About This Book

This book, *Inside Macintosh: PowerPC System Software*, describes the new process execution environment and system software services provided with the first version of the system software for Macintosh on PowerPC computers. It contains information you need to know to write applications and other software that can run on PowerPC processor-based Macintosh computers.

The first release of the system software for Macintosh on PowerPC computers provides a mixed or hybrid environment: the system software provides the ability to execute both applications that use the native instruction set of the PowerPC microprocessor and applications that use the 680x0 instruction set. It accomplishes this by providing a very efficient 68LC040 Emulator that emulates 680x0 instructions with PowerPC instructions. As a result, virtually all existing 680x0-based Macintosh applications and other software modules that conform to the programming interfaces and techniques documented in the *Inside Macintosh* suite of books will execute without modification on PowerPC processor-based Macintosh computers.

To take maximum advantage of the much greater processing speed of the PowerPC microprocessor, however, you'll need to recompile your application's source code into a PowerPC application. Apple Computer, Inc., provides MPW-based C and C++ compilers and other tools that you can use to create native PowerPC applications. In general, if your source code is already compliant with ANSI C standards or the de facto ANSI C++ standards, you should be able, with moderately little effort, to rework your source code so that it can be compiled and built using the Apple-supplied tools into a PowerPC application. This book is intended to provide much of the information you need to port your existing 680x0 application (or other software) to the PowerPC platform.

Note
There will also be third-party compilers and development environments capable of generating PowerPC code. ◆

Although the native run-time execution environment of the first version of the system software for PowerPC processor-based Macintosh computers is significantly different from the execution environment of current 680x0-based Macintosh computers, you won't need to worry about those differences unless your existing code relies on specific information about the 680x0 execution environment. For example, if for some reason you directly access information in your application's A5 world, you'll need to rewrite those parts of code when porting your application to the PowerPC environment. Similarly, you'll need to rewrite any parts of your code that depend on data being passed in certain 680x0 registers. VBL tasks, for instance, very

often depend on the fact that a pointer to the VBL task record is passed in register A0.

The first chapter in this book, "Introduction to PowerPC System Software," provides a general overview of the system software that runs on PowerPC processor-based Macintosh computers. It also describes in detail the mixed environment provided by the 68LC040 Emulator and the Mixed Mode Manager, as well as the new run-time environment used for native PowerPC applications. You should read this chapter for general information about porting your existing software to the PowerPC environment. Even if you do not intend to port your existing 680x0 software, you might still want to read this chapter for information about running under the 68LC040 Emulator.

The remaining chapters in this book provide reference material for the three new system software managers introduced in the first version of the system software for PowerPC processor-based Macintosh computers. You should read these chapters for specific information on using the services provided by those managers. The new system software managers are

- the Mixed Mode Manager, which manages the mixed environment of PowerPC processor-based Macintosh computers running 680x0-based code
- the Code Fragment Manager, which loads fragments (blocks of executable PowerPC code and their associated data) into memory and prepares them for execution
- the Exception Manager, which handles exceptions that occur during the execution of PowerPC applications or other software

IMPORTANT

Some of the system software services introduced in the first version of the system software for PowerPC processor-based Macintosh computers might in the future be available on Macintosh computers that are not based on the PowerPC microprocessor. For example, it's possible that the Code Fragment Manager (and the entire run-time environment based on fragments) will be included in future versions of the system software for 680x0-based Macintosh computers. As a result, some of the information in this book might eventually be more generally applicable than the title of this book might suggest. ▲

If you are new to programming for Macintosh computers, you should read the book *Inside Macintosh: Overview* for an introduction to general concepts of Macintosh programming. You should also read other books in the *Inside Macintosh* series for specific information about other aspects of the Macintosh Toolbox and the Macintosh Operating System. In particular, to benefit most from this book, you should already be familiar with the run-time environment of 680x0 applications, as described in the two books *Inside Macintosh: Processes* and *Inside Macintosh: Memory*.

Related Documentation

This book is part of a larger suite of books that contain information essential for developing PowerPC applications and other software.

- For information about the PPCC compiler that you can use to compile your source code into a PowerPC application, see the book *C/C++ Compiler for Macintosh With PowerPC*.
- For information about the PPCAsm assembler, see the book *Assembler for Macintosh With PowerPC*.
- For information about debugging and measuring the performance of PowerPC applications, see the book *Macintosh Debugger Reference*.
- For information about performing floating-point calculations in PowerPC applications, see the book *Inside Macintosh: PowerPC Numerics*.
- For information about building PowerPC applications and other kinds of PowerPC software for Macintosh computers, see *Building Programs for Macintosh With PowerPC*.

Format of a Typical Chapter

Almost all chapters in this book follow a standard structure. For example, the chapter "Mixed Mode Manager" contains these sections:

- "About the Mixed Mode Manager." This section describes the Mixed Mode Manager. You should read this section for a general understanding of what the Mixed Mode Manager does and when you might need to call it explicitly.
- "Using the Mixed Mode Manager." This section provides detailed instructions on using the Mixed Mode Manager. You should read this section if you need to use the services provided by the Mixed Mode Manager.
- "Mixed Mode Manager Reference." This section provides a complete reference to the constants, data structures, and routines provided by the Mixed Mode Manager. Each routine description also follows a standard format, which presents the routine declaration followed by a description of every parameter of the routine. Some routine descriptions also give additional descriptive information, such as circumstances under which you cannot call the routine or result codes.
- "Summary of the Mixed Mode Manager." This section provides the C interfaces for the constants, data structures, routines, and result codes associated with the Mixed Mode Manager.

Conventions Used in This Book

Inside Macintosh uses various conventions to present information. Words that require special treatment appear in specific fonts or font styles. Certain information, such as parameter blocks, appears in special formats so that you can scan it quickly.

Special Fonts

All code listings, reserved words, and the names of actual data structures, constants, fields, parameters, and routines are shown in Courier (`this is Courier`).

Words that appear in **boldface** are key terms or concepts and are defined in the glossary at the end of this book. Note that numerical entries (for example, **32-bit clean**) are sorted before all alphabetical entries in the glossary and in the index.

Types of Notes

There are several types of notes used in *Inside Macintosh*.

Note

A note like this contains information that is interesting but possibly not essential to an understanding of the main text. (An example appears on page 1-6.) ◆

IMPORTANT

A note like this contains information that is essential for an understanding of the main text. (An example appears on page 1-19.) ▲

▲ **WARNING**

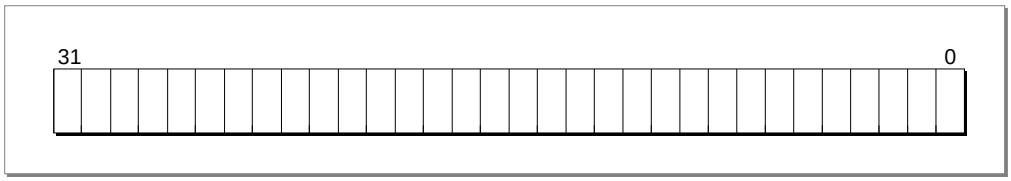
Warnings like this indicate potential problems that you should be aware of as you design your application. Failure to heed these warnings could result in system crashes or loss of data. (An example appears on page 1-8.) ▲

Bit Numbering and Word Size

This book departs from the conventions followed in previous *Inside Macintosh* books in regard to the numbering of bits within a range of data. Previously, for example, the bits in a 32-bit data type were numbered 0 to 31, from right to left, as shown in Figure P-1 on the following page. The least significant bit of a 32-bit data type was addressed as bit 0, and the most significant bit was addressed as bit 31. This convention was in accordance with that used by

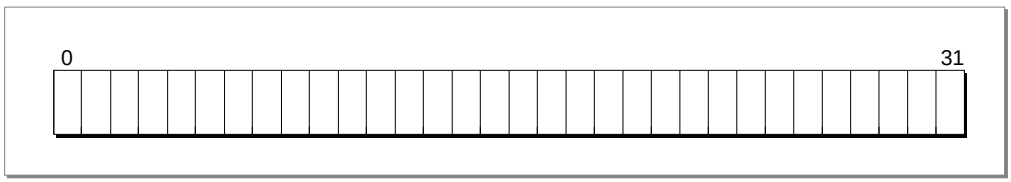
Motorola in the books documenting their 680x0 family of microprocessors (for example, the *MC68040 32-Bit Microprocessor User's Manual*).

Figure P-1 680x0 bit numbering



In this book, the bits in a 32-bit data type are numbered 0 to 31, from left to right. The most significant bit of a 32-bit data type is addressed as bit 0, and the least significant bit is addressed as bit 31. This convention, illustrated in Figure P-2, is in accordance with the bit-numbering conventions used by Motorola in the books documenting the PowerPC family of microprocessors (for example, the *PowerPC 601 RISC Microprocessor User's Manual*).

Figure P-2 PowerPC bit numbering



In addition, there are differences between 680x0 and the PowerPC terminology to describe the sizes of certain memory operands, as shown in Table P-1.

Table P-1 Sizes of memory operands

Size	680x0 terminology	PowerPC terminology
8 bits	Byte	Byte
2 bytes	Word	Half word
4 bytes	Long word	Word
8 bytes	N/A	Double word
16 bytes	N/A	Quad word

To avoid confusion, however, this book generally uses bytes to give the sizes of objects in memory.

Assembly-Language Information

Inside Macintosh presents information about the fields of a parameter block in this format:

Parameter block

↔	`inAndOut`	`Handle`	Input/output parameter.
←	`output1`	`Ptr`	Output parameter.
→	`input1`	`Ptr`	Input parameter.

The arrow in the far-left column indicates whether the field is an input parameter, output parameter, or both. You must supply values for all input parameters and input/output parameters. The routine returns values in output parameters and input/output parameters.

The second column shows the field name as defined in the MPW C interface files; the third column indicates the C data type of that field. The fourth column provides a brief description of the use of the field. For a complete description of each field, see the discussion that follows the parameter block or the description of the parameter block in the reference section of the chapter.

Development Environment

The system software routines described in this book are available using C or assembly-language interfaces. How you access these routines depends on the development environment you are using. This book shows system software routines in their C interface using the Macintosh Programmer's Workshop (MPW).

All code listings in this book are shown in C (except for listings that describe resources, which are shown in Rez-input format). They show methods of using various routines and illustrate techniques for accomplishing particular tasks. All code listings have been compiled and, in most cases, tested. However, Apple Computer does not intend that you use these code samples in your application. You can find the location of this book's code listings in the list of figures, tables, and listings.

To make the code listings in this book more readable, only limited error handling is shown. You need to develop your own techniques for detecting and handling errors.

This book occasionally illustrates concepts by reference to a sample application called *SurfWriter* and a sample import library called *SurfTools*; these are not actual products of Apple Computer, Inc.

For More Information

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 our 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	APDA
CompuServe	76666,2405
Internet	APDA@applelink.apple.com

If you provide commercial products and services, call 408-974-4897 for information on the developer support programs available from Apple.

For information on registering signatures, file types, Apple events, and other technical information, contact

Macintosh Developer Technical Support
Apple Computer, Inc.
20525 Mariani Avenue, M/S 303-2T
Cupertino, CA 95014-6299