What's Where in the APPLE

Enhanced Edition

A Complete Guide to the Apple II Computer

Volume 1: Guide

Original Edition:
William F. Luebbert & Phil Daley

Featuring Publishing Legend:
Robert Tripp

Enhanced Edition:
Brian Wiser & Bill Martens

Apple **P**ugetSound **P**rogram **L**ibrary **E**xchange

What's Where in the Apple – Enhanced Edition:
A Complete Guide to the Apple II Computer – "Volume 1: Guide"

Copyright © 2016, 2020 by Apple Pugetsound Program Library Exchange (A.P.P.L.E.). All Rights Reserved.

Published by Apple Pugetsound Program Library Exchange (A.P.P.L.E.): www.callapple.org

Second Edition: November 2020

Paperback ISBN: 9-781716-405273
Hardback ISBN: 9-781716-405310

ACKNOWLEDGEMENTS

We would like to thank the original authors and publishers for this incredible resource:
William F. Luebbert, Phil Daley, and Robert Tripp. Produced in coordination with Robert Tripp.

Special thanks to Antoine Vignau, Javier A. Rivera, Mike Willegal, Paul Hagstrom, and Sean Fahey
for technical feedback.

The Cover and Book were designed by Brian Wiser.

PRODUCTION

What's Where in the Apple – Enhanced Edition → © 2016, 2020 Apple Pugetsound Program Library Exchange
Brian Wiser → Design, Layout, Editing, Proofreading, Chapter Revisions
Bill Martens → Proofreading, Chapter Revisions, Additional IIe/IIc Chapters

What's Where in the Apple – eWWA → © 2012 FlexAble Systems, Inc.
Donna Tripp, Robert Tripp, Andrew Molloy, Brian Wiser, Peter Wong

What's Where in the Apple → © 1982, 1984 The Computerist, Inc.
William F. Luebbert, Phil Daley, Robert Tripp,
Emmalyn Bentley, Ford Cavallari, Paula Kramer, Marjorie Morse

DISCLAIMER

No part of this book may be reproduced, distributed or transmitted in any form or by any means, including photocopying, scanning, or other electronic or mechanical methods, without prior written permission of the publisher, except in the case of brief quotations contained in articles and reviews, and program listings which may be entered, stored and executed in a computer system, but not reproduced for publication.

What's Where in the Apple programs are available on a disk image from: www.callapple.org.
No warranty of disk images is made or implied and should be used at your own risk.

What's Where in the Apple – Enhanced Edition: A Complete Guide to the Apple II Computer is an independent publication and has not been authorized, sponsored, or otherwise approved by any institution, public or private.

All images are under copyright and the property of Apple Pugetsound Program Library Exchange, or as otherwise indicated. Use is prohibited without prior permission.

Apple and all Apple hardware and software brand names are trademarks of Apple Inc., registered in the United States and other countries. All other brand names and trademarks are the property of their respective owners.

While all possible steps have been taken to ensure that the information included within is accurate, the publisher, producers, and authors shall have no liability or responsibility for any errors or omissions, or for loss or damages resulting from the use of the information and programs contained herein.

AUTHORS

William F. Luebbert

William F. Luebbert was adjunct Professor of Engineering at Thayer School of Engineering, Dartmouth College, Hanover, New Hampshire. He was also president of the Computer Literacy Institute, an organization founded in 1980 to train educators in the uses and applications of computers in education. He received the Automation Educator of the Year Award from *Business Automation* magazine*,* the Certified Data Processor Award from the Data Processing Management Association, and the American Society for Engineering Education Award and Prize for excellence in teaching engineering students.

Professor Luebbert, a U.S. Army retired Colonel, served on the faculty of the U.S. Military Academy, West Point, New York, from 1960 to 1978, where he taught Electrical Engineering and headed the Academic Computer Center.

Phil Daley

Phil Daley is a software engineer who was the technical editor and in-house Apple specialist for Micro Ink, Inc., the publisher of *MICRO – The 6502 Journal* and the original publisher of *What's Where in the Apple.* He was a high school math and music teacher in Hillsborough, New Hampshire as well as an activist for the Boston Computer Society. In 1984, he created the Apple IIe Appendix (now a revised Chapter 21) for *What's Where in the Apple.* He holds a Bachelor of Science and a Master of Arts degree from the University of Connecticut.

Robert Tripp

Robert Tripp started The Computerist in 1976 and Micro Ink in 1979. As an original Apple-1 owner, he is in the unique position of having received technical support from both Steve Jobs and Steve Wozniak, the founders of Apple, Inc. He published *MICRO – The 6502 Journal* from 1978 through 1984, along with a dozen 6502-related technical books. The magazine was a programming mainstay with coverage of many 6502 computing platforms of the time.

Because of Robert's interest and perseverance, he convinced author Dr. William F. Luebbert to expand the original *What's Where in the Apple* 7-page article into a highly-detailed, complete book. It was this resulting book, with its accompanying Atlas and Gazetteer, that became such an important resource in Apple II programmer's libraries. From 1982 to 1984, he published several versions of *What's Where in the Apple*, selling over 40,000 copies before it went out of print. He revisited that book with the *eWWA* released as a PDF in 2012. Robert has also developed computer-based medical products, including the first FDA-approved device for measuring human tremors, and was an Adjunct Professor at Arizona State University.

Brian Wiser

Brian Wiser is a producer of books, films, games, and events, as well as a long-time consultant, enthusiast and historian of Apple, the Apple II and Macintosh. Steve Wozniak and Steve Jobs, as well as *Creative Computing*, *Nibble*, *InCider*, and *A+* magazines were early influences.

Brian designed, edited, and co-produced dozens of books including: *Nibble Viewpoints: Business Insights From The Computing Revolution*, *Cyber Jack: The Adventures of Robert Clardy and Synergistic Software*, *Synergistic Software: The Early Games*, *The Colossal Computer Cartoon Book: Enhanced Edition*, *All About Applesoft: Enhanced Edition*, *Graphically Speaking: Enhanced Edition, What's Where in the Apple: Enhanced Edition*, and *The WOZPAK: Special Edition* – an important Apple II historical book with Steve Wozniak's restored original, technical handwritten notes. Brian is also the author of *The Etch-a-Sketch and Other Fun Programs*.

He passionately preserves and archives all facets of Apple's history, and noteworthy companies such as Beagle Bros and Applied Engineering, featured on AppleArchives.com. His writing, interviews and books are featured on the technology news site CallApple.org and in *Call-A.P.P.L.E.* magazine that he co-produces as an A.P.P.L.E. board member. Brian also co-produced the retro iOS game *Structris*.

In 2005, Brian was cast as an extra in Joss Whedon's movie *Serenity*, leading him to being a producer and director for the documentary film *Done The Impossible: The Fans' Tale of Firefly & Serenity*. He brought some of the *Firefly* cast aboard his Browncoat Cruise and recruited several of the *Firefly* cast to appear in a film for charity. Throughout these experiences, he develops close personal relationships with many actors, authors, and computer industry luminaries. Brian speaks about his adventures to large audiences at conventions around the country.

Bill Martens

Bill Martens is a systems engineer specializing in office infrastructures and has been programming since 1976. The DEC PDP 11/40 with ASR-33 Teletypes and CRT's were his first computing platforms with his first forays in the Apple world coming with the Apple II computer.

Influences in Bill's computing life came from *Byte* magazine, *Creative Computing* magazine, and *Call-A.P.P.L.E.* magazine as well as his mentors Samuel Perkins, Don Williams, Joff Morgan, and Mike Christensen.

Bill is the author of *ApPilot/W1*, *Beyond Quest*, *The Anatomy of an EAMON*, and multiple EAMon adventure games, as well as a co-producer of many books including *What's Where in the Apple: Enhanced Edition*, *The WOZPAK: Special Edition*, *Nibble Viewpoints: Business Insights From The Computing Revolution*, and co-programmer for the iOS version of the retro game *Structris*. He has written many articles which have appeared in user group newsletters and magazines such as *Call-A.P.P.L.E.*.

Bill worked for Apple Pugetsound Program Library Exchange (A.P.P.L.E.) under Val Golding and Dick Hubert as a data manager and programmer in the 1980s, and is the current president of the A.P.P.L.E. user group established in 1978. He reorganized A.P.P.L.E. and restarted *Call-A.P.P.L.E.* magazine in 2002. He is the production editor for the A.P.P.L.E. website CallApple.org, writes science fiction novels in his spare time, and is a retired semi-pro football player.

CONTENTS

Preface: *by Brian Wiser and Bill Martens* $11

Author and Publisher Acknowledgements: 1982-1984 and 2012 Editions $12

Foreword: *by Robert Tripp* $15

The Evolution of *"What's Where in the Apple"*: *by Robert Tripp* $17

What's Where in the Apple – the 'Apple Memory Map': *by William F. Luebbert* $19

Introduction $1B

GUIDE

1. There's More In Your Apple II Than You Think

Overview of the Apple II Series 3

- Apple-1 (1975) 3
- Apple II (1977) 3
- Apple II Plus (1979) 4
- Apple III (1980-1983) 4
- Apple IIe (1983-1987) 4
- Apple IIc (1984-1988) 5
- Apple IIGS (1986-1989) 5

The Apple II System Environment – Hardware and Firmware 5

Making Hardware and Firmware Resources Accessible 6

2. System Specific Programming

BASIC Doesn't Have To Be a Straightjacket – Neither Does Assembly Language 9

A Programming Approach for BASIC and Assembly Language Programmers 9

A Step-by-Step Approach for BASIC Programmers –
Learning to Take Advantage of System-Specific Capabilities 10

When Should You Use Assembly Language? 11

When Should You Use System-Specific Quasi-BASIC? 12

Examples of System-Specific Quasi-BASIC Programming 13

3. PEEKing Can Be Informative

What Does a PEEK Do? 17

The BASIC Idea 17

Formal Statement and Hardware Implementation 18

What Can You Learn from PEEKing 18

Find Current Cursor Position 18

Find Currently Active Peripheral Slot 19

Determine Printout Speed-Delay 19

Case Study – Screen and Printout Status Inquiry 19

Double PEEKing 20

Finding Line Number of a BASIC Error 20

Finding Line Number of Current Data Statement 21

Finding Where You Transfer When 'RESET' is Pressed 21

More About Using Decimal Numbers –
Modulo 256 and Hexadecimal Numbers to Handle Double-Byte Information 22

Memory Pages & Magic Numbers – Decimal 256 or Hexadecimal $100 22

Decimal to Double-Decimal – Conversion Using Hexadecimal Tables 22

Table Look-Up Procedure – Hexadecimal-to-Decimal and Decimal-to-Hexadecimal 24

Table Look-Up Procedure – Integer or Decimal Address to Byte-Oriented Decimal
(Using Hex Tables) 24

Hexadecimal Addressing for PEEKs, POKEs, and CALLs 25

Hexadecimal PEEKing 26

The All-Hexadecimal PRINTed PEEK Capability of the System Monitor 26

4. POKEs Can Make Changes

What Does a POKE Do? 29

The BASIC Idea 29

Properties and Hardware Implementation Concepts 29

What a Single POKE Can Do 29

POKEing the Hardware – Changing Graphics Modes 29

POKEing the Software – Changing Printout Speed/Window,
Normal/Inverse/Flashing Mode, Cursor Position 31

POKEing Alphabetic/Numeric Information – Changing Screen Display Area 32

Sometimes You Should Double-POKE 34

Double-POKEing Without Aids 34

Other Ways of Double-POKEing 35

5. CALLs Can Make Things Happen

What Does a CALL Do? 37
The BASIC Idea 37
Formal Description of the CALL Statement 37
Subroutine Transfer of Control Diagram (Program Mixing GOSUBs, CALLs and JSRs) 38
CALLing System Subroutines 39
Communication Between BASIC and Machine Language Routines 40
Passing Parameters to Monitor Firmware 40
Passing Parameters from the Keyboard 40
Passing Monitor Parameters and CALLing Monitor Routines from Inside BASIC 42
Case Study: CALLing a Subroutine with Parameters in Monitor-Specified Locations 42
Passing Parameters Between BASIC and Machine Language 44
What Information is Passed 44
How Information is Passed 44
Passing Parameters via Predetermined Memory Locations 45
Overview 45
Areas Which Contain Standard Common-Agreement Locations 45
The Simulated Registers in Addressable Memory 46
Discussion of Passing Parameters via Hardware Registers 47
Hardware Registers in the Apple II 47
Options for Passing Parameters To / From Hardware Registers 47
The Monitor SAVE / RESTORE Approach 48
The Concept 48
Case Study: Animation Technique Using Fast Copying / Display Change & SAVE / RESTORE Method for Loading Hardware Registers 48
Direct Loading of Hardware Registers 51
The Concept 51
Case Study: Fast Data Copy Program Using Direct Method Calling a Subroutine Requiring Parameter Set-up in Hardware Registers 51
Modifying the CALL Statement to Include Parameters to be Passed 53
The Concept 53
The New Applesoft CALL 54
Utility Program Which Implements the New CALL 55
Using the New CALL with Monitor Subroutine PRTAX as a Quick Decimal-Hexadecimal Converter 57

6. Apple Architecture I

Architecture in Perspective – Not Just for Assembly Language Programmers 59

A Simplified Hardware Block Diagram and '6502 Programming Model' of the Apple II 60

Bit-Oriented Information Representation and Addressing
(Abandon Decimal Numbers All Ye Who Enter Here!) 62

Hexadecimal as a Convenient Human-Oriented Method of Abbreviation
(Not as a Strange Number System) 62

Hexadecimal and Negative Decimal Addresses 64

The Stored Program, Program Counter, FETCH / EXECUTE Cycle –
The Heart of the Stored-Program Computer 65

Instructions and Data Both Stored in Memory as Binary Bits 65

The FETCH / EXECUTE Cycle – How the Computer Distinguishes Instructions from Data 65

The Repertoire of Hardware-Implemented Instructions Built into the Apple II 66

The Total Repertoire 66

Most Important Instructions for BASIC Programmers 74

Data Handling Instructions Equivalent to BASIC PEEKs and POKEs 75

Load Accumulator Instruction – LDA the Machine Language Equivalent of PEEK 75

Store Accumulator Instruction – STA the Machine Language Equivalent of POKE 75

An Example of Data Movement Using PEEKs and POKEs and its
Machine Language Equivalent Using LDA and STA 75

Symbolic Instructions and Programming Machine Language in Symbolic Assembler Format 77

Instructions Which Change the Normal Sequence of Operation
(The Key to Repetition & the Computer's Decision-Making Capability) 81

7. Apple Architecture II: Addressing in the 6502 Microprocessor

Addressing Modes of the 6502 Microprocessor 83

Simple Addressing Modes of the 6502 Microprocessor 84

Implied Addressing (No Address Required) 84

Immediate Addressing (Data Value Included in Instruction) 84

Absolute Addressing (16-Bit Address Specifies Absolute Data Location) 85

Zero Page Addressing (8-Bit Absolute Addressing) 85

Relative Addressing (Relative to the Instruction) 85

Overview of Computed Address Concepts 86

The Key to Understanding Computed Addresses 86

Computing Addresses By Treating Them As Data 86

Computing Addresses By Hardware Indexing 86

Computing Addresses By Indirect Indexing Techniques 87

Elementary Indexing 87

Indexed Absolute Addressing and Indexed Zero-Page Addressing 87

Using Elementary Indexing Techniques for Moving Data 89

Using Indexing to Search and Print Data from a Table 92

Indirect Addressing 94

Overview of Indirect Addressing 94

Indirect Indexed Addressing – i.e. LDA ($06), Y 95

Indexed Indirect Addressing – i.e. LDA ($80,X) 99

8. Machine Language Programs Can Live Happily in a BASIC Environment

The Simple Approach – Using a Binary File for the
Machine Language Program Loaded and Called by the BASIC Program 101

POKEing Small Machine Language Programs into BASIC 102

POKEing Each Byte Individually 102

Using Read and Data Statements to Simplify POKEing 103

Tricking the Apple Monitor to Work Inside a BASIC Program 103

Imbedding Machine Language Transparently in Applesoft 105

When Relocatable Machine Language Code is Available 105

When Relocatable Machine Language Code is Unavailable 106

9. Apple System Memory Allocation

Ways to Look at Apple System Memory Organization 107

The First Cut – RAM, ROM and SPECIAL I/O 108

The Second Cut – Functional Allocation of Pages 110

10. The System Quick-Access Area – Memory Page 0 ($0000~$00FF)

Zero Page Addressing to Save Memory and Computation Cycles 115

Zero Page Usage 115

11. The System Stack Page – Memory Page 1 ($0100~$01FF)

Introduction to the System Stack – A Last-In, First-Out Storage Area 117
Subroutines, The Program Counter Store, Pushing and Popping 117
Combined Operations of the S-Register and Stack Page 118
Use of the Stack by the Programmer 121
Interrupts 123
- Overview 123
- What Happens When an Interrupt Occurs? 123
- Programming Concepts and Techniques 123

Interrupts for the BASIC Programmer 125

12. The Keyboard Input Buffer – Memory Page 2 ($0200~$02FF)

Introduction to the Keyboard Input Buffer and GETLN 127
- Keyboard Input in the Default Case 127
- Concept of Operation 127
- Variations on the Theme of Keyboard Input 128

GETLN and Related Routines 129
- Survey of Services Provided by GETLN 129
- The Routines 129
- Replacement of KEYIN 139
- Automatic Capitalization in GETLN 140

13. The Monitor and DOS Vector Page – Memory Page 3 ($0300~$03FF)

The Monitor Special Locations ($03F0~$03FF) 141
The DOS Vector Table ($03D0~$03FF) 142
Page 3 Space Available to Users ($0300~$03CF) and
How It is Typically Used for Machine Language Programming 144

14. Text and Low-Resolution Graphics Display – Memory Pages 4~7 ($0400~$07FF) and 8~11 ($0800~$0BFF)

Introduction to Text Output to the Screen 145
Representation of Text Characters – The ASCII Code 146
How Text Character Screen Locations and Memory Locations Map Into Each Other 148
Controlling What Appears Where on the Display Screen 151
The Low-Resolution Graphics Mode 152

15. **User Memory for BASIC Programs – Memory Pages 9~149 ($0800~$95FF)**

Overview of Available User Memory 157

The Default Case 157

Rationale for the Default Case 157

Variations in User Memory Availability in Different Hardware/Software Environments 158

Memory Allocation Theory 158

Terminology, Fundamentals, and a Pictorial Overview 158

HIMEM and the Top End of Available User Memory Space 159

LOMEM and the Bottom End of Available User Memory Space 161

Finding User Memory in Your BASIC Program 162

How Memory Is Allocated for Program Code (BASIC Statement Structure) 162

Method of Allocation 162

Sample Program and Analysis of How it Appears in User Memory 164

Modified Sample Program and Analysis 166

Another Sample Program for Analysis 168

Lessons from Analysis of the Three Sample Programs 169

Using Memory Allocation Information to Create Self-Modifying BASIC Programs 170

How User Memory is Allocated for Simple Variables (Variables Other Than Arrays) 171

Information Layout for Individual Variables 171

Analyzing Variable Allocation Information with the System Monitor 173

Locating Applesoft Variables Using an Applesoft Utility 173

Controlling the Locations Assigned to Variables 179

How User Memory is Allocated for Arrays 180

How Memory is Allocated for Strings 184

Ways to Handle Not Having Enough Applesoft User Memory 186

Memory Conservation 186

Making Unavailable Memory Available 187

Overlaying, Chaining, and Reusing the Same Memory 187

16. **High-Resolution Graphics Display – Memory Pages 32~63 ($2000~$3FFF) and 64~95 ($4000~$5FFF)**

Introduction 193
Using High-Resolution Graphics 194
The Simple Way – Using the Applesoft BASIC Commands 194
Information Useful in Pseudo-BASIC and Machine Language Programming 196
Similarities and Differences in Organization and Memory Mapping Between High-Res Graphics and Text / Low-Res Graphics 200
Getting 560-Position Horizontal Resolution 208
A Procedure for Overcoming Memory Allocation Conflicts 211
Overview 211
The Procedure Step-By-Step 211
Imbedding in Applesoft User Memory Space 216

17. **The Disk Operating System – Memory Pages 150~191 ($9600~$BFFF)**

Introduction 217
How Information is Organized on Apple II Floppy Disks 218
Introduction 218
Tracks 219
Sectors 219
Standard Overhead of Stored Information on Disks (Copy of DOS, VTOC, Directory) 220
Summary of Diskette Track/Sector/Byte Capacity 221
Diskette Organizational Concepts 221
Volume Table Of Contents (VTOC) 221
Bit Maps of Free Sectors in Each Track of the Diskette 224
Diskette Catalog (Directory) 224
Catalog File Descriptions and File Types 225
Track/Sector List 228
Text Files 229
Binary Files 230
BASIC Program Files (Applesoft and Integer) 231
Other File Types: R, S, A, B 232
Recap of the DOS Method of Finding Information on Diskette 233
Deleting and Resurrecting Files 233
Disk Space Allocation to Files and Simple Rules for Improving Disk Response 235
How Diskette User Space is Allocated to User Files (Programs, Text and Data) 236

18. **Specialized Input/Output Memory – Memory Pages 192~207 ($C000~$CFFF)**

Introduction to 16 Pages of Specialized Input/Output Addresses 237

The Strange Page – Built-in I/O Locations ($C000~$C07F), Slot (Peripheral Card) I/O Space ($C080~$C0FF) 238

The First Half ($C000~$C07F) in Context with the Rest of the Apple I/O System 238

Anomalous Characteristics of Locations in the First Half ($C000~$C07F) 238

Bit Versus Byte Anomaly 239

Full-byte Input Locations in the First Half ($C000~$C07F) Work Differently from Conventional Full-byte Memory Locations 239

The Incompletely Decoded Address Anomaly in the First Half ($C000~$C07F) 240

Data Change on Read Access Anomalies in the First Half ($C000~$C07F) 240

The Second Half ($C080~$C0FF) 242

The Strange Page In Depth 242

Tabular Summary/Overview 242

Hardware Perspective of the Strange Page 243

Keyboard Data Input ($C00x) and Clear Keyboard Strobe ($C01x) 245

Cassette Output Toggle ($C02x) and Speaker Output Toggle ($C03x) 246

Utility Strobe ($C040x) 246

Video Screen Display Mode Selection Soft Switches ($C050~$C057) 246

Annunciator Output Soft Switches ($C058~$C05F) 247

Cassette and Pushbutton/Flag Inputs ($C060~$C063 or $C068~$C06B) 248

Analog/Game Controller Inputs ($C064~$C067 or $C06C~$C06F), Analog Clear/Game Controller Strobe ($C070) 248

Slot or Peripheral Card I/O Space ($C080~$COFF) 249

Slot/Peripheral Card I/O Locations ($C100~$CFFF) 249

Overview of Memory Assigned to Each Peripheral Slot 250

Peripheral Slot Scratchpad RAM 250

Peripheral Card I/O Space 251

Peripheral Card ROM Page 252

Shared Exclusive-Use Expansion ROM 253

19. Applesoft BASIC Interpreter – Memory Pages 208~247 ($D000~$F7FF)

The Applesoft Dialect of BASIC 255
Features of the Applesoft Dialect 255
Variations in the Applesoft Interpreter for Different Hardware/Software Environments 255
How the Interpreter Functions 257
Overview 257
BASIC Cycle of Functional Operation for Applesoft Interpreter 257
Functional Utilization of Memory Space by Applesoft Programs 257
Structure of the Applesoft Interpreter 260
The Interpreter as Simulator of a Computer Whose Machine Language is BASIC 260
Program Structure of the Interpreter 261

20. The System Monitor – Memory Pages 248~255 ($F800~$FFFF)

Overview 265
Two Varieties of the System Monitor 265
Communicating with the System Monitor 266
Summary of System Monitor Commands 267

21. Apple IIe: 1983 – Memory Pages 192~207 and 248~255 ($C000~$CFFF and $F800~$FFFF)

Overview 271
A Third Apple Monitor 271
The Apple IIe Display 272
Hardware Locations 272
Software Status 273
Programming Considerations 273

22. Apple IIe: 1984-1988 – Revision B, Enhanced, Platinum

Overview 275
Special Features of Newer Apple IIe Models 275
Double Lo-Res Graphics 276
Double Hi-Res Graphics 276
MouseText Characters 276
Control Characters 279

Editing / ESC Characters 279
ProDOS 279
    Additional ProDOS Resources 281
ProDOS Disk Usage 281
    Volume Overhead Specification 281
    The Volume Bit Map 281
    The Volume Directory 283
    The File Entry Description 283
    The ProDOS Subdirectory 283
ProDOS Memory Usage 283
    ProDOS Zero Page Usage 283
    ProDOS Page 3 Usage – Interrupt Vectors 283
    Other ProDOS Page Usage 283
BASIC Commands 287
    Directory Control Commands 287
    Program Control Commands 288
    Programming Control Commands 288
    Binary File Control Commands 289
    Text File Control Commands 289
    Other ProDOS Commands 289
    Commands Not in ProDOS 291
    ProDOS 2.4 and 2.5 291
Apple IIe and IIc ASCII Chart 294
The Mini-Assembler 297
    Starting the Mini-Assembler 297
    Entering a Program in Mini-Assembler 298
Memory Bank Switching 298
Apple IIe Memory Map 301

23. Apple IIc and IIc Plus

Overview 303
Differentiating the Models 303
    Apple IIc Slot Assignments 304
Apple IIc ROM 255 304

Apple IIc ROM 0 304
Apple IIc ROM 3 305
Apple IIc ROM 4 305
Apple IIc Plus ROM 5 305
Clocks for the Apple IIc 306
No-Slot Clock Entry Points 306
Apple IIc Memory Map 306
Auxiliary Memory Soft Switch and Status Locations 309

ATLAS

Type-Use Guide and Abbreviations .... *Volume 2*
Numerical Atlas: Complete Memory Listing by Memory Location .... *Volume 2*
Numerical Atlas: Apple IIe and Apple IIc .... *Volume 2*

GAZETTEER

Alphabetical Gazetteer: Complete Memory Listing by Label Name .... *Volume 2*
Alphabetical Gazetteer: Apple IIe and Apple IIc .... *Volume 2*

APPENDICES

Appendix A: Determining Type of Apple II 313
Appendix B: Error Codes 315
Appendix C: ProDOS File Types 317
Appendix D: ProDOS Version Codes 321
Appendix E: BASIC.SYSTEM Versions 322
Appendix F: Glossary 323

INDEX

Figures Index 333
Guide Index 339

Preface

What's Where in the Apple is an incredible resource with a long history, and the monumental efforts of the original publishers and volunteers should not be overlooked. For the *Enhanced Edition*, we wanted to return the book to print, significantly improve upon it, and have something beautiful to hold.

The cover was designed by Brian Wiser, including redrawing and enhancing the original stylized apple. The new back cover has enhanced, hybrid text from the original publication and ads.

Wanting to make this invaluable reference even better, Brian designed the book with revised formatting to improve readability. Among other things like larger headers and better spacing, the descriptions in the Atlas and Gazetteer sections were converted from uppercase to sentence case, while keeping specific statements and routines all uppercase or title case. This was no small undertaking. Additionally, the Atlas, Gazetteer, and all program listings are now in a monospaced font that makes them more readable, and better distinguishes differences between characters like `0` and `O`.

With any work of this complexity, especially one that was OCR converted for the 2012 *eWWA* edition, there were many lingering errors to be found and fixed in the production files we received. Bill Marten's programming background enabled us to identify and correct many errors and restore missing data, including data missing in the original *WWA*. All pages were proofread numerous times over many months by Brian and Bill, using an original *WWA* for comparison. Countless spelling, OCR and formatting mistakes were corrected, abbreviations were expanded, and wording was improved.

With the Atlas/Gazetteer (in Volume 2), we've also clarified descriptions and added hexadecimal addresses to routine names, resulting in improvements to almost every line. Using multiple references, including Apple II ROM source code and the *Apple II Reference Manual*, we added missing addresses and expanded overly simplistic descriptions, resulting in a much more thorough and accurate Atlas/Gazetteer.

Additional material is included throughout the book. With the Guide (in Volume 1), we added an overview of Apple II models to Chapter 1 and updated the Apple IIe Appendix as Chapter 21. Bill wrote new Apple IIe and IIc chapters and their accompanying Atlas/Gazetteer, incorporating ProDOS updates. The Index was restored and updated, and we added new Appendices. In an effort to be historically complete, we've included the publisher acknowledgements from previous editions, and an article by William F. Luebbert that originally appeared in *MICRO*. This new *Enhanced Edition* is the most complete and accurate edition ever created.

Lastly, we want to acknowledge Robert Tripp who we had the pleasure of working with on the *MICRO* magazine archive, *eWWA*, and again with this new edition. He provided his *eWWA* production files that made this edition possible and gave us a starting point. Additionally for this edition, Robert has generously written *The Evolution of WWA* and a *Foreword* highlighting his personal computing history. Thank you Robert for originally bringing *WWA* to life and helping preserve a tangible and important piece of the Apple II's legacy.

As we are passionate about preserving and enhancing all that is Apple, we hope everyone enjoys our *Enhanced Edition* and that it serves to aid and inspire the next generation of Apple II programmers.

Brian Wiser and Bill Martens
Apple Pugetsound Program Library Exchange (A.P.P.L.E.)
March 2016