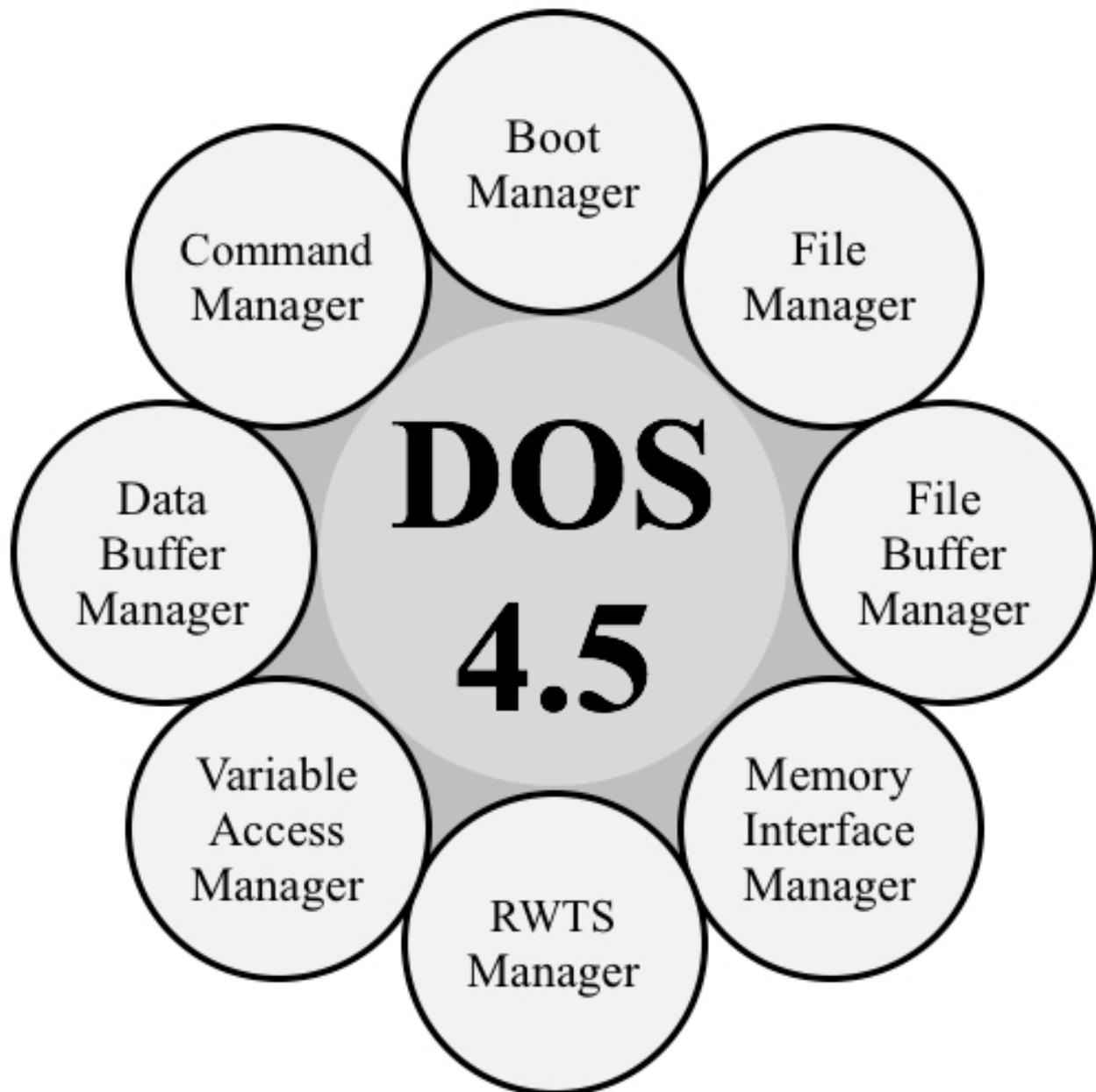


DOS 4.5 File and Volume Disk Management System



Walland Philip Vrbancic, Jr.

DOS 4.5

File and Volume Disk Management System

Walland Philip Vrbancic, Jr.



Copyright © 2022 January 1
Walland Philip Vrbancic, Jr.

All rights reserved.

No part of this publication may be reproduced,
stored in a retrieval system,
or transmitted, in any form or by any means,
electronic, mechanical, photocopying, recording, or otherwise,
without the prior written permission of the author.

DOS 4.5 File and Volume Disk Management System
and this Publication are the
Confidential and Proprietary Intellectual Properties
of
Walland Philip Vrbancic, Jr.

ISBN 978-0-578-38584-6

*I am proud to dedicate this Book on the
DOS 4.5 File and Volume Disk Management System
and all of my previous achievements
to my loving Parents Walland and Melba
who continuously nourished my intellectual curiosities
with games, toys, books, education, and unconditional love.*

*I am truly fortunate for all of the love and support
I have always received from my sister Marile.*

*I am also grateful to my partner
Carlton D. Wong
who delightfully pretends to understand
what the Hell I am talking about!*

Excellence is never an accident!
It is always the result of high intention, sincere effort, and intelligent execution;
it represents the wise choice of many alternatives;
choice, not chance, determines one's destiny.

~~~ Aristotle ~~~

If I have seen further than others it is because I have stood on the shoulders of giants.

~~~ Isaac Newton ~~~

Disclaimer of All Liability

Do not use the DOS 4.5 File and Volume Disk Management System software or this Book for any mission-critical applications or for any purpose in which a software error or a software failure could cause you financial or material loss. The DOS 4.5 File and Volume Disk Management System software and this Book are designed to enhance your Apple][computing experience, but they may contain design flaws that could inhibit the proper operation of your computer or they may result in the loss of recorded data on any storage device connected to your computer. You assume all risks associated with the operation of your computer and the potential loss of your recorded data when using the DOS 4.5 File and Volume Disk Management System software or this Book. If these terms are not acceptable to you, please do not use the DOS 4.5 File and Volume Disk Management System software or this Book.

Walland Philip Vrbancic, Jr., the administrator of www.applecored.net, makes no warranties either expressed or implied with respect to the DOS 4.5 File and Volume Disk Management System software or with respect to this Book, its quality, performance, or fitness for any particular purpose. Any risk of incidental or consequential damages resulting from the use of the DOS 4.5 File and Volume Disk Management System software or the use of information contained in this Book shall be assumed by you, the User. In no event shall Walland Philip Vrbancic, Jr., or www.applecored.net be liable for any direct, indirect, incidental, or consequential damages resulting from any defect, deficiency, or neglect in the DOS 4.5 File and Volume Disk Management System software or in this Book.

While all possible attempts have been made to ensure that the information contained within this publication is complete and accurate, the author shall have no liability or responsibility for any errors or omissions, or for any damages or the loss of recorded data resulting from the use of the information, circuit diagrams, and example software programs contained herein. The author reserves the right to implement any changes and/or any improvements to the DOS 4.5 File and Volume Disk Management System software or to the contents of this publication at any time and without any prior notice to you, the User, or to the general Apple][community.

Preface

When Brian Wiser and Bill Martens discovered my original DOS 4.1 software and documentation at www.applecored.net in 2017, they immediately contacted me and wanted Apple Pugetsound Program Library Exchange (*Call-A.P.P.L.E.*) to publish my DOS 4.1 Manual. Ha! If only this would have happened back in 1982. That's when my co-worker, Randy at Rockwell, and I were actively reading many publications featuring Apple software and hardware, and *Call-A.P.P.L.E.* was one of our favorite publications. Needless to say, to be published by any of those early computer journals would have been crazy exciting at that time, and it would have certainly been a cherished memory for a lifetime. I actually was very close to completing all of the capabilities I wanted in DOS 4.1 when I agreed to have *Call-A.P.P.L.E.* publish the DOS 4.1 Manual for Build 45. I also provided *Call-A.P.P.L.E.* with demo diskette images for both DOS 4.1L and DOS 4.1H. A month or so after Build 45 was published and available for purchase, I completed DOS 4.1 with Build 46. Only the DOS 4.1 Build 46 software and documentation in PDF form are available at www.applecored.net.

I remember telling Wiser and Martens that I wanted both versions of DOS 4.1 to provide the user with virtually the same computing experience, albeit the `HELP` command is found only in DOS 4.1H. This desire proved to be somewhat troublesome in that I was limited in memory for DOS 4.1L, whereas I had ample memory for DOS 4.1H. Actually, it was the unused memory in DOS 4.1H that was the impetus to introduce the `HELP` command in the first place. At the onset of our negotiations, I warned Wiser and Martens that I could not stop creating more functionality in DOS 4.1, but they were rather insistent on publishing the DOS 4.1 Manual for the Apple][community in its current stage of development. In 2021 I had the opportunity to republish the DOS 4.1 Manual as a book in the format that I originally intended. This publication is Build 46.

My next area of exploration for an Apple DOS was an attempt to port DOS 4.1H to Auxiliary memory. I was absolutely successful, I might add, but I could not successfully design an interface between *Lisa* (my most favorite 65C02 assembler) and this new DOS I created to reside in Auxiliary memory. Over the course of several months of significant effort, I could not realize a viable solution that would be elegant, save memory, and provide the roadmap for interfacing other utilities and tools to this unique DOS. But this effort was certainly not wasted! I documented what I had learned about Main and Auxiliary memory management and I moved forward to other areas of exploration.

I had already decided to use DOS 4.1H as my initial model for DOS 4.3. Yes, DOS 4.3 does retain the “H” designation for “High” memory. However, I did not want to develop a companion “Low” memory DOS version in parallel, so I did **not** develop a DOS 4.3L. I simply refer to this new DOS as DOS 4.3. The question then became, can *Lisa* be ported to and function in Auxiliary memory? The answer to that question turns out to be a resounding “Yes!” With DOS 4.3 in the memory of the Language Card partition in Main memory and *Lisa* in the memory of the Language Card partition in Auxiliary memory, the user has access to virtually all

of Main memory below 0xBE00 for source code, object code, and the complete symbol list. I saw this exciting configuration as the path to many new and potential possibilities.

Now, if I can relocate *Lisa* to Auxiliary memory, what about doing the same for *Big Mac*? I have to say that that challenge was a bit uneventful because relocating *Big Mac* to the memory of the Language Card partition in Auxiliary memory was even easier to accomplish. My main focus in *Big Mac* was to align *Sourceror* and *Big Mac* in terms of their *SWEET16* sourcing and assembling capabilities, though I do not believe *Big Mac* has ever been able to assemble all of its own *SWEET16* opcodes. This task turned out to be an extraordinarily massive undertaking since I wanted *Sourceror* and *Big Mac* to disassemble/assemble my personal version of the *SWEET16* opcodes. I realized that *Big Mac* could not even assemble its own unique *SWEET16* EVAL opcode because the EVAL opcode did not even exist. This tells me that Mr. Bredon probably could not even use *Big Mac* to assemble his own *Big Mac* source code unless he possibly hardcoded these undefined opcodes as `byt` statements. I have to confess that there still remain two *SWEET16* branches in my disassembled (and verified) *Big Mac* source code that are absolutely wrong, and I do not know their solution to this day. They occur at memory addresses 0xD2C1 and 0xD2DF. Furthermore, I have yet to discover how to force their execution in order to analyze the resulting behavior in *Big Mac*. I suspect these particular code sequences may be part of MACRO handling, something I have had no reason to use. After all is said and done, *Big Mac* and DOS 4.3 complemented each other beautifully.

While moving past DOS 4.3 to begin the development of the DOS 4.5 File and Volume Disk Management System, I discovered even more layers of File Manager functionality that I could code so much more effectively. DOS 4.3 certainly contains all the necessary solutions to properly close files and flush volumes when the DOS `CLOSE` command is issued from within an Applesoft program. Yet, the code resolutions found in DOS 4.3 still do not exhaust my list of all the additional capabilities I wanted in my final release of an Apple][DOS. All of the capabilities contained in DOS 4.5 are detailed in Section I.2.

I know the user will discover many, if not all of the fascinating developments I have included in the DOS 4.5 File and Volume Disk Management System. The user will be left wondering how he or she was able to accomplished anything useful in a timely fashion without having had those developments in any other version of a previous Apple][Disk Operating System. I would take that as my greatest compliment.

Enjoy the ride!

Table of Contents

| | |
|---|----------------|
| I. Designing Another New DOS | 1 |
| 1. Introduction..... | 1 |
| 2. Overview of the Improvements and Enhancements in DOS 4.5 | 2 |
| 3. DOS 4.5 Software Development Strategies..... | 4 |
| 4. Page-Zero Utilization..... | 7 |
| 5. DOS 4.5 VTOC Structure..... | 11 |
| 6. The VTOC Bitmap Definition..... | 15 |
| 7. The DOS 4.5 Catalog..... | 17 |
| 8. Booting DOS 4.5 Into Memory | 22 |
| 9. DOS 4.5 Memory Initialization | 32 |
| 10. The DOS 4.5 RWTS Interface..... | 39 |
| 11. The DOS 4.5 File Manager Interface | 57 |
| 12. DOS 4.5 Data Structures..... | 66 |
| 13. DOS 4.5 Clock Access | 71 |
| 14. DOS 4.5 Error Processing..... | 76 |
| 15. DOS 4.5 Chain Command..... | 77 |
| 16. ProDOS Disk I/O Algorithm..... | 82 |
| 17. Using DOS 4.5 Commands..... | 85 |
|
II. Apple][ROM Modifications..... |
87 |
| 1. Corrected HLIN Drawing Algorithm | 87 |
| 2. Soft Switches in the Apple][e | 90 |
| 3. SWEET16 Metaprocessor | 97 |
| 4. Applesoft Garbage Collector..... | 101 |
| 5. Building a New Apple][e ROM..... | 106 |
| 6. Apple][e Character Generator ROM..... | 109 |
| 7. Peripheral Slot Card Signature Bytes | 112 |
|
III. DOS 4.5 Commands |
117 |
| 1. System Commands | 122 |
| 2. File System Commands | 134 |
| 3. Applesoft File Commands | 149 |
| 4. Binary File Commands..... | 152 |
| 5. Sequential Text File Commands..... | 155 |
| 6. Random-Access Data File Commands | 162 |

IV. DOS 4.5 Assembly Language Routines..... 171

| | |
|--|-----|
| 1. DOSWARM Entry Address 0x3D0..... | 172 |
| 2. DOSCOLD Entry Address 0x3D3..... | 172 |
| 3. CALLFM Entry Address 0x3D6..... | 173 |
| 4. CALLRWTS Entry Address 0x3D9..... | 173 |
| 5. GETFMCB Routine at 0x3DC..... | 174 |
| 6. RDCLKVSN Vector Address at 0x3E1..... | 175 |
| 7. GETIOCB Routine at 0x3E3..... | 175 |
| 8. PRERRADR Vector Address at 0x3E8..... | 176 |
| 9. HOOKDOS Entry Address 0x3EA..... | 176 |
| 10. XFERADR Transfer Address at 0x3ED..... | 177 |
| 11. AUTOBRK Entry Address 0x3EF..... | 177 |
| 12. AUTORSET Vector Address at 0x3F2..... | 178 |
| 13. PWRSTATE Variable Byte at 0x3F4..... | 178 |
| 14. USRAHAND Entry Address 0x3F5..... | 179 |
| 15. USRYHAND Entry Address 0x3F8..... | 180 |
| 16. NMASKIRQ Entry Address 0x3FB..... | 180 |
| 17. MASKIRQ Vector Address at 0x3FE..... | 181 |
| 18. DOS 4.5H GOTOMON Notes..... | 182 |
| 19. GOTOMON2 Entry Address 0xBE00..... | 183 |
| 20. GOTOMON1 Entry Address 0xBE08..... | 183 |
| 21. BLDVRSN Variable Byte at 0xBFF0..... | 184 |
| 22. BLDNMBR Variable Byte at 0xBFF1..... | 184 |
| 23. MNGDISK Vector Address at 0xBFF2..... | 185 |
| 24. MNGVALS Vector Address at 0xBFF4..... | 186 |
| 25. MNGUSER Vector Address at 0xBFF6..... | 187 |
| 26. INITDOS Vector Address at 0xBFF8..... | 188 |
| 27. INITVAL Vector Address at 0xBFFA..... | 189 |
| 28. BCFGNDX Variable Byte at 0xBFFC..... | 189 |
| 29. NBUF1PG Address Byte at 0xBFFD..... | 190 |
| 30. BOOTADR Variable Byte at 0xBFFE..... | 191 |
| 31. BOOTPGS Variable Byte at 0xBFFF..... | 191 |

V. DOS 4.5 Operational Environment 193

| | |
|--|-----|
| 1. Disk Window..... | 194 |
| 2. EPROM Operating System (EOS) for SCRG quikLoader..... | 196 |
| 3. VTOC Manager (VMGR)..... | 210 |
| 4. Big Mac..... | 211 |
| 5. CFFA Card..... | 221 |
| 6. Volume Manager (VOLMGR)..... | 228 |
| 7. File Developer (FID)..... | 233 |
| 8. Lazer's Interactive Symbolic Assembler (Lisa)..... | 235 |
| 9. Program Global Editor (PGE)..... | 246 |
| 10. Global Program Line Editor (GPLE)..... | 249 |
| 11. Axlon RAM Disk 320..... | 250 |
| 12. RanaSystems EliteThree..... | 257 |

| | |
|--|-----|
| 13. First Class Peripherals Sider | 260 |
| 14. Sourceror | 266 |
| 15. Applesoft Formatter | 267 |
| 16. Binary File Installation (BFI) | 269 |
| 17. SCRG PROMGRAMER | 274 |
| 18. Real Time Clock | 276 |
| 19. JFD Parallel Printer Buffer | 282 |
| 20. Asynchronous Data Transfer (ADT) | 288 |
| 21. TrackScan | 289 |
| 22. ClientServer | 295 |
| 23. CHAR Editor and LENGTH | 312 |
| 24. ICON Maker | 316 |
| 25. Apple][+ Memory Upgrade | 330 |
| 26. Apple][+ Keyboard Modification | 335 |
| 27. Last Concluding Thoughts | 337 |

VI. Autobiographical Information..... 341

Index 349

SAMPLE

This page intentionally left blank.

List of Figures

| | |
|--|----|
| Figure I.4.1. Page-Zero Memory Utilization | 8 |
| Figure I.5.1. Disk Volume VTOC in DOS 3.3 | 11 |
| Figure I.5.2. Data Disk Volume VTOC in DOS 4.5 | 12 |
| Figure I.7.1. First Volume Catalog Sector in DOS 4.5 | 18 |
| Figure I.7.2. TSL Sector Structure in DOS 4.5 | 21 |
| Figure I.8.1. RWPAGES Routine in DOS 4.5H | 23 |
| Figure I.8.2. Attaching a Disk][Interface Card Handler in DOS 4.5 | 27 |
| Figure I.8.3. Requesting and Detaching a Slot Card Handler in DOS 4.5 | 27 |
| Figure I.8.4. Using MNGUSER to Manage DOS 4.5 | 28 |
| Figure I.8.5. Using MNGUSER in LOADMAC | 29 |
| Figure I.8.6. Accessing and Changing INITVALS in DOS 4.5 | 31 |
| Figure I.8.7. Changing the Boot Configuration Data Structure in DOS 4.5 | 32 |
| Figure I.9.1. Generating RDNIBLBT in Disk][Firmware | 33 |
| Figure I.9.2. Generating RDNIBL and WRNIBL in DOS 4.5 | 34 |
| Figure I.9.3. Reading the DOS Version in DOS 4.5 | 37 |
| Figure I.9.4. Reading the Date and Time in DOS 4.5 | 37 |
| Figure I.9.5. Printing a File Manager Error in Big Mac | 38 |
| Figure I.10.1. Disk][Carriage and Cam Table | 42 |
| Figure I.10.2. Four Half-Phase Track Separation | 43 |
| Figure I.10.3. Three Half-Phase Track Separation | 43 |
| Figure I.10.4. Inside View of a Stepper Motor | 44 |
| Figure I.10.5. Index Byte Format for the P6 ROM | 48 |
| Figure I.10.6. Setting the Disk][Interface Card to READ Mode | 49 |
| Figure I.10.7. Accessing the Disk][Interface Card to READ a Disk Byte | 49 |
| Figure I.10.8. Accessing the Disk][Interface Card to Sense Write Protect | 49 |
| Figure I.10.9. Accessing the Disk][Interface Card to Write Auto-Sync Bytes | 50 |
| Figure I.10.10. Accessing the Disk][Interface Card to Write Disk Bytes | 51 |
| Figure I.10.11. Adjusted Index Byte Format for the P6 ROM | 52 |
| Figure I.10.12. Functional Index Byte Format for the P6 ROM | 52 |
| Figure I.10.13. Functional ROM Read Index | 53 |
| Figure I.10.14. Functional ROM Read Data | 53 |
| Figure I.10.15. Functional ROM Write Index | 54 |
| Figure I.10.16. Functional ROM Write Data | 54 |
| Figure I.11.1. File Manager Command Parameter List in DOS 4.5 | 56 |
| Figure I.11.2. Using the File Manager Context Block for the INIT Command | 60 |
| Figure I.11.3. Using the File Manager Context Block for the URM Command | 63 |
| Figure I.11.4. Using the File Manager Context Block for the TOUCH Command | 63 |
| Figure I.11.5. Using the File Manager Context Block for the TS Command | 64 |
| Figure I.11.6. Using the File Manager Context Block for the WTS Command | 65 |
| Figure I.11.7. Using the File Manager Context Block for the CD Command | 65 |
| Figure I.12.1. Accessing and Changing 8-Bit CMDVALS in DOS 4.5 | 68 |
| Figure I.12.2. Reading a 16-Bit LOADLEN Value in Lisa | 71 |
| Figure I.13.1. Reading Generic Clock Data Using Applesoft | 73 |
| Figure I.13.2. Reading Generic Clock Data Using Assembly Language | 73 |

| | |
|---|-----|
| Figure I.13.3. Page-zero Wraparound in Read Clock Algorithm | 74 |
| Figure I.15.1. Example Applesoft Program Layout in Memory | 78 |
| Figure II.1.1. First HLIN Code Adjustment | 88 |
| Figure II.1.2. Second HLIN Code Adjustment | 88 |
| Figure II.1.3. HLIN Demonstration Program in Applesoft | 88 |
| Figure II.1.4. Original ROM HLIN Routine | 89 |
| Figure II.1.5. Modified ROM HLIN Routine | 89 |
| Figure II.2.1. Notes for Tables II.2.1 to II.2.6 | 94 |
| Figure II.6.1. International XML File | 109 |
| Figure II.6.2. Character Set Bitmap TIFF File | 110 |
| Figure II.6.3. Icon TIFF Bitmap File | 110 |
| Figure II.6.3. Load ROM File for EDITROM | 111 |
| Figure II.6.4. ROM Data Show Mode | 111 |
| Figure II.6.5. ROM Data Edit Mode | 111 |
| Figure II.6.6. Save ROM Data for EDITROM | 111 |
| Figure II.7.1. Video Firmware Note for the Apple //e | 114 |
| Figure III.1.1. CONFIG Command Line Display | 123 |
| Figure III.1.2. Set CONFIG Program Display | 123 |
| Figure III.1.3. DATE Command for Thunderclock | 125 |
| Figure III.1.4. Capturing DATE Data | 125 |
| Figure III.1.5. HELP Command Display | 126 |
| Figure III.1.6. HELP Command HELP Content | 126 |
| Figure III.1.7. HELP Command INIT Content | 126 |
| Figure III.1.8. HELP Command LSAVE Content | 126 |
| Figure III.1.9. DOS 4.5L MAXFILES Command | 128 |
| Figure III.1.10. DOS 4.5H MAXFILES Command | 128 |
| Figure III.1.11. MON Command Display | 129 |
| Figure III.1.12. NOMON Command Display | 129 |
| Figure III.1.13. PHASE Command Display | 131 |
| Figure III.1.14. PHASE 3 VTOC Display | 131 |
| Figure III.1.15. SV Command Display | 133 |
| Figure III.1.16. USER Command Usage | 133 |
| Figure III.2.1. CATALOG and CAT Command | 135 |
| Figure III.2.2. LS R Command Display | 135 |
| Figure III.2.3. CD Command Display | 137 |
| Figure III.2.4. DELETE Command Display | 137 |
| Figure III.2.5. DIFF Command Display | 138 |
| Figure III.2.6. GREP Command Display | 138 |
| Figure III.2.7. INIT Command for Boot Volume | 139 |
| Figure III.2.8. INIT Command for Data Volume | 139 |
| Figure III.2.9. LIST Text File Command | 141 |
| Figure III.2.10. LIST Applesoft File Command | 141 |
| Figure III.2.11. LOCK File Command Display | 142 |
| Figure III.2.12. LOCK Volume Command Display | 142 |
| Figure III.2.13. RENAME Command Display | 144 |
| Figure III.2.14. TOUCH Command Display | 144 |
| Figure III.2.15. TS Command of a VTOC | 144 |
| Figure III.2.16. TS Command in 80-Column | 144 |
| Figure III.2.17. UNLOCK File Command | 145 |

| | |
|---|-----|
| Figure III.2.18. UNLOCK Volume Command..... | 145 |
| Figure III.2.19. URM Command Display | 146 |
| Figure III.2.20. VERIFY Command Display..... | 146 |
| Figure III.2.21. WTS Command Display..... | 148 |
| Figure III.2.22. WTS Command in 80-Column..... | 148 |
| Figure III.3.1. Listing of START & PROGRAM2..... | 149 |
| Figure III.3.2. Output of START & PROGRAM2 | 149 |
| Figure III.3.3. LOAD and SAVE Commands..... | 151 |
| Figure III.3.4. SAVE Commands Display | 151 |
| Figure III.4.1. BLOAD and BSAVE Commands | 153 |
| Figure III.4.2. BRUN Command Display | 153 |
| Figure III.4.3. LLOAD and LSAVE Commands..... | 154 |
| Figure III.5.1. OPEN, WRITE, and CLOSE..... | 156 |
| Figure III.5.2. APPEND Command Display..... | 156 |
| Figure III.5.3. EXEC Command Display..... | 158 |
| Figure III.5.4. No PROMPT, EXEC,Rr Display | 158 |
| Figure III.5.5. POSITION and READ Commands | 159 |
| Figure III.5.6. READ,Bb Command Display..... | 159 |
| Figure III.5.7. TLOAD and TSAVE Commands..... | 161 |
| Figure III.5.8. TW Command Display..... | 161 |
| Figure III.6.1. OPEN, WRITE, and CLOSE..... | 164 |
| Figure III.6.2. Contents of RTEST.T Display..... | 164 |
| Figure III.6.3. READ Command..... | 165 |
| Figure III.6.4. Random-Access Data File CREATE..... | 166 |
| Figure IV.0.1. Direct and Indirect Approach for a Subroutine Call | 171 |
| Figure V.1.1. Disk Window Startup Display | 195 |
| Figure V.1.2. Select T/S Mode Display | 195 |
| Figure V.1.3. Edit Data Display..... | 195 |
| Figure V.1.4. Write Sector Data Display | 195 |
| Figure V.1.5. Print Sector Data Display | 195 |
| Figure V.1.6. Disk Window Error Message Display | 195 |
| Figure V.2.1. quikLoader Schematic with Circuit Modifications..... | 198 |
| Figure V.2.2. EOS Main Selection Menu | 202 |
| Figure V.2.3. EPROM 4 Catalog, Part 1..... | 202 |
| Figure V.2.4. EPROM 4 Catalog, Part 2..... | 202 |
| Figure V.2.5. EOS Catalog File Entry Values | 202 |
| Figure V.2.6. Example Code for QLBIENEOS Utilization | 207 |
| Figure V.3.1. VMGR Option Menu..... | 210 |
| Figure V.3.2. Show VTOC for Option 2..... | 210 |
| Figure V.3.3. Edit VTOC for Option 3..... | 210 |
| Figure V.3.4. Save VTOC for Option 3..... | 210 |
| Figure V.3.5. Show VTOC Bitmap Option 4 | 211 |
| Figure V.3.6. Edit VTOC Bitmap Option 5..... | 211 |
| Figure V.4.1. Big Mac Main Menu..... | 212 |
| Figure V.4.2. Big Mac Exit..... | 212 |
| Figure V.4.3. Big Mac Mnemonic Hashing Algorithm | 214 |
| Figure V.6.1. VOLMGR Product Warning Display | 228 |
| Figure V.6.2. VOLMGR Main Menu | 228 |
| Figure V.6.3. Manage Firmware Menu..... | 229 |

| | |
|---|-----|
| Figure V.6.4. Manage CompactFlash Menu | 229 |
| Figure V.6.5. CF Memory Utilization..... | 231 |
| Figure V.6.6. Device Identity Contents | 231 |
| Figure V.6.7. Manage Drives Menu | 231 |
| Figure V.6.8. Manage Volumes Menu..... | 231 |
| Figure V.6.9. Manage DOS Images Menu..... | 232 |
| Figure V.6.10. List DOS Images Display | 232 |
| Figure V.7.1. FID Main Menu | 233 |
| Figure V.7.2. FID Copy Files | 233 |
| Figure V.8.1. Lisa80 Startup Display | 236 |
| Figure V.8.2. Lisa80 Source Code List Display | 236 |
| Figure V.8.3. Lisa80 SETSCRN Routine | 237 |
| Figure V.8.4. Lisa80 READSCRN Routine | 237 |
| Figure V.8.5. Lisa80 SAVESCRN Routine..... | 238 |
| Figure V.8.6. Lisa80 SETUP80 Utility..... | 238 |
| Figure V.8.7. Lisa80 Screen Editing Definitions..... | 238 |
| Figure V.8.8. DOS 3.3 Source Code Volume..... | 241 |
| Figure V.8.9. EOS Image Segment Files | 241 |
| Figure V.8.10. EOS1 Image Creation | 241 |
| Figure V.8.11. EOS2 Image Creation | 241 |
| Figure V.9.1. Applesoft LIST Function 1 | 248 |
| Figure V.9.2. PGE LIST Function | 248 |
| Figure V.9.3. PGE RENUMBER Function | 248 |
| Figure V.9.4. Applesoft LIST Function 2 | 248 |
| Figure V.10.1. GPLE EDIT Function | 250 |
| Figure V.11.1. Original RAM Card Hardware Circuit Diagram | 253 |
| Figure V.11.2. Modified RAM Card Hardware Circuit Diagram..... | 254 |
| Figure V.11.3. RAM Card Hardware Modifications | 255 |
| Figure V.11.4. RAM Disk/Card Connect Program..... | 257 |
| Figure V.13.1. Sider Connect Program..... | 265 |
| Figure V.13.2. Sider Entry Points..... | 265 |
| Figure V.14.1. Sourceror Initialization..... | 266 |
| Figure V.14.2. Sourceror Startup/Help Display..... | 266 |
| Figure V.14.3. Sourceror Source Listing 1 | 266 |
| Figure V.14.4. Sourceror Source Listing 2 | 266 |
| Figure V.15.1. Applesoft Test Program Listing..... | 268 |
| Figure V.15.2. Applesoft Formatted Program | 268 |
| Figure V.16.1. BFI Splash Screen..... | 270 |
| Figure V.16.2. BFI Main Menu | 270 |
| Figure V.16.3. BFI Peripheral Selection..... | 270 |
| Figure V.16.4. Applesoft Program Directions | 270 |
| Figure V.16.5. Applesoft Program Selection..... | 271 |
| Figure V.16.6. Binary File Directions..... | 271 |
| Figure V.16.7. Binary File Selection | 271 |
| Figure V.16.8. Intermediate Installation Screen | 271 |
| Figure V.16.9. BFI Complete..... | 271 |
| Figure V.16.10. BFI Final Report..... | 271 |
| Figure V.17.1. PROmGRAMER Configuration..... | 275 |
| Figure V.17.2. PROmGRAMER Command Menu | 275 |

| | |
|---|-----|
| Figure V.18.1. Real Time Clock Circuit Diagram..... | 277 |
| Figure V.20.1. ADT Window | 288 |
| Figure V.20.2. ADT Configuration..... | 288 |
| Figure V.20.3. ADT Software Credits | 289 |
| Figure V.20.4. ADT Receive File Name Entry..... | 289 |
| Figure V.21.1. TrackScan Main Menu | 292 |
| Figure V.21.2. TrackScan Format Volume..... | 292 |
| Figure V.21.3. TrackScan Format Results..... | 293 |
| Figure V.21.4. TrackScan Scan Tracks..... | 293 |
| Figure V.21.5. TrackScan for Track 0x00 | 293 |
| Figure V.21.6. TrackScan for Track 0x17 | 293 |
| Figure V.22.1. Apple II Computer, Keyspan Adapter, MacBook Pro Computer..... | 295 |
| Figure V.22.2. Apple II Computer, Null Modem Adapter, Apple II Computer | 296 |
| Figure V.22.3. Selecting the USA19H142P1.1 | 296 |
| Figure V.22.4. Keyspan Connection Complete | 296 |
| Figure V.22.5. Waiting for Auto-Synchronization | 298 |
| Figure V.22.6. Waiting for Auto-Synchronization | 298 |
| Figure V.22.7. Auto-Synchronized to Server | 298 |
| Figure V.22.8. Auto-Synchronized to Client | 298 |
| Figure V.22.9. Client Volume-Verify Message..... | 301 |
| Figure V.22.10. Server Volume-Verify Message | 301 |
| Figure V.22.11. Calculation of GETPAGE | 306 |
| Figure V.23.1. Load CHAR File for EDITCHAR..... | 314 |
| Figure V.23.2. CHAR Data Show Mode | 314 |
| Figure V.23.3. CHAR Data Edit Mode..... | 314 |
| Figure V.23.4. Save CHAR Data for EDITCHAR..... | 314 |
| Figure V.23.5. Complete BFI CHAR Font | 316 |
| Figure V.23.6. Using LENGTH for String Draw | 316 |
| Figure V.24.1. DELTA Calculation in ICONPROC | 322 |
| Figure V.24.2. Initialization of ICON Maker | 323 |
| Figure V.24.3. ICON Maker Main Menu | 323 |
| Figure V.24.4. ICON Maker Draw Menu..... | 324 |
| Figure V.24.5. Draw Menu for Color | 324 |
| Figure V.24.6. Main Menu for Edit | 325 |
| Figure V.24.7. Moving an Icon in Edit..... | 325 |
| Figure V.24.8. Main Menu for Load..... | 325 |
| Figure V.24.9. Disk Errors in ICON Maker | 325 |
| Figure V.24.10. Main Menu for Save | 326 |
| Figure V.24.11. Entering Filename in Save..... | 326 |
| Figure V.24.12. ICON Maker Text Data for Icon | 326 |
| Figure V.24.13. Draw Menu for Horiz | 326 |
| Figure V.24.14. Drawing Lines in Horiz | 327 |
| Figure V.24.15. Drawing More Lines in Horiz..... | 327 |
| Figure V.24.16. Draw Menu for Vertical..... | 327 |
| Figure V.24.17. Drawing Lines in Vertical | 327 |
| Figure V.24.18. Draw Menu for Diagonal..... | 328 |
| Figure V.24.19. Drawing Lines in Diagonal..... | 328 |
| Figure V.24.20. Draw Menu for Curve..... | 328 |
| Figure V.24.21. Drawing Lines in Curve..... | 328 |

| | |
|---|-----|
| Figure V.24.22. Draw Menu for Box..... | 328 |
| Figure V.24.23. Draw Menu for Parallel | 328 |
| Figure V.24.24. Drawing Lines in Parallel | 329 |
| Figure V.24.25. Drawing Single Points in Dots | 329 |
| Figure V.25.1. Apple][+ Satellite Circuit Diagram | 331 |
| Figure V.26.1. Apple][+ Keyboard Modification Circuit..... | 334 |

List of Tables

| | |
|---|----|
| Table I.2.1. Major Improvements and Enhancements to DOS 4.5 | 3 |
| Table I.3.1. Apple II Memory Utilization | 5 |
| Table I.3.2. Apple II Memory Utilization with DOS 4.5L Installed | 6 |
| Table I.3.3. Apple II Memory Utilization with DOS 4.5H Installed | 6 |
| Table I.4.1. Available Page-Zero Locations Summary | 9 |
| Table I.4.2. Page-Zero Utilization in DOS 4.5 – Part 1 | 9 |
| Table I.4.3. Page-Zero Utilization in DOS 4.5 – Part 2 | 10 |
| Table I.5.1. VTOC Structure Block Definition in DOS 3.3 | 12 |
| Table I.5.2. VTOC Structure Block Definition in DOS 4.5 | 13 |
| Table I.5.3. Free Sector Bitmap for Each Track Having 16 Sectors | 14 |
| Table I.5.4. Free Sector Bitmap for Each Track Having 32 Sectors | 14 |
| Table I.5.5. Date and Time Definition in Variable Order in DOS 4.5 | 14 |
| Table I.6.1. Free Sector Bitmap for Each Track Having 32 Sectors in DOS 3.3 | 16 |
| Table I.7.1. Volume Catalog Entry in DOS 4.5 | 19 |
| Table I.7.2. Catalog Sector Data Offsets for File Entries in DOS 4.5 | 19 |
| Table I.7.3. File Type Byte Description in DOS 4.5 | 19 |
| Table I.7.4. TSL Structure Block Definition in DOS 4.5 | 21 |
| Table I.8.1. Boot and Data Management Structure Definition in DOS 4.5 | 22 |
| Table I.8.2. Boot Configuration Data Structure in DOS 4.5 | 23 |
| Table I.8.3. Disk Track/Sector Mapping to Memory Address in DOS 4.5L | 24 |
| Table I.8.4. Disk Track/Sector Mapping to Memory Address in DOS 4.5H | 24 |
| Table I.8.5. File Image Mapping to Memory Address in DOS 4.5L | 26 |
| Table I.8.6. File Image Mapping to Memory Address in DOS 4.5H | 26 |
| Table I.8.7. INITVALS Data Structure Definition in DOS 4.5 | 30 |
| Table I.9.1. RWTS Scratchpad RAM Locations in DOS 4.5 | 33 |
| Table I.9.2. Boot Sequence Steps in DOS 4.5 | 35 |
| Table I.9.3. Page 0x03 Interface Routines and Vectors in DOS 4.5 | 36 |
| Table I.10.1. RWTS Input/Output Context Block Definition in DOS 4.5 | 39 |
| Table I.10.2. RWTS Command Codes | 41 |
| Table I.10.3. RWTS Error Codes | 41 |
| Table I.10.4. Disk II Peripheral-card I/O Memory Definition | 48 |
| Table I.10.5. Selectable Logic State Sequencer Function | 48 |
| Table I.10.6. Shift/Storage Data Register Control Commands | 53 |
| Table I.11.1. File Manager Input/Output Context Block Definition | 57 |
| Table I.11.2. File Manager Command Codes | 58 |
| Table I.11.3. File Manager Read and Write Command Subcodes | 58 |
| Table I.11.4. File Manager SUBCODE Utilization | 59 |
| Table I.11.5. File Manager FMINITCD Boot Type for SUBCODE | 59 |
| Table I.11.6. File Manager VTOCVALS Substructure Initialization Data | 60 |
| Table I.11.7. Error Messages and Sources in DOS 4.5 | 62 |
| Table I.12.1. CMDVALS Data Structure Definition in DOS 4.5 | 67 |
| Table I.12.2. File Manager FMWORK Data Structure Definition in DOS 4.5 | 69 |
| Table I.12.3. File Manager File Buffer Definition in DOS 4.5 | 70 |
| Table I.13.1. Supported Clock Cards in DOS 4.5 | 72 |

| | |
|--|-----|
| Table I.15.1. Simple Variable Descriptor Definitions in Applesoft | 80 |
| Table I.15.2. Array Variable Descriptor Definitions in Applesoft | 80 |
| Table I.15.3. Single Array Element Descriptor Definitions in Applesoft | 80 |
| Table I.16.1. Comparison of DOS 4.5 and ProDOS RWTS | 83 |
| Table II.2.1. New Memory Management and Video Soft Switches | 91 |
| Table II.2.2. New Soft Switch Status Flags | 92 |
| Table II.2.3. Original Input/Output Control Soft Switches, Part 1 | 92 |
| Table II.2.4. Original Input/Output Control Soft Switches, Part 2 | 93 |
| Table II.2.5. Original Memory Management Soft Switches | 93 |
| Table II.2.6. Original Disk II Control Soft Switches | 94 |
| Table II.2.7. Zip Chip Control Soft Switches | 95 |
| Table II.2.8. CFFA Control Soft Switches | 95 |
| Table II.2.9. quikLoader Control Soft Switches | 96 |
| Table II.2.10. Sider, RAM Disk, RAM Card, and Rana Control Soft Switches | 96 |
| Table II.3.1. SWEET16 Register Descriptions | 98 |
| Table II.3.2. SWEET16 Non-Register Opcodes | 98 |
| Table II.3.3. SWEET16 Register Opcodes | 99 |
| Table II.4.1. Simple Variable Descriptor Processing in Bongers' Pass 1 | 102 |
| Table II.4.2. Array Variable Element Processing in Bongers' Pass 1 | 103 |
| Table II.4.3. Garbage Collector Comparison and Verification Timing Results (Minutes) | 105 |
| Table II.5.1. Disabled Applesoft Commands | 106 |
| Table II.5.2. Transformations to Build a New Apple IIe ROM | 108 |
| Table II.7.1. Peripheral Slot Card Signature Bytes | 113 |
| Table II.7.2. Revised Disk Drive Peripheral Slot Card Signature Bytes | 114 |
| Table III.0.1. Command Valid Keyword Table in DOS 4.5 | 117 |
| Table III.0.2. Command Table in DOS 4.5 | 118 |
| Table III.0.3. Keyword Name and Range Table in DOS 4.5 | 120 |
| Table III.0.4. Keywords and Their Definition in DOS 4.5 | 120 |
| Table III.1.1. System Commands in DOS 4.5 | 122 |
| Table III.1.2. CONFIG Command Bit Definitions | 123 |
| Table III.1.3. DOS 4.5L File Buffer Memory Locations | 128 |
| Table III.1.4. DOS 4.5H File Buffer Memory Locations | 128 |
| Table III.2.1. File System Commands in DOS 4.5 | 134 |
| Table III.2.2. Available Data Sectors for 36 Tracks, 16/32 Sectors/Track | 140 |
| Table III.2.3. Total Sectors in Initialized Volumes | 141 |
| Table III.3.1. Applesoft File Commands in DOS 4.5 | 148 |
| Table III.4.1. Binary File Commands in DOS 4.5 | 152 |
| Table III.5.1. Sequential Text File Commands in DOS 4.5 | 155 |
| Table III.6.1. Random-Access Data File Commands in DOS 4.5 | 162 |
| Table IV.17.1. Interrupt Handler System Status Byte Definition | 181 |
| Table V.2.1. quikLoader Bank Switching | 199 |
| Table V.2.2. quikLoader Firmware Entry Points | 200 |
| Table V.2.3. quikLoader EPROM 0 Containing EOS and Programs | 201 |
| Table V.2.4. EOS File Types | 201 |
| Table V.2.5. EOS Catalog File Entry Structure | 205 |
| Table V.2.6. BINEOS Catalog File Entry | 209 |
| Table V.4.1. Hash Table 1 – Assembly Directives | 215 |
| Table V.4.2. Hash Table 2 – Text Directives | 215 |
| Table V.4.3. Hash Table 3 – Macro Directives | 216 |

| | |
|---|-----|
| Table V.4.4. Hash Table 4 – Single Byte Instructions..... | 217 |
| Table V.4.5. Hash Table 5 – Branch Instructions..... | 218 |
| Table V.4.6. Hash Table 6 – Single Byte SWEET16 Register Instructions..... | 218 |
| Table V.4.7. Hash Table 7 – Multiple Addressing Mode Instructions..... | 219 |
| Table V.4.8. Format Byte Definition..... | 219 |
| Table V.5.1. CFFA Card Firmware Entry Points..... | 222 |
| Table V.5.2. CompactFlash Card CHS Geometry..... | 222 |
| Table V.5.3. CFFA Firmware Interface Control Registers..... | 223 |
| Table V.5.4. CFFA Firmware DVTS Variable Range..... | 223 |
| Table V.5.5. CompactFlash Card Drive/Volume Parameters..... | 224 |
| Table V.5.6. Block Utilization for a 1 GB CompactFlash Card..... | 225 |
| Table V.5.7. DOS 3.3 Patches for CFFA..... | 227 |
| Table V.8.1. Lisa USB Command..... | 238 |
| Table V.8.2. Lisa80 Command-Line Commands..... | 245 |
| Table V.11.1. RAM Card Memory Configuration Soft Switches..... | 252 |
| Table V.11.2. RAM Disk 320 Firmware Entry Points..... | 256 |
| Table V.12.1. Rana Disk Firmware Entry Points..... | 259 |
| Table V.13.1. Sider Logical Block Structure..... | 262 |
| Table V.13.2. Modified Sider Logical Block Structure..... | 262 |
| Table V.13.3. Sider Firmware Entry Points..... | 263 |
| Table V.18.1. SaRonix RTC58321 Real Time Clock Pinout..... | 276 |
| Table V.18.2. Real Time Clock Peripheral Interface Card I/O Addresses..... | 278 |
| Table V.18.3. Real Time Clock Configuration Register..... | 278 |
| Table V.18.4. Interrupt Rate Selection..... | 279 |
| Table V.18.5. Real Time Clock Registers..... | 279 |
| Table V.18.6. Clock Firmware Entry Points..... | 281 |
| Table V.19.1. 8035-Microprocessor Memory Map..... | 284 |
| Table V.19.2. User Flag 1 F1..... | 285 |
| Table V.19.3. Port Utilization..... | 285 |
| Table V.19.4. SELRB0 Utilization..... | 286 |
| Table V.19.5. SELRB1 Utilization..... | 286 |
| Table V.19.6. Primary R3 R/W Block Number Bits..... | 286 |
| Table V.19.7. Secondary R3 System Flag Bits..... | 287 |
| Table V.22.1. Client/Server Commands and Responses..... | 299 |
| Table V.22.2. CFDCBTBL Configuration Data Control Block Table Contents..... | 300 |
| Table V.22.3. TSDCBTBL Track/Sector Data Control Block Table Contents..... | 303 |
| Table V.22.4. TXDCBTBL Transfer Data Control Block Table Contents..... | 303 |
| Table V.22.5. RWDCBTBL Read/Write Data Control Block Table Contents..... | 307 |
| Table V.22.6. Error Handling Error Codes for Client Routines..... | 309 |
| Table V.22.7. Error Handling Error Codes for Server Routines..... | 309 |
| Table V.22.8. Client Source Routines for Error Codes..... | 310 |
| Table V.22.9. Server Source Routines for Error Codes..... | 311 |
| Table V.24.1. Drawing Commands Available in ICONPROC..... | 318 |
| Table V.24.2. Simple Colors Available in ICONPROC..... | 318 |
| Table V.24.3. Complex Colors Not Available in ICONPROC..... | 320 |
| Table V.24.4. Using ICONPROC Drawing Commands..... | 320 |
| Table V.25.1. Apple][+ Satellite Circuit Board Connections..... | 331 |
| Table V.25.2. Apple][+ Satellite Circuit Board Operation, Part 1..... | 332 |
| Table V.25.3. Apple][+ Satellite Circuit Board Operation, Part 2..... | 333 |

Table V.26.1. 74LS153 Truth Table 335

Table V.26.2. Generation of Unavailable Characters 336

SAMPLE

DOS 4.5 File and Volume Disk Management System

I. Designing Another New DOS

This publication describes the process and the products I created when I decided to design and program another enhanced Disk Operating System (DOS) for my Apple //e. Wherever I am able, I have included schematic diagrams, code samples, equations, figures, tables, and representative screen shots to help explain what I have created and many of the reasons why I did so. As in my previous designs of an Apple][DOS, i.e. DOS 4.1 and DOS 4.3, this has been an incredible journey for me. With DOS 4.5 I have again re-imagined that time when I mostly lived, breathed, and worked on Apple][computers, hardware, and software development continuously for a good period of my life many, many years ago.

1. Introduction

I have been an avid Apple][computer enthusiast, hobbyist, and professional software programmer since 1983 when I became the proud owner of an Apple][+ computer. Besides the Apple][+ computer, my initial system included an Apple][Language Card, an Apple Disk][Drive with a Disk][Interface Card, an Amdek color monitor, and an Epson MX100 printer with a Grappler+ Printer Interface card. During those early years I designed and built my own Apple][peripheral slot cards, I made electrical and hardware modifications to my Apple][+ motherboard and keyboard, and I wrote a substantial number of software programs initially using Applesoft BASIC (Applesoft hereafter) and then a few months later, I wrote software in 6502 assembly language. I soon acquired a Videx UltraTerm video display card and a Microsoft Z80 Softcard. With the Z80 Softcard I was able to write complex Fortran programs that analyzed tomographic reconstructions of the human spinal column. A year or so later I added the Southern California Research Group quikLoader and PROMGRAMMER cards to my system, a Johnathon Freeman Designs (JFD) Parallel Printer Buffer, and an Axlon RAM Disk 320 with its interface card.

I used C language in my professional programming career for the design and development of ultra-high-speed data collection systems for tactical radar and sensor development. Now that I am retired from the aerospace industry, I have always wanted to dig into, tear apart, and learn the intricacies of the last available version of DOS 3.3 for the Apple][series of computers. I thought the last DOS 3.3 version was published on August 25, 1980. Then I recently came across another DOS 3.3 version published years later on January 1, 1983. That later DOS contains even more patches for the DOS APPEND command and for Apple //e initialization. What I learned from the 1980 publication flabbergasted me: the software is exciting in its originality and concept vis-à-vis it was released just after the publication of Integer BASIC. However, I found the software to be somewhat juvenile in its structure and in its implementation. Apparently, very little attention was given to software design and review. It appeared to me that Apple made a strong push to release *something or anything* to consumers and vendors in

order to begin marketing software products on diskettes and hardware products to read and write those diskettes. And history does reveal that Apple Computer did outsource DOS and contracted for it to be delivered within thirty-five days for \$13,000 in April, 1978. Paul Laughton at Shepardson Microsystems wrote Apple's initial disk operating system using Hollerith cards, a card reader/writer, and a minicomputer.

Now that I have the time and the continuing curiosity to delve into Apple][DOS, I have the unique opportunity to create my own version of DOS that contains the power and the flexibility I always thought DOS ought to and could have. I call this version of Apple][DOS, the DOS 4.5 File and Volume Disk Management System, and it requires an Apple][that contains memory in the Language Card partition for its complete operation. This publication describes my fifth build of DOS 4.5. What a ride I have been on! Why? To see what I could do for this brilliant machine and its equally magnificent architecture! I hope that you find my journey into DOS 4.5 as fascinating as I found developing it.

2. Overview of the Improvements and Enhancements in DOS 4.5

I know there are a great many ProDOS users in the global Apple][community, but I never became at all interested in ProDOS. The work I did at Hughes Aircraft in the mid 1980's consisted of using assembly language for programming an operating system executive and hardware interface driver routines on Gould SEL 2780, 6780, and 9780 mainframe computers. These computers hosted a proprietary operating system that allowed our team to simulate the hardware of a Radar Digital Processor (RDP) traveling above the earth's surface in virtually real time. In order to accomplish that goal and simulate real time navigation, the computer's file system was essentially flat: every software developer had their own directory, and these user directories contained no subdirectories. I was very comfortable with the idea of a flat file system as it was very much like Apple's DOS 3.3. I was simply not comfortable with a slew of subdirectories exemplified by Apple's ProDOS. My thought was always "How does one recall the path to follow in order to find anything?" With the advent of the Macintosh computer and later when I became familiar with the UNIX file system, my subdirectory fears vanished and I cannot imagine a modern computer file system without subdirectories. However, I still remain passionate about Apple][DOS and I leave ProDOS to those who are comfortable with that operating system architecture. Though what I have seen of ProDOS recently, I believe it could definitely use a facelift, seriously. I also believe that ProDOS is far better suited on a machine with a 16-bit microprocessor much like that found in the Apple //GS.

I am sure many are curious and want to know what is new and different in DOS 4.5, and what makes this version of the DOS File and Volume Disk Management System so special. Looking back over my previous publications of DOS 4.1 and DOS 4.3, I realized that I should have included this enhancement information with every version and for each software build, if only for historical reasons. Like, which version and build did I solve the Track 0x00 utilization quest? Which version and build did I start labeling volumes? Which version and build did I solve the Disk Full logic error? Taken all together, I have done an incredible amount of research, writing, and software development to reach the level of perfection that is contained in DOS 4.5 with Build 5. And, to say the least, I have done an incredible amount of unit testing for each and every logical function under normal and abnormal (i.e. error) conditions. However small the list of improvements and enhancements unique to DOS 4.5 may seem, I have spent countless hours developing and testing those improvements and enhancements alone and in concert with the overall DOS 4.5 command repertoire and its system functional capabilities.

| Module | Description of Improvement or Enhancement |
|----------|---|
| CMD1 | Move CMDINDX to keyboard variables and clear keyboard variables before, not after parsing a new DOS command. No need to range check SV and USER parameters. |
| CMD2 | Modified TS to read sector data into caller's specified buffer. Added ENTRMON routine that calls INITPTRS before entering RAM Monitor. Removed COPYVALS after moving CD to File Manager with CDHNDLR. Moved DOADRINC to CMD2. |
| CMD3 | Rewrote GREP so that it no longer requires a character string marker for a multiple-word character string search, and less code was required (Aha! Moment) |
| CMD4 | FMDVR now copies VALSPHAS to FMPHASE, CONFIG is set to default if R keyword is included, PHASE is set to default if R keyword is included, MAXFILES is set to default if R keyword is included, SAVALRD writes CONFIG, PHASE, and MAXFILES values into the Value Read Buffer if DOS is in the RUN mode, SV writes its value into the Value Read Buffer if DOS is in the RUN mode (Aha! Moment), CD uses File Manager to read VTOC for CONFIG value, PRTSDV now prints volume lock status. |
| MNGR1 | External File Manager entry uses FILALC/NOFILALC values to set KEYWORD1 for file allocation control, now FMTBLJMP uses memory placement of File Manager routines to selectively copy FM Context Block SDV values to FMWORK since Byte Offset and Range Length are overloaded values in FMWORK. |
| MNGR2 | TOUCH only copies VALSCNFG to DOSCONFIG if R keyword is included to save the DOS configuration, DELETE handler rewritten and its logic reordered, CATALOG handler does not show build information if DOSVRN < 0x33 or DOSBUILD = 0, FREESECT now clears SECBTMAP before calling RORBITMP, not after. Added CDHNDLR. |
| MNGR3 | ALLOCSEC modified to perform up to 4 scans of the VTOC bitmap, ALLOCNTR moved to CMDVALS. |
| DATA1 | Moved CMDUSER before CMDHELP, set KWRANGEH for drive to 81. |
| BUFR1 | Moved CMDINDX after MONVAL in keyword variables and put ALLOCNTR after CMDLNIDX where CMDINDX was, removed TRKNUMBR from FMWORK, set VALRDBUF to 8 bytes. File buffer is now 0x0245 bytes in size. Added 7-byte SCRCHTBL table (Aha! Moment) and adjusted SPAREBUF size. |
| BUFR2 | Removed WATRKNUM in WORKAREA (parallel variable to TRKNUMBR). |
| RWTS1 | If slot number index to SCRCHTBL is zero, RWTS scratchpad RAM locations for track and phase are cleared for drives 1 and 2, and calls BLDNIBL when RWTS is called the first time for that slot number (Aha! Moment). BLDNIBL dynamically builds RDNIBL and WRNIBL tables in same page with NBUF2. |
| RWTS2 | PHASE value better managed and DISKfmt now initially formats track 0x00, reduces SYNCNT, then formats entire volume starting with track 0x00. |
| MNGEXVAL | Modified this function to return the Y-reg incremented twice when reading/writing 16-bit values and once when read/writing 8-bit values found in the CMDVALS and FMWORK Data structures. Removed access to the INITVALS Data structure. |
| MNGEXUSR | Removed the call to CLRVALS from this function. The call to CLRVALS is now properly made by DOSINIT just prior to the COLDSTRT entry point. |
| HELP | Any key press other than ESC, RTN, or arrow keys select the display of the DOS 4.5 Management screen. |
| USER | The USER handler now resides in Main memory, its call to the USER address returns to Main memory, and finally USER re-enables RAM memory in the Language Card partition to complete its processing. |

Table I.2.1. Major Improvements and Enhancements to DOS 4.5

DOS 4.5 is designed to reside either in Main memory or in the memory of the Language Card partition. The version of DOS 4.5 that resides fully in Main memory is DOS 4.5L, and its executable code begins at 0x9F00, its variables and data buffers begin at 0x9A8A, and its two file buffers begin at 0x9600 where HIMEM is set, the very same memory location as in DOS 3.3. In comparison, DOS 4.1 sets HIMEM to 0x9625 when MAXFILES is set to three file buffers. Thus, the additional code that comprises DOS 4.5L is essentially the size of one file buffer. Unlike DOS 3.3 or DOS 4.1L, DOS 4.5L is a single, continuous object code image that is 0x2100 bytes in size. On the other hand, the version of DOS 4.5 that resides in the memory of the Language Card partition is DOS 4.5H, and it occupies both

banks of memory from 0xD000 to 0xDFFF and the remaining memory of the partition from 0xE000 to 0xF7FF. The Apple][user has complete freedom to use all of Main memory below 0xBE00 where HIMEM is set.

The foundation for DOS 4.5 is DOS 4.3H which includes the HELP command. In order to better appreciate all of the improvements and the enhancements I have added to DOS 4.5, I have tabulated those changes according to each software module as shown in Table I.2.1. Please realize that nothing in DOS 4.3 was removed in order to provide the code space required by all of these improvements and enhancements in DOS 4.5. I simply spent a sizable amount of time and effort re-working certain modules to yield the same or better functionality in far less code space. I can truthfully say that I experienced several Aha! Moments that certainly assisted in making these improvements and enhancements to DOS 4.5 possible.

3. DOS 4.5 Software Development Strategies

Let's begin with some software design and development strategies. In order to design reliable and powerful software for a particular machine or platform, one must understand the complete architecture of that machine. I believe this design approach is fully applicable even to the Apple][computer: either code or data occupies fixed addressable memory where some defined memory locations are reserved for the stack, text, graphics, control, and peripheral slot cards. Code is further restricted in the Apple][by the rather limited 6502-microprocessor Instruction Set. My obvious goal strategy is to design software in such a way as to create the most functionality using the least amount of code and data space. I believe this methodology yields the greatest range of code effectiveness.

I highly recommend obtaining and referring to a number of reference publications for the Apple][hardware. I have used the Apple][*Reference Manual*, the Apple][e *Reference Manual*, the Apple][e *Technical Reference Manual*, the Apple][*The DOS Manual* Disk Operating System, *Beneath Apple DOS* by Don Worth and Pieter Lechner, *Understanding the Apple][* by Jim Sather, and *What's Where in the APPLE A Complete Guide to the Apple Computer* by William F. Luebbert to obtain much of my understanding in how the Apple][hardware functions. I have also referred to the *APPLESOFT][* manual to obtain my understanding of Apple's Applesoft programming language. It is an absolutely required reference manual to have in order to learn that BASIC programming language. These references provides the reader a fairly complete understanding of the architecture of the Apple][computer as well as how to create software programs using the Applesoft language. No reference manual is perfect and people do make mistakes, however these manuals contain very few errors.

In order to study how other people have approached the hardware architecture of the Apple][in creating their software applications, it is necessary to obtain their source code or generated source code from their object code, the code that actually executes from within the Apple][memory. Rarely have I ever found published source code. Glen Bredon designed *Sourceror* as a subsidiary tool to his assembler *Big Mac* that creates *Big Mac* source code files from assembly language object code files. *Big Mac* can also save source code as text files so that *Big Mac* source code can be migrated to other assemblers like *Lisa*.

For my assembly language programming I use Gerard Putter's application Virtual][, Version 9.3, to create my software applications, and that is the platform I use to perform my initial, though simulated software testing. Once I am satisfied with a software program or utility operating within the Virtual][

environment, I transfer the volume image containing that software program or utility to an Enhanced Apple //e. I have found some discrepancies between Virtual][and my Enhanced Apple //e particularly when I am enabling memory in the Language Card partition: two successive *writes* to memory address 0xC083 does **not** write enable Bank 2 in the Language Card partition in my Enhanced Apple //e as it does in Virtual][. Two successive *reads* of memory address 0xC083 function the same in both my Enhanced Apple //e and in Virtual][to enable Bank 2 memory. I have brought this to the attention of Mr. Putter. Also, Main memory is not initialized at power-up in quite the same way in my Enhanced Apple //e as it is in Virtual][. I believe DOS 3.3 always assumes that an Apple][powers up with all bytes in page-zero memory set to 0xFF. Virtual][also makes this same assumption. I know I have been caught unaware that all bytes in Auxiliary page-zero memory are not always set to 0xFF at power-up. Therefore, I have included a call to SETNORM during Boot Stage 2 to ensure that page-zero memory location 0x32 is, indeed, set to 0xFF. I have used AUXMOVE to manually *stash* some ProDOS code in Auxiliary memory from within Virtual][. The code disappears (becomes overwritten) when I then boot into DOS 4.5. This does not happen in the Enhanced Apple //e: the ProDOS code or any stashed code can still be safely found in Auxiliary memory even after a DOS reboot. Always, always, always make final tests using **real** hardware.

| Memory Page | Description | Description |
|-------------|---|-------------|
| 0x00 | Page-zero variables, pointers, routines, and special addressing modes | |
| 0x01 | Stack for the 6502-microprocessor | |
| 0x02 | INPUT buffer, Applesoft interpretation buffer | |
| 0x03 | User buffer, DOS vectors and routines | |
| 0x04–0x07 | Text or LORES graphics Page 1 | |
| 0x08–0x0B | Applesoft program start, Text or LORES graphics Page 2, or available for software | |
| 0x0C–0x1F | Available for software | |
| 0x20–0x3F | HIRES graphics Page 1, or available for software | |
| 0x40–0x5F | HIRES graphics Page 2, or available for software | |
| 0x60–0xBF | Available for software | |
| 0xC0 | System Soft Switches | |
| 0xC1–0xC7 | Peripheral-card ROM memory for slots 1-7, or CX ROM | |
| 0xC8–0xCF | Peripheral-card expansion ROM memory for slots 1-7, or CX ROM | |
| 0xD0–0xDF | Bank 2, ROM Applesoft Interpreter routines | Bank 1 |
| 0xE0–0xF7 | ROM Applesoft Interpreter routines | |
| 0xF8–0xFF | ROM Monitor routines | |

Table I.3.1. Apple][Memory Utilization

Before beginning any discussion of a complicated subject like a file and volume disk management system for the Apple][, it is usually easier to understand such a system if each component of that system is shown as part of a Big Picture. That Big Picture is shown in the following three tables, Tables I.3.1, I.3.2, and I.3.3. Though certainly not to any particular scale, Table I.3.1 shows how memory is utilized in the Apple][and where the basic Apple][system hardware and software components can be found in

Main memory and in the memory of the Language Card partition which is shown in the more shaded bottom lines in each of these three tables. The basic components shown in Table I.3.1 are the 6502-microprocessor memory requirements, the DOS vectors and routines, text and LORES graphic pages, HIRRES graphic pages, system Soft Switches, peripheral-card and CX ROM memory, where the ROM Applesoft interpreter is found, and where the ROM Monitor resides. If any of the components shown in Table I.3.1 are unfamiliar, it would be to your advantage now to locate and study one or more of the above referenced publications to refresh and increase your understanding of that component. Even the Apple][*Reference Manual* that came with my Apple][+ computer contains invaluable information applicable to the entire family of Apple][computers. I even own a few **SAMS** Publications that have provided me with enhanced understanding of many of the components shown in Table I.3.1.

| Memory Page | Description | Description |
|-------------|---|-------------|
| 0x00–0x03 | System, 6502-microprocessor memory utilization | |
| 0x04–0x07 | Text Page 1 | |
| 0x08–0x95 | Available for software | |
| 0x96–0x99 | DOS 4.5L HIMEM, DOS 4.5L file buffers | |
| 0x9A–0x9E | DOS 4.5L working variables, workarea buffer, VTOC and catalog buffers, nibble buffers | |
| 0x9F–0xBE | DOS 4.5L Command and File Manager, RWTS, and all other software routines | |
| 0xBF | DOS 4.5L bootstrap routines | |
| 0xC0 | System Soft Switches | |
| 0xC1–0xCF | Peripheral-card ROM memory for slots, CX ROM | |
| 0xD0–0xDF | Bank 2 ROM routines | Bank 1 |
| 0xE0–0xF7 | ROM routines | |
| 0xF8–0xFF | ROM Monitor routines | |

Table I.3.2. Apple][Memory Utilization with DOS 4.5L Installed

| Memory Page | Description | Description |
|-------------|--|----------------------|
| 0x00–0x03 | System, 6502-microprocessor memory utilization | |
| 0x04–0x07 | Text Page 1 | |
| 0x08–0xBD | Available for software | |
| 0xBE–0xBF | DOS 4.5H HIMEM, DOS 4.5H Language Card partition software interface, DOS 4.5H bootstrap routines | |
| 0xC0 | System Soft Switches | |
| 0xC1–0xCF | Peripheral-card ROM memory for slots, CX ROM | |
| 0xD0–0xDF | RAM Bank 2, DOS 4.5H Command and File Managers | RAM Bank 1, DOS RWTS |
| 0xE0–0xE6 | RAM DOS 4.5H Command and File Managers | |
| 0xE7–0xEC | RAM DOS 4.5H working variables and workarea buffer | |
| 0xED–0xF7 | RAM DOS 4.5H file buffers | |
| 0xF8–0xFF | RAM Monitor routines | |

Table I.3.3. Apple][Memory Utilization with DOS 4.5H Installed

Table I.3.2 is similar to Table I.3.1 except that the 6502-microprocessor and ROM components have been diminished from view in lieu of showing visually where DOS 4.5L and its components are placed in memory. DOS 4.5L is typically configured to have two file buffers, and those buffers begin at 0x9600 and end at 0x9A89 in memory. From 0x9A8A to 0x9EFF is where the working variables and workarea buffer, the VTOC and catalog buffers, and the nibble buffers, the write translate, and the read translate tables are defined. The DOS 4.5L software routines reside in the continuous span of memory from 0x9F00 to 0xBFFF.

Table I.3.3 is also similar to Table I.3.1 and it, like Table I.3.2, shows visually where DOS 4.5H and its components are placed in memory. In order to manage the DOS 4.5H routines located in the memory of both banks in the Language Card partition, a set of software interface routines that control the utilization of memory in the Language Card partition is located in Page 0xBE. Page 0xBF contains the DOS bootstrap routines which are similar in nature to those same routines found in DOS 4.5L.

The following sections discuss the utilization of Apple][memory in great detail. It may be helpful to occasionally refer to Tables I.3.1 to I.3.3 in order to fully understand how that memory utilization relates to the entire hardware and software management of the Apple][computer by either version of the DOS 4.5 File and Volume Disk Management System. The Apple][computer is truly a brilliantly designed machine and it has an equally magnificent architecture. I hope you find my presentation of the Apple][computer vis-à-vis DOS 4.5 interesting, enlightening, and useful in view of your own hardware and software experiences with this delightful machine.

4. Page-Zero Utilization

The Instruction Set for the 6502-microprocessor (as well as the 65C02-microprocessor) includes certain microprocessor instructions that utilize variables located in the first 256 bytes, or page, of addressable memory, that is, locations 0x0000 to 0x00FF. I designate this area of memory to be *page-zero*.

When Steve Wozniak designed the Apple ROM Monitor (a collection of low-level software routines), he allocated a number of page-zero locations for its variables and pointers. Similarly, Applesoft, DOS, and virtually all other assembly language programs use page-zero locations in order to utilize those specific instructions. The 6502-microprocessor contains an accumulator called the A-register and two index registers called the X-register and the Y-register. Page-zero instructions using these registers include load and store instructions, indexed load and store instructions, indexed indirect addressing instructions using the X-register, and indirect indexed addressing instructions using the Y-register. Page-zero wraparound occurs with indexed and indexed indirect addressing instructions using the X-register and indexed addressing instructions of the X-register using the Y-register, but page-zero wraparound does **not** occur with indirect indexed addressing instructions using the Y-register. Yes, initially, addressing modes can certainly be a little bit confusing.

When developing a new assembly language program, it is critical to select page-zero locations that do not conflict with the Apple ROM Monitor, Applesoft, or DOS depending on whether those ROM applications and the applications that reside in the memory of the Language Card partition are important to the new program. Knowing which page-zero locations are used by or critical to ROM and resident applications can greatly simplify the selection of unused or available page-zero locations. Because DOS 3.3 supports Integer BASIC, a few page-zero locations are used to process that file type. DOS 4.5 also

uses those same page-zero locations for processing the Applesoft **CHAIN** command, for example, and many other DOS command enhancements. There are definitely obvious page-zero locations that cannot be used except for how they were intended, like the horizontal and vertical cursor locations **CH** and **CV**, respectively. Then, there are less obvious, rather dubious page-zero locations from **0x00** to **0x1F** that are used by some Applesoft commands. These page-zero locations are fair game for new programs that do not use the Applesoft interpreter or Steve Wozniak's *SWEET16* interpreter. Figure I.4.1 shows all of the used and the unused page-zero locations, and the Key defines those applications that use those particular locations according to my references and the best of my ability to decipher the ROM routines that make use of page-zero memory. The shaded areas in Figure I.4.1 are unused page-zero locations that are most likely not used by the Apple //e Monitor or Applesoft, so they are more than likely the better page-zero locations to select. Table I.4.1 summarizes all of the available page-zero locations that are not utilized by the ROM routines and by DOS 4.5 shown in Figure I.4.1. Keep in mind that indirect indexed addressing mode instructions using the Y-register **do** require a page-zero byte-pair, so it is even more critical that neither address byte is clobbered by software external to a new assembly language program.

| 0x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|----|------|-----|------|------|------|------|------|------|------|------|-------|-------|-------|-------|--------|--------|
| 00 | 1234 | 134 | 34 | 34 | 34 | 4 | | | | | 4 | 4 | 4 | 4 | 4 | 4 |
| 10 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 24 | 2 | 3 |
| 20 | 134 | 134 | 134 | 134 | 1346 | 134 | 1456 | 1456 | 1346 | 1346 | 13456 | 13456 | 1456 | 1456 | 123456 | 123456 |
| 30 | 14 | 12 | 134 | 1246 | 123 | 1236 | 136 | 136 | 136 | 136 | 123 | 123 | 12345 | 12345 | 123456 | 123456 |
| 40 | 156 | 156 | 1236 | 1236 | 1236 | 1 | 1 | 1 | 1 | 1 | 56 | 56 | 6 | 6 | 13 | 13 |
| 50 | 346 | 346 | 4 | 4 | 4 | 4 | 24 | 24 | 24 | 24 | 46 | 6 | 6 | 6 | 4 | 4 |
| 60 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 46 | 46 | 46 | 46 | 46 | 46 | 346 | 346 | 346 |
| 70 | 346 | 4 | 4 | 346 | 346 | 4 | 46 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 80 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 90 | 4 | 4 | 4 | 4 | 34 | 34 | 4 | 4 | 4 | 4 | 4 | 34 | 34 | 4 | 4 | 4 |
| A0 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 46 |
| B0 | 46 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| C0 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | |
| D0 | 4 | 4 | 4 | 4 | 4 | 4 | 46 | | 46 | 6 | 4 | 4 | 4 | 4 | 4 | 4 |
| E0 | 4 | 4 | 4 | | 4 | 4 | 4 | 4 | 34 | 34 | 4 | | | | | |
| F0 | 4 | 4 | 4 | 14 | 14 | 4 | 4 | 4 | 4 | 4 | | | | | | 34 |

Figure I.4.1. Page-Zero Memory Utilization

Key

1 – used by the ROM Monitor
 2 – used by the Mini Assembler
 3 – used by the Apple //e CX ROM

4 – used by Applesoft
 5 – used by RWTS
 6 – used by DOS 4.5

| Start | End | Description |
|-------|------|--------------|
| 0x06 | 0x09 | 4 bytes free |
| 0xCE | 0xCF | 2 bytes free |
| 0xD7 | 0xD7 | 1 byte free |
| 0xE3 | 0xE3 | 1 byte free |
| 0xEB | 0xEF | 5 bytes free |
| 0xFA | 0xFE | 5 bytes free |

Table I.4.1. Available Page-Zero Locations Summary

Tables I.4.2 and I.4.3 list all of the page-zero locations utilized by DOS 4.5 and defined in the DOS 4.5 source code file `INCL.L`. There are certainly common page-zero locations that all software routines can use as temporary variables and pointers. The 6502-microprocessor is not time-shared and there is no context switching between routines, so if a routine uses some common page-zero locations, that routine should complete all calculations and processing using those locations and not expect to find those same results sometime later. Examples of common page-zero locations would be `A1L/A1H` at `0x3C/0x3D`, `A2L/A2H` at `0x3E/0x3F`, `A3L/A3H` at `0x40/0x41`, `A4L/A4H` at `0x42/0x43`, `OPRND` at `0x44` and `0x45`, and the first three bytes of `DSCTMP` at `0x9D:0x9F`. As is shown in Tables I.4.2 and I.4.3, all of these page-zero locations are defined in the DOS 4.5 file `INCL.L`. Using these page-zero locations to move or copy data would be safe and not interfere with the ROM Monitor or Applesoft processing. Actually, several ROM Monitor routines require that some of these page-zero locations just mentioned contain your data before using those routines. The ROM Monitor routine `MOVE` at `0xFE2C` is one such example that uses `A1L/A1H` and `A2L/A2H` to move data in memory. It is really up to the user to confirm and verify that the selected page-zero memory locations do not interfere with other routines external to and required by any new software developed by a user.

| Address | Parameter | Description |
|---------|-----------|---|
| 0x24 | CH | horizontal cursor location |
| 0x25 | CV | vertical cursor location |
| 0x26 | BUFRADRZ | ROM firmware boot data field buffer address |
| 0x26 | TEMPZ | RWTS temporary 8-bit variable |
| 0x27 | TEMP2Z | RWTS temporary 8-bit variable |
| 0x28 | BASEZ | text screen line address |
| 0x2A | ASPTRSAV | DOS CHAIN array descriptor addresses |
| 0x2A | CURTRKZ | RWTS requested track |
| 0x2B | SLOT16Z | boot slot * 16 |
| 0x2C | DRVFLAG | RWTS data-changing drive flag |
| 0x2C | ADRDATMK | RWTS address/data mark |
| 0x2C | ADRFIELD | RWTS sector address field array |
| 0x2D | SECFNDZ | RWTS sector address field sector found |
| 0x2E | TRKFNDZ | RWTS sector address field track found |
| 0x2F | VOLFNDZ | RWTS sector address field volume found |

Table I.4.2. Page-Zero Utilization in DOS 4.5 – Part 1

| Address | Parameter | Description |
|---------|-----------|--|
| 0x32 | INVFLG | text screen inverse/normal flag |
| 0x33 | PROMPT | text screen prompt character |
| 0x34 | PHASE | RWTS requested phase number |
| 0x35 | PAGECNT | boot/initialization DOS image page count |
| 0x35 | SAVXYREG | save X-reg or Y-reg 8-bit variable |
| 0x35 | SYNCNT | RWTS synchronization byte count |
| 0x36 | CSWL | output device handler address |
| 0x38 | KSWL | input device handler address |
| 0x3C | ROMTEMPZ | ROM firmware boot temporary 8-bit variable |
| 0x3C | MOTORTIM | RWTS motor on-time 16-bit count |
| 0x3C | A1 | general purpose temporary 16-bit variable |
| 0x3D | ROMSECTR | ROM firmware boot requested sector |
| 0x3E | BUFADR2Z | RWTS data field buffer address |
| 0x3E | ODDBITSZ | RWTS temporary 8-bit variable |
| 0x3E | A2 | general purpose temporary 16-bit variable |
| 0x3F | SECTORZ | RWTS address field sector |
| 0x40 | ROMDATA | ROM firmware boot address field track found |
| 0x40 | FILEBUFZ | file context block parameter buffer address |
| 0x40 | TRACKZ | RWTS address field track |
| 0x41 | ROMTRACK | ROM firmware boot requested track |
| 0x41 | VOLUMEZ | RWTS address field volume |
| 0x42 | A4 | general purpose temporary 16-bit variable |
| 0x42 | BUFADRZ | general purpose sector data buffer address |
| 0x44 | DIRINDX | VTOC and TSL data index |
| 0x4A | IOBADR | RWTS IOCB buffer address |
| 0x4C | DOSPTR | DOS general purpose pointer address |
| 0x50 | LINNUM | Applesoft line number 16-bit variable |
| 0x5A | DOSTEMP1 | DOS general purpose 8-bit variable |
| 0x5B | DOSTEMP2 | DOS general purpose 8-bit variable |
| 0x5C | DOSBUFR | DOS general purpose 16-bit variable/address |
| 0x67 | ASPGMST | Applesoft program start address |
| 0x69 | ASVARS | Applesoft simple variables pointer |
| 0x6B | ASARYS | Applesoft array pointer |
| 0x6D | ARYEND | Applesoft end of array pointer |
| 0x6F | ASSTRS | Applesoft end of character string storage pointer |
| 0x73 | ASHIMEM | Applesoft HIMEM address |
| 0x76 | ASRUN | Applesoft RUN flag |
| 0x9D | DSCTMP | Applesoft temporary character string descriptor data |
| 0xAF | ASPEND | Applesoft end of program address |
| 0xD6 | PROTECT | Applesoft program write-protect 8-bit flag |
| 0xD8 | ASONERR | Applesoft ONERR 8-bit error flag |
| 0xD9 | RKEYWORD | DOS R keyword 8-bit variable |

Table I.4.3. Page-Zero Utilization in DOS 4.5 – Part 2

Index

| | | | |
|---|----------|---------------------------------------|-----|
| APPEND Command | 156 | Binary File Installation Program..... | 269 |
| Apple][..... | 1 | BLDNMBR Variable Byte | 184 |
| Building a New ROM..... | 106 | BLDVRSN Variable Byte..... | 184 |
| Disabled ROM Routines..... | 106 | BLOAD Command | 152 |
| Memory Utilization | 5 | BOOTADR Variable Byte..... | 191 |
| Memory Utilization, DOS 4.5H..... | 6 | Bootling Process | 22 |
| Memory Utilization, DOS 4.5L..... | 6 | Configuration Data Structure..... | 23 |
| Memory Utilization, Page-Zero..... | 8 | Data Management Structure Block..... | 22 |
| ROM HLIN Drawing..... | 87 | DOS 4.5H Image Mapping..... | 26 |
| ROM Modifications..... | 87 | DOS 4.5H TS Mapping | 24 |
| Signature Bytes, Original..... | 113 | DOS 4.5L Image Mapping..... | 26 |
| Signature Bytes, Revised..... | 114 | DOS 4.5L TS Mapping..... | 24 |
| Transformations for a New ROM..... | 108 | RWPAGES Routine..... | 23 |
| Apple][+ Keyboard Modification | 335 | BOOTPGS Variable Byte..... | 191 |
| 74LS153 Truth Table..... | 335 | BRUN Command..... | 152 |
| Generation of Unavailable Characters..... | 336 | BSAVE Command..... | 153 |
| Modification Circuit | 334 | CALLFM Entry Address | 173 |
| Apple][+ Memory Upgrade | 330 | CALLRWTS Entry Address..... | 173 |
| Satellite Board Circuit Diagram | 331 | CAT Command..... | 134 |
| Satellite Board Connections | 331 | CATALOG Command..... | 134 |
| Satellite Board Operation | 332, 333 | Catalog Structure | 17 |
| Applesoft Formatter Program..... | 267 | Entry Definition | 19 |
| Assembly Routines | 171 | Entry Offsets | 19 |
| Direct Subroutine Call..... | 171 | File Type Definition..... | 19 |
| Indirect Subroutine Call..... | 171 | Sector Definition..... | 18 |
| Asynchronous Data Transfer Program..... | 288 | CD Command | 136 |
| AUTOBRK Entry Address | 177 | CFFA Data Storage..... | 221 |
| AUTORSET Vector Address..... | 178 | 1 GB Block Utilization | 225 |
| Axlon RAM Disk 320 Data Storage..... | 250 | DOS 3.3 Patches | 227 |
| Firmware Structure..... | 256 | DVTS Variable Range..... | 223 |
| Modified RAM Card Circuit Diagram..... | 254 | Firmware Structure | 222 |
| Original RAM Card Circuit Diagram..... | 253 | Interface Control Registers | 223 |
| Original RAM Card Soft Switches..... | 252 | CHAIN Command | 149 |
| RAM Card Circuit Modifications..... | 255 | CHAIN Processing..... | 77 |
| BCFGNDX Variable Byte | 189 | Array Descriptor Definition..... | 80 |
| DOS 4.5 Example Program | 32 | Element Descriptor Definition..... | 80 |
| Big Mac Program..... | 211 | Example Applesoft Program..... | 78 |
| Assembly Directives..... | 215 | Simple Descriptor Definition..... | 80 |
| Branch Instructions..... | 218 | CHAR Editor Program..... | 312 |
| Format Byte Definition..... | 219 | ClientServer Program..... | 295 |
| Macro Directives | 216 | A2V2/ADT System Components | 295 |
| Mnemonic Hashing Algorithm..... | 214 | Commands and Responses..... | 299 |
| Multiple Addressing Mode Instructions | 219 | Computer System Components | 296 |
| Single Byte Instructions..... | 217 | Configuration Data Control Block..... | 300 |
| SWEET16 Register Instructions..... | 218 | Error Codes, Client | 309 |
| Text Directives | 215 | | |

| | | | |
|--|----------|---|------------------------------|
| Error Codes, Server | 309 | Stepper Motor Illustration..... | 44 |
| Error Source Routines, Client..... | 310 | Write Auto-Sync Bytes..... | 50 |
| Error Source Routines, Server | 311 | Write Data Bytes..... | 51 |
| GETPAGE Calculation..... | 306 | Disk Window Program | 194 |
| Read/Write Data Control Block..... | 307 | DOS 4.5..... | 1 |
| Track/Sector Data Control Block | 303 | Commands | <i>See</i> Commands, DOS 4.5 |
| Transfer Data Control Block | 303 | Enhancements to DOS | 3 |
| Clock Access..... | 71 | Error Processing..... | 76 |
| Applesoft Clock Data Generation..... | 73 | Improvements to DOS | 3 |
| Assembly Clock Data Generation | 73 | Introduction..... | 1 |
| Supported Clock Indexes..... | 72 | Operational Environment..... | 193 |
| CLOSE Command | 157, 163 | Overview..... | 2 |
| Commands, DOS 4.5 | 117 | ProDOS Algorithm | 82 |
| Applesoft File Commands | 149 | Software Strategies | 4 |
| Binary File Commands..... | 152 | Using DOS Commands..... | 85 |
| Command List | 120 | DOSCOLD Entry Address..... | 172 |
| Command Table | 118 | DOSWARM Entry Address..... | 172 |
| Data File Design Considerations | 167 | EDITROM Program..... | 111 |
| Data Sizing Equations | 168 | EPROM Operating System | 196 |
| File System Commands..... | 134 | Catalog File Entry..... | 205 |
| Keyword Definitions | 120 | File Types | 201 |
| Keyword Minimum/Maximum Ranges..... | 120 | QLBINEOS Example Code..... | 207 |
| Random-Access Data File Commands | 162 | quikLoader Bank Switching | 199 |
| Random-Access Design Considerations..... | 167 | quikLoader EPROM 0 | 201 |
| Sequential Text File Commands..... | 155 | quikLoader Firmware Structure..... | 200 |
| System Commands | 122 | quikLoader Schematic | 198 |
| Valid Keyword Table | 117 | EXEC Command | 157 |
| CONFIG Command | 122 | File Developer Program..... | 233 |
| Bit Definitions | 123 | File Manager | 57 |
| Data Structures..... | 66 | CD Context Block..... | 65 |
| CMDVALS Definition | 67 | Command Codes..... | 58 |
| File Buffer Definition | 70 | Commands and Parameter List..... | 56 |
| FMWORK Definition..... | 69 | Error Messages | 62 |
| INITVALS Definition | 30 | INIT Boot Disk Types | 59 |
| VTOCVALS Definition | 60 | INIT Context Block | 60 |
| Date and Time..... | 14 | Input/Output Context Block..... | 57 |
| DATE Command | 124 | Input/Output Subcodes | 58 |
| DELETE Command..... | 136 | TOUCH Context Block | 63 |
| DIFF Command | 137 | TS Context Block..... | 64 |
| Disk][..... | 39 | URM Context Block | 63 |
| 3 Half-Phase Tracks..... | 43 | WTS Context Block..... | 65 |
| 4 Half-Phase Tracks..... | 43 | First Class Peripherals Sider Data Storage .. | 260 |
| Cam Table and Carriage | 42 | Firmware Structure | 263 |
| Function Write Indexing..... | 54 | Modified Logical Block Structure | 262 |
| Functional Read Indexing..... | 53 | Original Logical Block Structure..... | 262 |
| I/O Memory Switches..... | 48 | Garbage Collector | 101 |
| P5 ROM..... | 47 | Applesoft ROM Entry..... | 101 |
| P6 ROM..... | 47 | Array Element Processing | 103 |
| Read Disk Byte..... | 49 | Simple Descriptor Processing..... | 102 |
| Read Write Protect..... | 49 | Verification Timing Results..... | 105 |
| Sequencer Command Codes | 53 | GETFMCB Routine Address..... | 174 |
| Sequencer Control Switches | 48 | GETIOCB Routine Address | 175 |
| Set Read Mode..... | 49 | | |

| | | | |
|--|-----|---|----------|
| GETPAGE Routine in ClientServer | 307 | Building WRNIBL | 34 |
| Global Program Line Editor Program | 249 | Page 0x03 Routines | 36 |
| GOTOMON Description Notes | 182 | Scratchpad Definition | 33 |
| GOTOMON1 Entry Address | 183 | MNGDISK Vector Address | 185 |
| GOTOMON2 Entry Address | 183 | Attaching Example Program | 27 |
| GREP Command | 138 | Detaching Example Program | 27 |
| HELP Command | 125 | MNGUSER Vector Address | 187 |
| HOOKDOS Entry Address | 176 | DOS 4.5 Example Program | 28 |
| ICON Maker Program | 316 | LOADMAC Example Program | 29 |
| DELTA Calculation | 322 | MNGVALS Vector Address | 186 |
| ICON Complex Colors | 320 | 8-Bit Example Program | 68 |
| ICON Drawing Command Examples | 320 | LOADLEN Example Program | 71 |
| ICON Drawing Commands | 318 | MON Command | 129 |
| ICON Simple Colors | 320 | MORE Command | 141 |
| IN# Command | 127 | MV Command | 143 |
| INIT Command | 139 | NBUF1PG Address Byte | 190 |
| Available Data Sectors | 140 | NMASKIRQ Entry Address | 180 |
| INITDOS Vector Address | 188 | NOMON Command | 130 |
| INITVAL Vector Address | 189 | Null Modem Adapter | 296 |
| DOS 4.5 Example Program | 31 | OPEN Command | 159, 164 |
| JFD Parallel Printer Buffer | 282 | Page-Zero | 7 |
| 8035-Microprocessor Memory Map | 284 | Available Locations | 9 |
| Port Utilization | 285 | DOS 4.5 Utilization | 9 |
| Primary R3 R/W Block | 286 | General Utilization | 8 |
| Secondary R3 System Flag | 287 | PHASE Command | 130 |
| SELRB0 Utilization | 286 | POSITION Command | 159 |
| SELRB1 Utilization | 286 | PR# Command | 132 |
| User Flag F1 | 285 | PRERRADR Vector Address | 176 |
| Keyspan Adapter | 295 | Big Mac Example Program | 38 |
| Lazer's Interactive Symbolic Assembler | 235 | Program Global Editor Program | 246 |
| Command-Line Commands | 245 | PWRSTATE Variable Byte | 178 |
| READSCRN Routine | 237 | RanaSystems EliteThree Data Storage | 257 |
| SAVESCRN Routine | 238 | Firmware Structure | 259 |
| SETSCRN Routine | 237 | RDCLKVSN Vector Address | 175 |
| USR Command | 238 | Get Date and Time Program | 37 |
| LENGTH Program | 312 | Get DOS Version Program | 37 |
| LIST Command | 141 | READ Command | 160, 165 |
| LLOAD Command | 154 | Real Time Clock Card | 276 |
| LOAD Command | 150 | Circuit Diagram | 277 |
| LOCK Command | 142 | Clock Registers | 279 |
| LS Command | 134 | Configuration Register | 278 |
| LSAVE Command | 154 | Firmware Structure | 281 |
| MASKIRQ Vector Address | 181 | Interrupt Rate Selection | 279 |
| Interrupt Handler Status Byte | 181 | Peripheral Interface I/O Addresses | 278 |
| MAXFILES Command | 127 | RTC58321 Clock Pinout | 276 |
| DOS 4.5H File Buffers | 128 | RENAME Command | 143 |
| DOS 4.5L File Buffers | 128 | RM Command | 136 |
| Memory Initialization | 32 | RUN Command | 150 |
| Boot Sequence Steps | 35 | RWTS | 39 |
| Building RDNIBL | 34 | DOS 4.5 Routines Timing | 83 |
| Building RDNIBLBT | 33 | Error Codes | 41 |

| | | | |
|------------------------------------|-----------------------------------|------------------------------------|---------------------------|
| Error Messages | 62 | TS Command | 144 |
| I/O Command Codes | 41 | TSAVE Command | 161 |
| I/O Context Block Definition | 39 | TSL Structure | 21 |
| ProDOS Routines Timing | 83 | Block Definition | 21 |
| SAVE Command | 151 | TW Command | 161 |
| SCRG PROMGRAMER Card | 274 | UNLOCK Command | 145 |
| SCRG quikLoader | <i>See</i> EPROM Operating System | URM Command | 146 |
| Soft Switches | 90 | USA19H142P1.1 Keyspan Driver | 296 |
| CFFA Control | 95 | USER Command | 133 |
| Memory and Video | 91 | USRAHAND Entry Address | 179 |
| Original Control | 94 | USRYHAND Entry Address | 180 |
| Original Input/Output | 92, 93 | VERIFY Command | 147 |
| Original Management | 93 | Virtual][..... | 4, 73, 107, 131, 243, 288 |
| quikLoader Control | 96 | Character Bitmap File | 110 |
| RAM Card Control | 96 | Character Generator ROM | 109 |
| RAM Disk Control | 96 | Character Set XML File | 109 |
| Rana Control | 96 | Icon Bitmap File | 110 |
| Sider Control | 96 | Volume Manager Program | 228 |
| Status Switch Flags | 92 | VTOC | 11 |
| Zip Chip | 95 | 16 Sector Bitmap | 14 |
| Sourceror Program | 266 | 32 Sector Bitmap | 14 |
| SV Command | 132 | Bitmap Definition | 15 |
| SWEET16 Metaprocessor | 97 | DOS 3.3 Bitmap | 16 |
| Control Registers | 98 | DOS 3.3 Structure Block | 12 |
| Non-Register Opcodes | 98 | DOS 4.5 Structure Block | 13 |
| Register Opcodes | 99 | VTOC Manager Program | 210 |
| TLOAD Command | 160 | WRITE Command | 162, 166 |
| TOUCH Command | 143 | WTS Command | 147 |
| TrackScan Program | 289 | XFERADR Transfer Address | 177 |

```

DOS 4.5.05L
(C) 2022
Walland Philip Urbancic, Jr

B=4505L boot T=DOS 4.5.05L Demo Disk
M=4505L P=04 L=0x2022 01/01/22 08:28:48
S=6 D=01 U=000 F=0519 01/01/22 08:28:48
001 0x12,0x0F HELLO

```

```

DOS 4.5.05H
(C) 2022
Walland Philip Urbancic, Jr

B=4505H boot T=DOS 4.5.05H Demo Disk
M=4505H P=04 L=0x2022 01/01/22 08:28:48
S=6 D=01 U=000 F=0510 01/01/22 08:28:48
001 0x12,0x0F HELLO

```

This publication describes the process and the products I created when I decided to design and program an enhanced Disk Operating System (DOS) for my Apple][+ and for my Apple //e to manage files and volumes. Wherever I am able, I have included source code samples, schematic diagrams, equations, figures, tables, and representative screen shots to help explain what I have created and the reasons why I did so. As in my previous designs of an Apple][DOS, i.e. DOS 4.1L and DOS 4.1H that I released in 2018 (and again in 2020) and DOS 4.3H that I released in 2020, this has been an incredible journey for me. With DOS 4.5, I have again re-imagined that time when I mostly lived, breathed, and worked on Apple][hardware and software development continuously for a good period of my life many, many years ago. The list that follows are some of the features that I have engineered into the DOS 4.5 File and Volume Disk Management System.

DOS 4.5L boots directly into Main memory and sets HIMEM to 0x9600 whereas DOS 4.5H boots directly into Language Card memory and sets HIMEM to 0xBE00. Unlike the file buffers in DOS 4.5L, all five file buffers in DOS 4.5H are always available and they are fully contained in Language Card memory.

DOS 4.5 supports the full integration of the *Lisa* and the *Big Mac* assemblers into Auxiliary Memory. Both assemblers now support the full 65C02 instruction set. *Sourceror* and *Big Mac* are fully integrated and they both support all *SWEET16* opcodes. My C0:FF ROM image is totally compatible with DOS 4.5 and this image supports the 65C02 processor, *SWEET16*, and GARBAGE which is based on Bongers' algorithm. A fix for HLIN now draws perfect HIRES line graphics for my celebrated *ICON Maker* tool that can be paired with my HIRES *CHAR Editor* and *LENGTH* tools for stunning graphic presentations as those used in *BFI*.

DOS 4.5 supports all Clock Cards which plays a vital role in Volume initialization and the management of File timestamps. Volumes may be dated, titled, numbered, locked, and initialized to boot or to store data exclusively. Even track zero is used for data. Files are loaded and saved using an acceleration algorithm.

DOS 4.5 supports Applesoft CHAIN, File DIFF, File GREP, File LIST, File Undelete, Sector Display, Sector Modify, a programmable and argument-passing USER command, Variable Management, Syntactical HELP, DOS Processing Controls, Half-Phase Tracking, and full lowercase command and argument input.

DOS 4.5 can read DOS 3.3 Volumes and files.

DOS 4.5 can initialize Volumes with up to 48 tracks having either 16 or 32 sectors per track.

The utilities, tools, schematics, and Excel and Edraw files can be requested through www.applecored.net.

