

The DOS 4.1 Manual

Disk Operating System for the Apple II

Written & Programmed by:
Walland Philip Vrbancic, Jr.

Produced by:
Brian Wiser & Bill Martens



Apple PugetSound Program Library Exchange

The DOS 4.1 Manual: Disk Operating System for the Apple II

Copyright © 2019 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

Paperback ISBN: 978-0-359-39684-9

ACKNOWLEDGEMENTS

DOS 4.1 was Programmed and Designed by Walland Philip Vrbancic, Jr.

DOS 4.1 build 46 documentation inside this book is the Confidential and Proprietary Intellectual Property of Walland Philip Vrbancic, Jr. Copyright © 2019 Walland Philip Vrbancic, Jr. All Rights Reserved.

This book, produced in coordination with Walland Philip Vrbancic, Jr. and released with his permission, is copyright by A.P.P.L.E. as the publisher. No claim to copyright over DOS 4.1 is created outside of those portions created by A.P.P.L.E..

The Cover was designed by Brian Wiser.

PRODUCTION

Brian Wiser → Cover Design, Production

Bill Martens → Production

Walland Philip Vrbancic, Jr. → Documentation, Programming

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 that may be entered, stored and executed in a computer system, but not reproduced for publication. Thank you for respecting the intellectual property of the authors and publisher.

DOS 4.1 is available on disk images from the publisher: www.callapple.org. No warranty of disk images is made or implied and should be used at your own risk.

The DOS 4.1 Manual 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.

About Walland Philip Vrbancic, Jr.

I decided to join the team at Rockwell, in the Space Shuttle Simulation Laboratory about five months before the launch of STS-1, changing my hospital scrubs for a coat and tie, and a whole lot more money. About three months after I was hired at Rockwell, another engineer was hired, a Computer Science Engineer, to join the ranks of Initialization Engineers. He and I had the daunting task of initializing the computers and electronics that comprised the simulation of a Space Shuttle from Main Engine Cutoff (MECO) to landing at a few selected American landing sites. Our computers included a PDP-11 and two Xerox mainframes that were programmed using front-panel rocker switches: the Sigma 5 had 16 KB and the Sigma 9 had 64 KB of magnetic core memory. We used Hollerith cards to insert faults into the shuttle's General Purpose Computers (GPCs) and we used a color Eidophor projector to project visual images of our landing site runways into a shuttle cockpit simulator in which the astronauts trained. We used Nova computers by DEC and DEC word processing software to generate all required documentation, and to play *Adventure* I might add.

My colleague was an early Apple II owner when Integer BASIC was initially available in ROM. The following year Rockwell offered a home computer purchase program providing the choice between an IBM PC and the Apple II Plus. My colleague strongly encouraged me to request the Apple computer, and he assisted me in selecting the monitor, disk drive, and printer accessories. The total cost was a lot of money for me but Rockwell loaned me the money and paid the bill, and I repaid the interest-free loan through weekly payroll deductions. Thus began my dream of having my own personal computer.

I became fascinated with all aspects of the Apple II Plus computer, and I wanted to incorporate it into my Master's degree studies. My assigned advisor was analyzing tomographic reconstructions of the human spinal column and he thought perhaps I could assist him. He wanted to be able to make measurements between any two locations within the computer images even after rotating or enlarging the images. I was tasked with developing the Fortran programs that could be launched on a Microsoft Z80 peripheral slot card in an Apple II Plus that would provide him with these capabilities. I found an ingenious way to reduce the size of the three-dimensional rotational matrix in order to accelerate data processing and the mapping of those results to the screen. My professor was very pleased with my progress.

However, I was becoming increasingly interested in high-speed graphics animation and the only way I would learn that technology was to work for Ken Williams at Sierra On-Line. I terminated my work on my Master's degree, gave notice to Rockwell, packed my bags, and moved to Oakhurst, California. At Sierra I assisted a colleague in migrating *ScreenWriter* to the newly marketed Apple IIe, I wrote all the I/O routines and ICON drawing routines for *HomeWord Speller*, and I nearly finished *Goofy's Word Factory*, a children's game to teach English grammar. Williams had a license to display certain Disney characters on a computer screen per approval by Disney for visual likeness and color. I would have finished the product if the designer of the game (Williams' brother) could have developed the third game feature in a timely fashion. He apparently could not do so before I secured a position at Hughes Aircraft Company back in Los Angeles. I stayed all of 18 months at Sierra and I did utilize Williams' high-speed graphics animation algorithms in *Goofy*, which I had to redesign in order to include collision detection on a dithered background. No other computer game could detect collisions on a dithered background at that time. Williams was impressed, and it was really hard to impress Williams.

The major observation I made at Hughes Aircraft was how different the culture was to the culture I experienced at Rockwell. At Rockwell I found it exceedingly difficult to have anyone who had written a software tool explain to me how that tool worked and the algorithms the tool utilized, or exploited.

When I was tasked to migrate a software tool from Fortran to C language at Rockwell, I found some incorrect logic that affected the final data output. Given certain input parameters this tool could calculate a three-dimensional corridor in space and either interpolate points within or extrapolate points outside that corridor. I presented my findings to its author showing how I could insert the same incorrect logic into the C code and generate the same wrong data output. He told me to keep the incorrect logic and not disclose my findings. I refused.

This was totally unthinkable, and this would have never happened at Hughes. In fact, CIP awards were given to engineers for finding such errors in software and bringing those errors to the attention of management. The Hughes culture encouraged the aggressive sharing of knowledge and giving reward for making improvements, not the self-preservation culture of Rockwell where knowledge was thought to be job security and not to be shared, but kept secret. Hughes certainly provided me a great opportunity within the Digital Simulation Laboratory where I learned about real time executive software executing on Gould SEL mainframe computers (9720, 9760, and 9780), MIL-STD-1553 communication protocol, and real time software interface drivers to a host of various external data processors. Our purpose was to create a digital time frame in order to simulate in real time the environment for a tactical Radar Digital Processor flying above the surface of the earth.

In 1990, I returned to Rockwell believing that my knowledge in real time executive software executing on SEL mainframe computers would be my passport to a nice software engineering career closer to where I lived. How I regret this major blunder in judgment because my employment at Rockwell was terminated a few years later. I had co-authored a White Paper outlining the risks associated with using off-the-shelf RISC processors in certain applications, and the response from my colleagues was very unfavorable. Fortunately, my former Hughes management was able to reinstate my position and I was tasked to gain expertise in real time data collection software for tactical radar systems.

Hughes tactical radar systems are programmed to operate in many different modes depending upon various situations and needs faced by the pilot of a military aircraft. During the development of a radar mode, its processing is heavily instrumented which generates a large amount of output data as the mode progresses through its various processing stages. It is critical to capture all that generated data in order for the mode developer to ensure and verify the mode is behaving as expected and is generating data according to pre-established boundaries, much like comparing the data to some gold standard. My task was to capture all the in-phase and Quadrature (I/Q) components of radar data in real time, package the data according to source and timestamp, and save the resulting files to some recording device.

It is important to understand that there are many independent sources of data in a radar system whose timestamps are totally asynchronous. At a later time the data in those files would be analyzed to determine if, in fact, the processing modes operated as expected. Physically collecting this I/Q data during real tactical maneuvers was quite a challenge, and recorders designed to operate in this environment were costly. Preparing for a data collection session involved securing a military aircraft, a flight crew, a ground crew, and people to securely bring the recorded data back to my secure lab. This certainly added to my responsibilities, and my mantra was to neither add, subtract, nor modify any data word or data bit while that data was in my immediate possession and while my software algorithms processed that data.

I was thoroughly vetted and held maximum-security clearances that allowed me to process data from many different and independent programs not only here in Los Angeles, but also in other locations, even out of state. The general data collection software engines I began designing in the unclassified world served as my software library for every classified program I worked. Perhaps I was simply in the right place at

the right time that steered my career to become the sole resident expert in Transcription Software, that is, to process, encrypt, and store in real time at least a terabyte of data every second. Or, perhaps I was in the right place at the right time that allowed me to develop a task beyond its envisioned potential. There is a direct ancestral linkage between my unclassified software library of tools, routines, and transcription engines and every single classified program with which I was associated that required my tools, engines, and expertise.

I was practicing “code reuse” light-years before it became a topic that some managers thought could reduce software development costs. Really? Initially I was given the opportunity to host my Transcription Software on a newly acquired SGI Origin 300 having four bricks, or 16 CPUs. “Code reuse” made this task fairly straightforward, thus demonstrating the machine’s practicality in their feasibility study. After a fact-finding tour at the SGI facilities at Mountain View, California, I was given the momentous task of designing a Transcription Software engine for an SGI Origin 3000 having eight bricks (32 CPUs) running IRIX, and using Big Endian memory management. This turned out to be one of my greatest achievements. Little did my management know how effortlessly I could build my transcription engines by this time primarily because of “code reuse.”

I was extremely fortunate to have had one very intelligent manager who casually asked me to think about the possibility of building a digital playback system. Such a system did not exist. Some had tried building an analog playback system years earlier with very little success. Instead of “analyzing” the collected instrumented data, one could observe how the simulated Radar Digital Processor behaved when the high-speed (I/Q) data and the slow-speed (environment) data is injected back into its system. A few months later I presented my first digital playback recorder and pre-processing system, my last and greatest achievement for Raytheon. I was given the unique privilege to design and build a second digital playback recorder and pre-processing system for another classified program. As for the program that used my first digital playback recorder and pre-processing system, that next program saved countless hours of analysis time and mission costs before I scheduled my overdue retirement.

A few years after I retired I was presented with an astonishing diagnosis that seemed to explain the idiosyncrasies I have displayed my entire life as far back as elementary school: I may have been living with Asperger’s. Indeed, how does one know what is normal – that which falls under the umbrella of a Gaussian curve? We are all volitional beings and our behavior is internal to each of us. Our brain is composed of carbon-based synapses whose billions of connections compose the very person and personality we have become or have allowed ourselves to become. It is simply miraculous that any of our species reach total fulfillment of their dreams. I like to think I have come closer than most in reaching many of my aspirations.

I have the time and the continuing curiosity to delve into Apple II DOS now, and I have the 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 my version of Apple II DOS, DOS 4.1. And this is my 46th build of DOS 4.1 with more to come at www.applecored.net. What a ride I have been on! Why? To see what I could do for this wonderful machine and its magnificent architecture!

Table of Contents

I. Designing a New DOS	1
1. Introduction.....	1
2. General Software Design Strategy	3
3. DOS Wish List	4
4. DOS 4.1 Software Development.....	5
5. Page-Zero Utilization	8
6. VTOC Structure	14
7. DOS 4.1 Catalog.....	18
8. Booting DOS 4.1	21
9. DOS 4.1 Initialization.....	27
10. DOS 4.1 Data Structures.....	41
11. DOS 4.1 Clock Access.....	47
12. DOS 4.1 Error Processing.....	49
13. DOS 4.1 Chain Command	50
14. The VTOC Bitmap Definition.....	55
15. ProDOS Disk I/O Algorithm.....	57
16. Building and Installing DOS 4.1 Images.....	60
17. Using DOS 4.1 Commands.....	61
II. Apple ROM Modifications	63
1. Apple ROM Modification for Correct HLINE Drawing Algorithm	63
2. Apple ROM Modification for Delete Key Utilization.....	66
3. Apple //e 80 Column Text Card and ROM Monitor	67
4. Sweet 16 Metaprocessor	84
5. Applesoft Garbage Collector	90
6. Apple Character Generator ROM.....	95
7. Peripheral Slot Card Signature Bytes	98
III. DOS 4.1 Commands.....	101
1. File System Commands	106
2. System Commands	128
3. Applesoft File Commands	131
4. Binary File Commands	135
5. Sequential Text File Commands	139
6. Random-Access Data File Commands	148

IV. DOS 4.1 Operational Environment.....	155
1. Applesoft Formatter.....	156
2. Binary File Installation (BFI).....	158
3. Apple][+ Memory Upgrade	161
4. Real Time Clock Card	165
5. Disk Window	171
6. EPROM Operating System (EOS) for quikLoader	176
7. VTOC Manager (VMGR).....	191
8. Asynchronous Data Transfer (ADT).....	194
9. Big Mac.....	197
10. PROmGRAMER	199
11. CFFA Card.....	201
12. Volume Manager (VOLMGR).....	206
13. File Developer (FID)	211
14. Lazer's Interactive Symbolic Assembler (Lisa).....	213
15. Program Global Editor (PGE).....	221
16. Global Program Line Editor (GPL).....	222
17. RamDisk 320.....	223
18. RanaSystems EliteThree	230
19. The Sider.....	233
20. Sourceror.....	237
21. Parallel Printer Buffer.....	240
22. Last Concluding Thoughts.....	244

List of Figures

Figure I.5.1. Page-Zero Memory Usage.....	13
Figure I.6.1. DOS 4.1L Data Disk Volume VTOC.....	14
Figure I.7.1. DOS 4.1 First Volume Catalog Sector	17
Figure I.7.2. DOS 4.1L TSL Sector	20
Figure I.8.1. Attaching a Slot Card Handler to DOS 4.1.....	23
Figure I.9.1. Using USERADR and CMDVAL in DOS 4.1	28
Figure I.9.2. Reading the DOS Version in DOS 4.1	30
Figure I.9.3. Reading the Date and Time in DOS 4.1	30
Figure I.9.4. Big Mac Printing a File Manager Error in DOS 4.1	31
Figure I.9.5. Using the File Manager Context Block in DOS 4.1.....	37
Figure I.9.6. File Manager Command Parameter List.....	39
Figure I.10.1. Lisa Using LOADLEN from KEYVALS in DOS 4.1	44
Figure I.13.1. Example Applesoft Program Layout in Memory.....	51
Figure II.1.1. Applesoft HLIN Demonstration Program	64
Figure II.1.2. Original ROM HLIN Routine Display	65
Figure II.1.3. Modified ROM HLIN Routine Display	65
Figure II.3.1. Notes for Tables II.3.1 to II.3.5	71
Figure II.6.1. International XML File	96
Figure II.6.2. New Character Set TIFF Bitmap File	96
Figure II.6.3. Icon TIFF Bitmap File	97
Figure II.6.4. Binary Character Set LORES Editor	97
Figure III.1.1. CATALOG and CAT Command Display.....	107
Figure III.1.2. LS R Command Display	108
Figure III.1.3. CD Command Display.....	109
Figure III.1.4. DATE Command for Thunderclock Card Display.....	110
Figure III.1.5. DELETE Command Display.....	111
Figure III.1.6. DIFF Command Display.....	112
Figure III.1.7. GREP Command Display	113
Figure III.1.8. HELP HELP Command Display 1	114
Figure III.1.9. HELP HELP Command Display 2	115
Figure III.1.10. HELP HELP Command Display 3	115
Figure III.1.11. HELP HELP Command Display 4	116
Figure III.1.12. HELP INIT Command Display.....	116
Figure III.1.13. INIT Command Display 1.....	118
Figure III.1.14. INIT Command Display 2.....	118
Figure III.1.15. LIST Command Display	121
Figure III.1.16. LOCK Command Display.....	122
Figure III.1.17. RENAME Command Display.....	122
Figure III.1.18. SV Command Display	124
Figure III.1.19. TS Command of a Data Disk VTOC Display.....	124
Figure III.1.20. UNLOCK Command Display	126
Figure III.1.21. URM Command Display.....	126
Figure III.1.22. VERIFY Command Display	127
Figure III.2.1. MAXFILES, MON, and NOMON Command Display	129

Figure III.3.1. Listing of START and PROGRAM2 Programs Display	132
Figure III.3.2. Output of Programs START and PROGRAM2 Display	132
Figure III.3.3. LOAD and SAVE Commands Display	133
Figure III.4.1. BLOAD and BSAVE Commands Display	136
Figure III.4.2. BRUN Command Display	136
Figure III.4.3. LLOAD and LSAVE Commands Display	138
Figure III.5.1. OPEN, WRITE, and CLOSE Commands Display	140
Figure III.5.2. APPEND Command Display	141
Figure III.5.3. EXEC Command Display	142
Figure III.5.4. EXEC,Rr Command Display	142
Figure III.5.5. POSITION and READ Commands Display	144
Figure III.5.6. READ,Bb Command Display	144
Figure III.5.7. TLOAD and TSAVE Command Display	146
Figure III.5.8. TW Display	146
Figure III.6.1. OPEN, WRITE, and CLOSE Commands Display	149
Figure III.6.2. Contents of RTEST.T Display	150
Figure III.6.3. READ and RUN Command Display	151
Figure III.6.4. Example Random-Access Data File CREATE	152
Figure IV.1.1. Applesoft Program Listing	156
Figure IV.1.2. Applesoft Program Programmatically Formatted	157
Figure IV.2.1. BFI Main Menu	159
Figure IV.2.2. BFI Peripheral Selection	159
Figure IV.2.3. BFI Installation Report on BFI	160
Figure IV.3.1. Apple][+ Satellite Circuit Diagram	161
Figure IV.4.1. Real Time Clock Circuit Diagram	166
Figure IV.5.1. Disk Window Startup Screen	172
Figure IV.5.2. Select T/S Mode	172
Figure IV.5.3. Edit Data Screen	173
Figure IV.5.4. Write Sector Data Screen	173
Figure IV.5.5. Print Sector Data Screen	174
Figure IV.5.6. Disk Window Error Message Display	174
Figure IV.6.1. quikLoader Circuit Diagram with Modifications	177
Figure IV.6.2. EOS Commands at RESET	181
Figure IV.6.3. EOS Catalog for EPROM 0, Part 1	181
Figure IV.6.4. EOS Catalog for EPROM 0, Part 2	182
Figure IV.7.1. VMGR Option Menu	191
Figure IV.7.2. VTOC Contents	192
Figure IV.7.3. VTOC Sector Bitmap Contents	192
Figure IV.8.1. ADT Window	194
Figure IV.8.2. ADT Configuration	195
Figure IV.8.3. ADT Software Credits	195
Figure IV.9.1. Big Mac Main Menu	197
Figure IV.10.1. PROmGRAMER Configuration	200
Figure IV.10.2. PROmGRAMER Command Menu	200
Figure IV.12.1. VOLMGR Product Warning Screen	206
Figure IV.12.2. VOLMGR Command Menu	207
Figure IV.12.3. VOLMGR Manage Firmware Menu	207
Figure IV.12.4. VOLMGR Manage CompactFlash Menu	208
Figure IV.12.5. VOLMGR Device Identity Contents	208

Figure IV.12.6. VOLMGR Manage Drives Menu	209
Figure IV.12.7. VOLMGR Manage Volumes Menu	209
Figure IV.12.8. VOLMGR Manage User DOS Images Menu	210
Figure IV.13.1. FID Main Menu.....	211
Figure IV.14.1. Lisa Startup Screen.....	213
Figure IV.14.2. Lisa Setup Utility.....	214
Figure IV.14.3. DOS 4.1H Source Code Volume.....	216
Figure IV.14.4. EOS Image Segment Files	216
Figure IV.14.5. EOS1 Image Creation.....	217
Figure IV.14.6. EOS2 Image Creation.....	217
Figure IV.17.1. Original RamCard Hardware Circuit Diagram	225
Figure IV.17.2. Modified RamCard Hardware Circuit Diagram.....	226
Figure IV.17.3. RamCard Hardware Modifications.....	227
Figure IV.20.1. Sourceror Initialization	237
Figure IV.20.2. Sourceror Startup/Help Screen.....	238
Figure IV.20.3. Sourceror Monitor Source Listing.....	238

List of Tables

Table I.4.1. Apple][Memory Utilization	7
Table I.5.1. Page-Zero Memory Locations 0x00-0x3F	9
Table I.5.2. Page-Zero Memory Locations 0x40-0x7F	10
Table I.5.3. Page-Zero Memory Locations 0x80-0xBF	11
Table I.5.4. Page-Zero Memory Locations 0xC0-0xFF	12
Table I.6.1. DOS 4.1 VTOC Structure Block Definition	15
Table I.6.2. Free Sector Bitmap for Each Track	16
Table I.6.3. DOS 4.1 Date and Time Definition and Variable Order	16
Table I.7.1. DOS 4.1 Volume Catalog Entry	18
Table I.7.2. DOS 4.1 Catalog Sector Data Offsets for File Entries	19
Table I.7.3. DOS 4.1 File Type Byte Description	19
Table I.7.4. DOS 4.1 TSL Structure Block Definition	20
Table I.8.1. DOS 4.1 RWTS Slot Interface Structure Definition	21
Table I.8.2. DOS 4.1L Boot Configuration Table	22
Table I.8.3. DOS 4.1L Disk Track/Sector Mapping to Memory Address	23
Table I.8.4. DOS 4.1H Disk Track/Sector Mapping to Memory Address	24
Table I.8.5. DOS 4.1L File Image Mapping to Memory Address	24
Table I.8.6. DOS 4.1H File Image Mapping to Memory Address	25
Table I.8.7. DOS 4.1 Initial Address Table Definition	26
Table I.9.1. DOS 4.1 Page 0x03 Interface Routines	29
Table I.9.2. RWTS I/O Context Block Definition	32
Table I.9.3. RWTS Command Codes	32
Table I.9.4. RWTS Error Codes	33
Table I.9.5. File Manager Context Block Definition	33
Table I.9.6. File Manager Command Codes	34
Table I.9.7. File Manager Read and Write Command Subcodes	34
Table I.9.8. DOS 4.1 Error Messages and Sources	35
Table I.9.9. File Manager INIT DOS Flags (SUBCODE)	36
Table I.9.10. File Manager Initialization Data, VTOCVALS	36
Table I.10.1. File Manager File Buffer Definition	40
Table I.10.2. CMDVALS Data Structure Definition	42
Table I.10.3. KEYVALS Data Structure Definition	43
Table I.10.4. File Manager Workarea Structure Definition	45
Table I.11.1. Supported Clock Cards in DOS 4.1	47
Table I.13.1. Applesoft Simple Variable Descriptor Definition	52
Table I.13.2. Applesoft Array Variable Descriptor Definition	52
Table I.14.1. Free Sector Bitmap for 32 Sector Tracks in DOS 3.3	55
Table I.15.1. DOS 4.1 and ProDOS RWTS Routines, Tables, and Buffers	57
Table II.3.1. New Memory Management and Video Soft Switches	68
Table II.3.2. New Soft Switch Status Flags	68
Table II.3.3. Original Input/Output Control Soft Switches	69
Table II.3.4. Original Memory Management Soft Switches	70
Table II.3.5. Original Disk][Control Soft Switches	70
Table II.3.6. Zip Chip Control Soft Switches	71

Table II.3.7. CFFA Control Soft Switches	72
Table II.3.8. quikLoader Control Soft Switches	72
Table II.3.9. Sider, RamDisk, RamCard, and Rana Control Soft Switches	72
Table II.3.10. Disabled Applesoft Commands	83
Table II.4.1. Sweet 16 Register Descriptions	85
Table II.4.2. Sweet 16 Non-Register Opcodes	85
Table II.4.3. Sweet 16 Register Opcodes	86
Table II.5.1. Simple Variable Descriptor Processing in Pass 1	91
Table II.5.2. Array Variable Element Processing in Pass 1	91
Table II.5.3. Garbage Collector Timing Results	94
Table II.7.1. Peripheral Slot Card Signature Bytes	99
Table II.7.2. Revised Disk Drive Peripheral Slot Card Signature Bytes	100
Table III.0.1. DOS 4.1 Command Valid Keyword Table	101
Table III.0.2. DOS 4.1 Command Table	102
Table III.0.3. DOS 4.1 Keyword Name and Range Table	104
Table III.0.4. DOS 4.1 Keywords and Keyword Value Items	104
Table III.1.1. DOS 4.1 File System Commands	106
Table III.1.2. Initialized Catalog Size for 35 Tracks, 16 Sectors/Track	119
Table III.1.3. Initialized Catalog Size for 35 Tracks, 32 Sectors/Track	119
Table III.1.4. Total Sectors for Volumes	120
Table III.2.1. DOS 4.1 System Commands	128
Table III.2.2. MAXFILES Memory Locations	130
Table III.3.1. DOS 4.1 Applesoft File Commands	131
Table III.4.1. DOS 4.1 Binary File Commands	135
Table III.5.1. DOS 4.1 Sequential Text File Commands	139
Table III.6.1. DOS 4.1 Random-Access Data File Commands	148
Table IV.3.1. Apple][+ Satellite Circuit Board Connections	162
Table IV.3.2. Apple][+ Satellite Circuit Board Operation Part 1	163
Table IV.3.3. Apple][+ Satellite Circuit Board Operation Part 2	164
Table IV.4.1. Real Time Clock Peripheral-Card I/O Addresses	166
Table IV.4.2. Real Time Clock Configuration Register	167
Table IV.4.3. Interrupt Rate Selection	167
Table IV.4.4. Real Time Clock Registers	168
Table IV.4.5. Clock Firmware Entry Points	169
Table IV.6.1. quikLoader Bank Switching	178
Table IV.6.2. quikLoader Firmware Entry Points	179
Table IV.6.3. EPROM 0 Containing EOS and Programs	179
Table IV.6.4. EOS File Types Used in Optional Parameter Array	184
Table IV.6.5. EOS Catalog File Entry Structure	185
Table IV.6.6. BINEOS Catalog File Entry	189
Table IV.11.1. CFFA Card Firmware Entry Points	201
Table IV.11.2. DOS 3.3 Patches for CFFA	204
Table IV.14.1. Lisa USR Command	214
Table IV.17.1. RamCard Memory Configuration Soft Switches	224
Table IV.17.2. RamDisk 320 Firmware Entry Points	228
Table IV.18.1. Rana Disk Firmware Entry Points	231
Table IV.19.1. Sider Logical Structure	234
Table IV.19.2. Modified Sider Logical Structure	234
Table IV.19.3. Sider Firmware Entry Points	235