSpeed** 4  Processor can be started up at a reduced speed in order to extend battery life.

**dynamicSpeedChange** 5  Processor speed can be switched dynamically between its full and reduced speed at any time, rather than only at startup time.

**hasSCSIDiskMode** 6  The SCSI disk mode is supported.

**canGetBatteryTime** 7  The computer can provide an estimate of the battery time remaining.

**canWakeupOnRing** 8  The computer supports waking up from the sleep state when an internal modem is installed and the modem detects a ring.

### ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for PMFeatures is 1 ($01) in the low word of register D0. The 32-bit field of supported features is returned in register D0.

---

### GetSleepTimeout

You can use the `GetSleepTimeout` routine to find out how long the computer will wait before going to sleep.

```cpp
unsigned char GetSleepTimeout();
```

**DESCRIPTION**

The `GetSleepTimeout` routine returns the amount of time that the computer will wait after the last user activity before going to sleep. The value of `GetSleepTimeout` is expressed as the number of 15-second intervals that the computer will wait before going to sleep.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is _PowerMgrDispatch ($A09E). The selector value for GetSleepTimeout is 2 ($02) in the low word of register D0. The sleep timeout value is returned in the low word of register D0.
SetSleepTimeout

You can use the SetSleepTimeout routine to set how long the computer will wait before going to sleep.

```c
void SetSleepTimeout(unsigned char timeout);
```

**DESCRIPTION**

The SetSleepTimeout routine sets the amount of time the computer will wait after the last user activity before going to sleep. The value of SetSleepTimeout is expressed as the number of 15-second intervals making up the desired time. If a value of 0 is passed in, the routine sets the timeout value to the default value (currently equivalent to 8 minutes).

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is _PowerMgrDispatch ($A09E). The selector value for SetSleepTimeout is 3 ($03) in the low word of register D0. The sleep timeout value to set is passed in the high word of register D0.

GetHardDiskTimeout

You can use the GetHardDiskTimeout routine to find out how long the computer will wait before turning off power to the internal hard disk.

```c
unsigned char GetHardDiskTimeout();
```

**DESCRIPTION**

The GetHardDiskTimeout routine returns the amount of time the computer will wait after the last use of a SCSI device before turning off power to the internal hard disk. The value of GetHardDiskTimeout is expressed as the number of 15-second intervals the computer will wait before turning off power to the internal hard disk.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is _PowerMgrDispatch ($A09E). The selector value for GetHardDiskTimeout is 4 ($04) in the low word of register D0. The hard disk timeout value is returned in the low word of register D0.
### SetHardDiskTimeout

You can use the `SetHardDiskTimeout` routine to set how long the computer will wait before turning off power to the internal hard disk.

```c
void SetHardDiskTimeout(unsigned char timeout);
```

**DESCRIPTION**

The `SetHardDiskTimeout` routine sets how long the computer will wait after the last use of a SCSI device before turning off power to the internal hard disk. The value of `SetHardDiskTimeout` is expressed as the number of 15-second intervals the computer will wait before turning off power to the internal hard disk. If a value of 0 is passed in, the routine sets the `timeout` value to the default value (currently equivalent to 4 minutes).

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `SetHardDiskTimeout` is 5 ($05) in the low word of register D0. The hard disk timeout value to set is passed in the high word of register D0.

### HardDiskPowered

You can use the `HardDiskPowered` routine to find out whether the internal hard disk is on.

```c
Boolean HardDiskPowered();
```

**DESCRIPTION**

The `HardDiskPowered` routine returns a Boolean value indicating whether or not the internal hard disk is powered up. A value of `true` means that the hard disk is on, and a value of `false` means that the hard disk is off.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `HardDiskPowered` is 6 ($06) in the low word of register D0. The Boolean result is returned in the low word of register D0.
**SpinDownHardDisk**

You can use the SpinDownHardDisk routine to force the hard disk to spin down.

```c
void SpinDownHardDisk();
```

**DESCRIPTION**

The SpinDownHardDisk routine immediately forces the hard disk to spin down and power off if it was previously spinning. Calling SpinDownHardDisk will not spin down the hard disk if spindown is disabled by calling SetSpindownDisable (defined later in this section).

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is _PowerMgrDispatch ($A09E). The selector value for SpinDownHardDisk is 7 ($07) in the low word of register D0.

**IsSpindownDisabled**

You can use the IsSpindownDisabled routine to find out whether hard disk spindown is enabled.

```c
Boolean IsSpindownDisabled();
```

**DESCRIPTION**

The IsSpindownDisabled routine returns a Boolean true if hard disk spindown is disabled, or false if spindown is enabled.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is _PowerMgrDispatch ($A09E). The selector value for IsSpindownDisabled is 8 ($08) in the low word of register D0. The Boolean result is passed in the low byte of register D0.

**SetSpindownDisable**

You can use the SetSpindownDisable routine to disable hard disk spindown.

```c
void SetSpindownDisable(Boolean setDisable);
```
DESCRIPTION
The SetSpindownDisable routine enables or disables hard disk spindown, depending on the value of setDisable. If the value of setDisable is true, hard disk spindown will be disabled; if the value is false, spindown will be enabled.

Disabling hard disk spindown affects the SpinDownHardDisk routine, defined earlier, as well as the normal spindown that occurs after a period of hard disk inactivity.

ASSEMBLY-LANGUAGE INFORMATION
The trap is _PowerMgrDispatch ($A09E). The selector value for SetSpindownDisable is 9 ($09) in the low word of register D0. The Boolean value to set is passed in the high word of register D0.

HardDiskQInstall
You can use the HardDiskQInstall routine to notify your software when power to the internal hard disk is about to be turned off.

OSErr HardDiskQInstall(HDQueueElement *theElement);

DESCRIPTION
The HardDiskQInstall routine installs an element into the hard disk power down queue to provide notification to your software when the internal hard disk is about to be powered off. For example, this feature might be used by the driver for an external battery-powered hard disk. When power to the internal hard disk is turned off, the external hard disk could be turned off as well.

The structure of HDQueueElement is as follows.

typedef pascal void (*HDSpindownProc)(HDQueueElement *theElement);
struct HDQueueElement {
    Ptr hdQLink; /* pointer to next queue element */
    short hdQType; /* queue element type (must be HDQType) */
    short hdFlags; /* miscellaneous flags (reserved) */
    HDSpindownProc hdProc; /* pointer to routine to call */
    long      hdUser; /* user-defined (variable storage, etc.) */
} HDQueueElement;

When power to the internal hard disk is about to be turned off, the software calls the routine pointed to by the hdProc field so that it can do any special processing. The routine will be passed a pointer to its queue element so that, for example, the routine can reference its variables.
Before calling `HardDiskQInstall`, the calling program must set the `hdQType` field to

```c
#define HDPwrQType 'HD'  /* queue element type */
```

or the queue element won’t be added to the queue and `HardDiskQInstall` will return an error.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `HardDiskQInstall` is 10 ($0A) in the low word of register D0. The pointer to the `HDQueue` element is passed in register A0. The result code is returned in the low word of register D0.

---

**HardDiskQRemove**

You can use the `HardDiskQRemove` routine to discontinue notification of your software when power to the internal hard disk is about to be turned off.

```c
OSErr HardDiskQRemove(HDQueueElement *theElement);
```

**DESCRIPTION**

The `HardDiskQRemove` routine removes a queue element installed by `HardDiskQInstall`. If the `hdQType` field of the queue element is not set to `HDPwrQType`, `HardDiskQRemove` simply returns an error.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `HardDiskQRemove` is 11 ($0B) in the low word of register D0. The pointer to the `HDQueue` element is passed in register A0. The result code is returned in the low word of register D0.

---

**GetScaledBatteryInfo**

You can use the `GetScaledBatteryInfo` routine to find out the condition of the battery or batteries.

```c
void GetScaledBatteryInfo(short whichBattery, BatteryInfo*theInfo);
```
DESCRIPTION

The GetScaledBatteryInfo routine provides a generic means of returning information about the battery or batteries in the system. Instead of returning a voltage value, the routine returns the battery level as a fraction of the total possible voltage.

Note

New battery technologies such as NiCad (nickel cadmium) and nickel metal hydride (NiMH) have replaced the sealed lead acid batteries of the original Macintosh Portable. The algorithm for determining the battery voltage that is documented in the Power Manager chapter of Inside Macintosh, Volume VI, is no longer correct for all PowerBook models.

The value of whichBattery determines whether GetScaledBatteryInfo returns information about a particular battery or about the total battery level. The value of GetScaledBatteryInfo should be in the range of 0 to BatteryCount(). If the value of whichBattery is 0, GetScaledBatteryInfo returns a summation of all the batteries, that is, the effective battery level of the whole system. If the value of whichBattery is out of range, or the selected battery is not installed, GetScaledBatteryInfo will return a result of 0 in all fields. Here is a summary of the effects of the whichBattery parameter:

<table>
<thead>
<tr>
<th>Value of whichBattery</th>
<th>Information returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Total battery level for all batteries</td>
</tr>
<tr>
<td>From 1 to BatteryCount()</td>
<td>Battery level for the selected battery</td>
</tr>
<tr>
<td>Less than 0 or greater than BatteryCount</td>
<td>0 in all fields of theInfo</td>
</tr>
</tbody>
</table>

The GetScaledBatteryInfo routine returns information about the battery in the following data structure:

```c
typedef struct BatteryInfo {
    unsigned char flags; /* misc flags (see below) */
    unsigned char warningLevel; /* scaled warning level (0-255) */
    char reserved; /* reserved for internal use */
    unsigned char batteryLevel; /* scaled battery level (0-255) */
} BatteryInfo;
```

The flags character contains several bits that describe the battery and charger state. If a bit value is 1, that feature is available or is operating; if the bit value is 0, that feature is not operating. Unused bits are reserved by Apple for future expansion.

Field descriptions

<table>
<thead>
<tr>
<th>Bit name</th>
<th>Bit number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>batteryInstalled</td>
<td>7</td>
<td>A battery is installed.</td>
</tr>
<tr>
<td>batteryCharging</td>
<td>6</td>
<td>The battery is charging.</td>
</tr>
<tr>
<td>chargerConnected</td>
<td>5</td>
<td>The charger is connected.</td>
</tr>
</tbody>
</table>
The value of \texttt{warningLevel} is the battery level at which the first low battery warning message will appear. The routine returns a value of 0 in some cases when it’s not appropriate to return the warning level.

The value of \texttt{batteryLevel} is the current level of the battery. A value of 0 represents the voltage at which the Power Manager will force the computer into sleep mode; a value of 255 represents the highest possible voltage.

\textbf{ASSEMBLY-LANGUAGE INFORMATION}

The trap is \_\_PowerMgrDispatch (\$A09E). The selector value for \texttt{GetScaledBatteryInfo} is 12 (\$0C) in the low word of register D0. The \texttt{BatteryInfo} data are returned in the low word of register D0 as follows:

\begin{itemize}
  \item Bits 31–24 Flags
  \item Bits 23–16 Warning level
  \item Bits 15–8 Reserved
  \item Bits 7–0 Battery level
\end{itemize}

\textbf{AutoSleepControl}

You can use the \texttt{AutoSleepControl} routine to turn the automatic sleep feature on and off.

\begin{verbatim}
void AutoSleepControl(Boolean enableSleep);
\end{verbatim}

\textbf{DESCRIPTION}

The \texttt{AutoSleepControl} routine enables or disables the automatic sleep feature that causes the computer to go into sleep mode after a preset period of time. When \texttt{enableSleep} is set to \texttt{true}, the automatic sleep feature is enabled (this is the normal state). When \texttt{enableSleep} is set to \texttt{false}, the computer will not go into the sleep mode unless it is forced to either by some user action—for example, by the user’s selecting Sleep from the Special menu of the Finder—or in a low battery situation.

\textbf{IMPORTANT}

Calling \texttt{AutoSleepControl} with \texttt{enableSleep} set to \texttt{false} multiple times increments the auto sleep disable level so that it requires the same number of calls to \texttt{AutoSleepControl} with \texttt{enableSleep} set to \texttt{true} to reenable the auto sleep feature. If more than one piece of software makes this call, auto sleep may not be reenabled when you think it should be. \(\blacktriangleup\)
ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for AutoSleepControl is 13 ($0D) in the low word of register D0. The Boolean value is passed in the high word of register D0.

GetIntModemInfo

You can use the GetIntModemInfo routine to find out information about the internal modem.

unsigned long GetIntModemInfo();

DESCRIPTION

The GetIntModemInfo routine returns a 32-bit field containing information that describes the features and state of the internal modem. It can be called whether or not a modem is installed and will return the correct information.

If a bit is set, that feature or state is supported or selected; if the bit is cleared, that feature is not supported or selected. Undefined bits are reserved by Apple for future expansion.

Field descriptions

<table>
<thead>
<tr>
<th>Bit name</th>
<th>Bit number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasInternalModem</td>
<td>0</td>
<td>An internal modem is installed.</td>
</tr>
<tr>
<td>intModemRingDetect</td>
<td>1</td>
<td>The modem has detected a ring on the telephone line.</td>
</tr>
<tr>
<td>intModemOffHook</td>
<td>2</td>
<td>The internal modem has taken the telephone line off hook (that is, you can hear the dial tone or modem carrier).</td>
</tr>
<tr>
<td>intModemRingWakeEnb</td>
<td>3</td>
<td>The computer will come out of sleep mode if the modem detects a ring on the telephone line and the computer supports this feature (see the canWakeupOnRing bit in PMFeatures).</td>
</tr>
<tr>
<td>extModemSelected</td>
<td>4</td>
<td>The external modem is selected (if this bit is set, then the modem port will be connected to port A of the SCC; if the modem port is not shared by the internal modem and the SCC, then this bit can be ignored).</td>
</tr>
</tbody>
</table>

Bits 15–31 contain the modem type, which will take on one of the following values:

-1  Modem is installed but type not recognized.
0   No modem is installed.
1   Modem is a serial modem.
2   Modem is a PowerBook Duo-style Express Modem.
3   Modem is a PowerBook 160/180-style Express Modem.
CHAPTER 6

Power Manager Interface

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for GetIntModemInfo is 14 ($0E) in the low word of register D0. The bit field to set is passed in the high word of register D0.

SetIntModemState

You can use the SetIntModemState routine to set some parts of the state of the internal modem.

void SetIntModemState(short theState);

DESCRIPTION

The SetIntModemState routine configures some of the internal modem’s state information. Currently the only items that can be changed are the internal/external modem selection and the wakeup-on-ring feature.

To change an item of state information, the calling program sets the corresponding bit in theState. In other words, to change the internal/external modem setting, set bit 4 of theState to 1. To select the internal modem, bit 15 should be set to 0; to select the external modem, bit 15 should be set to 1. Using this method, the bits may be set or cleared independently, but they may not be set to different states at the same time.

Note

In some PowerBook computers, there is a hardware switch to connect either port A of the SCC or the internal modem to the modem port. The two are physically separated, but software emulates the serial port interface for those applications that don’t use the Communications Toolbox. You can check the hasSharedModemPort bit returned by PMFeatures to determine which way the computer is set up.

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for SetIntModemState is 15 ($0F) in the low word of register D0. The bit field is returned in register D0.

MaximumProcessorSpeed

You can use the MaximumProcessorSpeed routine to find out the maximum speed of the computer’s microprocessor.

short MaximumProcessorSpeed();
CHAPTER 6

Power Manager Interface

DESCRIPTION

The `MaximumProcessorSpeed` routine returns the maximum clock speed of the computer’s microprocessor, in MHz.

ASSEMBLY-LANGUAGE INFORMATION

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `MaximumProcessorSpeed` is 16 ($10) in the low word of register D0. The processor speed value is returned in the low word of register D0.

CurrentProcessorSpeed

You can use the `CurrentProcessorSpeed` routine to find out the current clock speed of the microprocessor.

```c
short CurrentProcessorSpeed();
```

DESCRIPTION

The `CurrentProcessorSpeed` routine returns the current clock speed of the computer’s microprocessor, in MHz. The value returned will be different from the maximum processor speed if the computer has been configured to run with a reduced processor speed to conserve power.

ASSEMBLY-LANGUAGE INFORMATION

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `CurrentProcessorSpeed` is 17 ($11) in the low word of register D0. The processor speed value is returned in the low word of register D0.

FullProcessorSpeed

You can use the `FullProcessorSpeed` routine to find out whether the computer will run at full speed the next time it restarts.

```c
Boolean FullProcessorSpeed();
```

DESCRIPTION

The `FullProcessorSpeed` routine returns a Boolean value of `true` if, on the next restart, the computer will start up at its maximum processor speed; it returns `false` if the computer will start up at its reduced processor speed.
Power Manager Interface

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for FullProcessorSpeed is 18 ($12) in the low word of register D0. The Boolean result is returned in the low byte of register D0.

SetProcessorSpeed

You can use the SetProcessorSpeed routine to set the clock speed the microprocessor will use the next time it is restarted.

Boolean SetProcessorSpeed(Boolean fullSpeed);

DESCRIPTION

The SetProcessorSpeed routine sets the processor speed that the computer will use the next time it is restarted. If the value of fullSpeed is set to true, the processor will start up at its full speed (the speed returned by MaximumProcessorSpeed, described on page 109). If the value of fullSpeed is set to false, the processor will start up at its reduced speed.

For PowerBook models that support changing the processor speed dynamically, the processor speed will also be changed. If the speed is actually changed, SetProcessorSpeed will return true; if the speed isn’t changed, it will return false.

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for SetProcessorSpeed is 19 ($13) in the low word of register D0. The Boolean value to set is passed in the high word of register D0. The Boolean result is returned in register D0.

GetSCSIDiskModeAddress

You can use the GetSCSIDiskModeAddress routine to find out the SCSI ID the computer uses in SCSI disk mode.

short GetSCSIDiskModeAddress();

DESCRIPTION

The GetSCSIDiskModeAddress routine returns the SCSI ID that the computer uses when it is started up in SCSI disk mode. The returned value is in the range 1 to 6.
When the computer is in SCSI disk mode, the computer appears as a hard disk to another computer.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `GetSCSIDiskModeAddress` is 20 ($14) in the low word of register D0. The SCSI ID is returned in the low word of register D0.

### SetSCSIDiskModeAddress

You can use the `SetSCSIDiskModeAddress` routine to set the SCSI ID for the computer to use in SCSI disk mode.

```c
void SetSCSIDiskModeAddress(short scsiAddress);
```

**DESCRIPTION**

The `SetSCSIDiskModeAddress` routine sets the SCSI ID that the computer will use if it is started up in SCSI disk mode.

The value of `scsiAddress` must be in the range of 1 to 6. If any other value is given, the software sets the SCSI ID for SCSI disk mode to 2.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `SetSCSIDiskModeAddress` is 21 ($15) in the low word of register D0. The SCSI ID to set is passed in the high word of register D0.

### GetWakeupTimer

You can use the `GetWakeupTimer` routine to find out when the computer will wake up from sleep mode.

```c
void GetWakeupTimer(WakeupTime *theTime);
```
About the Power Manager Interface

**Description**

The `GetWakeupTimer` routine returns the time when the computer will wake up from sleep mode.

If the PowerBook model doesn't support the wakeup timer, `GetWakeupTimer` returns a value of 0. The time and the enable flag are returned in the following structure:

```c
typedef struct WakeupTime {
    unsigned long wakeTime;  /* wakeup time (same format as the time) */
    char      wakeEnabled;    /* 1=enable timer, 0=disable timer */
} WakeupTime;
```

**Assembly-Language Information**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `GetWakeupTimer` is 22 ($16) in the low word of register D0. The pointer to `WakeupTime` is passed in register A0.

**SetWakeupTimer**

You can use the `SetWakeupTimer` routine to set the time when the computer will wake up from sleep mode.

```c
void SetWakeupTimer(WakeupTime *theTime);
```

**Description**

The `SetWakeupTimer` routine sets the time when the computer will wake up from sleep mode and enables or disables the timer. On a PowerBook model that doesn’t support the wakeup timer, `SetWakeupTimer` does nothing.

**Assembly-Language Information**

The trap is `_PowerMgrDispatch ($A09E)`. The selector value for `SetWakeupTimer` is 23 ($17) in the low word of register D0. The pointer to `WakeupTime` is passed in register A0.
IsProcessorCyclingEnabled

You can use the IsProcessorCyclingEnabled routine to find out whether processor cycling is enabled.

Boolean IsProcessorCyclingEnabled();

DESCRIPTION

The IsProcessorCyclingEnabled routine returns a Boolean value of true if processor cycling is currently enabled, or false if it is disabled.

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for IsProcessorCyclingEnabled is 24 ($18) in the low word of register D0. The Boolean result is returned in register D0.

EnableProcessorCycling

You can use the EnableProcessorCycling routine to turn the processor cycling feature on and off.

void EnableProcessorCycling(Boolean enable);

DESCRIPTION

The EnableProcessorCycling routine enables processor cycling if a value of true is passed in, and disables it if false is passed.

WARNING

You should follow the advice of the mustProcessorCycle bit in the feature flags when turning processor cycling off. Turning processor cycling off when it’s not recommended can result in hardware failures due to overheating. ▲

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for EnableProcessorCycling is 25 ($19) in the low word of register D0. The Boolean value to set is passed in the high word of register D0.
BatteryCount

You can use the BatteryCount routine to find out how many batteries the computer supports.

```c
short BatteryCount();
```

**DESCRIPTION**

The BatteryCount routine returns the number of batteries that are supported internally by the computer. The value of BatteryCount returned may not be the same as the number of batteries currently installed.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is _PowerMgrDispatch ($A09E). The selector value for BatteryCount is 26 ($1A) in the low word of register D0. The number of batteries supported is returned in the low word of register D0.

GetBatteryVoltage

You can use the GetBatteryVoltage routine to find out the battery voltage.

```c
Fixed GetBatteryVoltage(short whichBattery);
```

**DESCRIPTION**

The GetBatteryVoltage routine returns the battery voltage as a fixed-point number. The value of whichBattery should be in the range 0 to BatteryCount() - 1. If the value of whichBattery is out of range, or the selected battery is not installed, GetBatteryVoltage will return a result of 0.0 volts.

**ASSEMBLY-LANGUAGE INFORMATION**

The trap is _PowerMgrDispatch ($A09E). The selector value for GetBatteryVoltage is 27 ($1B) in the low word of register D0. The battery number is passed in the high word of register D0. The 32-bit value of the battery voltage is returned in register D0.
GetBatteryTimes

You can use the GetBatteryTimes routine to find out about how much battery time remains.

```c
void GetBatteryTimes (short whichBattery, BatteryTimeRec *theTimes);
```

DESCRIPTION

The GetBatteryTimes routine returns information about the time remaining on the computer’s battery or batteries. The information returned has the following data structure:

```c
typedef struct BatteryTimeRec {
    unsigned long expectedBatteryTime; /* estimated time remaining */
    unsigned long minimumBatteryTime; /* minimum time remaining */
    unsigned long maximumBatteryTime; /* maximum time remaining */
    unsigned long timeUntilCharged; /* time until full charge */
} BatteryTimeRec;
```

The time values are in seconds. The value of expectedBatteryTime is the estimated time remaining based on current usage patterns. The values of minimumBatteryTime and maximumBatteryTime are worst-case and best-case estimates, respectively. The value of timeUntilCharged is the time that remains until the battery or batteries are fully charged.

The value of whichBattery determines whether GetBatteryTimes returns the time information about a particular battery or the total time for all batteries. The value of GetScaledBatteryInfo should be in the range of 0 to BatteryCount(). If the value of whichBattery is 0, GetBatteryTimes returns a total time for all the batteries, that is, the effective battery time for the whole system. If the value of whichBattery is out of range, or the selected battery is not installed, GetBatteryTimes will return a result of 0 in all fields. Here is a summary of the effects of the whichBattery parameter:

<table>
<thead>
<tr>
<th>Value of whichBattery</th>
<th>Information returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Total battery time for all batteries</td>
</tr>
<tr>
<td>From 1 to BatteryCount()</td>
<td>Battery time for the selected battery</td>
</tr>
<tr>
<td>Less than 0 or greater than BatteryCount</td>
<td>0 in all fields of theTimes</td>
</tr>
</tbody>
</table>

ASSEMBLY-LANGUAGE INFORMATION

The trap is _PowerMgrDispatch ($A09E). The selector value for GetBatteryTimes is 28 ($1C) in the low word of register D0. The pointer to BatteryTimeRec is passed in register A0.
/*************************************************************************************/
// Power Manager Interface

/** Header File for Power Manager Dispatch **/

Here is a sample header file for access to the Power Manager.

/* file: PowerMgrDispatch.h */
/* contains: header file for access to the Power Manager */
* Copyright © 1992-1993 by Apple Computer, Inc. All rights reserved. */

#ifndef __PowerMgrDispatch__
#define __PowerMgrDispatch__
#endif

#ifndef __TYPES__
#include <Types.h>
#endif

#ifndef gestaltPMgrDispatchExists
#define gestaltPMgrDispatchExists 4 /* gestaltPowerMgrAttr bit:
1=PowerMgrDispatch exists */
#endif

/* bits in bitfield returned by PMFeatures */
#define hasWakeupTimer 0 /* 1=wakeup timer is supported */
#define hasSharedModemPort 1 /* 1=modem port shared by SCC and internal modem */
#define hasProcessorCycling 2 /* 1=processor cycling is supported */
#define mustProcessorCycle 3 /* 1=processor cycling should not be turned off */
#define hasReducedSpeed 4 /* 1=processor can be started up at reduced speed */
#define dynamicSpeedChange 5 /* 1=processor speed can be switched dynamically */
#define hasSCSIDiskMode 6 /* 1=SCSI disk mode is supported */
#define canGetBatteryTime 7 /* 1=battery time can be calculated */
#define canWakeupOnRing 8 /* 1=can wake up when the modem detects a ring */
/* bits in bitfield returned by GetIntModemInfo and set by SetIntModemState */
#define hasInternalModem 0 /* 1=internal modem installed */
#define intModemRingDetect 1 /* 1=internal modem has detected a ring */
#define intModemOffHook 2 /* 1=internal modem is off hook */
#define intModemRingWakeEnb 3 /* 1=wake up on ring is enabled */
#define extModemSelected 4 /* 1=external modem selected */
#define modemSetBit 15 /* 1=set bit, 0=clear bit (SetIntModemState) */

/* information returned by GetScaledBatteryInfo */

struct BatteryInfo {
    unsigned char flags; /* misc flags (see below) */
    unsigned char warningLevel; /* scaled warning level (0-255) */
    char reserved; /* reserved for internal use */
    unsigned char batteryLevel; /* scaled battery level (0-255) */
};

typedef struct BatteryInfo BatteryInfo;

/* bits in BatteryInfo.flags */
#define batteryInstalled 7 /* 1=battery is currently connected */
#define batteryCharging 6 /* 1=battery is being charged */
#define chargerConnected 5 /* 1=charger is connected to the PowerBook */
    /* (this doesn't mean the charger is plugged in) */

/* hard disk spindown notification queue element */
typedef struct HDQueueElement HDQueueElement;

typedef pascal void (*HDSpindownProc)(HDQueueElement *theElement);
CHAPTER 6

Power Manager Interface

struct HDQueueElement {
    Ptr    hdQLink;    /* pointer to next queue element */
    short hdQType;    /* queue element type (must be HDQType) */
    short hdFlags;    /* miscellaneous flags */
    HDSpindownProc hdProc;    /* pointer to routine to call */
    long    hdUser;    /* user-defined (variable storage, etc.) */
};

#define HDPwrQType'HD'    /* queue element type */

/* wakeup time record */
typedef struct WakeupTime {
    unsigned long wakeTime;    /* wakeup time (same format as current time) */
    char    wakeEnabled;    /* 1=enable wakeup timer, 0=disable wakeup timer */
} WakeupTime;

/* battery time information (in seconds) */
typedef struct BatteryTimeRec {
    unsigned long expectedBatteryTime;    /* estimated battery time remaining */
    unsigned long minimumBatteryTime;    /* minimum battery time remaining */
    unsigned long maximumBatteryTime;    /* maximum battery time remaining */
    unsigned long timeUntilCharged;    /* time until battery is fully charged */
} BatteryTimeRec;

#ifdef __cplusplus
extern "C" {
#endif

About the Power Manager Interface 119
About the Power Manager Interface

#pragma parameter __D0 PMSelectorCount(__D0)
short PMSelectorCount()
    = {0x7000, 0xA09E};

#pragma parameter __D0 PMFeatures
unsigned long PMFeatures()
    = {0x7001, 0xA09E};

#pragma parameter __D0 GetSleepTimeout
unsigned char GetSleepTimeout()
    = {0x7002, 0xA09E};

#pragma parameter __D0 SetSleepTimeout(__D0)
void SetSleepTimeout(unsigned char timeout)
    = {0x4840, 0x303C, 0x0003, 0xA09E};

#pragma parameter __D0 GetHardDiskTimeout
unsigned char GetHardDiskTimeout()
    = {0x7004, 0xA09E};

#pragma parameter __D0 SetHardDiskTimeout(__D0)
void SetHardDiskTimeout(unsigned char timeout)
    = {0x4840, 0x303C, 0x0005, 0xA09E};

#pragma parameter __D0 HardDiskPowered
Boolean HardDiskPowered()
    = {0x7006, 0xA09E};
Power Manager Interface

#pragma parameter __D0 SpinDownHardDisk

void SpinDownHardDisk()
    = (0x7007, 0xA09E);

#pragma parameter __D0 IsSpindownDisabled

Boolean IsSpindownDisabled()
    = (0x7008, 0xA09E);

#pragma parameter __D0 SetSpindownDisable(__D0)

void SetSpindownDisable(Boolean setDisable)
    = (0x4840, 0x303C, 0x0009, 0xA09E);

#pragma parameter __D0 HardDiskQInstall(__A0)

OSErr HardDiskQInstall(HDQueueElement *theElement)
    = (0x700A, 0xA09E);

#pragma parameter __D0 HardDiskQRemove(__A0)

OSErr HardDiskQRemove(HDQueueElement *theElement)
    = (0x700B, 0xA09E);

#pragma parameter __D0 GetScaledBatteryInfo(__D0,__A0)

void GetScaledBatteryInfo(short whichBattery, BatteryInfo *theInfo)
    = (0x4840, 0x303C, 0x000C, 0xA09E, 0x2080);

#pragma parameter __D0 AutoSleepControl(__D0)

void AutoSleepControl(Boolean enableSleep)
    = (0x4840, 0x303C, 0x000D, 0xA09E);
Power Manager Interface

#pragma parameter __D0 GetIntModemInfo(__D0)
unsigned long GetIntModemInfo()
    = {0x700E, 0xA09E};

#pragma parameter __D0 SetIntModemState(__D0)
void SetIntModemState(short theState)
    = {0x4840, 0x303C, 0x000F, 0xA09E};

#pragma parameter __D0 MaximumProcessorSpeed
short MaximumProcessorSpeed()
    = {0x7010, 0xA09E};

#pragma parameter __D0 CurrentProcessorSpeed
short CurrentProcessorSpeed()
    = {0x7011, 0xA09E};

#pragma parameter __D0 FullProcessorSpeed
Boolean FullProcessorSpeed()
    = {0x7012, 0xA09E};

#pragma parameter __D0 SetProcessorSpeed(__D0)
Boolean SetProcessorSpeed(Boolean fullSpeed)
    = {0x4840, 0x303C, 0x0013, 0xA09E};

#pragma parameter __D0 GetSCSIDiskModeAddress
short GetSCSIDiskModeAddress()
    = {0x7014, 0xA09E};
Power Manager Interface

#pragma parameter __D0 SetSCSIDiskModeAddress(__D0)

void SetSCSIDiskModeAddress(short scsiAddress)
    = {0x4840, 0x303C, 0x0015, 0xA09E};

#pragma parameter __D0 GetWakeupTimer(__A0)

void GetWakeupTimer(WakeupTime *theTime)
    = {0x7016, 0xA09E};

#pragma parameter __D0 SetWakeupTimer(__A0)

void SetWakeupTimer(WakeupTime *theTime)
    = {0x7017, 0xA09E};

#pragma parameter __D0 IsProcessorCyclingEnabled

Boolean IsProcessorCyclingEnabled()
    = {0x7018, 0xA09E};

#pragma parameter __D0 EnableProcessorCycling(__D0)

void EnableProcessorCycling(Boolean enable)
    = {0x4840, 0x303C, 0x0019, 0xA09E};

#pragma parameter __D0 BatteryCount

short BatteryCount()
    = {0x701A, 0xA09E};

#pragma parameter __D0 GetBatteryVoltage(__D0)

Fixed GetBatteryVoltage(short whichBattery)
    = {0x4840, 0x303C, 0x001B, 0xA09E};
Power Manager Interface

#pragma parameter __D0 GetBatteryTimes(__D0, __A0)

void GetBatteryTimes(BatteryTimeRec *theTimes)
{
    = {0x4840, 0x303C, 0x001C, 0xA09E};

#ifdef __cplusplus
}
#endif
#endif
# Index

## A
- AC adapter 5
- access to internal components 22
- active matrix display 24
- ADB (Apple Desktop Bus) port 31
- ADB connector 51
- appearance 3
- Ariel CLUT-DAC IC 14
- AutoRemounter control panel 80
- AutoSleepControl routine 107

## B
- back view 4
- batteries 5
- BatteryCount routine 115

## C
- compatibility 6
  - sound sample rates 7
- configurations 5
- connectors
  - ADB 31
  - DAA card 66
  - Ethernet 30
  - external video 26
  - floppy disk 21
  - hard disk 18, 19, 20
  - modem card 21
  - PDS card 53
  - RAM expansion 35, 36
  - SCSI 29
  - serial port 28
  - video 26
- control panels
  - AutoRemounter 80
  - Control Strip 80
  - PowerBook 77
  - PowerBook Display 79
  - PowerBook Setup 78
  - Trackpad 79
- control strip 75
  - adding modules to
  - module files 81
  - module interface 82
  - module reentrancy 82
- Control Strip control panel 77
- control strip Gestalt selectors 82
- control strip module messages
  - sdevCloseModule 83
  - sdevDrawStatus 85
  - sdevFeatures 83
  - sdevGetDisplayWidth 84
  - sdevInitModule 83
  - sdevMouseClicked 85
  - sdevPeriodicTickle 84
  - sdevSaveSettings 86
  - sdevShowBalloonHelp 86
- control strip module utility routines 86
  - SBDrawBarGraph 91
  - SBGetBarGraphWidth 91
  - SBGetDetachedIndString 89
  - SBGetDetachedIconSuite 89
  - SBIsControlStripVisible 87
  - SBLoadPreferences 88
  - SBModalDialogInContext 93
  - SBOpenModuleResourceFile 88
  - SBSafeToAccessStartupDisk 87
  - SBSavePreferences 88
  - SBShowHelpString 81
  - SBShowHideControlStrip 87
  - SBTrackPopupMenu 90
  - SBTrackSlider 80
- country key 68
- CSC custom IC 14
- CurrentProcessorSpeed routine 110
- custom ICs
  - Ariel 14
  - Combo 14
  - CSC 14
  - Keystone 14
  - Pratt 10, 12
  - Singer 31
### Index

#### D
- DAA card: 64–68
- connector on 66
- for multiple countries: 68
- displays:
  - active matrix: 23, 24
  - backlighting: 23
  - controller IC: 14
  - dual mode: 25
  - DualScan: 24
- external monitors: 26
- external video monitors:
  - adapter cable: 26
  - sense codes: 26
- flat panel types:
  - FSTN: 24
  - maximum size: 26
  - mirror mode: 25
  - NuBus card emulation: 24
  - number of colors: 6, 24
  - passive matrix: 23
  - selecting modes: 26
  - supertwist: 23, 24
  - TFT: 24
- dual mode: 25
- DualScan display: 24

#### E
- EnableProcessorCycling routine: 114
- Ethernet driver: 73
- Ethernet port:
  - connectors: 30
- Express Modem II: 6, 56
- external video port:

#### F
- features summary: 2
- floppy disk drive:
  - connector: 21
- FPU (floating-point unit): 10
- FullProcessorSpeed routine: 110
- function-key software: 74

#### G
- gestaltMachineType value: 75
- gestaltPowerMgrAttr selector: 97
- GetBatteryTimes routine: 116
- GetBatteryVoltage routine: 115
- GetHardDiskTimeout routine: 101
- GetModemInfo routine: 108
- GetScaledBatteryInfo routine: 105
- GetSleepTimeout routine: 100
- GetWakeupTimer routine: 112

#### H
- hard disk capacity: 5
- hard disk drive: 16
  - carrier bracket: 17
  - connector: 18, 19, 20
  - dimensions: 18
  - power budget: 16
  - SCSI ID: 19
- HardDiskPowered routine: 102
- HardDiskQInstall routine: 104
- HardDiskQRemove routine: 105
- HDI-30 connector: 28, 29

#### I, J
- identifying the computers: 75
- input/output subsystem:
  - MC68030 bus: 13
- Internal Modem Manager: 61
- I/O ports:
  - Ethernet: 50
  - SCSI: 28
  - serial: 28
  - video: 25, 26
- IsProcessorCyclingEnabled routine: 114
- IsSpindownDisabled routine: 103

#### K, L
- keyboards: 22
  - function keys: 74
  - ISO layout: 23
  - removing: 23
  - U.S. layout: 22
- Keystone video timing IC: 14
M

MacsBug, LCD buffer incompatibility 7
MaximumProcessorSpeed routine 109
MC68040 microprocessor 10
MC68LC040 microprocessor 10
MC68LC040 microprocessor support 72
memory controller software 72
memory expansion 5, 12
microphone 32
microprocessor
  clock speed 10
  type 10
mirror mode 25
modem
  country key 6
  DAA module 6
  specifications 6
modem card connector 57
DAA separate from 57
Express Modem II 56
NuBus emulation for 56
signals on 57, 59, 60
modem drivers 61
multi-country DAA 6, 68

N, O

new control panels 77

P, Q

PCMCIA cards 42
PDS card connector 53
design guidelines 43
electrical characteristics 55
inrush current 49, 51
logic design of 52
mechanical design 52
memory access from power budget 49
power control of selection circuit 52
signal descriptions 45, 48
software support for 74
user installation of peripheral devices 3
PMFeatures routine 99
PMSelectorCount routine 99
pointing device 22
PowerBook control panel 77
PowerBook Display control panel 79
PowerBook Setup control panel 78
Power Manager IC 13
  track pad registers in 74
Power Manager interface routines 97-116
AutoSleepControl 107
BatteryCount 115
CurrentProcessorSpeed 110
EnableProcessorCycling 114
FullProcessorSpeed 110
GetBatteryTimes 116
GetBatteryVoltage 115
GetHardDiskTimeout 101
GetModemInfo 108
GetScaledBatteryInfo 105
GetSCSIDiskModeAddress 111
GetSleepTimeout 100
GetWakeupTimer 112
HardDiskPowered 102
HardDiskQInstall 104
HardDiskQRemove 105
IsProcessorCyclingEnabled 114
IsSpindownDisabled 103
MaximumProcessorSpeed 109
PMFeatures 99
PMSelectorCount 99
SetHardDiskTimeout 102
SetIntModemState 109
SetProcessorSpeed 111
SetSCSIDiskMode 112
SetSleepTimeout 101
SetSpindownDisable 103
SetWakeupTimer 113
SpinDownHardDisk 103
Power Manager software
  checking for routines 97
  data structures 7
  dispatching 117
  interface routines 96, 97-116
  unsafe assumptions 7, 96
PowerPC upgrade 10
Pratt memory controller software for processor clock speed 12
processor/memory subsystem 10
RAM
- address range
- built in
- expansion
- addressing
- connector
- DRAM devices
- RAM banks
- signals
- expansion card
- refresh
- size of
- RAM expansion
- RAM expansion card
- design guide
- dimensions
- DRAM devices
- electrical limits
- RAM banks
- reentrancy in control strip modules
ROM
- address range
- implementation of
- overlay at startup
- software features
- ROM software features

SetSpindownDisable routine
SetWakeUpTimer routine
Singer sound IC
sound circuits
sound features
sound IC
sound sample rates
sound system
- microphone
- speakers
- speakers
SpinDownHardDisk routine
Supertwist display
SVGA monitors
System 7.1
system enabler

telephone line interface. See DAA card
TFT display
trackball
trackpad
- software support for
- Trackpad control panel

SVGA monitors
- adapter cable for
- sense codes for
- video port

video
- connector
- external monitors
- video adapter cable
- video connector
- video mirror mode
- video modes
dual
mirror
video monitors
- adapter cable for
- sense codes for
This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Proof pages and final pages were created on an Apple LaserWriter Pro printer. Line art was created using Adobe Illustrator™ and Adobe Photoshop™. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

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

WRITER
Allen Watson
DEVELOPMENTAL EDITORS
Wendy Krafft, Beverly Zegarski
ILLUSTRATORS
Deb Dennis, Sandee Karr

Special thanks to Larry Agbulos, Mark Baumwell, Faith Bradford, Steve Christensen, Gary Embler, Eric Gradeler, Ted Hammer, and Rosemary Mylod