MACINTOSH C

A Hobbyist's Guide to Programming the Mac OS in C

CLASSIC EDITION — Version 2.3

K. J. Bricknell

MACINTOSH C:
A Hobbyist's Guide to Programming the Mac OS in C
CLASSIC EDITION — Version 2.3

CONTENTS

CHANGES AND ADDITIONS IN VERSION 2.3

The following summarises the changes and additions in Version 2.3 as compared with Version 2.2.

Chapter		Change or Addition
All	Text	**All instances of N Helvetica Narrow font and Helvetica font changed to Arial Narrow.**
	Demos	All projects updated to CodeWarrior Pro 5.3. All demos now assume use of Universal Headers and Libraries Version 3.3 or later. Header files included by each demo revised to reflect Universal Headers 3.3.
Preface		Updated to reflect current status of the Mac OS, current development tools and resources, etc. States that, in future, there will be two editions of Macintosh C, namely, the Classic Edition and the Carbon Edition.
3 Menus	Text	New paragraph describing how, when Apple Help is used to provide online help for an application, an item is added to the Help menu. (Applies to Mac OS 8.6 and later.)
	Demo	Menus2 now demonstrates how an item is added to the Help menu when Apple Help is used to provide online help for an application. Choosing this item invokes Menus Help, which itself describes the basics of providing online help for an application using Apple Help. (Applies to Mac OS 8.6 and later.)
16 B More on Files — Navigation Services	Text	Minor updates to text for Navigation Services Version 2.0. Modified demo listing and comments.
	Demo	Coding of the functions doOpenCommand and doSaveAsCommand modified to accord with "how to" examples in the latest Apple Navigation Services documentation.
21 Floating Windows — Mac Os 8.5 and Earlier	Demo	In the source file FloatingWindows.c, in the function doMouseDown, a coding error at the inContent case has been fixed. (The variable frontWindowPtr is now assigned a pointer to the front-non-floating window.)
23 Miscellany	Text	Text modified to reflect the fact that, in Mac OS 9.x, notification alerts are displayed in a floating window, not a modal alert box.
	Demo	Code relating to system-based VBL task removed. Stop push button removed from the dialog associated with the progress indicator demonstration. (The way in which a click on the Stop push button was detected and handled did not reflect normal button behaviour.)

In addition, Version 2.3 is the first to include the words "Classic Edition —" before the version number. This is to distinguish this publication from the projected (as of February 2000) Carbon Edition.

PREFACE

Macintosh C: A Hobbyist's Guide to Programming the Mac OS in C

CLASSIC EDITION — Version 2.3

This book relies heavily on information contained in the principal volumes of the Addison-Wesley publication **Inside Macintosh**. Some demonstration programs include the author's translations into C of Pascal code examples in that publication. In addition, parts of Chapters 21 and 22 rely on information contained in Issues No 11, 15, and 17 of **develop** (The Apple Technical Journal). Apple Computer, Inc, which holds the copyright to those publications, has kindly consented to the author distributing Macintosh C on the Internet as a free publication.

Introduction

Macintosh C represents my attempt to provide a reasonably comprehensive entry point to Macintosh programming for the beginning hobbyist.

Version 1.0 of Macintosh C was published on the Internet in early 1996, and Version 1.1 followed in early 1997. Version 2.0, published in August 1998, was a major revision which addressed new features introduced by Mac OS 8, including an important new component of the system software known as the Appearance Manager. Because I found it all but impossible to satisfactorily address the old System 7 world and the new Mac OS 8 world in the same book (the cumbersome mishmash of alternative text, screen-shots, source code, etc., would have rapidly exhausted the patience of the reader), I decided to "freeze" the old Version 1 at Version 1.2 and make it separately available for those who, for one reason or another, needed to stay exclusively in the System 7 world, or who needed to write programs that were backwards-compatible with the old ways and means.[1]

Version 2.1, published in March 1999, brought things up-to-date with Mac OS 8.5 less the Mac OS 8.5 Window Manager. Version 2.2, published in June 1999, brought things up-to-date with Mac OS 8.6, including the Mac OS 8.5 Window Manager. [2]

1 Most, though not all, of the new features introduced by Mac OS 8.0 were provided by the Appearance Manager, the earlier versions of which were delivered as an extension. Appearance Version 1.0 shipped with, and supported, Mac OS 8.0 only. However, as of Version 1.0.1, the extension works with System 7.1 through Mac OS 8.1 on Macintoshes and Power Macintoshes. As of Mac OS 8.5, the Appearance Manager was included in the System file. This was Version 1.1. Mac OS 8.0 and 8.1 run only on Power Macintoshes (PowerPC processors) and Macintoshes with Motoroloa 68040 processors. Mac OS 8.5 and later runs only on Power Macintoshes.

2 Shortly after the release of Mac OS 8.5, Apple advised of certain bugs in that part of the Window Manager concerned with floating windows support and recommended that the associated Application Programming Interfaces (APIs) not be used

This version (Version 2.3) brings things up to date with Mac OS 9. It is the first issue to be subtitled "Classic Edition", this to differentiate it from a projected (as of February 2000) separate edition to be known as the "Carbon Edition". The new Carbon edition will address the Carbon Application Programming Interface (API) (see below) associated with the introduction of Mac OS X. In essence:

- The Classic edition is for those who wish to learn to write programs intended to run on Mac OS 8 and 9, that is, programs written using what is now referred to as the Classic API.[3]

- The Carbon edition is for those who wish to learn to write programs using the Carbon API, that is, programs which will run on both Mac OS 8/9 and Mac OS X and which, when run on Mac OS X, will take advantage of all of Mac OS X's advanced features and appear in the Mac OS X Aqua interface "look".

Classic Edition Version 2.3 contains some minor additions and changes, including updates to the demonstration programs to accommodate CodeWarrior Pro 5.3 and Version 3.3 of the Universal Headers (see below).

About the Two Mac OSs and APIs

Apple announced the intended future introduction of a new operating system, to be known as Mac OS X (pronounced "Mac OS Ten") at the 1998 World Wide Developers Conference.

Mac OS X is not just another Mac OS update; it is a completely new operating system complete with "modern" operating system features such as pre-emptive multitasking and protected memory. It features a completely new user interface, called Aqua, whose "look" and behaviour differs significantly from that of the original Mac OS (represented, in its latest — and probably last — incarnation, by Mac OS 9.x).

At the time of writing (February 2000), Mac OS X was scheduled to be released in mid-2000 and was expected to run on G3 and G4 PowerPC machines only, meaning that Power Macintoshes based on PowerPC 604 and 603 microprocessors must necessarily remain with Mac OS 8/9. A large installed base of these latter machines will no doubt remain for many years to come, making it highly desirable for programs written to take advantage of Mac OS X's "modern" features to be also capable of being run on Mac OS 8/9 *without modification*. Fortunately, Apple has devised the means whereby this may be achieved.

Mention was made previously of APIs (Application Programming Interfaces), specifically, the Classic API and the Carbon API. An API is simply a collection of system software calls like `GetNewCWindow` or `FspOpenDF` that programmers call to, for example, create a window or open a file's data fork. Mac OS 8/9 includes a collection of more than 8,000 such system software functions, which are now known collectively as the Classic API. Apple determined that about 2000 of these APIs were incompatible with a modern operating system like Mac OS X. The remaining 6000 or so "clean" APIs were isolated and, together with a few additions, included in the Carbon API.

It turns out, therefore, that programmers will not need to learn a completely new API in order to program the completely new operating system that is Mac OS X[4]. Any programmer familiar with the Classic API will be able to transition to the Carbon API with very little effort.

The upshot of all this is that, if you decide to use this edition of Macintosh C, rather than the Carbon edition, to learn Macintosh programming, that effort will in no way be wasted

---

until the bugs were eliminated. The bugs were eliminated in Mac OS 8.6.

[3] Note that a program written using the Classic API will actually run on Mac OS X in what may be regarded as a Mac OS 8/9 emulator; however such programs will not be capable of taking advantage of certain advanced Mac OS X features, nor will they inherit the "look" of Mac OS X's Aqua interface.

[4] That said, there is another API which may be used to write programs for Mac OS X. This API is called Cocoa, and it must be learned from scratch. Cocoa's advantage is that it allows applications to be written far more quickly that is the case using the Carbon API; however, such applications will not run on Mac OS 8/9.

should you decide, in the future, to make the transition to the Carbon API. The vast bulk of what you learn from this edition will remain valid when you make that move, given that some 92% of the system software calls used in this edition's demonstration programs are supported by Carbon.

The First Task — Learn the C Language

The main assumption made by Macintosh C is that you have already learned the C language. Accordingly, if you do not already know C, learning that language will be your first task.

Since the computer in question is a Macintosh, and since the development system assumed by Macintosh C is Metrowerks CodeWarrior, there is probably no better way to for you to learn C than to work through the book Learn C on the Macintosh by Dave Mark. This book, together with its associated example programs, is included on the Metrowerks CodeWarrior CD-ROMs.

As you are learning C, do not spend too much time on the subject of console input/output, since this has limited application in the world of the graphical user interface. In addition, you can afford to gloss over file input/output at this stage, since Macintosh C examples utilise Macintosh system software routines, rather than C standard library routines, to effect file input/output. (Indeed, it is entirely possible that you will never need to use the C standard library routines.)

The Macintosh C Phase

When you have learned the C language, you are ready to open Macintosh C. As you move through this second phase of the journey, you will quickly discover that learning C was by far the easiest part!

Essentially, Macintosh C covers all of the territory which, in my judgement, needs to be covered before you write your first serious application. This includes, for example, how to create and manage all elements of the user interface (menus, windows, controls, dialogs, alerts, lists, etc.), how to ensure that your application observes the house rules of the Macintosh graphical user interface and cooperative multitasking environment, how to perform file input/output, how to print files, how to draw text and graphics, and so on.

Considerable thought has been given to the sequence in which each topic is introduced, the content of most chapters relying to some extent on a full understanding of what has gone before. Accordingly, you should note that Macintosh C is not intended to be a randomly-accessed reference work; rather, is should be regarded as more in the nature of a course of study in which each chapter should be worked through in sequence[5].

General Structure of Macintosh C

The general structure of most chapters of Macintosh C is the same: first comes the information, then a list of constants, data types and functions relevant to the subject of that chapter, then the source code listing of one or more demonstration programs related to the subject of that chapter, and, finally, line-by-line comments which explain the workings of the source code.

The book itself is supported by the CodeWarrior project and source code files, and Resorcerer resource files, for all demonstration programs.

---

5 The one exception to this general rule is Chapter 4B — More on Windows — Mac OS 8.5 Window Manager. You might like to skip this chapter and return to it only after you have acquired a basic understanding of files and saving resources (Chapters 16 and 17) and TextEdit (Chapter 19).

What You Will Need

Development System

Apart from Macintosh C you will, of course, require a development system. Macintosh C assumes that that system will be Metrowerks CodeWarrior.

The Metrowerks product **Discover Programming For Macintosh** includes a C/C++ compiler that produces code that will run on 680x0-based Macintoshes and (in emulation) on the PowerPC-based Power Macintosh. Since all Macintosh C (Classic Edition) demonstration programs are capable of being compiled as either 680x0 or PowerPC code, and since the project files are "multi-target" (PowerPC *and* 680x0), the Discover Programming For Macintosh package will be quite adequate for your purposes.

The significantly more expensive Metrowerks product **CodeWarrior Pro**[6], adds, amongst other things, a compiler capable of producing code which will run native on PowerPC-based Macintoshes.

On-Line Reference

An on-line reference enables you to quickly and easily access information relating to the system software, and is thus quite indispensable. You can choose between **THINK Reference**, which is included on the MacTech Magazine CD-ROM (http://www.mactech.com/) and Apple's CD-ROM-based **Macintosh Programming Toolbox Assistant** (http://www.devdepot.com/descpage.html?PCODE=STBASST).

THINK Reference is somewhat out-of-date but still quite useful. If you decide on THINK Reference, be aware that the spelling of many of the constants and function names listed therein is now quite out-of-date, and that many new constants, data types and functions have been introduced since the last version of THINK Reference was published. The most up-to-date references in these matters are the Universal Header files produced by Apple and included in the Metrowerks CodeWarrior packages.[7]

Resource Editor

A resource editor allows you to create resources for programs and files. A copy of Apple's resource editor **ResEdit** is included with the CodeWarrior package; however, Apple ceased developing ResEdit some time ago, and it is simply not up to the task of creating the new resources introduced with Mac OS 8 and the Appearance Manager. The resource editor you will need is **Resorcerer**, which is produced by Mathemæsthetics, Inc (http://www.mathemaesthetics.com/).

Other Tools

Another useful tool is **ZoneRanger**, a dynamic memory inspection tool that allows you to investigate how effectively and efficiently your application uses memory. ZoneRanger is included with the CodeWarrior package.

You will also find a programmer's calculator very useful for converting between decimal, hexadecimal and binary values, the nicely-presented shareware program **CalcWorks** (http://sitelink.net/jbrochu/) being ideal for that purpose.

Human Interface Guidelines

Useful additions to your library when you get a little further down the track would be the publications **Macintosh Human Interface Guidelines** and **Mac OS 8 Human Interface**

---

6 Specially-priced academic versions of CodeWarrior Pro are available for students. Information on Metrowerks CodeWarrior products, including system requirements, is available at http://www.metrowerks.com/

7 The Universal Headers were introduced at the same time as the Power Macintosh. Amongst other things, they enable you to write source code capable of being compiled as either 680x0 code or PowerPC code — hence the term "Universal".

Guidelines, both of which are available at http://developer.apple.com/techpubs/mac/HIGuidelines/HIGuidelines-2.html.

Universal Headers and Libraries Version 3.3

Version 2.3 of Macintosh C assumes the use of **Version 3.3 or later of the Universal Headers and Libraries**. If the version of CodeWarrior you are using included an earlier version, you can download Version 3.3 (or later) from http://developer.apple.com/sdk/index.html.

Special Requirements — Chapter 16B Demonstration Program

If you are running Mac OS 8.1 or earlier, the demonstration program associated with Chapter 16B (Files2) requires that you acquire and install a shared library (**Navigation**) not included with Mac OS 8.1 and earlier. Download the Navigation Services SDK (Software Development Kit) from http://developer.apple.com/sdk/index.html and copy the shared library to your extensions folder. In addition, if you are using a 680x0 system, replace **OpenTransportLib.68K** in your Extensions folder with the one supplied in the SDK.

Demonstration Programs

All of the demonstration programs may be run from within CodeWarrior with the exception of the program that accompanies Chapter 10 — Apple Events. By its nature, this program must be run as a built (that is, double-clickable) application. The demonstration programs at Chapter 16A — Files and 16B — More on Files — Navigation Services may be run within CodeWarrior, although certain aspects of the programs can only be explored by running them as built applications.

The demonstration program that accompanies Chapter 4B — More on Windows — Mac OS 8.5 Window Manager can only be run on Power Macintoshes running Mac OS 8.6 or later.

You should read the top section of the source code comments in each chapter before running each program. For most programs, this explains what to do, what to expect, and what to note.

As far as is possible, each demonstration program avoids making calls to system software routines that are only explained in a later chapter. However, achieving that ideal has not been possible in the demonstration programs associated with the earlier chapters. For example, the demonstration program associated with Chapter 1 must, of necessity, make calls to system software routines relating to windows (the subject of Chapter 4) and drawing in a graphics port (the subject of Chapter 12). Where this occurs, you should simply accept, on faith, that the associated source code does as is stated in the demonstration program comments section. The important thing is to concentrate on that part of the source code pertaining to the subject of the chapter with which the program is associated.

Sorry, No MacHeaders

If you already know C, you will be familiar with the concept of header files and with the lines of code at the top of a source code file which explicitly include the header files required. CodeWarrior relieves the programmer of the requirement to explicitly include the most commonly used header files by automatically including a pre-compiled header file titled MacHeaders68K (680x0 projects) or MacHeadersPPC (PowerPC projects).

Ordinarily, these precompiled headers are summoned up by the file MacHeaders.h which, by default, appears in the Prefix File editable text item in the Settings dialog box (Language Settings/C/C++ Language section). MacHeaders.h has, however, been deleted as the Prefix File in all Macintosh C CodeWarrior projects, and all required header files are explicitly included in all source code files. Although the deletion of this precompiled

header adds to compilation time, there is a sound reason for this approach. Familiarising yourself with the contents of relevant header files should be considered to be an integral part of the process of learning to program the Mac OS in C. Accordingly, I would recommend that, every time you see a new header file at the top of a Macintosh C source code listing, you open that file and peruse its contents.

Terminology

All but the latest volumes of the official Mac OS reference work (Inside Macintosh) are Pascal-oriented, reflecting the fact that system software routines were originally designed to be called from a Pascal program. Because of this historically cosy relationship between Pascal and the system software, the C programmer will often be confronted by Pascal terminology. For example:

- The Pascal terms *procedure* and *record* are sometimes used in C-oriented programming books and other publications in a context where the C programmer would ordinarily expect the terms *function* and *structure*.[8]

- The names of many system software data structures end in Rec (for *record*) rather than, say, Struc (for *structure*).

- The names of certain fields in many system software data structures end in Proc (for *procedure*) rather than, say, Func (for *function*).

As a reflection of the fact that the later additions to Apple's Inside Macintosh series of publications are C-oriented rather than Pascal-oriented, Macintosh C uses C terminology exclusively. Hence *structure* is used rather than *record* and *function* is used rather than *procedure*.

There are a few other terms (or, rather, words) in this book which, depending on your country of residence, may seem only vaguely familiar. Bear in mind that this book was compiled in Australia, a civilised land where spelling conventions equate with those of the country that invented the language. Hence the word *colour* is generally spelled with a *u*. That said, the *u* has been removed where appropriate — for example, when reference is made to a component of the system software known, officially, as Color QuickDraw. In this way, and at the risk of being accused of inconsistency, I seek to offend nobody.

Comments Welcome

The author welcomes comments and suggestions on Macintosh C, which may be addressed to: brick@spirit.com.au

K. J. Bricknell, AM
Canberra
Australia
February 2000

[8] In C, a routine which returns a value and a routine which does not return anything are both called *functions*. In Pascal, a routine which returns a value is also called a *function*; however, a routine which does not return anything is called a *procedure*.