

MACINTOSH C

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

Version 2.0

K. J. Bricknell

MACINTOSH C:

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

Version 2.0

©1998, K. J. Bricknell

Portions of **Macintosh C: A Hobbyist's Guide to Programming the Mac OS in C** were adapted from the Inside Macintosh series of books and **Develop** magazine, © Apple Computer, Inc. All rights reserved. Used with the permission of Apple Computer, Inc.

Apple, the Apple logo, LaserWriter, and Macintosh are trademarks of Apple Computer Inc., registered in the United States and other countries.

Finder and QuickDraw are trademarks of Apple Computer Inc.

Metrowerks is a registered trademark of Metrowerks Inc.

CodeWarrior is a trademark of Metrowerks, Inc..

PostScript is a trademark of Adobe Systems incorporated, which may be registered in certain jurisdictions.

No warranty or representation is made, either express or implied, with respect to this manual, its quality, accuracy, or fitness for a particular purpose. As a result, this manual is distributed "as is", and you, the distributee, are assuming the entire risk as to its quality and accuracy. In no event will the author be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect or inaccuracy in this manual.

CONTENTS

- 1 *System Software, Memory, and Resources***
- 2 *Low-Level and Operating System Events***
- 3 *Menus***
- 4 *Windows***
- 5 *Microprocessor Matters***
- 6 *The Appearance Manager***
- 7 *Introduction to Controls***
- 8 *Dialogs and Alerts***
- 9 *Finder Interface***
- 10 *Required Apple Events***
- 11 *QuickDraw Preliminaries***
- 12 *Drawing With QuickDraw***
- 13 *Offscreen Graphics Worlds, Pictures, Cursors, and Icons***
- 14 *More on Controls***
- 15 *Printing***
- 16 *Files***
- 16B *More on Files — Navigation Services***
- 17 *More on Resources***
- 18 *Scrap***
- 19 *Text and TextEdit***
- 20 *Lists and Custom List Definition Functions***
- 21 *Floating Windows***
- 22 *Sound***
- 23 *Miscellany***

PREFACE

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

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.

Purpose and Evolution of Macintosh C

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. As Version 1.1 was being developed, Koryn Grant (a New Zealander then at the University of Kent in Canterbury, England) completed a translation of that version to the Pascal language. The result was Version 1.1 of Macintosh C's sister publication Macintosh Pascal.

A few months after the release of Mac OS 8, Koryn and I decided that we should do a further upgrade to accommodate the new features and changes ushered in by Mac OS 8 and, more particularly, a new component of the system software known as the Appearance Manager. After some discussion, we concluded that producing a new version that addressed Mac OS 8 and the Appearance Manager while continuing to look backward to System 7 minus the Appearance Manager would not be a good idea.¹ Our reasoning was that the cumbersome mishmash of alternative text, screen-shots, source code, source code files, resource files, etc., that would have resulted from any attempt to address both worlds in the one book would have rapidly exhausted the patience of the reader — not to mention the sanity of the writer! Our guess was that the majority of hobbyists would be quite happy to learn to program exclusively for Mac OS 8 (or System 7 plus the Appearance Manager), would not need their programs to be backward-compatible to System 7 minus the Appearance Manager, and thus would not want to have their attention continually diverted to older and (to them) irrelevant ways and means.

Version 2.0 is intended for that particular audience. For those who, for one reason or another, need to stay exclusively in the System 7 minus Appearance Manager world, or who need to write programs that are backwards-compatible with the old ways and means, Version 1.2 (Frozen) remains available. Version 1.2 (Frozen) is essentially Version 1.1 with the demonstration program files updated for Metrowerks CodeWarrior Pro 3 and the latest Universal Headers.

¹ Most, but not all, of the new features introduced by Mac OS 8 are provided by the Appearance Manager, which is delivered as an Extension. Mac OS 8 runs only on Power Macintoshes (PowerPC processors) and Macintoshes with Motorola 68040 processors. Appearance Version 1.0 shipped with, and supported, Mac OS 8 only. However, as of Appearance Version 1.0.2, the Extension works with System 7.1 and later.

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 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 with the CodeWarrior package.

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, it should be regarded as more in the nature of a tutorial in which each chapter should be worked through in sequence.

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 files, source code files, and resource files for all demonstration programs.

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* 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**², 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 Apple's CD-ROM-based **Macintosh Programming Toolbox Assistant** (<http://www.devdepot.com/SBDisplayItem.qry?ItemCode=STBASST>) and **THINK Reference** (<http://www.xplain.com/thinkreference/>).

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.³

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** 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 Guidelines**, both of which are available at http://developer.apple.com/techpubs/mac/user_interface.html.

Special Requirements — Chapter 16B Demonstration Program

The demonstration program associated with Chapter 16B (Files2) gets a little ahead of Codewarrior Pro 3 and Mac OS 8.1 in that it requires a Universal Header file and two libraries not included with CodeWarrior Pro 3 and a shared library not included with versions of the system software earlier than Mac OS 8.5. To determine whether you have the files required to support this particular demonstration program, proceed as follows:

- In the folder Metrowerks CodeWarrior/MacOS Support/Headers/Universal Headers, check whether the header file **Navigation.h** is present.
- In the folder Metrowerks CodeWarrior/MacOS Support/Libraries/MacOS Common, check whether the library **NavigationLib** is present. (This file is required for the PowerPC project.)
- In the folder Metrowerks CodeWarrior/MacOS Support/Libraries/MacOS 68K/MacOS Files, check whether the library **Navigation.o** is present. (This file is required for the 680x0 project.)

² All Macintosh C demonstration programs were written using CodeWarrior Pro 3. 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/>

³ 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".

- In your Extensions folder, check whether the shared library **Navigation** is present.

If one or more of these files are not present, download the Navigation Services SDK (Software Development Kit) from <http://developer.apple.com/sdk/> and copy the relevant file to the location indicated above. In addition, if you are using a 680x0 system, replace **OpenTransportLib.68K** in your Extensions folder with the one supplied in the SDK.

Special Requirements — System 7 Users

If you are using System 7, you will need System 7.5.5 or later for the demonstration program associated with Chapter 16B (Files2). Your most important special requirement, however, is the **Appearance extension Version 1.0.2** or later. Version 1.0.2 of the Appearance extension is included in the Appearance SDK (Software Development Kit), which may be downloaded from <http://developer.apple.com/sdk/>.

Demonstration Programs

All of the demonstration programs may be run from within CodeWarrior with the exception of the program that accompanies Chapter 10 — Required Apple Events. By its nature, this program should 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.

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 modules. 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 Macintosh 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*.⁴
- 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 Version 2.0 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.

Future of the Mac OS

In early 1997, when Apple announced the intended introduction of a completely new operating system codenamed **Rhapsody**, the days of the Mac OS appeared to be numbered. All that has now changed. As announced by Apple at the 1998 World Wide Developers Conference, the new light on the horizon is now **Mac OS X** (pronounced "Mac OS Ten"), which is probably best described as an offspring of both Mac OS 8 and Rhapsody.

Mac OS X, which is due in the second half of 1999, will not be just another operating-system update; it will be, to all intents and purposes, a new operating system with "modern" operating system features such as pre-emptive multitasking and protected memory. That said, Mac OS X will take its look and feel from today's Mac OS, and will provide a level of compatibility with current Macintosh applications that Rhapsody itself could not offer.

All this is excellent news for the Macintosh programmer, including the beginners for whom this book is intended, the reason being that the vast bulk of what you learn about programming the Mac OS today will remain applicable in the Mac OS X era.

The Mac OS of today includes a collection of more than 8,000 system software functions (usually called **application programming interfaces (APIs)**) that programmers use to do such things as create a window or open a file. Apple has determined that about 2000 of these APIs are incompatible with a modern operating system like Mac OS X. The remaining 6000 or so "clean" APIs, together with a few modifications and additions, have been isolated and codenamed **Carbon**. Carbon represents the core set of APIs that will be used to build Mac OS X applications.

The remaining good news is that Carbon applications will run on Mac OS 8 as well. (At present, it is expected that Mac OS X will run on G3 Power Macintoshes only.) Because Carbon supports both the Mac OS 8 and Mac OS X runtime environments, you will not need to produce separate versions of your application's source code for each of these environments.

All Macintosh C demonstration programs have been analysed by Apple's Carbon-compatibility facility. Of the APIs used in those programs, 92% are supported by Carbon, 2% are supported but will be modified in some way, 4% are supported but not recommended, 0.3% are currently being evaluated (and may or may not be included in Carbon), and 1.5% are not supported. In those circumstances, I anticipate that my future task of rendering these programs fully Carbon-compatible will involve

⁴ 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*.

nothing more than a few hour's work, the upshot being that you may approach this book in the certain knowledge that the greater part of the information contained therein will remain valid well into the future.

Acknowledgments

I mentioned earlier that my associate Koryn Grant does the Pascal translation of Macintosh C, thus producing the sister publication Macintosh Pascal. Koryn has also undertaken the onerous task of proof reading these 950 pages. I am most grateful for his assistance, and for his valuable suggestions and comments. The collaboration has been all the more remarkable when you consider that this is the first time in recorded history that an Australian and a New Zealander have ever agreed on anything.

Towards Version 2.1

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

K. J. Bricknell, AM
Canberra
Australia
August 1998