SUPPLEMENT TO

Beneath Apple ProDOS

For ProDOS 8, Versions 1.2 and 1.3

by Don D. Worth and Pieter M. Lechner

QUALITY SOFTWARE
21610 Lassen Street #7
Chatsworth, California 91311
Apple Books from Quality Software

Beneath Apple ProDOS
by Don Worth & Pieter Lechner

Supplement to Beneath Apple ProDOS for Versions 1.0.1, 1.0.2
by Don Worth & Pieter Lechner

Supplement to Beneath Apple ProDOS for Version 1.1.1
by Don Worth & Pieter Lechner

Beneath Apple DOS
by Don Worth & Pieter Lechner

Understanding the Apple II
by Jim Sather

Understanding the Apple IIe
by Jim Sather

Apple Utility Software from Quality Software

Bag of Tricks 2 (includes diskette)
by Don Worth & Pieter Lechner

Universal File Conversion (includes diskette)
by Gary Charpentier

See the last two pages of this book for information about how to order Quality Software products.

Illustrations by George Garcia

(c)1987 Quality Software. All rights reserved. No part of this book may be reproduced, in any way or by any means, without permission in writing from the Publisher. No liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

"Apple" is a registered trademark of Apple Computer, Inc. This manual was not prepared nor reviewed by Apple Computer, Inc., and use of the term "Apple" should not be construed to represent any endorsement, official or otherwise, by Apple Computer, Inc.
## CONTENTS

<table>
<thead>
<tr>
<th>PAGE</th>
<th>TOPIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Introduction</td>
</tr>
<tr>
<td>5</td>
<td>Understanding the Listings</td>
</tr>
</tbody>
</table>

### PRODOS 8, VERSIONS 1.2 AND 1.3

<table>
<thead>
<tr>
<th>PAGE</th>
<th>TOPIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>How ProDOS 8 is Loaded and Relocated (for both Version 1.2 and 1.3)</td>
</tr>
<tr>
<td>7</td>
<td>ProDOS 8 Loader (for both 1.2 and 1.3)</td>
</tr>
<tr>
<td>10</td>
<td>ProDOS 8 Relocator, Version 1.2</td>
</tr>
<tr>
<td></td>
<td>Relocation routines</td>
</tr>
<tr>
<td></td>
<td>RAMdrive Device Driver</td>
</tr>
<tr>
<td></td>
<td>SYSTEM File Loader</td>
</tr>
<tr>
<td>26</td>
<td>ProDOS 8 Relocator, Version 1.3</td>
</tr>
<tr>
<td>32</td>
<td>ProDOS 8 MLI (Kernel), Version 1.2</td>
</tr>
<tr>
<td>67</td>
<td>ProDOS 8 MLI, Version 1.3</td>
</tr>
<tr>
<td>75</td>
<td>ProDOS 8 System Global Page (for both 1.2, 1.3)</td>
</tr>
<tr>
<td>77</td>
<td>ProDOS 8 Quit Code (for both 1.2 and 1.3)</td>
</tr>
<tr>
<td>81</td>
<td>ProDOS 8 Disk II Device Driver, Version 1.2</td>
</tr>
<tr>
<td>88</td>
<td>ProDOS 8 Disk II Device Driver, Version 1.3</td>
</tr>
<tr>
<td>89</td>
<td>ProDOS 8 IRQ Handler (for both 1.2 and 1.3)</td>
</tr>
<tr>
<td>90</td>
<td>ProDOS 8 Thunderclock Code (for both 1.2, 1.3)</td>
</tr>
<tr>
<td>92</td>
<td>ProDOS 8 IIGS Clock Code (for both 1.2 and 1.3)</td>
</tr>
</tbody>
</table>

### BASIC.SYSTEM, VERSION 1.1

<table>
<thead>
<tr>
<th>PAGE</th>
<th>TOPIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>How BASIC.SYSTEM is Loaded and Relocated</td>
</tr>
<tr>
<td>94</td>
<td>BI Relocator</td>
</tr>
<tr>
<td>97</td>
<td>BASIC Interpreter (BI)</td>
</tr>
<tr>
<td>132</td>
<td>BI Global Page</td>
</tr>
</tbody>
</table>

### DISK II BOOT ROM

<table>
<thead>
<tr>
<th>PAGE</th>
<th>TOPIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>134</td>
<td>Disk II Controller ROM--Apple II/II+/IIe</td>
</tr>
<tr>
<td>136</td>
<td>Disk II Boot Logic--Apple IIE</td>
</tr>
<tr>
<td>139</td>
<td>Disk II Boot Logic--Apple IIGS</td>
</tr>
</tbody>
</table>

### APPENDIX A -- Differences Between ProDOS 8 Versions

### APPENDIX B -- Errata to Beneath Apple ProDOS
INTRODUCTION

This supplement documents the actual ProDOS 8 logic at nearly a byte by byte level. It is intended to aid experienced programmers in designing customized interfaces to ProDOS 8, and to provide implicit documentation of the ProDOS 8 functions. All assembly language programmers will find this supplement useful in learning about how an operating system works. This information is presented in the spirit of helping the user to understand ProDOS 8 better. The authors do not endorse indiscriminant modification of the ProDOS components. Whenever possible, standardized interfaces to ProDOS should be used to avoid the uncontrolled modifications which plagued Apple's previous operating system, DOS 3.3.

External system programs and utilities such as the Apple II System Utilities are not covered here, nor are disk controller ROM's covered other than the Disk II controllers available from Apple.

The information presented here is for the release of the ProDOS operating system called ProDOS 8, Versions 1.2 and 1.3. Previous supplements to Beneath Apple ProDOS documented the structure of Versions 1.0.1, 1.0.2, and 1.1.1 of ProDOS.

UNDERSTANDING THE LISTINGS

The listings which follow describe the major ProDOS 8 components in great detail. Each module is presented separately and consists of a section defining external addresses referenced by the program (such as zero page usage, I/O select addresses, and global page fields) followed by a section describing the instructions and data in the module. Divisions between major sections and subroutines are indicated with a row of asterisks (*) and additional comments.

Each detail line gives the address of the instruction or data field being described, followed by comments. Within the comments, the following notation is used to indicate references by instructions:

(address) A store or load reference to a memory or I/O location.
>>address A branch or jump to an address.
<address> A call to a subroutine at the indicated address.
-->address A pointer to an address.

Page titles give the address of the next instruction or data area in the module to be described. These may be used to quickly locate a particular area within the component.
HOW PRODOS 8 Versions 1.2 and 1.3 ARE LOADED AND RELOCATED

3 Copy to High RAM:
   IRQ Handler
   System Global Page
   MLI Kernel
   Disk II Device Driver

1 PQUIT, the ProDOS Loader, or a "-" command loads the "P8"
   file to memory address $2000
   and jumps to the Relocator.

2 Copy from within Relocotor to low memory:
   SYSTEM FILE LOADER
   PAGE 3 IMAGE
   80-COL CARD CHECKER

4 Final moves:
   FUNCTION FROM TO LENGTH
   Clock code 5100* D742 7D
   QUIT code 5900 D100** 300
   RAM drive...
   Caller 2E00 FF00 9A
   Driver 2C00 200***200

*5C00 if IIGS  **BANK2  ***AUX MEMORY
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>MODULE STARTING ADDRESS</td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
<tr>
<td></td>
<td>* PRODOS LOADER</td>
</tr>
<tr>
<td></td>
<td>* THIS CODE IS LOADED FROM BLOCK $0. *</td>
</tr>
<tr>
<td></td>
<td>* INTO MEMORY AT $800. *</td>
</tr>
<tr>
<td></td>
<td>* ITS PURPOSE IS TO LOAD THE &quot;PRODOS&quot;</td>
</tr>
<tr>
<td></td>
<td>* FILE INTO $2000 AND JUMP TO IT. *</td>
</tr>
<tr>
<td></td>
<td>* (PRODOS RELOCATOR IS AT $2000) *</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.2 -- 6 SEP 86 *</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.1 -- 2 DEC 86 *</td>
</tr>
<tr>
<td></td>
<td>* (THE LOADER IS STILL THE SAME AS IT *</td>
</tr>
<tr>
<td></td>
<td>* WAS IN VERSION 1.0.1) *</td>
</tr>
<tr>
<td></td>
<td>* ***********************</td>
</tr>
</tbody>
</table>

---

**EXTERNAL ADDRESSES**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0027</td>
<td>ROM BOOT SUBRTN BUFFER PAGE ADUR</td>
</tr>
<tr>
<td>0028</td>
<td>ROM BOOT SUBRTN SLOT * 16</td>
</tr>
<tr>
<td>003D</td>
<td>ROM BOOT SUBRTN SECTOR TO READ</td>
</tr>
<tr>
<td>0040</td>
<td>ROM BOOT SUBRTN CURRENT TRACK</td>
</tr>
<tr>
<td>0041</td>
<td>ROM BOOT SUBRTN TRACK TO READ</td>
</tr>
<tr>
<td></td>
<td>-- BLOCK READ PARAMETER LIST --</td>
</tr>
<tr>
<td>0042</td>
<td>COMMAND NUMBER (1 = READ)</td>
</tr>
<tr>
<td>0043</td>
<td>SLOT * 16</td>
</tr>
<tr>
<td>0044</td>
<td>I/O BUFFER ADDRESS ($44/$45)</td>
</tr>
<tr>
<td>0045</td>
<td></td>
</tr>
<tr>
<td>0046</td>
<td>BLOCK TO READ ($46/$47)</td>
</tr>
<tr>
<td>0047</td>
<td></td>
</tr>
<tr>
<td>0048</td>
<td>POINTER TO BLOCK READ ROUTINE</td>
</tr>
<tr>
<td>0049</td>
<td></td>
</tr>
<tr>
<td>004A</td>
<td>VOL DIR ENTRY POINTER/FRST INDEX PAGE</td>
</tr>
<tr>
<td>004B</td>
<td></td>
</tr>
<tr>
<td>004C</td>
<td>ADDR OF SECOND PAGE OF INDEX BLOCK</td>
</tr>
<tr>
<td>004D</td>
<td></td>
</tr>
<tr>
<td>004E</td>
<td>INDEX INTO INDEX BLOCK PAGES</td>
</tr>
<tr>
<td>0050</td>
<td>TRACK SEEK PHASE-ON INDEX</td>
</tr>
<tr>
<td>0051</td>
<td>TRACK PHASE WANTED</td>
</tr>
<tr>
<td>0052</td>
<td>BLOCK READER RETRY COUNT</td>
</tr>
<tr>
<td>0053</td>
<td>CURRENT TRACK PHASE/PHASE-OFF INDEX</td>
</tr>
<tr>
<td>0054</td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>BUFFER POINTER</td>
</tr>
<tr>
<td>0061</td>
<td></td>
</tr>
<tr>
<td>005E</td>
<td>SCREEN CENTER LINE</td>
</tr>
<tr>
<td>0080</td>
<td>LOAD POINT FOR RELOCATOR</td>
</tr>
<tr>
<td>C000</td>
<td>DISK ARM PHASED</td>
</tr>
<tr>
<td>C008</td>
<td>TURN DISK DRIVE OFF</td>
</tr>
<tr>
<td>C089</td>
<td>TURN DISK DRIVE ON</td>
</tr>
</tbody>
</table>

---

**LOAD PRODOS**

(ENTIRE LOADER IN MEMORY NOW)

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0801</td>
<td>MAIN ENTRY ***************</td>
</tr>
<tr>
<td></td>
<td>ON ENTRY, X = SLOT*16</td>
</tr>
<tr>
<td></td>
<td>A = SECTOR NUMBER</td>
</tr>
<tr>
<td>0802</td>
<td>ENTRY POINT FOR APPLE II</td>
</tr>
<tr>
<td>0803</td>
<td>ALWAYS TAKE TRACK (APPLE II) &gt;&gt;0807</td>
</tr>
<tr>
<td>0804</td>
<td>JUMP TO APPLE /// LOGIC &gt;&gt;A132</td>
</tr>
<tr>
<td>0807</td>
<td>SAVE SLOT*16</td>
</tr>
<tr>
<td>0809</td>
<td>READING SECTOR 3 NEXT?</td>
</tr>
<tr>
<td>080B</td>
<td>REMEMBER THIS...</td>
</tr>
<tr>
<td>080D</td>
<td>MAKE $C FROM SLOT*16</td>
</tr>
<tr>
<td>0815</td>
<td>AND SAVE AT $49</td>
</tr>
<tr>
<td>0819</td>
<td>$48/49 --&gt; $CXFF IN ROM BOOT</td>
</tr>
<tr>
<td>081C</td>
<td>CHECK $CXFF</td>
</tr>
<tr>
<td>081D</td>
<td>BOOT ROM FOR DISK II?</td>
</tr>
<tr>
<td>081F</td>
<td>NO, NOT A 5.25&quot; FLOPPY &gt;&gt;085B</td>
</tr>
<tr>
<td>0821</td>
<td>GUT BOTH SECTORS OF LOADER? &gt;&gt;0831</td>
</tr>
<tr>
<td>0823</td>
<td>NO, STOP AT SECTOR 3</td>
</tr>
<tr>
<td>0825</td>
<td>STORE ON PARM (0808)</td>
</tr>
<tr>
<td>0828</td>
<td>SKIP SECTOR 1 (GET SEC 2)</td>
</tr>
<tr>
<td>082A</td>
<td>DUMMY UP $CX5C AS RETURN ADDRESS</td>
</tr>
<tr>
<td>0830</td>
<td>AND CALL ROM SECTOR READ SUBRTN</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0831</td>
<td>CURRENT TRACK IS ZERO</td>
</tr>
<tr>
<td>0833</td>
<td>$48/49 --&gt; $CXX0</td>
</tr>
<tr>
<td>0837</td>
<td>COPY A PORTION OF DISKETTE BOOT ROM</td>
</tr>
<tr>
<td>0839</td>
<td>TO MY BLOCK READER SUBROUTINE (0994)</td>
</tr>
<tr>
<td>083D</td>
<td>FROM $2077 TO $27E</td>
</tr>
<tr>
<td>0843</td>
<td>MODIFY SOME BRANCHES IN THE COPIED CODE (091D)</td>
</tr>
<tr>
<td>0846</td>
<td>TO SUIT MY ERROR HANDLING TASTES (0924)</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>084C</td>
<td>AND COPY SECTOR READ SUBROUTINE EXIT CODE (092B)</td>
</tr>
<tr>
<td>084F</td>
<td>TO $A7F TO $A85 (0A7F)</td>
</tr>
<tr>
<td>0855</td>
<td>$48/49 --&gt; DISKETTE BLOCK READER SUBRTN</td>
</tr>
<tr>
<td>0859</td>
<td>AT $0906</td>
</tr>
<tr>
<td>085B</td>
<td>---</td>
</tr>
<tr>
<td>085D</td>
<td>LEGAL DISK ROM?</td>
</tr>
<tr>
<td>085F</td>
<td>NO, ERROR &gt;&gt;0890</td>
</tr>
<tr>
<td>0861</td>
<td>STORE LSB OF BLOCK READER</td>
</tr>
<tr>
<td>0863</td>
<td>STORE ZEORS IN SEVERAL THINGS</td>
</tr>
<tr>
<td>086E</td>
<td>COMMAND = 1 (READ BLOCK)</td>
</tr>
<tr>
<td>0871</td>
<td>BLOCK NUMBER = 2 (VOL DIRECTORY)</td>
</tr>
<tr>
<td>0875</td>
<td>$60/61 --&gt; $C00 (BUFFER)</td>
</tr>
<tr>
<td>0877</td>
<td>$4A/48 --&gt; $C00 (FIRST ENTRY)</td>
</tr>
<tr>
<td>0879</td>
<td>READ VOLUME DIRECTORY BLOCKS &lt;0912&gt;</td>
</tr>
<tr>
<td>087C</td>
<td>ERROR? &gt;&gt;0866</td>
</tr>
<tr>
<td>087E</td>
<td>MOVE UP TWO PAGES IN MEMORY</td>
</tr>
<tr>
<td>0882</td>
<td>NEXT BLOCK NUMBER</td>
</tr>
<tr>
<td>0886</td>
<td>NOW AT BLOCK 67</td>
</tr>
<tr>
<td>0888</td>
<td>NO, GO READ NEXT ONE &gt;&gt;0879</td>
</tr>
<tr>
<td>088A</td>
<td>YES, CHECK LINK FOR VALIDITY ($C00)</td>
</tr>
<tr>
<td>0890</td>
<td>IF SHOULD BE ZERO FOR VOL DIR ($C01)</td>
</tr>
<tr>
<td>0899</td>
<td>BAD VOLUME DIR IF NOT ZERO &gt;&gt;08FF</td>
</tr>
<tr>
<td>0892</td>
<td>NO, INDEX PAST LINK AND VOL HDR</td>
</tr>
<tr>
<td>0894</td>
<td>AND BEGIN &gt;&gt;0893</td>
</tr>
<tr>
<td>0896</td>
<td>IF ALREADY PROCESSING, USE ENTRY LSB</td>
</tr>
<tr>
<td>0898</td>
<td>---</td>
</tr>
<tr>
<td>0899</td>
<td>ADD ENTRY LENGTH TO FIND NEXT ENTRY ($C23)</td>
</tr>
<tr>
<td>089D</td>
<td>STILL IN SAME PAGE? &gt;&gt;08AC</td>
</tr>
<tr>
<td>089F</td>
<td>NO, BUMP ENTRY MSB</td>
</tr>
<tr>
<td>08A3</td>
<td>IS IT ODD? (SECOND PAGE OF A BLOCK?)</td>
</tr>
<tr>
<td>08A4</td>
<td>YES... &gt;&gt;08AC</td>
</tr>
<tr>
<td>08A6</td>
<td>NO, JUST FINISHED LAST BLOCK?</td>
</tr>
<tr>
<td>08A8</td>
<td>YES, ERROR -- FILE NOT FOUND &gt;&gt;08FF</td>
</tr>
<tr>
<td>08A9</td>
<td>ELSE, START JUST PAST LINKS</td>
</tr>
<tr>
<td>08AC</td>
<td>UPDATE LSB OF ENTRY POINTER</td>
</tr>
<tr>
<td>08AE</td>
<td>GET NAME LENGTH (0902)</td>
</tr>
<tr>
<td>08B1</td>
<td>MASK OFF STORAGE TYPE</td>
</tr>
<tr>
<td>08B4</td>
<td>COMPARE NAME WITH &quot;PRODOS&quot;</td>
</tr>
<tr>
<td>08B9</td>
<td>NOT A MATCH? &gt;&gt;0896</td>
</tr>
<tr>
<td>08BE</td>
<td>IF NAME MATCHES, IS IT A SAVING FILE?</td>
</tr>
<tr>
<td>08BC</td>
<td>IF NOT, I CAN'T HANDLE IT &gt;&gt;08FF</td>
</tr>
<tr>
<td>08C0</td>
<td>GET FILE TYPE</td>
</tr>
<tr>
<td>08CB</td>
<td>SHOULD BE A PRODOS SYS FILE</td>
</tr>
<tr>
<td>08CA</td>
<td>IF NOT, I GIVE UP &gt;&gt;08FF</td>
</tr>
<tr>
<td>08CD</td>
<td>ALL IS WELL, COPY KEY BLOCK NUMBER</td>
</tr>
<tr>
<td>08CF</td>
<td>TO $46/47</td>
</tr>
<tr>
<td>08D6</td>
<td>$4A/48 AND $60/61 --&gt; $1E00</td>
</tr>
<tr>
<td>08DB</td>
<td>(BUFFER TO HOLD KEY BLOCK)</td>
</tr>
<tr>
<td>08E1</td>
<td>$4C/4D --&gt; $1F00 (SECOND PAGE)</td>
</tr>
<tr>
<td>08E3</td>
<td>READ A BLOCK &lt;0912&gt;</td>
</tr>
<tr>
<td>08E6</td>
<td>ERROR? &gt;&gt;08FF</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>08EA</td>
<td>BUMP TO NEXT BLOCK BUFFER</td>
</tr>
<tr>
<td>08EE</td>
<td>$4E = OFFSET INTO INDEX BLOCK</td>
</tr>
<tr>
<td>08F0</td>
<td>GET NEXT BLOCK NUMBER FROM INDEX BLOCK</td>
</tr>
<tr>
<td>08F8</td>
<td>BLOCK NUMBER = 07 (END OF FILE)</td>
</tr>
<tr>
<td>08FA</td>
<td>NOT YET, READ A BLOCK &gt;&gt;08E3</td>
</tr>
<tr>
<td>08FC</td>
<td>ELSE, JUMP TO RELOCATOR AT $2000 &gt;&gt;2000</td>
</tr>
<tr>
<td>08FF</td>
<td>ERROR JUMP &gt;&gt;093F</td>
</tr>
<tr>
<td>0902</td>
<td>*********** KERNEL NAME ***********************</td>
</tr>
<tr>
<td>0902</td>
<td>LENGTH OF KERNEL'S NAME</td>
</tr>
<tr>
<td>0903</td>
<td>&quot;PRODOS&quot; &quot;PRODOS&quot; (KERNEL NAME)</td>
</tr>
<tr>
<td>0912</td>
<td>*********** COPY BLOCK READ BUFFER PTR ***********************</td>
</tr>
<tr>
<td>0912</td>
<td>COPY $60/61 --&gt; $44/45</td>
</tr>
<tr>
<td>0914</td>
<td>(BLOCK READ BUFFER POINTER)</td>
</tr>
<tr>
<td>091A</td>
<td>THEN GO TO BLOCK I/O ROUTINE &gt;&gt;0048</td>
</tr>
<tr>
<td>091D</td>
<td>*********** ROM SECTOR READ OFFSETS ***********************</td>
</tr>
<tr>
<td>091D</td>
<td>OFFSETS INTO ROM SECTOR READ SUBROUTINE</td>
</tr>
<tr>
<td>091D</td>
<td>TO BRANCH DISPLACEMENTS WHICH NEED TO</td>
</tr>
<tr>
<td>091D</td>
<td>BE CHANGED FOR LOADER'S PURPOSES</td>
</tr>
<tr>
<td>091D</td>
<td>*********** NEW BRANCH OFFSETS FOR ABOVE ***</td>
</tr>
<tr>
<td>0924</td>
<td>---</td>
</tr>
<tr>
<td>092B</td>
<td>*********** SECTOR READ EXIT CODE ***********************</td>
</tr>
<tr>
<td>092B</td>
<td>COPIED TO END OF DISKETTE SECTOR READ CODE</td>
</tr>
<tr>
<td>092B</td>
<td>GET SLOT*16</td>
</tr>
<tr>
<td>092D</td>
<td>AND EXIT NORMALLY</td>
</tr>
<tr>
<td>092E</td>
<td>RETURN</td>
</tr>
<tr>
<td>092F</td>
<td>RESTART BLOCK READ OPERATION &gt;&gt;09BC</td>
</tr>
<tr>
<td>0912</td>
<td>*********** APPLE /// BOOT CODE ***********************</td>
</tr>
<tr>
<td>A132</td>
<td>THIS IS $A132 WHEN BOOTED ON APPLE ///</td>
</tr>
<tr>
<td>0912</td>
<td>MAKE IT LOOK LIKE A JSR FROM $A000</td>
</tr>
<tr>
<td>093B</td>
<td>LOAD IN BLOCK 1 (WE WANT SOS, NOT PRODOS)</td>
</tr>
<tr>
<td>093C</td>
<td>GO TO APPLE /// BLOCK READ ROUTINE &gt;&gt;P479</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>093F</td>
<td>ERROR HANDLER</td>
</tr>
<tr>
<td>093F</td>
<td>HOME_CURSOR/CLEAR_SCREEN &lt;FC50&gt;</td>
</tr>
<tr>
<td>0944</td>
<td>COPY &quot;UNABLE TO LOAD PRODOS&quot; MESSAGE (0950)</td>
</tr>
<tr>
<td>0947</td>
<td>TO SCREEN (05AE)</td>
</tr>
<tr>
<td>094D</td>
<td>THEN GO TO SLEEP FOREVER &gt;&gt;094D</td>
</tr>
<tr>
<td>0950</td>
<td>*** UNABLE TO LOAD PRODOS ***</td>
</tr>
<tr>
<td>0950</td>
<td>---</td>
</tr>
<tr>
<td>096D</td>
<td>MOVE ARM TO NEXT PHASE</td>
</tr>
<tr>
<td>096F</td>
<td>GET CURRENT PHASE</td>
</tr>
<tr>
<td>0972</td>
<td>CONVERT TO NEXT ARM PHASE</td>
</tr>
<tr>
<td>0975</td>
<td>SELECT NEXT ARM PHASE THIS DRIVE (C8B0)</td>
</tr>
<tr>
<td>097A</td>
<td>---</td>
</tr>
<tr>
<td>097C</td>
<td>DELAY LONG ENOUGH FOR ARM TO MOVE</td>
</tr>
<tr>
<td>0963</td>
<td>WHEN FINISHED, RETURN WITH X = SLOT*16</td>
</tr>
<tr>
<td>0965</td>
<td>RETURN</td>
</tr>
<tr>
<td>0986</td>
<td>DISKETTE BLOCK READ ROUTINE</td>
</tr>
<tr>
<td>0986</td>
<td>$44/$45 --&gt; BUFFER</td>
</tr>
<tr>
<td>0986</td>
<td>$46/$47 = BLOCK NO.</td>
</tr>
<tr>
<td>0986</td>
<td>GET BLOCK NO. LSB</td>
</tr>
<tr>
<td>0988</td>
<td>ISOLATE SECTOR REMAINDER</td>
</tr>
<tr>
<td>098C</td>
<td>SKEW SECTOR BY 2</td>
</tr>
<tr>
<td>0992</td>
<td>AND STORE SECTOR WANTED</td>
</tr>
<tr>
<td>0994</td>
<td>GET MSB</td>
</tr>
<tr>
<td>0996</td>
<td>AND HIGH BIT OF TRACK</td>
</tr>
<tr>
<td>0999</td>
<td>MERGE WITH LOW PART OF TRACK</td>
</tr>
<tr>
<td>099C</td>
<td>STORE TRACK WANTED</td>
</tr>
<tr>
<td>099F</td>
<td>TRACK*2 IS PHASE WANTED</td>
</tr>
<tr>
<td>09A3</td>
<td>SET PAGE ADDRESS OF BUFFER</td>
</tr>
<tr>
<td>09A7</td>
<td>TURN DRIVE MOTOR ON (C8B9)</td>
</tr>
<tr>
<td>09A6</td>
<td>READ SECTOR &lt;09BC&gt;</td>
</tr>
<tr>
<td>09AD</td>
<td>NEXT PAGE</td>
</tr>
<tr>
<td>09B1</td>
<td>SKEW TO NEXT SECTOR</td>
</tr>
<tr>
<td>09B5</td>
<td>READ SECOND SECTOR OF BLOCK &lt;09BC&gt;</td>
</tr>
<tr>
<td>09B8</td>
<td>THEN TURN MOTOR OFF AND EXIT (C883)</td>
</tr>
<tr>
<td>09BB</td>
<td>RETURN</td>
</tr>
<tr>
<td>09BC</td>
<td>GET CURRENT TRACK</td>
</tr>
<tr>
<td>09BF</td>
<td>CONVERT TO PHASE</td>
</tr>
<tr>
<td>09C5</td>
<td>GET CURRENT PHASE</td>
</tr>
<tr>
<td>09C7</td>
<td>STORE FOR PHASE OFF</td>
</tr>
<tr>
<td>09CA</td>
<td>SUBTRACT PHASE WANTED TO DETERMINE...</td>
</tr>
<tr>
<td>09CC</td>
<td>DIRECTION -- ON CORRECT TRACK NOW? &gt;&gt;09E2</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>04B8</td>
<td>SCREEN BUFFER ROW 10</td>
</tr>
<tr>
<td>05A9</td>
<td>SCREEN BUFFER ROW 12</td>
</tr>
<tr>
<td>05AD</td>
<td>SCREEN BUFFER ROW 12</td>
</tr>
<tr>
<td>06D6</td>
<td>SCREEN BUFFER ROW 14</td>
</tr>
<tr>
<td>07AB</td>
<td>SCREEN BUFFER ROW 16</td>
</tr>
<tr>
<td>07AD</td>
<td>SCREEN BUFFER ROW 16</td>
</tr>
<tr>
<td>07D0</td>
<td>SCREEN BUFFER ROW 24</td>
</tr>
<tr>
<td>0800</td>
<td>ENTRY OF INTERP LOADER</td>
</tr>
<tr>
<td>0C00</td>
<td>VOLUME DIRECTORY BUFFER</td>
</tr>
<tr>
<td>0C23</td>
<td>ENTRY LENGTH -- RAMDRIVE VOLUME DIRECTORY --</td>
</tr>
<tr>
<td>0E94</td>
<td>VOLUME HDR, VOLUME NAME</td>
</tr>
<tr>
<td>0E22</td>
<td>VOLUME HDR, ACCESS TOTAL BLOCKS</td>
</tr>
<tr>
<td>2800</td>
<td>START OF SYSTEM PROGRAMS</td>
</tr>
<tr>
<td>2C90</td>
<td>RAMDRIVE DEVICE DRIVER LOAD ADDRESS</td>
</tr>
<tr>
<td>2A60</td>
<td>DIFFERENCE OF RAMDRIVE LOAD AND RUN LOCATIONS</td>
</tr>
<tr>
<td>SFFF</td>
<td>TOP OF 4K RAM</td>
</tr>
<tr>
<td>BF00</td>
<td>ENTRY POINT FOR MLI</td>
</tr>
<tr>
<td>BF03</td>
<td>QUIT VECTOR</td>
</tr>
<tr>
<td>BF06</td>
<td>DATE/TIME</td>
</tr>
<tr>
<td>BF10</td>
<td>DEVICE HANDLER TABLES</td>
</tr>
<tr>
<td>BF30</td>
<td>LAST DEVICE USED</td>
</tr>
<tr>
<td>BF31</td>
<td>NUMBER OF ACTIVE DISK DEVICES</td>
</tr>
<tr>
<td>BF32</td>
<td>ACTIVE DISKS SEARCH LIST</td>
</tr>
<tr>
<td>BF98</td>
<td>MACHINE TYPE FLAGS</td>
</tr>
<tr>
<td>BF99</td>
<td>SLOT WHICH CONTAINS CARDS WITH ROM</td>
</tr>
<tr>
<td>SFFF</td>
<td>MLI VERSION NUMBER</td>
</tr>
<tr>
<td>C000</td>
<td>80 STORE OFF</td>
</tr>
<tr>
<td>C001</td>
<td>80 STORE ON</td>
</tr>
<tr>
<td>C002</td>
<td>READ MAIN RAM</td>
</tr>
<tr>
<td>C003</td>
<td>READ AUX RAM</td>
</tr>
<tr>
<td>C004</td>
<td>WRITE MAIN RAM</td>
</tr>
<tr>
<td>C005</td>
<td>WRITE AUX RAM</td>
</tr>
<tr>
<td>C008</td>
<td>MAIN STACK/ZERO PAGE</td>
</tr>
<tr>
<td>C009</td>
<td>ALTERNATE STACK/ZERO PAGE</td>
</tr>
<tr>
<td>C00A</td>
<td>INTERNAL SLOT 3 ROM</td>
</tr>
<tr>
<td>C00B</td>
<td>PERIPHERAL SLOT 3 ROM</td>
</tr>
<tr>
<td>C00C</td>
<td>80 COLUMN DISPLAY OFF</td>
</tr>
<tr>
<td>C018</td>
<td>READ 80 STORE SWITCH</td>
</tr>
</tbody>
</table>
### ProDOS Relocator -- V1.2 -- 6 SEP 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C030</td>
<td>SPEAKER</td>
</tr>
<tr>
<td>C054</td>
<td>USE MAIN MEMORY PART OF 80-COL CARD</td>
</tr>
<tr>
<td>C055</td>
<td>USE AUX MEMORY PART OF 80-COL CARD</td>
</tr>
<tr>
<td>C068</td>
<td>IIGS STATEROG STATUS BYTE</td>
</tr>
<tr>
<td>C081</td>
<td>WRITE-ENABLE HIGH RAM</td>
</tr>
<tr>
<td>C082</td>
<td>MOTHERBOARD ROM READ ENABLE</td>
</tr>
<tr>
<td>C083</td>
<td>READ/WRITE RAM 2ND 4K BANK</td>
</tr>
<tr>
<td>C08B</td>
<td>READ/WRITE RAM 1ST 4K BANK</td>
</tr>
</tbody>
</table>

**INTERNAL C3ROM ADDRESSES**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C311</td>
<td>MOVE TO/FROM AUXMEM SUBROUTINE</td>
</tr>
<tr>
<td>C314</td>
<td>TRANSFER TO/FROM AUXMEM SUBROUTINE</td>
</tr>
</tbody>
</table>

**SLOT ROM ADDRESSES**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C305</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C307</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C308</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C30C</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C30E</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>CFFFF</td>
<td>RESET 1/O CARD ROMS</td>
</tr>
</tbody>
</table>

**PRODUCT ADDRESSES**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D000</td>
<td>START OF QUITCODE MEMORY AREA (BANK2)</td>
</tr>
<tr>
<td>DFF1</td>
<td>ENHANCED ROM FLAG</td>
</tr>
<tr>
<td>F000</td>
<td>VERSION NUMBER (FOR SUBDIRECTORIES)</td>
</tr>
<tr>
<td>FF0F</td>
<td>GS VIDEO FLAG</td>
</tr>
<tr>
<td>FFF0</td>
<td>RAMDRIVE CALLER ADDRESS</td>
</tr>
</tbody>
</table>

**MONITOR ROM**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB1E</td>
<td>PADDLE READ SUBROUTINE</td>
</tr>
<tr>
<td>FB2F</td>
<td>MONITOR INIT ROUTINE</td>
</tr>
<tr>
<td>FBB3</td>
<td>ROM VERSION BYTE</td>
</tr>
<tr>
<td>FBC0</td>
<td>SECONDARY VERSION BYTE (0-3)</td>
</tr>
<tr>
<td>F50B</td>
<td>CLEAR SCREEN</td>
</tr>
<tr>
<td>F10F</td>
<td>THIS ROUTINE CHECKS FOR IIGS</td>
</tr>
<tr>
<td>FBB4</td>
<td>SET NORMAL VIDEO</td>
</tr>
<tr>
<td>FBE9</td>
<td>IN#0</td>
</tr>
<tr>
<td>FE93</td>
<td>PR#0</td>
</tr>
</tbody>
</table>

**PRODUCTS RELOCATOR MAIN ENTRY**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>JUMP OVER QUIT ENTRY &gt;&gt;2006</td>
</tr>
<tr>
<td>2003</td>
<td>SET FLAG INDICATING QUIT ENTRY (IIGS) (21D1)</td>
</tr>
<tr>
<td>2006</td>
<td>STORE SLOT IN MLI ONLINE PARMS</td>
</tr>
<tr>
<td>2008</td>
<td>PRINT &quot;APPLE II PRODOS...&quot; &lt;25B1&gt;</td>
</tr>
<tr>
<td>2008</td>
<td>SET UP FOR COMMON MOVES (226E)</td>
</tr>
<tr>
<td>2014</td>
<td>RELOCATE SOME ROUTINES &amp; DATA TO LOW MEMORY &lt;2880&gt;</td>
</tr>
</tbody>
</table>

**RELOCATE PRODOS**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017</td>
<td>RELOCATION ERROR &gt;&gt;203C</td>
</tr>
<tr>
<td>201E</td>
<td>BE SURE 48K OF MAIN MEMORY EXISTS (BFF)</td>
</tr>
<tr>
<td>2024</td>
<td>IF NOT, ERROR &gt;&gt;204E</td>
</tr>
<tr>
<td>2029</td>
<td>MAKE DOUBLY SURE (BFF)</td>
</tr>
<tr>
<td>202E</td>
<td>ERROR THIS TIME &gt;&gt;204E</td>
</tr>
<tr>
<td>202F</td>
<td>SELECT MOTHERBOARD ROMS (C082)</td>
</tr>
<tr>
<td>2030</td>
<td>DETERMINE MACHINE TYPE &lt;251F&gt;</td>
</tr>
<tr>
<td>2036</td>
<td>PICK UP CONFIGURATION BYTE</td>
</tr>
<tr>
<td>2038</td>
<td>64K OR MORE MEMORY?</td>
</tr>
<tr>
<td>203A</td>
<td>YES, WE HAVE 64K RAM &gt;&gt;203F</td>
</tr>
<tr>
<td>203C</td>
<td>ERROR. MUST HAVE 64K OR MORE!! &gt;&gt;2227</td>
</tr>
</tbody>
</table>

**RELOCATE PRODOS**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>203F</td>
<td>SET UP FOR MLI MOVE (2278)</td>
</tr>
<tr>
<td>2045</td>
<td>COPY/RELOCATE PRODOS ITSELF &lt;208B</td>
</tr>
<tr>
<td>2048</td>
<td>GET PRODOS VERSION NUMBER (BFF)</td>
</tr>
<tr>
<td>204B</td>
<td>AND PUT IT IN MLI DATA AREA. (FDBB)</td>
</tr>
<tr>
<td>204E</td>
<td>RELOCATION ERROR! &gt;&gt;20AC</td>
</tr>
<tr>
<td>2050</td>
<td>ENABLE MOTHERBOARD ROMS AGAIN (C082)</td>
</tr>
<tr>
<td>2053</td>
<td>CHECK ROM I.D. BYTE (FBB3)</td>
</tr>
<tr>
<td>2056</td>
<td>APPLE //e FAMILY?</td>
</tr>
<tr>
<td>2058</td>
<td>NO, LEAVE I.D. BYTE AS IS &gt;&gt;208E</td>
</tr>
<tr>
<td>205C</td>
<td>TEST ANOTHER ROM I.D. BYTE (FBCU)</td>
</tr>
<tr>
<td>205F</td>
<td>SAVE BIT TEST RESULTS</td>
</tr>
<tr>
<td>2060</td>
<td>GET MACHID</td>
</tr>
<tr>
<td>2062</td>
<td>STRIP BITS THAT IDENTIFY MODEL</td>
</tr>
<tr>
<td>2067</td>
<td>IT'S A //e IF BITS 6 &amp; 7 ARE HIGH &gt;&gt;2075</td>
</tr>
<tr>
<td>2069</td>
<td>---</td>
</tr>
<tr>
<td>206A</td>
<td>EITHER A //c OR A FUTURE SYSTEM</td>
</tr>
<tr>
<td>206C</td>
<td>CHECK HIGH BITS OF FBCU AGAIN</td>
</tr>
<tr>
<td>206D</td>
<td>BIT 7 ON &gt;&gt;2073</td>
</tr>
<tr>
<td>206F</td>
<td>YES, FUTURE SYSTEM.</td>
</tr>
<tr>
<td>2073</td>
<td>IF BIT 6 ON, IT'S A FUTURE SYSTEM. &gt;&gt;2077</td>
</tr>
<tr>
<td>2075</td>
<td>---</td>
</tr>
<tr>
<td>2077</td>
<td>REPLACE UPDATED MACHID</td>
</tr>
<tr>
<td>207D</td>
<td>LOOK AT ROM. THIS A IIGS? &lt;F10F</td>
</tr>
<tr>
<td>2080</td>
<td>NO, CARRY STILL SET. &gt;&gt;208E</td>
</tr>
<tr>
<td>2082</td>
<td>YES, SET IIGS FLAG. (2278)</td>
</tr>
<tr>
<td>2085</td>
<td>ENTER FROM QUIT? (21D1)</td>
</tr>
<tr>
<td>2088</td>
<td>YES, THIS IS NOT INITIAL BOOT. &gt;&gt;208E</td>
</tr>
<tr>
<td>2089</td>
<td>NOW SET OS BOOT TO ZERO, INDICATING THAT PRODOSB</td>
</tr>
<tr>
<td>208A</td>
<td>WAS THE OPERATING SYSTEM INITIALLY BOOTED.</td>
</tr>
<tr>
<td>208B</td>
<td>TE1006 INSTRUCTION: STA $E100BB</td>
</tr>
<tr>
<td>208E</td>
<td>COPY BOOT DEVICE ID TO READ BLOCK PARMS (2262)</td>
</tr>
<tr>
<td>2094</td>
<td>AND AS LAST DEVICE USED (BF3E)</td>
</tr>
<tr>
<td>2097</td>
<td>DETERMINE PERIPHERAL CARD CONFIGURATION &lt;265F&gt;</td>
</tr>
<tr>
<td>2098</td>
<td>BOOT DEVICE TO... (2269)</td>
</tr>
<tr>
<td>209D</td>
<td>GLOBAL PAGE LAST DEVICE USED (WF3E)</td>
</tr>
<tr>
<td>20A0</td>
<td>ENABLE READ/WRITE HIGH RAM, BANK 1 &lt;2518&gt;</td>
</tr>
<tr>
<td>20A9</td>
<td>COPY CLOCK CODE TO DEVICE DRIVER AREA &lt;2880&gt;</td>
</tr>
</tbody>
</table>
### ProDOS Relocator -- V1.2 -- 6 SEP 86

**NEXT OBJECT ADDR: 20AC**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20AC</td>
<td>ERROR? &gt;&gt;20DA</td>
</tr>
<tr>
<td>20AE</td>
<td>CHECK MACHINE TYPE AGAIN (BF98)</td>
</tr>
<tr>
<td>20B1</td>
<td>GOT 64K OR MORE?</td>
</tr>
<tr>
<td>20B5</td>
<td>NO &gt;&gt;20DD</td>
</tr>
<tr>
<td>20B7</td>
<td>YES, QUIT VECTOR --&gt; $FCA9</td>
</tr>
<tr>
<td>20C1</td>
<td>WRITE TO HIGH RAM (BANK2) (C003)</td>
</tr>
<tr>
<td>20CA</td>
<td>POINT TO QUIT CODE TABLE (2275)</td>
</tr>
<tr>
<td>20CD</td>
<td>MOVE QUIT CODE TO HIGH RAM &lt;28B0&gt;</td>
</tr>
<tr>
<td>20D2</td>
<td>STORE QUIT VECTOR START PAGE (D000)</td>
</tr>
<tr>
<td>20D5</td>
<td>ENABLE READ/WRITE HIGH RAM, BANK 1 &lt;2518&gt;</td>
</tr>
<tr>
<td>20DA</td>
<td>RELOCATION ERROR &gt;&gt;2227</td>
</tr>
<tr>
<td>20DD</td>
<td>GET MACHID YET AGAIN (BF96)</td>
</tr>
<tr>
<td>20E0</td>
<td>128K?</td>
</tr>
<tr>
<td>20E4</td>
<td>NO... &gt;&gt;20FC</td>
</tr>
<tr>
<td>20E6</td>
<td>YES, SET UP AUX RAM</td>
</tr>
<tr>
<td>20EA</td>
<td>DISABLE INTERRUPTS</td>
</tr>
<tr>
<td>20BB</td>
<td>PREPARE TO WRITE TO AUX STACK AREA (C009)</td>
</tr>
<tr>
<td>20EE</td>
<td>AUX STACK POINTER SET TO $FF (0161)</td>
</tr>
<tr>
<td>20F1</td>
<td>BACK TO MAIN 2-PAGE, STACK (C008)</td>
</tr>
<tr>
<td>20F4</td>
<td>RESTORE STATUS REG</td>
</tr>
<tr>
<td>20F9</td>
<td>ESTABLISH RAM DRIVE IN AUX MEM &lt;28B0&gt;</td>
</tr>
</tbody>
</table>

******** SET UP FOR IRQ (ENHANCED ROM) ********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20FC</td>
<td>READ ROM (C081)</td>
</tr>
<tr>
<td>20FF</td>
<td>GET ROM'S IRQ VECTOR (FFE0)</td>
</tr>
<tr>
<td>2105</td>
<td>ENABLE READ/WRITE HIGH RAM, BANK 1 &lt;2518&gt;</td>
</tr>
<tr>
<td>2108</td>
<td>CARRY CLEAR IF IRQ VECTOR IN CJ ROM</td>
</tr>
<tr>
<td>210A</td>
<td>FLAG FOR &quot;OLD ROM&quot;</td>
</tr>
<tr>
<td>210C</td>
<td>IT'S AN OLD ROM &gt;&gt;2127</td>
</tr>
<tr>
<td>210E</td>
<td>SWITCH TO AUX STACK &amp; HIGH RAM (C009)</td>
</tr>
<tr>
<td>2113</td>
<td>INITIALIZE AUX STACK POINTER TO $0F (0101)</td>
</tr>
<tr>
<td>2116</td>
<td>PUT IRQ VECTOR IN AUX HIGH RAM (FFE0)</td>
</tr>
<tr>
<td>211C</td>
<td>BACK TO MAIN HIGH RAM &amp; 2-PAGE (C008)</td>
</tr>
<tr>
<td>211F</td>
<td>PUT IRQ VECTOR IN MAIN HIGH RAM (FFE0)</td>
</tr>
<tr>
<td>2125</td>
<td>INDICATE ENHANCED IRQ LOGIC ON BOARD</td>
</tr>
<tr>
<td>2127</td>
<td>STORE FLAG IN MLI DATA AREA (DF1)</td>
</tr>
</tbody>
</table>

******** LOOK FOR SLOT 3 VIDEO CARD ********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>212A</td>
<td>---</td>
</tr>
<tr>
<td>212C</td>
<td>SET GS VIDEO FLAG=0 (FFE0)</td>
</tr>
<tr>
<td>212F</td>
<td>THIS A IIGS? (2278)</td>
</tr>
<tr>
<td>2132</td>
<td>NO. &gt;&gt;2133</td>
</tr>
<tr>
<td>2134</td>
<td>YES, SET GS VIDEO FLAG (IN MLI) (FFE0)</td>
</tr>
<tr>
<td>2137</td>
<td>AND DON'T BOTHER SEARCHING SLOT 3. &gt;&gt;21A5</td>
</tr>
<tr>
<td>213A</td>
<td>ENABLE INTERNAL VIDEO Firmware (C00A)</td>
</tr>
<tr>
<td>213D</td>
<td>CHECK FOR ROM (BF99)</td>
</tr>
<tr>
<td>2140</td>
<td>IN SLOT 3.</td>
</tr>
<tr>
<td>2142</td>
<td>ROM EXISTS. &gt;&gt;2147</td>
</tr>
</tbody>
</table>

---

### ProDOS Relocator -- V1.2 -- 6 SEP 86

**NEXT OBJECT ADDR: 2144**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2144</td>
<td>NO ROM. &gt;&gt;21AD</td>
</tr>
<tr>
<td>2147</td>
<td>LOOK AT THE SLOT 3 ROM (C008)</td>
</tr>
<tr>
<td>214A</td>
<td>AT OFFSET +$05 (C005)</td>
</tr>
<tr>
<td>214D</td>
<td>THERE MUST BE A $38</td>
</tr>
<tr>
<td>2151</td>
<td>AND AT OFFSET +$07 (C307)</td>
</tr>
<tr>
<td>2154</td>
<td>THERE MUST BE AN $18</td>
</tr>
<tr>
<td>2158</td>
<td>AND AT OFFSET +$08 (C308)</td>
</tr>
<tr>
<td>215B</td>
<td>THERE MUST BE A 1</td>
</tr>
<tr>
<td>215F</td>
<td>AND AT OFFSET +$8C (C38C)</td>
</tr>
<tr>
<td>2164</td>
<td>INDICATE AN 80-COL CARD.</td>
</tr>
<tr>
<td>2168</td>
<td>CHECK MACHINE TYPE (BF98)</td>
</tr>
<tr>
<td>216D</td>
<td>IS THIS AN APPLE III?</td>
</tr>
<tr>
<td>216F</td>
<td>OK, IT'S GOT 80-COL CAPABILITY &gt;&gt;21A5</td>
</tr>
<tr>
<td>2171</td>
<td>OTHER MANUFACTURERS MUST FOLLOW THE RULES! (C3FA)</td>
</tr>
<tr>
<td>2174</td>
<td>MUST HAVE BIT INSTRUCTION AT $C3FA</td>
</tr>
<tr>
<td>2176</td>
<td>TO TURN ON 80-COL (C001)</td>
</tr>
<tr>
<td>2177</td>
<td>CHECK FOR AUX MEM. (C055)</td>
</tr>
<tr>
<td>2183</td>
<td>PUT A BYTE AT AUX $400 (0400)</td>
</tr>
<tr>
<td>2186</td>
<td>THE ACCUMULATOR LEFT</td>
</tr>
<tr>
<td>2187</td>
<td>AND THE SAME WITH $400 (0400)</td>
</tr>
<tr>
<td>218A</td>
<td>STILL THE SAME? (0400)</td>
</tr>
<tr>
<td>218D</td>
<td>NO, NO 80-COL MEMORY &gt;&gt;2196</td>
</tr>
<tr>
<td>218F</td>
<td>SHIFT TO THE RIGHT</td>
</tr>
<tr>
<td>2193</td>
<td>STILL THE SAME? (0400)</td>
</tr>
<tr>
<td>2196</td>
<td>BACK TO MAIN MEMORY (C054)</td>
</tr>
<tr>
<td>2199</td>
<td>TURN OFF 80-COL (C000)</td>
</tr>
<tr>
<td>219C</td>
<td>WAS 80-COL MEMORY FOUND? &gt;&gt;21A5</td>
</tr>
<tr>
<td>219E</td>
<td>NO. SO TURN OFF 80-COL FLAG (BF98)</td>
</tr>
<tr>
<td>21A1</td>
<td>IN MACHINE I.D. BYTE.</td>
</tr>
<tr>
<td>21A3</td>
<td>ALWAYS BRANCH &gt;&gt;21AA</td>
</tr>
<tr>
<td>21A5</td>
<td>TURN ON BOOTING (BF98)</td>
</tr>
<tr>
<td>21AD</td>
<td>THIS A IIGS? (2278)</td>
</tr>
<tr>
<td>21B0</td>
<td>NO. &gt;&gt;21C8</td>
</tr>
<tr>
<td>21B2</td>
<td>YES, ENABLE IIGS CLOCK DRIVER</td>
</tr>
<tr>
<td>21B7</td>
<td>GET ADDRESS OF RELOCATE TABLE (2276)</td>
</tr>
<tr>
<td>21BA</td>
<td>FOR IIGS CLOCK CODE (2277)</td>
</tr>
<tr>
<td>21B8</td>
<td>AND PUT THE CODE AT $D742 &lt;208B&gt;</td>
</tr>
<tr>
<td>21C8</td>
<td>INDICATE CLOCK EXISTS IN MACHID (BF98)</td>
</tr>
<tr>
<td>21C4</td>
<td>ENTER FROM PQUIT? (21D1)</td>
</tr>
<tr>
<td>21C9</td>
<td>NO. &gt;&gt;21D2</td>
</tr>
<tr>
<td>21CD</td>
<td>YES, ENABLE ROM FOR READ (C082)</td>
</tr>
<tr>
<td>21D0</td>
<td>RETURN</td>
</tr>
<tr>
<td>21D1</td>
<td>PQUIT FLAG. (4 = PRODOS 8 WAS INITIAL BOOT)</td>
</tr>
</tbody>
</table>
ProDOS Relocator — V1.2 — 6 SEP 86

ADDR DESCRIPTION/CONTENTS

********* GET VOL LABEL ***********

21D2 MLI: ONLINE DEVICE CALL <BF00>
21D8 ERROR? >>2227
21DD VALID VOLUME NAME?
21DF IF NOT, ERROR >>2227
21E2 ELSE, BUMP LENGTH BY ONE
21E7 AND PREFIX NAME BY A "/"
21EC MLI: SET PREFIX <BF00>
21F2 ERROR? >>2227

********* READ VOLUME DIRECTORY **********

21F4 ---
21F5 $14/15 --- $C00
21FB ---
2200 BLOCK = 2 (VOLUME DIRECTORY) (226C)
2206 MLI: READ BLOCK <BF00>
220C ERROR? >>2227
2210 GET NEXT BLOCK NUMBER
2216 IF ZERO, END OF VOLUME DIRECTORY >>2224
2212 ADD TWO PAGES (ONE BLOCK) TO POINTER
2220 AND STOP AT $1400 IN ANY CASE
2222 ELSE, READ NEXT BLOCK AS WELL >>21FB
2224 WHEN DONE, JUMP TO SYSTEM FILE LOADER >>0300

2227 ********** ERROR HANDLER ******************

2227 ENABLE MOTHERBOARD ROMS (C082)
222A CLEAR SCREEn <PC58>
222F PRINT "RELOCATION/CONFIG ERROR" (223B)
2236 THEN SLEEP FOREVER >>2238

223B ********* DATA ******************

223B ---
223B "RELOCATION / CONFIGURATION ERROR"

2261 MLI: ONLINE PARMS
2262 SLOT*16 AND DRIVE
2263 READ THEM TO $281

2265 MLI: SET PREFIX PARMS
2266 PREFIX IS AT $280

2268 MLI: READ BLOCK PARMS
2269 DEVICE
226A BUFFER
226C BLOCK NUMBER

********* ADDRESS OF COMMON MOVES RELOC TABLE **************

226E ADDRESS OF COMMON MOVES RELOC TABLE
2270 ADDRESS OF PRODOS RELOC TABLE
2272 ADDRESS OF THUNDERCLOCK DRIVER RELOC TABLE
2274 ADDRESS OF QUIT CODE RELOC TABLE
2276 ADDRESS OF IIGS CLOCK DRIVER RELOC TABLE
2278 IIGS FLAG. IF NON-ZERO, THIS IS A IIGS.

2279 ********** RELOCATION TABLES ***********************

+0: 00 - ZERO BLOCK OF MEMORY
+0: 01 - COPY BLOCK
+0: 02 - RELOCATE MSB ADDRESSES
+0: 03 - RELOCATE 2 BYTES ADDRESSES
+0: 04 - RELOCATE INSTRUCTIONS
+1/2: ADDR OF OUTPUT BLOCK
+3/4: LENGTH OF BLOCK IN BYTES
+5/6: ADDR OF INPUT BLOCK (IF ANY)
+7: ADD MESSAGES TO INPUT BLOCKS (IF ANY)
+8: START PAGES
+8: COUNT: END PAGE ADDRESSES
+8: COUNT+COUNT: ADDITIVE CORRECTION FACTOR

********* COMMON MOVES TABLE **********

2279 COPY (SYSTEM FILE LOADER)
227A TO =$800
227C LEN=$213
227E FRM=$220B
2280 COPY (PAGE 3 IMAGE)
2281 TO =$5DB
2283 LEN=$2A
2285 FRM=$24EE
2287 COPY (CHECKSUM)
2288 TO =$9A
228A LEN=$02
228C FRM=$14
228E COPY (CHECK FOR 80-COL CARD)
228F TO =$90
2291 LEN=$46
2293 FRM=$256B
2295 END OF TABLE

********* QUIT CODE MOVE TABLE **********

2296 COPY (QUIT CODE)
2297 TO =$100
2299 LEN=$300
229B FRM=$5900
229D END OF TABLE
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>229E</td>
<td>COPY (IRQ HANDLER)</td>
</tr>
<tr>
<td>229F</td>
<td>TO =FF9B</td>
</tr>
<tr>
<td>22A1</td>
<td>LEN=65</td>
</tr>
<tr>
<td>22A3</td>
<td>FRM=519B</td>
</tr>
<tr>
<td>22A5</td>
<td>COPY (SYSTEM GLOBAL PAGE)</td>
</tr>
<tr>
<td>22A6</td>
<td>TO =BF00</td>
</tr>
<tr>
<td>22A8</td>
<td>LEN=100</td>
</tr>
<tr>
<td>22AA</td>
<td>FRM=50000</td>
</tr>
<tr>
<td>22AC</td>
<td>ZERO (PRODOS KERNEL DATA AREA)</td>
</tr>
<tr>
<td>22AD</td>
<td>ADM=$D700</td>
</tr>
<tr>
<td>22AF</td>
<td>LEN=$700</td>
</tr>
<tr>
<td>22B1</td>
<td>COPY (PRODOS KERNEL)</td>
</tr>
<tr>
<td>22B2</td>
<td>TO =DE00</td>
</tr>
<tr>
<td>22B4</td>
<td>LEN=2100</td>
</tr>
<tr>
<td>22B6</td>
<td>FRM=2F00</td>
</tr>
<tr>
<td>22B8</td>
<td>COPY (DISKETTE DRIVER)</td>
</tr>
<tr>
<td>22B9</td>
<td>TO =04000</td>
</tr>
<tr>
<td>22BB</td>
<td>LEN=7800</td>
</tr>
<tr>
<td>22BD</td>
<td>FRM=52000</td>
</tr>
<tr>
<td>22BF</td>
<td>END OF TABLE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>22C0</td>
<td>COPY (THUNDERCLOCK CODE)</td>
</tr>
<tr>
<td>22C1</td>
<td>TO =D742</td>
</tr>
<tr>
<td>22C3</td>
<td>LEN=7D</td>
</tr>
<tr>
<td>22C5</td>
<td>FRM=5100</td>
</tr>
<tr>
<td>22C7</td>
<td>RELOCATE INSTRUCTIONS</td>
</tr>
<tr>
<td>22C8</td>
<td>TO =D742</td>
</tr>
<tr>
<td>22CA</td>
<td>LEN=69</td>
</tr>
<tr>
<td>22CC</td>
<td>FRM=D742</td>
</tr>
<tr>
<td>22CE</td>
<td>FOR ADDR=8C1XX-$C1XX</td>
</tr>
<tr>
<td>22D1</td>
<td>ADJUST BY=$S0</td>
</tr>
<tr>
<td>22D2</td>
<td>END OF TABLE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>22D3</td>
<td>COPY (IIGS CLOCK CODE)</td>
</tr>
<tr>
<td>22D4</td>
<td>TO =D742</td>
</tr>
<tr>
<td>22D6</td>
<td>LEN=7D</td>
</tr>
<tr>
<td>22D8</td>
<td>FRM=55C00</td>
</tr>
<tr>
<td>22DA</td>
<td>END OF TABLE</td>
</tr>
</tbody>
</table>

22DB ******* SYSTEM FILE LOADER ***********************
(COPIED TO AND RUN AT $00)

22DB ML: GET FILE INFO CALL <BF00>
22DF FIRST SEE IF THERE IS AN "ATINIT" FILE
22E1 NO ERRORS >>22EA
22E3 IS ERROR "FILE NOT FOUND"?
22E5 YES, THAT'S OK. >>232C
22E7 NO, OTHER ERROR >>232F
22EA GET FILE TYPE OF FILE FOUND (088D)
22ED IS IT "ATINIT" FILE?
22EF NO, ERROR >>232F
22F1 ML: OPEN CALL <BF00>
22F7 FILE DOESN'T OPEN >>232F
22F9 ML: GET EOF CALL <BF00>
22FA CAN'T FIND EOF >>232F
2301 HIGH BYTE OF EOF (0A01)
2304 FILE TOO BIG >>232F
2306 MEDIUM BYTE OF EOF (0A06)
2309 MAX FILE SIZE IS $9800
230B FILE TOO BIG >>232F
230D PUT IN READ PARMS (0A07)
2310 GET LOW BYTE OF EOF (09FF)
2313 PUT IN READ PARMS (0A06)
2316 ML: READ CALL <BF00>
231C READ ERROR >>232F
231E ML: CLOSE CALL <BF00>
2324 CLOSE ERROR >>232F
2326 READ ROM (0902)
2329 GO TO APPLICATION <2000>
232C NOW LOOK FOR SYSTEM FILE >>08AB
232F PRINT ERROR MESSAGE: (233D)
2332 "UNABLE TO LOAD ATINIT FILE" (233D)
233B SLEEP FOREVER >>233B
233D MSG LENGTH
233E *** UNABLE TO LOAD ATINIT FILE ***
2364 GET FILE INFO PARMS (FOR ATINIT FILE)
(LOCATED AT $889 WHEN EXECUTED)
2365 PATHNAME ADDRESS
2368 FILE TYPE
2376 OPEN PARMS FOR ATINIT FILE
(AT $89B WHEN EXECUTED)
2377 PATHNAME ADDRESS
2379 I/O BUFFER AT $1400
237B REFNUM=1
**ProDOS Relocator -- V1.2 -- 6 SEP 86**

---

**NEXT OBJECT ADDR: 2378**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>237C</td>
<td>---</td>
</tr>
<tr>
<td>237D</td>
<td>&quot;ATINIT&quot;</td>
</tr>
</tbody>
</table>

2383 **LOOK FOR NORMAL SYSTEM FILE**

(This code executes at $8A88)

2383 $10/11 --> VOLUME DIRECTORY ENTRIES

2385 INITIALLY AT $C000

2387 OFFSET BEYOND LINKS (+4)

2389 JUMP OVER NEXT INSTRUCTION >>238D

---

**SCAN DIRECTORY FOR SYSTEM FILE**

238B PICK UP LSB

238D ---

238E BUMP BY ENTRY LENGTH ($C23)

2391 UPDATE LSB

2393 PAGE OVERFLOW? >>23A7

2395 NO, ROOM FOR ONE MORE ENTRY? ($C23)

239A NO, CHECK MSB

239D START OF A BLOCK? >>23A9

239F NO, AT END OF DIRECTORY?

23A1 YES, FILE NOT FOUND IN DIRECTORY >>23C1

23A3 NO, START NEW BLOCK AT +4

23A5 AND UPDATE LSB

23A7 BUMP MSB

23A9 ---

23AD "SYSTEM" FILE TYPE?

23AF NO, TRY ANOTHER. >>23B8

23B2 INACTIVE ENTRY?

23B4 IF SO, SKIP IT >>23B8

23BB SAVE NAME LENGTH AT $280 ($280)

23BD MUST BE AT LEAST 8 CHARLS LONG >>23BB

23BF JUMP AROUND ERROR CODE >>23C3

23C1 ERROR - SYSTEM FILE NOT FOUND >>2430

---

23C3 ---

23C6 IS THIS ".SYSTEM"?

23CB (SEE $24E7) ($0ABC)

23CC NO, SKIP ENTRY >>23B8

23D0 CHECK ALL CHARACTERS IN NAME >>23C6

---

**LOAD SYSTEM FILE AT $2000**

---

23D2 ---

23D4 ---

23D5 COPY NAME TO $281

23DC AND TO "UNABLE TO LOAD" MSG ($9E2)

23E4 ADD BLANK AT END OF NAME

23E6 IN MESSAGE ($9E3)

---

**ProDOS Relocator -- V1.2 -- 6 SEP 86**

**NEXT OBJECT ADDR: 23EA**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>23A</td>
<td>NAMELEN + ERRORMSGLEN</td>
</tr>
</tbody>
</table>

23B4 SAVE AT $24E1 ($9F6)

23BF MLI: OPEN .SYSTEM FILE <BF00>

23F5 OPEN ERROR >>243D

23F7 MLI: GET EOF <BF00>

23F9 CAN'T GET EOF >>243D

23FF GET HIGH BYTE ($A01)

2400 FILE TOO BIG >>2457

2404 GET MEDIUM BYTE ($A00)

2407 MUST BE LESS THAN $9000 BYTES

2409 TOO BIG. >>2437

240B STORE LENGTH IN MLI READ PARMS ($A07)

240E GET LOW BYTE ($0FF)

2411 AND STORE IN READ PARMS. ($A06)

2414 MLI: READ SYSTEM FILE INTO $2000 <BF00>

241A NO READ ERRORS >>2422

241C ERROR, BAD BUFFER?

241E YES, FILE WAS TOO LARGE >>2457

2420 ELSE, "UNABLE TO LOAD ..." >>243D

2422 MLI: CLOSE SYSTEM FILE <BF00>

2428 CLOSE ERROR >>243D

242A ENABLE MOTHERBOARD RUN (C002)

242D AND JUMP TO BEGINNING OF FILE >>2000

---

243B ---

2432 PRINT "UNABLE TO FIND A .SYSTEM FILE" ($9F9)

243B THEN GO TO SLEEP >>2462

243D GET NAME LENGTH ($9F6)

2440 LINE LENGTH

2443 LESS NAME LENGTH ($9F6)

2446 DIVIDED BY 2

2447 GIVES OFFSET TO CENTER THE LINE ($9F6)

244B PRINT "UNABLE TO LOAD ..." ($9D1)

2455 GO TO SLEEP FOREVER >>2462

2457 ---

2459 PRINT "SYSTEM PROGRAM TOO LARGE" ($9B1)

2462 GO TO SLEEP FOREVER >>2462

---

2464 **DATA AREA**

---

2464 "UNABLE TO FIND A .SYSTEM FILE"

246C **SYSTEM PROGRAM TOO LARGE**

24AC "UNABLE TO LOAD X.SYSTEM" **

24D1 NAME LEN +13H (LEN OF MSG)
ProDOS Relocator -- V1.2 -- 6 SEP 86

PRODOS RELATORE -- V1.2 -- 6 SEP 86

NEXT OBJECT ADDR: 24D1

ADDR  DESCRIPTION/CONTENTS

24D2  MLI: OPEN PARM LIST (AT $9F7 WHEN EXECUTING)
24D3  PATHNAME AT $280
24D5  I/O BUFFER AT $1400
24D7  REFINUM=1

24D8  MLI: GET EOF PARM LIST (AT $9FD)
24D9  REFINUM=1
24DA  EOF MARK POSITION

24DD  MLI: READ LIST (AT $A02)
24DE  REFINUM=1
24DF  READ INTO $2800

24E1  LENGTH (FROM EOF MARK)
24E3  ACTUAL LENGTH READ

24E5  MLI: CLOSE LIST (AT $A0A)
24E6  REFINUM=0, CLOSE ALL FILES

24E7  '.SYSTEM'

24EE  ---------END OF SYSTEM FILE LOADER-------------

24EE  -----------PAGE 3 VECTOR IMAGE-----------------

24ED $3DE-$3FF

(INCLUDES A ROUTINE AT $306 THAT IS USED
TO CALL A DEVICE DRIVER IN AUX HIGH RAM)

24FE FROM MAIN 2-PAGE, (C008)
24F1 GET X+1 VALUES STARTING AT $42
24F3 AND PUT IN AUX 2-PAGE (C009)
24F6 AT SAME LOCATION.

24FB "NO DEVICE CONNECTED" ERROR (IF NOT INSTALLED)
24FE BACK TO MAIN 2-PAGE (C008)
2501 RETURN

2502 ADDRESS OF MLI ROUTINE THAT CALLS
THE RAM-BASED DEIVCE DRIVER. THIS
ADDRESS IS USED AS THE INSTALLED
DRIVER'S ADDRESS IN THE GLOBAL PAGE.

250B BRK HANDLER AT $FA59
250A RESET AT $FF59
250C POWER UP BYTE
250D & VECTOR TO $FF59 >>FF59
2510 CTRL-X VECTOR TO $FF59 >>FF59
2513 NMI VECTOR TO $FF59 >>FF59
2516 IRQ HANDLER AT $8FEB (GLOBAL PAGE)

2518 ***** SET HIGH RAM FOR READ/WRITE ***************

2518 TWO READS OF $C888 (C888)
251E RETURN

251F ***** DETERMINE MACHINE ID *********************

$0C=6B, 0...APPLE II
01...0...APPLE II+
10...0...APPLE IIe or IIgs
10...1...APPLE IIc
11...0...APPLE /// EMULAT.
10...48K RAM
10...64K RAM
11...128K RAM
11...1 80 COL CARD
11...1 COMPATIBLE CLOCK

251F ASSUME NOTHING AT FIRST
2533 GET A ROM BYTE (F883)
2526 APPLE 117
252B YES, SET BIT >>254B
252A NO,
252C APPLE IIe OR IIgs?
252E YES, SET BIT >>254B
2530 NO,
2532 APPLE II+?
2534 NO, STRANGE ROMH >>2545
2539 REALLY A II+?
253B YES >>254B
253F /// EMULATION MODE?
2543 ---
2544 RETURN
2545 OTHERWISE, UNKNOWN MACHINE
2547 CREATE INVALID INSTR AT $80
2549 AND GO THERE >>2568
254B UPDATE MACHID
254D READ/WRITE ENABLE HIGH RAM (BANK1) <251B>
2552 SEE IF HIGH RAM EXISTS (D000)
2564 IF PRESENT, MARK IN MACHID

2568 ***** LOOK FOR AUXILIARY RAM ***************

(CODE MOVED TO $80 TO ALLOW BANK SWITCH)
(ENTERED WITH MACHID IN ACCUMULATOR)

256B UPDATE MACHID
256D IF PLUS, IS II OR II+ >>254A
256F IT'S A IIe OR LATER. CHECK FOR 128K.
2571 BANK TO AUX MEMORY (EO05)
2577 STORE A PATTERN AT $C000 (0C00)
257A AND AT $800 (0800)
ProDOS Relocator -- VI.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

2580 MAKE SURE PATTERN STAYS THERE
2582 IT DIDN'T!! >>2592
2584 NOW SHIFT $C00 TO THE LEFT ($C00)
2586 AND SHIFT THE ACCUM TO THE LEFT
2588 ARE THEY STILL THE SAME? ($C00)
258B NO, AUX RAM NOT THERE. >>2592
258D DID $800 MOVE TOO? ($000)
2590 NO, SO WE HAVE FULL 128K >>2595
2592 DON'T HAVE 128K
2595 ---
2596 BANK BACK TO MAIN MEMORY ($C004)
259C ONLY 64K >>25A4
25A0 NO, INDICATE 128K
25A2 IN MACHID
25A4 SET UP SA/B --> "APPLE II"
25A6 IN MOTHERBOARD ROM AT $FB09
25A9 $B NOW $FB
25AB SOMETHING WRONG >>25AF
25AD $A NOW $89
25AF ---
25B0 RETURN TO CALLER

25B1 ********** DISPLAY LOAD MESSAGE ****************************

25B1 CLICK SPEAKER ($C030)
25B4 STORE IN MAIN MEMORY ($C00C)
25B7 $B COL DISPLAY OFF ($C008)
25BD CALL MONITOR INITIALIZATION <FB2F>
25C0 SET VIDEO PR0 <F93>
25C1 SET KEYBD INH <F89>
25C6 OUT OF DECIMAL MODE
25C7 CLEAR SCREEN <$C58>
25CC PRINT "APPLE /" (25FA)
25D5 PRINT "PRODOS 8 V1.2 ETC. ON ROW 12 (2602)
25E2 PRINT 12 BLANKS ON ROW 14 (2620)
25ED PRINT "COPYRIGHT" ETC. ON ROW 24 (262C)
25F0 CLICK SPEAKER AGAIN ($C030)
25F9 DONE

25FA ********** DATA AREA ****************************

25FA 'APPLE /'
2602 'PRODOS 8 V1.2 06-SEP-86'
2620 'COPYRIGHT APPLE COMPUTER, INC., 1983-86'

2653 8 BYTES FOR SMARTPORT STATUS CALL
265B DRIVER ADDRESS
265D SPACES LEFT ON DEVICE LIST
265E SLOT 2 FLAG (0 = PRODOS STORAGE DEVICE IN SLOT 2)

265F ********** DETERMINE SLOT CONFIGURATION ****************************

265F ---
2661 ZERO SOME THINGS
2666 DEVCNT=$FF (NO DEVICES YET) (BF31)
2668 ALL 14 DEVICES ARE UNASSIGNED
2673 FIRST CHECK SLOT 2
2674 IS A STORAGE DEVICE IN SLOT 27 <2898>
2677 IF NOT, SET A FLAG (265E)
267A NOW POINT TO SLOT 7
267E STORAGE DEVICE IN SLOT? <2898>
2681 NO. >>26DF
2683 GET $CSFF BYTE
2685 LOOKS LIKE 16 SECTOR DISK II. >>26AC
2687 LOOK LIKE 13 SECTOR DISK II?
2689 YES, DON'T USE IT. >>26DF

***** NON-DISK II STORAGE DEVICE *****

2688 CSFF BYTE = LOW BYTE OF DEVICE ADDRESS (265B)
26BE CHECK BYTE AT OFFSET 7
2690 TO SEE IF IT'S A SMARTPORT
2692 NOT A SMARTPORT INTERFACE >>2697
2694 DO SMARTPORT STUFF >>2844
2697 ---
2699 GET $SFFE (STATUS BYTE)
269B CAN WE AT LEAST READ STATUS AND DATA?
269F ANTICIPATE FAILURE
26A0 CAN'T READ IT. NO SENSE USING IT. >>26DF
26A2 PUT LEFTNibBLE OF STATUS BYTE IN $12 <286D>
26A6 PUSH CLC, INDICATING ONE DRIVE
26A7 CARRY SET IF 2 OR 4 DRIVES
26A8 GET HIGH BYTE OF SLOT ROM
26AA ALWAYS BRANCH INTO DISK II PROCESSING >>26B9

***** LOOKS LIKE A DISK II **********

26AC $12 = 0 FOR DISK II
26AF PUSH SEC ON STACK (DISK II HAS 2 DRIVES)
26B8 GET LOW BYTE OF DISK II DRIVER ($00) (27DB)
26B3 SAVE IT IN RELOC DATA AREA (265B)
26B6 GET HIGH BYTE OF DISK II DRIVER ($00) (27DB)
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.2 -- 6 SEP 86

27D8 ********** DATA AREA ***********************

270A DISK II DEVICE DRIVER ENTRY POINT
270C +0,+2,+4,+6 = THUNDERCLOCK
270E +1,+3,+5,+7 = DISK
2700 (+7 NOT CHECKED)

27E2 BIT POSITION TABLE FOR SLOTS
27E5 (ALSO USED IN CHECKSUM CALCULS)

27EA ********** COMPUTE AUTOSTART RAM CHECKSUM ***********************

27EB --
27EE GET ZERO IN INDEX REGISTER (27E2)
27FC POINT TO $FB99 ("APPLE II" IN ROM)
27FE MAKE SURE UPPER CASE
27FF UPDATE CHECKSUM (27E2)
27F3 PUT HIGH BIT IN CARRY (27E2)
27F2 DO 8 BYTES IN ALL (27E5)
2801 ACCUM = $00
2805 ACCUM = $00
2807 TURN ON HIGH BIT (27E2)
280A ADD A FUDGE FACTOR
280C ON W1 A CLONE! >>2811
280E PASSED THE TEST...RETURN WITH MACHID
2810 RETURN
2811 ELSE, RETURN WITH ZERO MACHID
2813 RETURN

2814 ***DEVICE DRIVER IN GLOBAL PAGE ***********************

2814 SAVE CARRY (NUMBER OF DRIVES)
2815 GET HIGH BYTE OF SLOT ADDRESS
2817 MAKE IT SLOT NUMBER
2819 TIMES 2
281A USE LATER IN Y-REG
281B NOW GET SLOT*16 IN ACCUM
281D NOW HAVE 00SSS000 (DRIVE 1)
281E PUT DEVICE ID ON DEVICE LIST <27D5>
2821 GET BACK CARRY (NUMBER OF DRIVES)
2822 ROLL CARRY INTO ACCUM
2823 ONLY ONE DRIVE. >>2829
2825 TWO DRIVES. BUMP DEVICE COUNT
2826 AND PUT SECOND DRIVE ON SEARCH LIST (BF32)
2828 STORE FINAL DEVICE COUNT (BF31)
282C SHIFT DRIVE INDICATOR BACK TO CARRY
282D GET LOW BYTE OF DEVICE DRIVER ADDRESS (265B)
2830 PUT IN GLOBAL PAGE FOR DRIVE 1 (BF10)

ProDOS Relocator -- V1.2 -- 6 SEP 86

2833 ONLY ONE DRIVE >>283B
2835 PUT IN GLOBAL PAGE FOR DRIVE 2 (BF20)
2838 GET HIGH BYTE OF DEVICE DRIVER ADDRESS (265C)
283B PUT IN GLOBAL PAGE FOR DRIVE 1 (BF11)
283E ONLY ONE DRIVE >>2843
2840 PUT IN GLOBAL PAGE FOR DRIVE 2 (BF21)
2843 RETURN

2844 ********** HANDLE SMART PORT ***********************

2844 PUT LEFT NIBBLE OF STATUS BYTE IN $12 (<288D)
2847 GET HIGH BYTE OF SLOT ROM ADDRESS
2849 STORE IT IN RELOC DATA AREA (265C)
284C GET PRODOS ENTRY, LOW BYTE (2658)
2850 ADD THREE TO GET SMARTPORT ENTRY
2852 POKE INTO SMARTPORT CALL (285C)
2858 POKE THE HIGH BYTE, TOO. (285D)
285B SELF MODIFIED TO CALL THE SMARTPORT (<0000)
285E WITH A STATUS COMMAND.
285F PARMLIST AT $28AB
2861 GET NUMBER OF DEVICES ON LINE (2653)
2864 NONE ON LINE! >>288A
2866 INDICATE IF DRIVE 2 EXISTS.
2868 PUT DRIVER ADDRESS IN GLOBAL PAGE <2814>
286D IS THIS SLOT 5?
286F NO. >>28BA
2871 SLOT 2 BEING USED BY STORAGE DEVICE? (265E)
2874 YES, TWO DRIVES IS ALL YOU GET! >>288A
2876 GET NUMBER OF DEVICES AGAIN (2653)
2879 MORE THAN TWO DRIVES?
287B NO. >>288A
287D SET CARRY IF DRIVE 4 EXISTS.
287F PUT THEM IN SLOT 2
2881 PUT DRIVER ADDRESS IN GLOBAL PAGE <2814>
288A GO PROCESS NEXT SLOT >>26DE

288D ********** CONVERT STATUS FOR ID BYTE ***********************

288D GET STATUS BYTE
2891 SHIFT LEFT NIBBLE TO RIGHT NIBBLE
2895 PUT IT IN $12
2897 RETURN

289F ********** CHECK FOR PRODOS STORAGE DEVICE ***********************

289B RESET I/O CARD ROMS (CFFF)
289D CHECK 3 BYTES ON CONTROLLER ROM
28A2 ANTICIPATE FAILURE
28A3 NOT A PRODOS STORAGE DEVICE >>28AA
28AA SUCCESS--THIS IS A PRODOS STORAGE DEVICE.
28AA RETURN
**28AB ****** COMMAND LIST FOR SMARTPORT CALL ***********************

28AB 3 PARAMETERS
28AC OVERALL STATUS CALL
28AD PUT STATUS DATA AT $2653
28AF STATUS CODE IS $00

**28B0 ********** RELOCATION ROUTINE ***********************

(X/Y REGS CONTAIN TABLE ADDR)

28B0 SAVE PASSED TABLE ADDRESS
28B4 ACCESS IIGS STATEREG BYTE TO (C068)
28B7 TURN OFF SLOT ROM, ENSURE ROM BANK 0
28BC ---
28BE GET OPERATION CODE
28C0 VALID OPERATION? (4 OR LESS)
28C2 NO, ERROR >>2936
28C6 $14/15 --> OUTPUT BLOCK
28D0 $16/17 --> LENGTH
28D9 NEGATIVE LENGTH? >>2938
28DB CHECK OPERATION CODE
28DC ZERO BLOCK? >>2941
28DF NO, $12/13 = $18/19 --> INPUT BLOCK
28E9 $1A/1B --> END OF INPUT BLOCK
28F6 COPY BLOCK ONLY? >>2965
28F8 SAVE RELOCATION OPERATION CODE (2A8F)
28FE SAVE NUMBER OF RANGES TO CHECK (2A90)
2902 ---
2903 COPY START PAGES TO TABLE
290E ---
290F AND END PAGES
291A ---
291B AND FINALLY, RELOCATION FACTORS
2923 BUMP TO NEXT TABLE ENTRY <296B>
2926 RESTORE OPERATION CODE (2A8F)
292B RELOCATE INSTRUCTIONS? >>2933

**292D ********** 2/3 - RELOCATE ADDRESSES ***********************

292D NO, RELOCATE ADDRESS <29CD>
2930 COPY BLOCK <2976>
2933 AND CONTINUE IF ALL WENT WELL >>28BC
2936 NORMAL EXIT
2937 RETURN
2938 JUMP TO ERROR EXIT >>2A03

**2938 ******* 4 - RELOCATE INSTRUCTIONS ***********************

2938 RELOCATE INSTRUCTIONS <29DF>
293E AND THEN COPY BLOCK >>2938

**2941 ******* 0 - ZERO BLOCK ***********************

2941 BUMP TABLE POINTER TO NEXT ENTRY <296B>
2946 GET NUMBER OF PAGES TO DO
2948 NO FULL PAGES? >>2956
294B ZERO AN ENTIRE PAGE
2950 BUMP PAGE POINTER
2952 AND DECREMENT LENGTH
2956 GET LENGTH OF PARTIAL LAST PAGE
2958 NO PARTIAL PAGE? >>2962
295B ZERO PARTIAL PAGE TOO
2962 DONE, GET NEXT TABLE ENTRY >>28BC

**2965 ******* 1 - COPY BLOCK ***********************

2965 BUMP TABLE POINTER <296A>
2968 AND GO COPY BLOCK >>2930

**296B ******* ADVANCE TABLE POINTER ***********************

296B ADD FINAL ENTRY INDEX...
296F TO TABLE ENTRY ADDRESS
2975 RETURN

**2976 ******* COPY BLOCK ***********************

2976 ---
297A INPTR < OUTPTR? >>2987
297C NO, GREATER? >>29AA
297E MSB'S ARE EQUAL, CHECK LSB'S ALSO
2986 EXIT IF EQUAL
2987 INPTR < OUTPTR, COPY LAST PAGES FIRST
298B BUMP BOTH INPTR AND OUTPTR BY...
29BD LENGTH-1 TO POINT AT LAST BYTE
2995 START WITH SHORT LAST PAGE LENGTH
2999 ---
299A COPY BYTES BACKWARDS THROUGH MEMORY
29A1 DROP ADDRESSES AND LENGTH BY 256
29A7 AND CONTINUE UNTIL FINISHED >>2999
29A9 RETURN
29AA INPTR > OUTPTR, COPY PAGES FORWARD
29AC HOW MANY FULL PAGES LEFT?
29AE MONEY >> 29BF
29B0 COPY A FULL PAGE
29B7 AND BUMP ADDRESSES
29BB DECREMENT LENGTH BY 256
29BD AND DO ALL PAGES >> 29B0
29BF GET LENGTH OF LAST PAGE
29C1 EVEN PAGE BOUNDARY? >> 29CC
29C3 NO, COPY SHORT LAST PAGE
29CC RETURN

29CD ********** ADDR/PAGE RELOCATE ********************

29DF ********** INSTRUCTIONS RELOCATE **************

2A03 RETURN WITH POINTER
2A07 EXIT WITH ERROR CODE
2A0A RETURN

2A0B ********** RELOCATE ABSOLUTE ADDRESS ********************

2A0F GET PAGE NUMBER TO CHECK
2A10 GET NUMBER OF RANGES (LESST ONE) (2A0F)
2A11 IS IT PRIOR TO START OF THIS RANGE? (2A11)
2A13 YES? >> 2A1C
2A15 NO, IS IS AFTER END OF RANGE? (2A99)
2A18 NO >> 2A20
2A1C ---
2A1D CHECK EACH RANGE >> 2A16
2A1F RETURN

2A20 ---
2A21 ADD FUDGE FACTOR TO ADDRESS (2A1B)
2A24 AND UPDATE IT
2A26 RETURN

2A27 ********** BUMP POINTER TO NEXT ADDR ******************

2A27 ---
2A2D ADD LENGTH TO POINTER
2A2F CHECK TO SEE IF WE ARE DONE
2A35 ---
2A39 RETURN

2A3A ********** COMPUTE INSTRUCTION LENGTH ***************

2A3A A-REG CONTAINS_OPCODE
2A3B ISOLATE LAST TWO BITS FOR LATER
2A40 USE LAST 6 BITS AS TABLE_INDEX
2A42 GET_BYTE WITH 4 LENGTHS IN IT (2A4F)
2A45 ---
2A46 USING_TOP_TWO_BITS AS_INDEX... >> 2A4C
2A4B SHIFT_DOWN THE PROPER LENGTH
2A4C AND ISOLATE IT IN A-REG
2A4E RETURN

2A4F ********** 6582 OP LENGTH TABLE *******************

2A4F ---
ProDOS Relocator -- V1.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

2A8F ********** RELOCATION DATA ****************************
2A90 RELOCATION CODE (3,2,1)
2A91 NUMBER OF RANGES
2A92 START OF RANGE PAGES
2A93 END OF RANGE PAGES +1
2AA1 ADDITIVE FACTORS

2AA9 ********* 2AA9-2AFF NOT USED ****************************

2AEC

2B00 ********** SET UP RAMDRIVE IN AUX/MEM ********************

2B02 RELOCATE RAMDRIVE CALLER NOW AT.. (2E00)
2B05 TO HIGH RAM AT.. (F800)
2B0D NOW PREPARE TO MOVE
2B0F RAMDRIVE DEVICE DRIVER
2B11 INTO AUX RAM AT $200.
2B14 $3C/$3D --> $200
2B18 $3F/$3F --> $2DFF
2B1D $42/43 --> $200
2B23 COPY MAIN MEM TO AUX MEM
2B24 USE AUXMOVE TO COPY IT <C311>
2B29 SLOT 3, DRIVE 2 DEVICE DRIVER.. (BF20)
2B2C IS AT $FF00
2B31 BUMP DEVICE COUNT (BF31)
2B37 ADD DEVICE TO ONLINE DEVICE LIST
2B3C RETURN

2B3D ********* 2B3D-2BFF NOT USED ****************************

2B7D

2C00 ******** RAMDRIVE (/RAM) DEVICE DRIVER **********************

2C04 SAVE THE $0STORE SETTING (C018)
2C06 FORCE RAM READ/WRITE (C008)
2C08 COPY INPUT PARAMETERS
2C0B TO AUX PAGE 3. (03B)
2C11 FIRST TIME IN OR FORMAT COMMAND? (03C)
2C14 NO, SKIP FORMAT LOGIC >>2C4F

********* FORMAT RAMDRIVE *************

2C16 YES, SAVE BLOCK WANTED
2C18 PAGES $E AND $F ARE ACTUAL DIRECTORY
2C1A ZERO THE DIRECTORY BLOCK <0333>
2C1F COPY VOLUME NAME ($F3, "RAM") (03D2)
2C22 TO VOLUME DIRECTORY BLOCK (0E04)
2C28 LAST BYTE IN VOLUME BITMAP
2C2A IS AN $F (03D1)
2C2D $FF TO ACCUM.
2C30 14 $FF'S TO BITMAP (03C2)
2C36 SET FIRST BITMAP BYTE TO ZERO (03C2)
2C39 COPY 8 BYTES
2C3B OF DIRECTORY DATA (03D6)
2C3E TO VOLUME DIRECTORY BLOCK (0E22)
2C44 WAS THIS A FORMAT COMMAND? (03BC)
2C47 YES, DONE. >>2CAA
2C49 NO, SET FLAG & CONTINUE WITH READ/WRITE (03BC)
2C4C RESTORE BLOCK NUMBER (03C1)

********* READ/WRITE RAMDRIVE BLOCK ******

2C4F CONVERT BLOCK NUMBER TO PAGE NUMBER (03C1)
2C55 THIS PAGE IN HIGH RAM?
2C57 YES >>2C63
2C59 NO, IS IT BLOCK 37 (VOLUME BIT MAP)
2C5B NO >>2C66
2C5D YES, DUMP A PHONY BITMAP BLOCK >>03BC
2C60 ELSE, NORMAL READ/WRITE >>0342

********* READ/WRITE IN AUX HIGH RAM ******

2C63 SAVE PAGE NUMBER
2C64 FIND IT IN MEMORY <02E5>
2C67 REMEMBER READ/WRITE STATUS
2C68 WRITING? >>2C68
2C6A GET SAVED PAGE NUMBER
2C6B DOES OPERATION INVOLVE BANK1?
2C6D NO, USE BANK2 >>2C73
2C6F YES, FORCE IT TO $0XXX
2C71 AND USE BANK1 OF AUX HIGH RAM >>2C79
2C73 USE BANK2 OF AUX HIGH RAM (CB83)
2C76 AND WRITE ENABLE IT (CB83)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2C79</td>
<td>SAVE PAGE NUMBER IN BLOCK (03C1)</td>
</tr>
<tr>
<td>2C7C</td>
<td>PRESERVE HIS BUFFER ADDR (03C0)</td>
</tr>
<tr>
<td>2CB0</td>
<td>DURING THE FOLLOWING TRANSFER... (03BF)</td>
</tr>
<tr>
<td>2CB3</td>
<td>SELECT AUX HIGH RAM (C099)</td>
</tr>
<tr>
<td>2CB8</td>
<td>USE RAMDRIVE BUFFER AS AN &quot;IN BETWEEN&quot; (03C8)</td>
</tr>
<tr>
<td>2CBB</td>
<td>AREA WHEN TRANSFERING TO/FROM AUX HIGH RAM.</td>
</tr>
<tr>
<td>2CBD</td>
<td>PRETEND THAT WAS CALLER'S BUFFER (03BF)</td>
</tr>
<tr>
<td>2CC9</td>
<td>AND SET UP POINTERS AGAIN &lt;02E5&gt;</td>
</tr>
<tr>
<td>2CD4</td>
<td>COPY BLOCK TO OR FROM RAMDRIVE BUFFER</td>
</tr>
<tr>
<td>2CF0</td>
<td>THEN BACK TO MAIN ZERO PAGE (C099)</td>
</tr>
<tr>
<td>2CA2</td>
<td>RESTORE CALLER'S BUFFER ADDRESS (03BF)</td>
</tr>
<tr>
<td>2CA9</td>
<td>READING OR WRITING?</td>
</tr>
<tr>
<td>2CAA</td>
<td>IF WRITING, DONE &gt;&gt;2C65</td>
</tr>
<tr>
<td>2C6C</td>
<td>AND COPY RAMDRIVE BUFFER TO HIS BUFFER &lt;02BE&gt;</td>
</tr>
<tr>
<td>2CB5</td>
<td>THEN EXIT &gt;&gt;03DE</td>
</tr>
<tr>
<td>2CBB</td>
<td>IF WRITING, COPY HIS BLOCK TO RAMDRIVE BUFFER &lt;02BE&gt;</td>
</tr>
<tr>
<td>2CBB</td>
<td>THEN COPY RAMDRIVE BUFFER TO AUX HIGH RAM &gt;&gt;026A</td>
</tr>
<tr>
<td>2CBE</td>
<td>************ COPY BLOCK IN MAIN 48K ************</td>
</tr>
<tr>
<td>2CBF</td>
<td>THIS ENTRY IS FOR THE RAMDRIVE BUFFER</td>
</tr>
<tr>
<td>2CC0</td>
<td>THIS ENTRY ASSUMES AUX MEM PAGE NUMBER IN ACCUM (03C1)</td>
</tr>
<tr>
<td>2CC3</td>
<td>THIS ENTRY ASSUMES PAGE NUMBER ALREADY SET &lt;02E5&gt;</td>
</tr>
<tr>
<td>2C6C</td>
<td>WRITING TO RAMDISK? &gt;&gt;2CDB</td>
</tr>
<tr>
<td>2CBB</td>
<td>NO, WRITE TO MAIN 48K RAM (C094)</td>
</tr>
<tr>
<td>2CCC</td>
<td>COPY BLOCK AUX MEM --&gt; MAIN MEM</td>
</tr>
<tr>
<td>2C77</td>
<td>WRITE TO AUX MEM AGAIN (C005)</td>
</tr>
<tr>
<td>2COD</td>
<td>DONE (RETURN HERE AFTER FOLLOWING JUMP)</td>
</tr>
<tr>
<td>2CDD</td>
<td>GO BACK TO MAIN MEM PART OF DRIVER (03ED)</td>
</tr>
<tr>
<td>2CE0</td>
<td>TO COPY MAIN MEM --&gt; AUX MEM</td>
</tr>
<tr>
<td>2CBE5</td>
<td>********** SET BUFFER AND BLOCK ADDRESSES ***************</td>
</tr>
<tr>
<td>2CE5</td>
<td>GET COMMAND (03BD)</td>
</tr>
<tr>
<td>2CE6</td>
<td>READ OR WRITE?</td>
</tr>
<tr>
<td>2CE9</td>
<td>WRITE? &gt;&gt;2DB8</td>
</tr>
<tr>
<td>2CEB</td>
<td>NO, GET HIGH BYTE OF BUFFER TO BE READ (03C8)</td>
</tr>
<tr>
<td>2CF2</td>
<td>AND LOW BYTE OF BUFFER ADDRES (03BF)</td>
</tr>
<tr>
<td>2CF5</td>
<td>$42/43 --&gt; FIRST PAGE OF BUFFER</td>
</tr>
<tr>
<td>2CF7</td>
<td>$40/41 --&gt; SECOND PAGE OF BUFFER</td>
</tr>
<tr>
<td>2CF9</td>
<td>GET PAGE NUMBER (03C1)</td>
</tr>
<tr>
<td>2CFE</td>
<td>$3C/3D --&gt; BLOCK IN RAMDRIVE</td>
</tr>
<tr>
<td>2D00</td>
<td>$3E/3F --&gt; SECOND PAGE OF SAME</td>
</tr>
<tr>
<td>2D06</td>
<td>ALWAYS BRANCH AROUND WRITE CODE &gt;&gt;2D23</td>
</tr>
<tr>
<td>2D06</td>
<td>WRITE, (03C0)</td>
</tr>
<tr>
<td>2D0F</td>
<td>$3C/3D --&gt; MAIN MEMORY ADDRESS OF BUFFER TO BE WRITTEN (03BF)</td>
</tr>
<tr>
<td>2D12</td>
<td>$3E/3F --&gt; SECOND PAGE OF SAME</td>
</tr>
<tr>
<td>2D19</td>
<td>$42/43 --&gt; BLOCK IN RAMDRIVE</td>
</tr>
<tr>
<td>2D1B</td>
<td>$40/41 --&gt; SECOND PAGE OF SAME</td>
</tr>
<tr>
<td>2D23</td>
<td>SET SECOND PAGE ADDRESSES</td>
</tr>
<tr>
<td>2D27</td>
<td>EXIT</td>
</tr>
<tr>
<td>2D28</td>
<td>********** SEND HIM A DUMMY BLOCK OF ZEROS**********</td>
</tr>
<tr>
<td>2D2B</td>
<td>ZERO RAMDRIVE BUFFER IN CASE READING &lt;0331&gt;</td>
</tr>
<tr>
<td>2D2C</td>
<td>COPY BETWEEN RAMDRIVE BUFFER AND HIS BUFFER &lt;02C3&gt;</td>
</tr>
<tr>
<td>2D2E</td>
<td>AND EXIT &gt;&gt;03DE</td>
</tr>
<tr>
<td>2D31</td>
<td>********** ZERO BLOCK BUFFER **********</td>
</tr>
<tr>
<td>2D31</td>
<td>ZERO RAMDRIVE BUFFER</td>
</tr>
<tr>
<td>2D33</td>
<td>ZERO BLOCK INDICATED BY ACCUM. (03C1)</td>
</tr>
<tr>
<td>2D36</td>
<td>SET UP BUFFER POINTERS &lt;02E5&gt;</td>
</tr>
<tr>
<td>2D3A</td>
<td>ZERO BOTH PAGES OF BLOCK</td>
</tr>
<tr>
<td>2D40</td>
<td>AND EXIT</td>
</tr>
<tr>
<td>2D42</td>
<td>********** READ/WRITE IN LOW 48K **********</td>
</tr>
<tr>
<td>2D42</td>
<td>BLOCK 2 (VOLUME DIRECTORY)?</td>
</tr>
<tr>
<td>2D44</td>
<td>NO &gt;&gt;2D4A</td>
</tr>
<tr>
<td>2D46</td>
<td>YES, CONVERT IT BLOCK 7</td>
</tr>
<tr>
<td>2D4B</td>
<td>AND GO DO 1/0 NOW &gt;&gt;2D5B</td>
</tr>
<tr>
<td>2D4A</td>
<td>LESS THAN BLOCK 8?</td>
</tr>
<tr>
<td>2D4C</td>
<td>YES, RETURN WITH DUMMY ZERO BLOCK. &gt;&gt;2D28</td>
</tr>
<tr>
<td>2D4E</td>
<td>START MSB AT ZERO</td>
</tr>
<tr>
<td>2D50</td>
<td>GET ORIGINAL BLOCK NUMBER</td>
</tr>
<tr>
<td>2D52</td>
<td>BLOCK $5D THROUGH $5F?</td>
</tr>
<tr>
<td>2D54</td>
<td>NO &gt;&gt;2D5B</td>
</tr>
<tr>
<td>2D56</td>
<td>YES, ADJUST TO $D THROUGH $F</td>
</tr>
<tr>
<td>2D5B</td>
<td>AND USE $1A00 THRU $1FF IN RAMDRIVE. &gt;&gt;03B5</td>
</tr>
<tr>
<td>2D5B</td>
<td>ELSE, FOR BLOCKS $8 THRU $5C</td>
</tr>
<tr>
<td>2D5C</td>
<td>SUBTRACT 8</td>
</tr>
<tr>
<td>2D5E</td>
<td>AND DIVIDE BY 17 ($11)</td>
</tr>
<tr>
<td>2D64</td>
<td>XREG IS QUOTIENT</td>
</tr>
<tr>
<td>2D65</td>
<td>HAS TO BRANCH!! &gt;&gt;2D5E</td>
</tr>
<tr>
<td>2D68</td>
<td>AND XREG IS REMAINDER</td>
</tr>
<tr>
<td>2D69</td>
<td>REMAINDER OF 17</td>
</tr>
<tr>
<td>2D6B</td>
<td>NO &gt;&gt;2D73</td>
</tr>
<tr>
<td>2D6D</td>
<td>YES, EVERY 17TH BLOCK GOES</td>
</tr>
<tr>
<td>2D6E</td>
<td>IN $1000-$1BFF AREA</td>
</tr>
<tr>
<td>2D6F</td>
<td>BY ADDING 8 TO QUOTIENT</td>
</tr>
<tr>
<td>2D71</td>
<td>AND DO IT &gt;&gt;2D5B</td>
</tr>
<tr>
<td>2D73</td>
<td>BUMP QUOTIENT (START AT $2XXX)</td>
</tr>
<tr>
<td>2D75</td>
<td>SHIFT IT TO TOP NIBBLE OF BYTE</td>
</tr>
</tbody>
</table>
ProDOS Relocator -- V1.2 -- 6 SEP 86

PRODOS Relocator -- V1.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

2D7D GOT A REMAINDER? >>2D81
2D7F IF SO, DECREMENT IT (NOT USING 1)
2D81 THEN ADD INTO TOP NIBBLE
2D82 TO FORM $10 THRU $5F ($3C1)
2D85 BLOCK*2 FOR PAGE NUMBER
2D86 COPY THE BLOCK ($2CB)
2D89 THEN EXIT >>03DE

2DBC ********** READ/WRITE BITMAP BLOCK **************

2DBC USE RAMDRIVE BUFFER (NO ACTUAL BITMAP BLOCK)
2D91 SET UP BUFFER POINTERS ($2E5)
2D94 WRITING? >>2DA9
2D96 NO, READING: ZERO THE RAMDRIVE BUFFER ($336)
2DB0 COPY BITMAP IMAGE TO RAMDRIVE BUFFER ($3C2)
2DA3 COPY BLOCK BACK TO CALLER'S BUFFER ($2C3)
2D96 THEN EXIT >>03DE

2DA9 WRITING, COPY HIS BUFFER TO RAMDRIVE BUFFER ($2C3)
2DAC SET UP BUFFER POINTERS ($2E5)
2DB0 COPY 16 BITMAP BYTES FROM RAMDRIVE BUFFER
2DB3 INTO PAGE 3 BITMAP IMAGE ($3C2)
2D96 THEN EXIT >>03DE

2DBC ********** RAM DRIVE DATA (AT $3BC) ***************

2DBC FIRST TIME ENTRY FLAG
2DBD COMMAND FROM PARM LIST
2DBE UNIT NUMBER FROM PARM LIST
2DBF BUFFER ADDRESS FROM PARM LIST
2DC1 BLOCK NUMBER FROM PARM LIST

2DC2 BIT MAP IMAGE FOR RAM DRIVE

2DD2 RAMDRIVE VOLUME NAME
2DD3 'RAM'
2DD6 ACCESS, ENTRY LENGTH
2DD8 NUMBER OF ENTRIES
2DD9 FILE COUNT
2DDB BIT MAP BLOCK POINTER
2DDD BLOCKS ON DISK

2DDE ********** EXIT TO MAIN MEMORY *******************

2DE5 RESTORE $36 TO $36
2DE7 $36 WAS ON ($001)
2DEA LOW-ORDER BYTE AND
2DEE HIGH-ORDER BYTE USED BY XFER ROUTINE
2DF0 RETURN TO $FF44 (NORMAL EXIT)

ProDOS Relocator -- V1.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

2DFB USE ROM XFER ROUTINE TO DO IT >>C314
2DFE TWO BYTES NOT USED

2E00 ********** RAMDRIVE CALLER (RUNS AT $FF68) *************

(USED TO CALL MAIN PART OF RAMDRIVE DEVICE
DRIVER WHICH IS AT $2B0 IN AUX MEMORY.
ROUTINE AT $FF65 IS USED TO TRANSFER DATA
FROM MAIN TO AUX MEMORY.)

2E00 ---
2E03 SAVE 2PAGE STUFF I WILL Clobber
2E05 FROM $3C THRU $47 (FF84)
2E0D SAVE $3ED/E THAT XFER ROUTINE WILL Clobber ($3ED)
2E16 COMMAND = STATUS?
2E18 IF SO, SIMPLE EXIT WILL DO >>2E44
2E1A ELSE, TOO BIG A COMMAND NUM?
2E1C IF SO, ERROR >>2E3B
2E1E ELSE, INVERT BITS OF CMD
2E20 AND SAVE IT
2E22 FORMAT? >>2E2C
2E24 NO, CHECK BLOCK NUMBER
2E28 MUST BE <128 FOR RAMDRIVE
2E2C GOING TO $2B0 IN AUX MEMORY

2E33 USE XFER ROUTINE TO GET THERE >>C314

2E38 I/O ERROR RETURN CODE
2E3D EXIT >>2E41
2E3F WRITE PROTECTED RETURN CODE
2E41 ---
2E42 ERROR EXIT >>2E47
2E44 NORMAL EXIT, RETURN CODE IS 0
2E47 ---
2E4B RESTORE ZERO PAGE (FF84)
2E51 AND $3ED/E (FF82)
2E56 HARMLESS INSTRUCTION MAKES SURE $FF58 IS AN RTS ($060)
    NOTE: THERE ARE ONLY THREE TRUE THINGS IN LIFE:
    DEATH, TAXES, AND AN RTS AT $FF58.
2E64 AND EXIT TO CALLER WHEN THRU

2E65 ********** COPY MAIN TO AUX BLOCK *******************

(CALLED FROM AUX MEM HANDLER)

2E65 WRITE IN AUX 48K ($0B5)
2E6A COPY BOTH PAGES OF BLOCK
2E75 WRITE IN MAIN 48K AGAIN ($0B4)
2E7A GO TO $2DA IN AUX MEMORY TO RETURN ($3ED)
2E7F RETURN TO AUX MEM HANDLER AGAIN >>FF33
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2E82</td>
<td>********** DATA AREA</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FF82</td>
<td>2E82 SAVE $3ED,$3EE</td>
</tr>
<tr>
<td>FF83</td>
<td></td>
</tr>
<tr>
<td>FF84</td>
<td>2E84 ZERO PAGE SAVE AREA</td>
</tr>
<tr>
<td>2E90</td>
<td>********** $2E90-$2EFF NOT USED ***********************</td>
</tr>
<tr>
<td></td>
<td>(NOTE: THE AREA FROM $FF90-FF99 IS RESERVED FOR THE RAM CALLER. FROM $FF90 TO $FFFF IS RESERVED FOR THE IRQ HANDLER.)</td>
</tr>
<tr>
<td>2E98</td>
<td>NOT USED</td>
</tr>
<tr>
<td>2F00</td>
<td>********** START OF MLI LOAD IMAGE *******************</td>
</tr>
<tr>
<td>2F00</td>
<td>MLI LOAD IMAGE AT $2F00</td>
</tr>
</tbody>
</table>
THE 1.3 VERSION OF THE PRODOS RELOCATOR IS
INSTRUCTION FOR INSTRUCTION THE SAME FROM
$2600 TO $26F5 AND FROM $2600 TO $26F5.
SOME ADDRESSES IN THESE AREAS CHANGE BECAUSE
THEY ARE ADDRESSES WITHIN THE MODIFIED PORTION
OF THE RELOCATOR OR THE MLI.

ONLY THE MODIFIED PORTION OF THE RELOCATOR
($25B1 TO $2AFF) IS DOCUMENTED HERE FOR VERSION 1.3.
REFER TO THE 1.2 VERSION IN OTHER PARTS OF
THE RELOCATOR.

25B1 ---

25B1 *********** DISPLAY LOAD MESSAGE ***********

25B1 CLICK SPEAKER (C038)
25B4 STORE IN MAIN MEMORY (C08C)
25B7 80 COL DISPLAY OFF (C090)
25BA SET NORMAL VIDEO <FB84>
25BD CALL MONITOR INITIALIZATION <FB2F>
25C0 SET VIDEO PR# <FB93>
25C3 SET KEYBD IN# <FB99>
25C6 OUT OF DECIMAL MODE
25C7 CLEAR SCREEN <FC58>
25CC PRINT "APPLE / " (2605)
25D7 PRINT "PRODOS 8 " ETC. ON ROW 12 (260D)
25E2 PRINT 12 BLANKS ON ROW 14 (262B)
25ED PRINT "COPYRIGHT" ETC. ON ROW 23 (2637)
25F8 PRINT "ALL RIGHTS RESERVED" ON ROW 24 (265E)
2681 CLICK SPEAKER AGAIN (C038)
2684 DONE

2685 ********** DATA AREA **********

2685 'APPLE II'
268D 'PRODOS 8 V1.3 2-DEC-86'
26B2 '
2637 'COPYRIGHT APPLE COMPUTER, INC., 1983-86'
265E 'ALL RIGHTS RESERVED.'
2672 8 BYTES FOR SMARTPORT STATUS CALL
267A DRIVER ADDRESS
267C SPACES LEFT ON DEVICE LIST
267D SLOT 2 FLAG (0 = PRODOS STORAGE DEVICE IN SLOT 2)

267E ********** DETERMINE SLOT CONFIGURATION **********

267E ---
2680 ZERO SOME THINGS
2687 DEVCNT=$FF (NO DEVICES YET) (BF31)
268A ALL 14 DEVICES ARE UNASSIGNED
268F FIRST CHECK SLOT 2
2693 IS A STORAGE DEVICE IN SLOT 2? <28D4>
2696 IF NOT, SET A FLAG (267D)
2699 NOW POINT TO SLOT 7
269D STORAGE DEVICE IN SLOT? <28D4>
26A0 NO. >>26F8
26A2 GET $CSFF BYTE
26A4 LOOKS LIKE 16 SECTOR DISK II. >>26CB
26A6 LOOK LIKE 13 SECTOR DISK II?
26A8 YES, DON'T USE IT. >>26FE

***** NON-DISK II STORAGE DEVICE *****

26AA $AFF BYTE = LOW BYTE OF DEVICE ADDRESS (267A)
26AD CHECK BYTE AT OFFSET 7
26AF TO SEE IF IT'S A SMARTPORT
26B1 NOT A SMARTPORT INTERFACE >>26B6
26B3 GO DO SMARTPORT STUFF >>26E3
26B6 ---
26BB GET $CSFE (STATUS BYTE)
26BA CAN WE AT LEAST READ STATUS AND DATA?
26BF ANTICIPATE FAILURE
26BF CAN'T READ IT. NO SENSE USING IT. >>26FE
26C1 PUT LEFT NIBBLE OF STATUS BYTE IN $12 <<28C9>
26C5 PUSH CLC, INDICATING ONE DRIVE
26C6 CARRY SFT IF 2 OR 4 DRIVES
26C7 GET HIGH BYTE OF SLOT ROM
26C9 ALWAYS BRANCH INTO DISK II PROCESSING >>26DB
ProDOS Relocator -- V1.3 -- 2 DEC 86  NEXT OBJECT ADDR: 26C9
PRODOS RELOCATOR

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>26CB</td>
<td>$12 = 0 FOR DISK II</td>
</tr>
<tr>
<td>26CE</td>
<td>PUSH SEC ON STACK (DISK II HAS 2 DRIVES)</td>
</tr>
<tr>
<td>26CF</td>
<td>GET LOW BYTE OF DISK II DRIVER ($50) (27F7)</td>
</tr>
<tr>
<td>26D2</td>
<td>SAVE IT IN RELOC DATA AREA (267A)</td>
</tr>
<tr>
<td>26D5</td>
<td>GET HIGH BYTE OF DISK II DRIVER ($50) (27F8)</td>
</tr>
</tbody>
</table>

***** COMMON PROCESSING ***************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>26DB</td>
<td>SAVE DEVICE ADDRESS HIGH BYTE (267B)</td>
</tr>
<tr>
<td>26DB</td>
<td>EMBLISHED DEVICE DRIVER IN GLOBAL PAGE (2833)</td>
</tr>
<tr>
<td>26ED</td>
<td>ONLY ONE DRIVE?</td>
</tr>
<tr>
<td>26E7</td>
<td>YES, GO TO NEXT SLOT &gt;&gt;26F9</td>
</tr>
<tr>
<td>26F7</td>
<td>IF TWO DRIVES WERE ASSIGNED, MOVE THEM TO THE BOTTOM OF THE LIST IN REVERSE ORDER</td>
</tr>
<tr>
<td>271D</td>
<td>CARRY IS NOW CLEAR IF A PRODOS STORAGE DEVICE WAS FOUND IN THIS SLOT. OTHERWISE, CARRY IS SET.</td>
</tr>
<tr>
<td>271E</td>
<td>GO MARK SLTBY TO SHOW ROMS IN SLT (2787)</td>
</tr>
<tr>
<td>272A</td>
<td>MOVE DOWN ONE SLOT</td>
</tr>
<tr>
<td>272B</td>
<td>WE'VE DONE ALL SLOTS &gt;&gt;2780</td>
</tr>
<tr>
<td>272C</td>
<td>CHECK NEXT SLOT &gt;&gt;269D</td>
</tr>
<tr>
<td>272D</td>
<td>STASHED ANY DEVICES AT BOTTOM OF LIST? (267C)</td>
</tr>
<tr>
<td>272E</td>
<td>NO. &gt;&gt;2739</td>
</tr>
<tr>
<td>272F</td>
<td>YES, MOVE THEM BACK</td>
</tr>
<tr>
<td>2730</td>
<td>IN REVERSE ORDER.</td>
</tr>
<tr>
<td>2741</td>
<td>DONE WHEN X=Y (267C)</td>
</tr>
<tr>
<td>2787</td>
<td>WE ALREADY FOUND ROM IN THIS SLOT &gt;&gt;2788</td>
</tr>
<tr>
<td>278B</td>
<td>CHECK SIGNATURE ON CARD FOR THUNDERCLOCK</td>
</tr>
<tr>
<td>2790</td>
<td>NOT IT &gt;&gt;27AC</td>
</tr>
<tr>
<td>2796</td>
<td>THUNDERCLOCK, WHICH SLOT?</td>
</tr>
<tr>
<td>2798</td>
<td>SAVE SLOT NUMBER (LESS 1)</td>
</tr>
<tr>
<td>279A</td>
<td>IN CLOCK CODE RELOCATION TABLE (22D1)</td>
</tr>
<tr>
<td>279F</td>
<td>ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF06)</td>
</tr>
<tr>
<td>27A0</td>
<td>NO MACHIDI &gt;&gt;2770</td>
</tr>
<tr>
<td>27A1</td>
<td>INDICATE THAT A CLOCK IS PRESENT</td>
</tr>
<tr>
<td>27A2</td>
<td>AND UPDATE MACHID</td>
</tr>
<tr>
<td>27A3</td>
<td>GO MARK ROM IN THIS SLOT &gt;&gt;2788</td>
</tr>
<tr>
<td>278C</td>
<td>CHECK FOR FASCAL 1.1 PROTOCOL</td>
</tr>
<tr>
<td>27A0</td>
<td>$CAS5 = $387</td>
</tr>
<tr>
<td>27B2</td>
<td>DOESN'T GET TO FIRST BASE &gt;&gt;27D7</td>
</tr>
<tr>
<td>27B8</td>
<td>$CAS7 = $187</td>
</tr>
<tr>
<td>27BA</td>
<td>NO. &gt;&gt;27D7</td>
</tr>
<tr>
<td>27CA</td>
<td>$CASB = $817</td>
</tr>
<tr>
<td>27C2</td>
<td>NO, BAD SIGNATURE &gt;&gt;27D7</td>
</tr>
<tr>
<td>27C7</td>
<td>YES, GET LEFT NIBBLE</td>
</tr>
<tr>
<td>27C9</td>
<td>88 COLUMN CARD?</td>
</tr>
<tr>
<td>27CB</td>
<td>NO, UNKNOWN CARD &gt;&gt;27D7</td>
</tr>
<tr>
<td>27C6</td>
<td>NO MACHIDI &gt;&gt;2770</td>
</tr>
<tr>
<td>27D0</td>
<td>MARK 88 COLUMN CARD PRESENT</td>
</tr>
<tr>
<td>27D3</td>
<td>AND UPDATE MACHID</td>
</tr>
<tr>
<td>27D5</td>
<td>GO MARK ROM IN THIS SLOT &gt;&gt;2788</td>
</tr>
</tbody>
</table>

27D7 | UNKNOWN CARD, CHECK ROM TO |
<p>| 27DB | SEE IF IT WILL HOLD A VALUE |
| 27E1 | FOR SOME TIME. |</p>
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>27E8</td>
<td>WE FOUND ROM IN THIS SLOT</td>
</tr>
<tr>
<td>27EA</td>
<td>CONVERT SLOT NUMBER</td>
</tr>
<tr>
<td>27ED</td>
<td>TO A BIT POSITION (2001)</td>
</tr>
<tr>
<td>27F0</td>
<td>AND OR INTO SLTBYT (BF99)</td>
</tr>
<tr>
<td>27F6</td>
<td>RETURN TO CALLER</td>
</tr>
<tr>
<td>27F7</td>
<td>******************** DATA AREA ********************</td>
</tr>
<tr>
<td>27F7</td>
<td>DISK II DEVICE DRIVER ENTRY POINT</td>
</tr>
<tr>
<td>27F9</td>
<td>DEVICE SIGNATURE FOR:</td>
</tr>
<tr>
<td>27FB</td>
<td>+0,+2,+4,+6 = THUNDERCLOCK</td>
</tr>
<tr>
<td>27FD</td>
<td>+1,+3,+5,+7 = DISK</td>
</tr>
<tr>
<td>27FF</td>
<td>(+7 NOT CHECKED)</td>
</tr>
<tr>
<td>2801</td>
<td>BIT POSITION TABLE FOR SLOTS</td>
</tr>
<tr>
<td>2804</td>
<td>(ALSO USED IN CHECKSUM CALCS)</td>
</tr>
<tr>
<td>2889</td>
<td>******************** COMPUTE AUTOSTART ROM CHECKSUM ********************</td>
</tr>
<tr>
<td>2889</td>
<td>---</td>
</tr>
<tr>
<td>288A</td>
<td>GET ZERO IN INDEX REGISTER (2001)</td>
</tr>
<tr>
<td>288D</td>
<td>POINT TO $FB89 (&quot;APPLE II&quot; IN ROM)</td>
</tr>
<tr>
<td>288F</td>
<td>MAKE SURE UPPER CASE</td>
</tr>
<tr>
<td>2814</td>
<td>UPDATE CHECKSUM (2801)</td>
</tr>
<tr>
<td>2817</td>
<td>PUT HIGH BIT IN CARRY (2001)</td>
</tr>
<tr>
<td>281B</td>
<td>DO B BYTES IN ALL (2804)</td>
</tr>
<tr>
<td>2820</td>
<td>ACCUM = $98</td>
</tr>
<tr>
<td>2824</td>
<td>ACCUM = $98</td>
</tr>
<tr>
<td>2826</td>
<td>TERN ON HIGH BIT (2001)</td>
</tr>
<tr>
<td>2829</td>
<td>ADD A FUDGE FACTOR</td>
</tr>
<tr>
<td>282B</td>
<td>OH NO! A CLONE! &gt;&gt;2830</td>
</tr>
<tr>
<td>282D</td>
<td>PASSED THE TEST...RETURN WITH MACHID</td>
</tr>
<tr>
<td>282F</td>
<td>RETURN</td>
</tr>
<tr>
<td>2830</td>
<td>ELSE, RETURN WITH ZERO MACHID</td>
</tr>
<tr>
<td>2832</td>
<td>RETURN</td>
</tr>
<tr>
<td>2833</td>
<td>******************** DEVICE DRIVER IN GLOBAL PAGE ********************</td>
</tr>
<tr>
<td>2833</td>
<td>SAVE CARRY (NUMBER OF DRIVES)</td>
</tr>
<tr>
<td>2834</td>
<td>GET HIGH BYTE OF SLOT ADDRESS</td>
</tr>
<tr>
<td>2836</td>
<td>MAKE IT SLOT NUMBER</td>
</tr>
<tr>
<td>2838</td>
<td>TIMES 2</td>
</tr>
<tr>
<td>2839</td>
<td>USE LATER IN Y-REG</td>
</tr>
<tr>
<td>283A</td>
<td>NOW GET SLOT*16 IN ACCUM</td>
</tr>
<tr>
<td>283C</td>
<td>NOW HAVE $$SS$$0000 (DRIVE 1)</td>
</tr>
<tr>
<td>283D</td>
<td>PUT DEVICE ID ON DEVICE LIST &lt;277C&gt;</td>
</tr>
<tr>
<td>2840</td>
<td>GET BACK CARRY (NUMBER OF DRIVES)</td>
</tr>
<tr>
<td>2841</td>
<td>ROLL CARRY INTO ACCUM</td>
</tr>
<tr>
<td>2842</td>
<td>ONLY ONE DRIVE. &gt;&gt;2848</td>
</tr>
<tr>
<td>2944</td>
<td>TWO DRIVES, BUMP DEVICE COUNT</td>
</tr>
<tr>
<td>2945</td>
<td>AND PUT SECOND DRIVE ON SEARCH LIST (BF32)</td>
</tr>
<tr>
<td>2948</td>
<td>STORE FINAL DEVICE COUNT (BF31)</td>
</tr>
<tr>
<td>294B</td>
<td>SHIFT DRIVE INDICATOR BACK TO CARRY</td>
</tr>
<tr>
<td>294C</td>
<td>GET LOW BYTE OF DEVICE DRIVER ADDRESS (267A)</td>
</tr>
<tr>
<td>294F</td>
<td>PUT IN GLOBAL PAGE FOR DRIVE 1 (BF10)</td>
</tr>
<tr>
<td>2952</td>
<td>ONLY ONE DRIVE &gt;&gt;2957</td>
</tr>
<tr>
<td>2954</td>
<td>PUT IN GLOBAL PAGE FOR DRIVE 2 (BF20)</td>
</tr>
<tr>
<td>2957</td>
<td>GET HIGH BYTE OF DEVICE DRIVER ADDRESS (267B)</td>
</tr>
<tr>
<td>295A</td>
<td>PUT IN GLOBAL PAGE FOR DRIVE 1 (BF11)</td>
</tr>
<tr>
<td>295D</td>
<td>ONLY ONE DRIVE &gt;&gt;2962</td>
</tr>
<tr>
<td>295F</td>
<td>PUT IN GLOBAL PAGE FOR DRIVE 2 (BF21)</td>
</tr>
<tr>
<td>2962</td>
<td>RETURN</td>
</tr>
<tr>
<td>2963</td>
<td>******************** HANDLE SMART PORT ********************</td>
</tr>
<tr>
<td>2963</td>
<td>PUT LEFT NIBBLE OF STATUS BYTE IN $12 &lt;28C9&gt;</td>
</tr>
<tr>
<td>2966</td>
<td>GET HIGH BYTE OF SLOT ROM ADDRESS</td>
</tr>
<tr>
<td>2968</td>
<td>STORE IT IN RELOC DATA AREA (267B)</td>
</tr>
<tr>
<td>296B</td>
<td>GET PRODOS ENTRY, LOW BYTE (267A)</td>
</tr>
<tr>
<td>296E</td>
<td>POKE INTO PRODOS CALL (2995)</td>
</tr>
<tr>
<td>2972</td>
<td>ADD THREE TO GET SMARTPORT ENTRY</td>
</tr>
<tr>
<td>2974</td>
<td>POKE INTO SMARTPORT CALL (2996)</td>
</tr>
<tr>
<td>297A</td>
<td>POKE IN HIGH BYTE TO SMARTPORT CALL (2999)</td>
</tr>
<tr>
<td>297D</td>
<td>AND TO PRODOS CALL (2996)</td>
</tr>
<tr>
<td>2980</td>
<td>CONVERT HIGH Byte TO UNIT NUMBER</td>
</tr>
<tr>
<td>2984</td>
<td>STORE UNIT NUMBER</td>
</tr>
<tr>
<td>2986</td>
<td>STORE DOS STATUS CALL</td>
</tr>
<tr>
<td>2988</td>
<td>STORE AS COMMAND CODE</td>
</tr>
<tr>
<td>298C</td>
<td>ALSO ZERO BLOCK NUMBER</td>
</tr>
<tr>
<td>2990</td>
<td>SET BUFFER ADDRESS SET TO $1000</td>
</tr>
<tr>
<td>2992</td>
<td>JUST IN CASE IT'S NEEDED.</td>
</tr>
<tr>
<td>2994</td>
<td>SELF-MODIFIED TO CALL PRODOS DEVICE DRIVER. &lt;0000&gt;</td>
</tr>
<tr>
<td>2997</td>
<td>SELF-MODIFIED TO CALL THE SMARTPORT &lt;0000&gt;</td>
</tr>
<tr>
<td>299A</td>
<td>WITH A STATUS COMMAND.</td>
</tr>
<tr>
<td>299B</td>
<td>PARMLST AT $28AB</td>
</tr>
<tr>
<td>299D</td>
<td>GET NUMBER OF DEVICES ON LINE (2672)</td>
</tr>
<tr>
<td>29A0</td>
<td>NONE ON LINE1 &gt;&gt;28C6</td>
</tr>
<tr>
<td>29A2</td>
<td>INDICATE IF DRIVE 2 EXISTS.</td>
</tr>
<tr>
<td>29A4</td>
<td>PUT DRIVER ADDRESS IN GLOBAL PAGE &lt;2833&gt;</td>
</tr>
<tr>
<td>29A9</td>
<td>IS THIS SLOT 57</td>
</tr>
<tr>
<td>29AB</td>
<td>NO. &gt;&gt;28C6</td>
</tr>
<tr>
<td>29AD</td>
<td>SLOT 2 BEING USED BY A STORAGE DEVICE? (267D)</td>
</tr>
<tr>
<td>29B0</td>
<td>YES, TWO DRIVES IS ALL YOU GET! &gt;&gt;28C6</td>
</tr>
<tr>
<td>29B2</td>
<td>GET NUMBER OF DEVICES AGAIN (2672)</td>
</tr>
<tr>
<td>29B5</td>
<td>MORE THAN TWO DRIVES?</td>
</tr>
<tr>
<td>29B7</td>
<td>NO. &gt;&gt;28C6</td>
</tr>
<tr>
<td>29B9</td>
<td>SET CARRY IF DRIVE 4 EXISTS.</td>
</tr>
<tr>
<td>29BB</td>
<td>PUT THEM IN SLOT 2</td>
</tr>
<tr>
<td>29BF</td>
<td>PUT DRIVER ADDRESS IN GLOBAL PAGE &lt;2833&gt;</td>
</tr>
<tr>
<td>29C6</td>
<td>GO PROCESS NEXT SLOT &gt;&gt;26FD</td>
</tr>
</tbody>
</table>
28C9 ******** CONVRT STAT FOR ID BYTE *********************
    28C9 GET STATUS BYTE
    28CD SHIFT LEFT NIBBLE TO RIGHT NIBBLE
    28D1 PUT IT IN $12
    28D3 RETURN

28D4 ******** CHECK FOR PRODOS STORAGE DEVICE **********************
    28D4 RESET I/O CARD ROMS (CFF);
    28D9 CHECK 3 BYTES ON CONTROLLER ROM
    28DE ANTICIPATE FAILURE
    28DF NOT A PRODOS STORAGE DEVICE >>28E6
    28E5 SUCCESS--THIS IS A PRODOS STORAGE DEVICE.
    28E6 RETURN

28E7 ******** COMMAND LIST FOR SMARTPORT CALL ***********************
    28E7 3 PARAMETERS
    28B8 OVERALL STATUS CALL
    28E9 PUT STATUS DATA AT $2653
    28EB STATUS CODE IS $90

28EC ********** RELOCATION ROUTINE *******************************
    (X/Y REGS CONTAIN TABLE ADDR)
    28F0 ACCESS IIGS STATEREG BYTES TO (C968)
    28F3 TURN OFF SLOT ROM, ENSURE ROM BANK $1
    28F8 ---
    28F0 GET OPERATION CODE
    28FC VALID OPERATION? (4 OR LESS)
    28FE NO, ERROR >>2972
    2902 $14/15 --> OUTPUT BLOCK
    290C $16/17 --> LENGTH
    2915 NEGATIVE LENGTH? >>2974
    2917 CHECK OPERATION CODE
    2918 ZERO BLOCK? >>297D
    291B NO, $12/13 = $18/19 --> INPUT BLOCK
    2925 $1A/1B --> END OF INPUT BLOCK
    2922 COPY BLOCK ONLY? >>29A1
    2934 SAVE RELOCATION OPERATION CODE (2AC8)
    293A SAVE NUMBER OF RANGES TO CHECK (2ACC)
    293E ---
    293F COPY START PAGES TO TABLE
    2942 ---
    294B AND END PAGES
    2956 ---
    2957 AND FINALLY, RELOCATION FACTORS
    295F BUMP TO NEXT TABLE ENTRY <29A7>

2962 ********** 2/3 - RELOCATE ADDRESSES ***********************
    2969 NO, RELOCATE ADDRESS <2A99>
    296C COPY BLOCK <29B2>
    2972 BUMP BOTH INPTR AND OUTPTR BY...
    29C9 LENGTH-1 TO POINT AT LAST BYTE
ProDOS Relocator -- V1.3 -- 2 DEC 86  
NEXT OBJECT ADDR: 29D1

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>29D1</td>
<td>START WITH SHORT LAST PAGE LENGTH</td>
</tr>
<tr>
<td>29D5</td>
<td></td>
</tr>
<tr>
<td>29D6</td>
<td>COPY BYTES BACKWARDS THROUGH MEMORY</td>
</tr>
<tr>
<td>29D9</td>
<td>DROP ADDRESSES AND LENGTH BY 256</td>
</tr>
<tr>
<td>29E3</td>
<td>AND CONTINUE UNTIL FINISHED &gt;&gt;29D5</td>
</tr>
<tr>
<td>29E5</td>
<td>RETURN</td>
</tr>
<tr>
<td>29E6</td>
<td>INPTR &gt; OUTPTR, COPY PAGES FORWARD</td>
</tr>
<tr>
<td>29E8</td>
<td>HOW MANY FULL PAGES LEFT?</td>
</tr>
<tr>
<td>29EA</td>
<td>NONE? &gt;&gt;29FB</td>
</tr>
<tr>
<td>29EC</td>
<td>COPY A FULL PAGE</td>
</tr>
<tr>
<td>29F3</td>
<td>AND BUMP ADDRESSES</td>
</tr>
<tr>
<td>29F7</td>
<td>DECREMENT LENGTH BY 256</td>
</tr>
<tr>
<td>29F9</td>
<td>AND DO ALL PAGES &gt;&gt;29EC</td>
</tr>
<tr>
<td>29FB</td>
<td>GET LENGTH OF LAST PAGE</td>
</tr>
<tr>
<td>29FD</td>
<td>EVEN PAGE BOUNDARY? &gt;&gt;2A08</td>
</tr>
<tr>
<td>29FF</td>
<td>NO, COPY SHORT LAST PAGE</td>
</tr>
<tr>
<td>2A08</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

2A09 ******* ADDR/PAGE RELOCATE ********************

2A09  GET TABLE ENTRY TYPE (2ACA)
2A0D  GET PAGE TO RELOCATE
2A0F  RELOCATE A SINGLE ADDRESS <2A47>
2A12  BUMP BY 1 OR 2 BYTES (2ACB)
2A15  ADVANCE POINTER <2A63>
2A18  AND CONTINUE UNTIL COMPLETE >>2A09
2A1A  RETURN

2A1B ******* INSTRUCTIONS RELOCATE ********************

2A1B  ---
2A1D  GET 6502 OPCODE
2A1F  COMPUTE INSTRUCTION LENGTH <2A76>
2A22  INVALID OPCODE? >>2A35
2A24  3 BYTE INSTRUCTIONS?
2A26  NO >>2A2F
2A28  YES, 3 BYTE ADDRESS TO CORRECT
2A2A  RELOCATE ADDRESS <2A47>
2A2D  AND ADVANCE BY 3 BYTES
2A2F  NEXT INSTRUCTION <2A63>
2A32  CONTINUE UNTIL FINISHED >>2A1B
2A34  RETURN

******* INVALID OPCODE *************

2A35  POP THE STACK
2A37  RETURN WITH POINTER TO BAD INSTRUC.
2A3B  DIE HORRIBLY
2A3E  RETURN

ProDOS Relocator -- V1.3 -- 2 DEC 86  
NEXT OBJECT ADDR: 2A3E

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A3F</td>
<td>********** ERROR RETURN *******************</td>
</tr>
<tr>
<td>2A3F</td>
<td>RETURN WITH POINTER</td>
</tr>
<tr>
<td>2A43</td>
<td>EXIT WITH ERROR CODE</td>
</tr>
<tr>
<td>2A46</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

2A47 ******* RELOCATE ABSOLUTE ADDRESS *************

2A47  GET PAGE NUMBER TO CHECK
2A49  GET NUMBER OF RANGES (LESS ONE) (2ACC)
2A4C  IS IT PRIOR TO START OF THIS RANGE? (2ACD)
2A4F  YES? >>2A58
2A51  NO, IS IS AFTER END OF RANGE? (2AD5)
2A54  NO? >>2A5C
2A58  ---
2A59  CHECK EACH RANGE >>2A4C
2A5B  RETURN
2A5C  ---
2A5D  ADD FUDGE FACTOR TO ADDRESS (2ADD)
2A60  AND UPDATE IT
2A62  RETURN

2A63 ******* BUMP POINTER TO NEXT ADDR *************

2A63  ---
2A64  ADD LENGTH TO POINTER
2A6B  CHECK TO SEE IF WE ARE DONE
2A71  ---
2A75  RETURN

2A76 ******* COMPUTE INSTRUCTION LENGTH *************

2A76  A-REG CONTAINS OPCODE
2A77  ISOLATE LAST TWO BITS FOR LATER
2A7C  USE LAST 6 BITS AS TABLE INDEX?
2A7E  GET BYTE WITH 4 LENGTHS IN IT (2A88)
2A81  ---
2A82  USING TOP TWO BITS AS INDEX... >>2A8B
2A84  SHIFT DOWN THE PROPER LENGTH
2A88  AND ISOLATE IT IN A-REG
2A8A  RETURN

2A8B ******* 6502 OP LENGTH TABLE *************

Each byte contains four 2-bit lengths
ProDOS Relocator -- V1.3 -- 2 DEC 86

NEXT OBJECT ADDR: 2A8B

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
</table>

2A8B  ---

2ACB  ********** RELOCATION DATA  **********************

2ACB  RELOCATION CODE (3,2,1)
2ACC  NUMBER OF RANGES
2ACD  START OF RANGE PAGES
2ADD  END OF RANGE PAGES +1
2ADE  ADDITIVE FACTORS

2AE5  ********** 2AE5-2AFF NOT USED  **********************

2AE5  NOT USED

THE REST OF THE RELOCATOR IS IDENTICAL TO VERSION 1.2
**Beneath Apple ProDOS Supplement**

**ProDOS MLI -- V1.2 -- 6 SEP 86**

**NEXT OBJECT ADDR: D700**

**ADRR DESCRIPTION/CONTENTS**

---

**D700** MODULE STARTING ADDRESS

* * *

**PRODOS MACHINE LANGUAGE INTERFACE**

* THIS CODE IS MOVED INTO HIGH RAM ($D000-$DFEF) BY THE PRODOS RELOCATOR.

* IT PERFORMS ALL FILE MANAGEMENT AND SUPPORTS THE HARDWARE IN A DEVICE INDEPENDENT WAY.

* VERSION 1.2 -- 6 SEP 86

* *

---

**D700 ZERO PAGE USAGE**

**0040** Pointer to caller's parmlist

**0041** -- device driver parmlist --

**0042** Command

**0043** Unit Number

**0044** Buffer Pointer

**0045**

**0046** Block Number

**0047**

**0048** I/O Pointer - Index Block or.

**0048** pointer into $F000 work buffer pointer

**0049** caller's pathname buffer pointer

**004A** I/O Pointer - Data Block

**004B**

**004C** I/O Pointer - Data Block

**004D**

**004E** I/O Pointer - Caller's Data or.

**004E** buffer pointer passed in parmlist or.

**004E** old I/O buffer

---

**D700** SCREEN LOCATIONS

**0750** For direct movement of text to screen

**0760**

**0770**

**0780** Slot in use

---

**D700** RELOCATOR VARIABLE

**2278** Flag=1 when running on a 112S

---

**D700** SYSTEM GLOBAL PAGE EQUATES

**BFOO** Jump to MLI entry point

**BF03** JSPARE (Jump to $E8CF, QUIT code)

**BF06** DATETIME vector

**BF09** Jump to System Error

**BFOC** Jump to System Death Handler

**BF0F** System Error number

**BF10** Device Driver address table

**BF30** Slot/Drive last device
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.2 -- 6 SEP 86

ADDRES DESCRIPTION/CONTENTS

---

BP31 Count (-) active devices
BP32 List of active devices by DEVID
BP38 Memory BITMAP for low 48K
BP70 Open file 1 buffer address
BP7E Open file 8 buffer address
BP90 Interrupt handler 1
BP92 Interrupt handler 2
BP94 Interrupt handler 3
BP96 Interrupt handler 4
BP98 A reg save during interrupt
BP99 X reg save during interrupt
BP9A Y reg save during interrupt
BP9B S reg save during interrupt
BP9C P reg save during interrupt
BP9E Interrupt return address
BP9F Date/Time
BP9F File open LEVEL
BP9F Backup bit
BP9F6 Temporary storage
BP9F9 Prefix flag (0 = no prefix)
BP9F8 MLI active flag
BP9FC Last MLI call return address
BP9E MLI X reg savearea
BP9F MLI Y reg savearea
BFA0 High RAM entry/exit routines
BFA3 Interrupt entry/exit routines
BFA4 Bank switch saved state ($D000 byte)

D700 ********** SOFT SWITCHES **************

C00C Reset 8d column mode
C029 IIGS NEWVIDEO register
C051 Set TEXT mode
C053 Set Mixed text/graphics
C05A Display Primary page
C05B Set LORIES graphics mode
C0BA Read/Write RAM 2nd 4K Bank
C0BB Read/Write RAM 1st 4K Bank
C0F8 Reset alternate I/O ROMs

D700 ********** PATHNAME - DATA AREA ***************

| Li | NAME1 | L2 | NAME2 | ... | O

Prefix is at top of buffer such that a negative index may be used to use it, wrapping around to the pathname again.

D700 path name buffer

D800 ********** FILE CONTROL BLOCKS ***************

File Control Block (FCB0) starts here.

D800 Reference Number

THE FOLLOWING 6 BYTES ARE THE FILE ID
D801 Device Number
D802 Dir Block HDR for Dir describing this File
D804 Dir Block containing entry itself
D806 File entry # in this Directory

D807 Storage Type

Flags
1XXX XXXX Index Block Buffer Changed
1XXX XXXX Data Block Buffer Changed
XX1XXX XXXX Unused
XXX1 XXXX Directory entry needs update
XXXX 1XXX Storage Type Changed
XXX1 XXXX Allocate new Master Index Block
XXX1 XXXX Allocate new Sub Index Block
D808 XXXX XXX1 Allocate new Data Block
D809 Access Byte
D80A Newline Character
D80B Buffer Number (REF_NUM * 2)
D80C Master Index/Key Block Number
D80E Current Index Block
D810 Current Data Block
D812 Mark
D815 End of File
D818 Blocks Used
D81A not used
D81B Local
D81C Flag - Write occurred if MSB on
D81D not used
D81F Newline Enable Mask

D820 FCB1 through FCB7

D900 ********** VOLUME CONTROL BLOCKS ***************

Volume Control Block (VCB0) starts here.

D900 Length ($0000000000)
D901 File Name (Max 15)
D910 Unit Number
D911 Files Open Flag (if $FF)
D912 Total Blocks
D914 Blocks Free
D916 Block Number of Vol Dir Key Block
### ProDOS MLI -- V1.2 -- 6 SEP 86

**ProDOS MLI -- V1.2 -- 6 SEP 86**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D918</td>
<td>not used</td>
</tr>
<tr>
<td>D919</td>
<td>not used</td>
</tr>
<tr>
<td>D91A</td>
<td>Bit Map Pointer</td>
</tr>
<tr>
<td></td>
<td>Block offset into multi-block bitmap of</td>
</tr>
<tr>
<td>D91C</td>
<td>next free bit.</td>
</tr>
<tr>
<td>D91E</td>
<td>Count of open files</td>
</tr>
<tr>
<td>D920</td>
<td>VCB1 through VCB7</td>
</tr>
</tbody>
</table>

**DA00 ******** BITMAP BUFFER ***************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA00</td>
<td>Buffer 1st half</td>
</tr>
<tr>
<td>DB00</td>
<td>Buffer 2nd half</td>
</tr>
</tbody>
</table>

**DC00 ******** PRIMARY BUFFER ****************************

(Used for several things. DIRECTORY block offsets are mapped into it below)

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC00</td>
<td>Pointer Fields</td>
</tr>
<tr>
<td></td>
<td>*** DIRECTORY HEADER ***</td>
</tr>
<tr>
<td>DC04</td>
<td>Type/Length (TTTTLLLL)</td>
</tr>
<tr>
<td>DC05</td>
<td>Volume Name (Max 15)</td>
</tr>
<tr>
<td>DC14</td>
<td>Reserved</td>
</tr>
<tr>
<td>DC1C</td>
<td>Creation Datetime</td>
</tr>
<tr>
<td>DC20</td>
<td>Version</td>
</tr>
<tr>
<td>DC21</td>
<td>Min Version</td>
</tr>
<tr>
<td>DC22</td>
<td>Access Byte</td>
</tr>
<tr>
<td>DC23</td>
<td>Entry Length</td>
</tr>
<tr>
<td>DC24</td>
<td>Entries per Block</td>
</tr>
<tr>
<td>DC25</td>
<td>File Count</td>
</tr>
<tr>
<td>DC27</td>
<td>Bitmap Pointer</td>
</tr>
<tr>
<td>DC29</td>
<td>Entry number within parent's block</td>
</tr>
<tr>
<td>DC2A</td>
<td>Total Blocks</td>
</tr>
<tr>
<td>DC2B</td>
<td>Length of entries in parent</td>
</tr>
<tr>
<td>DC2B</td>
<td>(remainder of first page of block)</td>
</tr>
<tr>
<td>DD00</td>
<td>(second page of block)</td>
</tr>
</tbody>
</table>

**DE00 ******** MLI MAIN ENTRY POINT **************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE00</td>
<td>Clear decimal mode</td>
</tr>
<tr>
<td>DE01</td>
<td>Retrieve status byte from stack</td>
</tr>
<tr>
<td>DE02</td>
<td>and store it in global page. (BF96)</td>
</tr>
<tr>
<td>DE05</td>
<td>Save Registers (BF9F)</td>
</tr>
<tr>
<td>DE0B</td>
<td>Set ($48) -&gt; Address of function code -1</td>
</tr>
<tr>
<td>DE0F</td>
<td>Set $MDADR -&gt; True return address</td>
</tr>
<tr>
<td>DE1C</td>
<td>Retrieve status byte, (BF9F)</td>
</tr>
<tr>
<td>DE1F</td>
<td>push it onto the stack,</td>
</tr>
<tr>
<td>DE20</td>
<td>and pull it into status register.</td>
</tr>
<tr>
<td>DE24</td>
<td>Init Global Page System error to Ø (BF6F)</td>
</tr>
<tr>
<td>DE28</td>
<td>Get Function Code</td>
</tr>
<tr>
<td>DE2B</td>
<td>Build hash index into Command Table (X reg)</td>
</tr>
<tr>
<td>DE34</td>
<td>Is this function code valid?</td>
</tr>
<tr>
<td>DE39</td>
<td>No &gt;&gt;DEB1</td>
</tr>
<tr>
<td>DE3C</td>
<td>Set ($48) -&gt; Parameter list</td>
</tr>
<tr>
<td>DE49</td>
<td>Get parameter count required (FD4D)</td>
</tr>
<tr>
<td>DE4C</td>
<td>None? &gt;&gt;DE6A</td>
</tr>
<tr>
<td>DE4E</td>
<td>Is parameter count correct?</td>
</tr>
<tr>
<td>DE50</td>
<td>No &gt;&gt;DEB5</td>
</tr>
<tr>
<td>DE52</td>
<td>Check class of function (FD2D)</td>
</tr>
<tr>
<td>DE55</td>
<td>Quit?</td>
</tr>
<tr>
<td>DE57</td>
<td>yes &gt;&gt;DE67</td>
</tr>
<tr>
<td>DE59</td>
<td>no,</td>
</tr>
<tr>
<td>DE5A</td>
<td>$8X - Calls to I/O Drivers &gt;&gt;DE70</td>
</tr>
<tr>
<td>DE5C</td>
<td>$CX/DX - Non System calls &gt;&gt;DE7B</td>
</tr>
<tr>
<td>DE5E</td>
<td>Else, $4X - Interrupt support</td>
</tr>
<tr>
<td>DE5F</td>
<td>Isolate type (0=ALLOC, 1=DEALLOC, 2=SPECIAL)</td>
</tr>
<tr>
<td>DE61</td>
<td>Call Interrupt Support &lt;DEFD&gt;</td>
</tr>
<tr>
<td>DE64</td>
<td>Then Exit to Caller &gt;&gt;DEB2</td>
</tr>
<tr>
<td>DE67</td>
<td>Go to quit code via global page &gt;&gt;BF03</td>
</tr>
</tbody>
</table>

**DE6A *************** MLI GET TIME CALL ***************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE6A</td>
<td>Call Date/Time driver &lt;BF06&gt;</td>
</tr>
<tr>
<td>DE6D</td>
<td>and exit to caller &gt;&gt;DEB2</td>
</tr>
</tbody>
</table>

**DE70 *************** MLI READ BLOCK CALL ***************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE70</td>
<td>$80 - Read Block</td>
</tr>
<tr>
<td>DE71</td>
<td>$81 - Write Block</td>
</tr>
</tbody>
</table>

**DE7B *************** MLI WRITE BLOCK CALL ***************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE7B</td>
<td>$CX and $DX CALLS</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEB2</td>
<td>Clear Backup</td>
</tr>
<tr>
<td>DE8A</td>
<td>Error occurred?</td>
</tr>
<tr>
<td>DE8D</td>
<td>Save test results</td>
</tr>
<tr>
<td>DB8E</td>
<td>Disable interrupts</td>
</tr>
<tr>
<td>DB8F</td>
<td>Roll out most recent &quot;active&quot; bit (BF9B)</td>
</tr>
<tr>
<td>DE92</td>
<td>Get test results back</td>
</tr>
<tr>
<td>DE93</td>
<td>Store in X reg</td>
</tr>
<tr>
<td>DE94</td>
<td>Set up Return Address on stack (BF9D)</td>
</tr>
<tr>
<td>DE9C</td>
<td>Put test results on stack</td>
</tr>
<tr>
<td>DE9E</td>
<td>Put error code in A reg</td>
</tr>
<tr>
<td>DB9F</td>
<td>Restore X reg (BF9E)</td>
</tr>
<tr>
<td>DEA2</td>
<td>Restore Y reg (BF9I)</td>
</tr>
<tr>
<td>DEA5</td>
<td>Put error code on stack</td>
</tr>
<tr>
<td>DEA6</td>
<td>Get RAM/ROM orientation (BFF4)</td>
</tr>
<tr>
<td>DEA9</td>
<td>Exit via RAM Global Page &gt;&gt;BFA0</td>
</tr>
</tbody>
</table>

DEAC ********** NO DEVICE CONNECTED **************************************

DEAC ---
DEAE Call System Error Handler (Global Page) <BF09>

DEB1 ********** BAD SYSTEM CALL NUMBER *******************************

DEB1 ---
DEB3 Branch always taken >>DEB7

DEB5 ********** BAD PARAMETER COUNT *******************************

DEB5 ---
DEB7 Call System Error Handler <DEE1>
DEBA Exit to Caller >>DEB2

DEBC ********** BLOCK I/O SETUP *******************************

DEBC ---
DEBE Save Old Processor Flags
DEBF Disable Interrupts
DECO Copy Parameters to $43-$47
DECD Save Starting Buffer Page in $4F
DEDE Find last page + 1
DED0 Round up if Buffer not page aligned >>DED3
DEDD Is this Memory already in use? <PC63>
DEDE Yes, then exit with error >>DEE0
DED8 No, do Block I/O <DEE4>
DEDB Error? >>DEE0
DEDD No, then exit normally
DEDF RETURN
DEE0 Error Exit
DEE1 Call System Error Handler <BF09>

DEE4 -********** Block I/O ***********************************************

DEE4 ---
DEE6 Force off unused UNIT bits
DEE9D Put drive number in X reg
DEF1 Put Device Handler Address in Jump Vector (FEBD)
DEFA Exit through Device Handler >>FEBD

DEFD ********** Interrupt Handler ***************************************

DEFD Save Call Type
DEFF Install unclaimed interrupt handler?
DEF1 No, normal ALLOC/DEALLOC >>DF09
DF09 Yes, install a handler for unclaimed interrupts. <FD23>
DF06 Error? >>DF35
DF08 No, done.
DF09 Test bit 0
DF0A l=DEALLOC >>DF38

DF0C ---
DF0E Look for empty slot (BF7E)
DF15 His Address better be non-zero
DF19 Store Address of His routine in Global Page (BF7E)
DF21 And return the position number we used
DF28 Exit
DF29 Skip this Vector
DF2B Last one? 
DF2D No, check another >>DF0E
DF2F Yes, Table Full Error
DF31 Always taken >>DF35
DF33 Bad Parameter Error
DF35 Call System Error Handler <BF09>

DEALOC

DF38 ---
DF3A Get Position Number
DF3C Can't be zero >>DF33
DF40 Or greater than 4 >>DF33
DF43 Make Index into Table from it
DF46 And zero His Vector (BF7E)
DF4D Then Exit
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF4E ************ IRQ Handler ***************************</td>
<td></td>
</tr>
<tr>
<td>DF50</td>
<td>Save a reg from Monitor (BP08)</td>
</tr>
<tr>
<td>DF55</td>
<td>And X,Y,S and P (BP9)</td>
</tr>
<tr>
<td>DF5D</td>
<td>Is this ROM enhanced? (DFP1)</td>
</tr>
<tr>
<td>DF60</td>
<td>Yes, skip three pulls &gt;&gt;DF6E</td>
</tr>
<tr>
<td>DF67</td>
<td>And RTI Address (BP9E)</td>
</tr>
<tr>
<td>DF6E</td>
<td>Replace stack to original condition</td>
</tr>
<tr>
<td>DF72</td>
<td>Save active slot index (DF77)</td>
</tr>
<tr>
<td>DF75</td>
<td>In bottom half of stack?</td>
</tr>
<tr>
<td>DF78</td>
<td>Yes, pop off 16 bytes and save them</td>
</tr>
<tr>
<td>DF7A <strong>---</strong></td>
<td></td>
</tr>
<tr>
<td>DF81</td>
<td>Save $FA - $FF (top of zero page)</td>
</tr>
<tr>
<td>DF83</td>
<td><strong>---</strong></td>
</tr>
<tr>
<td>DF8B</td>
<td>Is there a User Vector #1 (BP81)</td>
</tr>
<tr>
<td>DF8E</td>
<td>No &gt;&gt;DF95</td>
</tr>
<tr>
<td>DF90</td>
<td>Yes, call it &lt;DF93&gt;</td>
</tr>
<tr>
<td>DF93</td>
<td>His interrupt? &gt;&gt;DFBD</td>
</tr>
<tr>
<td>DF95</td>
<td>Is there a User Vector #2 (BP83)</td>
</tr>
<tr>
<td>DF98</td>
<td>No &gt;&gt;DF9F</td>
</tr>
<tr>
<td>DF9A</td>
<td>Yes, call it &lt;DF96&gt;</td>
</tr>
<tr>
<td>DF9D</td>
<td>His interrupt? &gt;&gt;DFBD</td>
</tr>
<tr>
<td>DF9F</td>
<td>Is there a User Vector #3 (BP85)</td>
</tr>
<tr>
<td>DFA2</td>
<td>No &gt;&gt;DFA9</td>
</tr>
<tr>
<td>DFA4</td>
<td>Yes, call it &lt;DF99&gt;</td>
</tr>
<tr>
<td>DFA7</td>
<td>His interrupt? &gt;&gt;DFBD</td>
</tr>
<tr>
<td>DFA9</td>
<td>Is there a User Vector #4 (BP87)</td>
</tr>
<tr>
<td>DFAE</td>
<td>No, didn't find service routine. &gt;&gt;DF83</td>
</tr>
<tr>
<td>DFBA</td>
<td>Yes, call it &lt;DFC&gt;</td>
</tr>
<tr>
<td>DFBD</td>
<td>Call System Death Handler. &lt;BP0C&gt;</td>
</tr>
<tr>
<td>DFBF</td>
<td>Allow 256 tries. (DFP2)</td>
</tr>
<tr>
<td>DF88</td>
<td>then indicate error type 1 and</td>
</tr>
<tr>
<td>DF8A</td>
<td>call System Death Handler. &lt;BP0C&gt;</td>
</tr>
<tr>
<td>DFBD</td>
<td>Interrupt Serviced</td>
</tr>
<tr>
<td>DFBF</td>
<td>Restore zero page (DFBD)</td>
</tr>
<tr>
<td>DFCC</td>
<td>And stack (BP98)</td>
</tr>
<tr>
<td>DFDA</td>
<td>Is this enhanced ROM? (DFP1)</td>
</tr>
<tr>
<td>DFDD</td>
<td>Yes, skip some stuff we used to have to do &gt;&gt;DF8E</td>
</tr>
<tr>
<td>DFDC</td>
<td>Reload X and Y (BP8A)</td>
</tr>
<tr>
<td>DFDE</td>
<td>Disable I/O ROMS (CPFP)</td>
</tr>
<tr>
<td>DFDE</td>
<td>Replace active slot number (Clw8)</td>
</tr>
<tr>
<td>DF8E</td>
<td>Exit from Interrupt &gt;&gt;BP0D</td>
</tr>
<tr>
<td>DFF1</td>
<td>ENHANCE FLAG. Set to 1 by RELOCATOR if new type ROM found. (That is, if ROM IRQ Vector jumps below SD090)</td>
</tr>
<tr>
<td>DFF2</td>
<td>Unclaimed IRQ Count. Incremented when an interrupt is unclaimed (256 tries are allowed).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFF3</td>
<td>User Interrupt Handler #1 &gt;&gt;DF88</td>
</tr>
<tr>
<td>DFF6</td>
<td>User Interrupt Handler #2 &gt;&gt;BP82</td>
</tr>
<tr>
<td>DFF9</td>
<td>User Interrupt Handler #3 &gt;&gt;BP84</td>
</tr>
<tr>
<td>DFFC</td>
<td>User Interrupt Handler #4 &gt;&gt;BP86</td>
</tr>
<tr>
<td>DFFF ************ SYSTEM ERROR HANDLER ***************************</td>
<td></td>
</tr>
<tr>
<td>DFFP</td>
<td>Save Error Code (BP0F)</td>
</tr>
<tr>
<td>E003</td>
<td>Pop out of subroutine</td>
</tr>
<tr>
<td>E004</td>
<td>Exit to caller with Error Code (BP0F)</td>
</tr>
<tr>
<td>E006</td>
<td>RETURN</td>
</tr>
<tr>
<td>E009 ************ SYSTEM DEATH HANDLER ***************************</td>
<td></td>
</tr>
<tr>
<td>E009</td>
<td>Save Error number in X-REG</td>
</tr>
<tr>
<td>E00A</td>
<td>Turn off 80 column card (ClwC)</td>
</tr>
<tr>
<td>E00D</td>
<td>Select standard Text display (Clw5)</td>
</tr>
<tr>
<td>E010</td>
<td>Are we running on a IIGS? (2278)</td>
</tr>
<tr>
<td>E013</td>
<td>No. &gt;&gt;B1A</td>
</tr>
<tr>
<td>E015</td>
<td>Yes, initialize IIGS video</td>
</tr>
<tr>
<td>E018</td>
<td>by clearing NEWVIDEO. (Clw9)</td>
</tr>
<tr>
<td>E01F <strong>---</strong></td>
<td></td>
</tr>
<tr>
<td>E021</td>
<td>Blank next to last row of screen and (0750)</td>
</tr>
<tr>
<td>E024</td>
<td>print &quot;INSERT SYSTEM DISK AND RESTART&quot; (DFB6)</td>
</tr>
<tr>
<td>E027</td>
<td>on bottom row of screen. (07D8)</td>
</tr>
<tr>
<td>E02D</td>
<td>Get error number back</td>
</tr>
<tr>
<td>E02E</td>
<td>Expect errors in range 00 to 0F</td>
</tr>
<tr>
<td>E030</td>
<td>Make it ASI</td>
</tr>
<tr>
<td>E038</td>
<td>Put error number on screen (07F7)</td>
</tr>
<tr>
<td>E038</td>
<td>Infinite loop &gt;&gt;E038</td>
</tr>
<tr>
<td>E03E ************ PERFORM FILING OR *************** *** HOUSEKEEPING FUNCTIONS ***</td>
<td></td>
</tr>
<tr>
<td>E03E</td>
<td>Save function index (FE7F)</td>
</tr>
<tr>
<td>E041</td>
<td>Get INFO flags for this command (FD95)</td>
</tr>
<tr>
<td>E044</td>
<td>Times 2</td>
</tr>
<tr>
<td>E045</td>
<td>Store Command Number times 2 (FE7B)</td>
</tr>
<tr>
<td>E04A</td>
<td>And use it to index into Address Table</td>
</tr>
<tr>
<td>E04E</td>
<td>Set up Jump Vector with this function's (FEBD)</td>
</tr>
<tr>
<td>E051</td>
<td>.handler address (F06E)</td>
</tr>
<tr>
<td>E057</td>
<td>Signal Backup required after call</td>
</tr>
<tr>
<td>E05C</td>
<td>PATHNAME not required? &gt;&gt;E063</td>
</tr>
<tr>
<td>E05E</td>
<td>Required - parse and validity check &lt;E081</td>
</tr>
<tr>
<td>E061</td>
<td>Bad Name? &gt;&gt;E07A</td>
</tr>
<tr>
<td>E063</td>
<td>Reference Number in list? (FE7B)</td>
</tr>
<tr>
<td>E066</td>
<td>No &gt;&gt;E06D</td>
</tr>
<tr>
<td>E06d</td>
<td>Yes - check it out &lt;E1C7</td>
</tr>
<tr>
<td>E068</td>
<td>Bad Number? &gt;&gt;E07A</td>
</tr>
<tr>
<td>E06D</td>
<td>Date/Time in list? (FE7B)</td>
</tr>
</tbody>
</table>
**Pro DOS MLI -- V1.2 -- 6 SEP 86**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
E070 | No >>E075
E072 | Yes - set System date just in case <8F06>
E075 | Call Function Handler <E07E>
E076 | If no errors then exit >>E07D
E077A | Else - call System error handler <BF09>
E07D | Return to caller
E07E | Indirect JUMP to Handler >>FEBD

**E081**

******* CHECK CALLER'S PATHNAME *******

******* COPY TO MY AREA *******

E081 | Set ($48) -> Pathname
E086 | ---
E090 | Assume partial Pathname (FE84)
E093 | No Pathname in my area yet (D700)
E096 | Check length of caller's Pathname
E098 | Zero is no good >>E0F2
E09C | Nor is 65 or more >>E0F2
E09E | Save length (FE66)
E0A1 | Length + 1 (FE66)
E0A5 | Get first character of his name
E0A9 | Is it "/"?
E0AB | No >>E0B1
E0AD | Yes - indicate fully qualified name (FE84)
E0B0 | Bump past "/
E0B1 | ---
E0B3 | Length of Index level is -1 initially (D700)
E0B6 | First character of Index level (counter) (FE80)
E0B9 | Start of upcoming Index level in name (FE82)
E0BC | At end of name yet? (FE66)
E0BF | Yes >>E0F6
E0C1 | No - get next character in his name
E0C7 | Is it "/"?
E0C9 | Yes >>E10B
E0CB | No - lower case?
E0CD | No >>E0D1
E0CF | Yes - force upper case
E0D1 | Copy to my Pathname buffer (D700)
E0D4 | Increment Index level counter (FE80)
E0D7 | Subsequent characters may be A-Z,0-9 or . >>E0DE
E0D9 | Increment Index level counter (FE80)
E0DC | First character must be alphabetic >>E0EA
E0DE | Is it "/"?
E0E0 | Yes - get next character >>E0BC
E0E2 | No - is it special or control character
E0E4 | Yes - Bad Pathname then >>E0F2
E0E6 | Is it numeric?
E0E8 | Yes - get next character >>E0BC
E0EA | Is it Alphabetic?
E0F0 | If so get next character >>E0BC
E0F2 | Else

---

**E0F3** | Bad Pathname
**E0F5** | RETURN
**E0F6** | ---
**E0F8** | Any characters in last Index level? (FE80)
**E0F8** | Yes >>E101
**E0FD** | No, zero characters in it (FE80)
**E100** | And toss out last "/".
**E101** | ---
**E102** | Mark end of name with $80 (D700)
**E105** | Name too long? >>E0F2
**E107** | No - save final length (FE66)
**E10A** | Set X -> 0
**E10E** | Last Index more than 15 characters?
**E110** | Yes - then no good >>E0F2
**E112** | Save output Index (FE85)
**E115** | Store length of previous Index level (FE82)
**E118** | Just before it in buffer (D700)
**E11B** | Restore output index (FE85)
**E11E** | And continue >>E0B1
**E120** | End of Name
**E121** | Fully qualified name? (FE84)
**E124** | Yes >>E125
**E126** | No - Got a Prefix (BF9A)
**E129** | No - error >>E0F2
**E12B** | Else, okay to exit

---

**E12C**

****************************************************

******* MLI SET PREFIX CALL *******

****************************************************

**E12C** | Copy Pathname <E0B1>
**E12F** | It's okay >>E13B
**E131** | Check length of Volume name (D700)
**E136** | If zero - no Prefix wanted (BF9A)
**E139** | Exit with no error
**E13A** | RETURN

**E13B** | Get File entry for last Index <E593>
**E13E** | Okay? >>E144
**E140** | Invalid Pathname?
**E142** | No - Out now! >>E132
**E144** | Sub Directory file? (FE27)
**E148** | No, error >>E130
**E14D** | Fully qualified path? (FE84)
**E150** | Yes >>E155
**E152** | No - use old Prefix also (BF9A)
**E155** | ---
**E157** | Compute new Prefix Index (FE66)
**E15A** | Does new Prefix exceed 64 characters?
**E15C** | Yes - Bad Path error >>E0F2
**E15F** | Store new Prefix pointer (BF9A)
ProDOS MLI -- VI.2 -- 6 SEP 86
NEXT OBJECT ADDR: E165

ADDR DESCRIPTION/CONTENTS

E165 Set Device Number of Prefix Directory (F567)
E16B Save Keyblock for Prefix Directory (F668)
E174 Copy Prefix to top of Path buffer (D708)
E177 (preceded by old Prefix if one exists) (D708)
E17F Exit normally
E180 Bad File Type Error
E182 ---
E183 RETURN

E184 ******* MLI GET PREFIX CALL *******

****** MLI ONLINE CALL *******

E184 Set ($4E) -> Data Buffer
E190 Set Length = 64 (max)
E19A Validity check buffer storage <PC46>
E19D Error? >>E182
E1A1 Get Prefix index (F9A)
E1A5 No Prefix? - Length = 0 >>E1AB
E1A7 Complement for length
E1AB Store in first byte of buffer
E1AD If null Prefix exit >>E1C5
E1AF ---
E1B0 Copy Prefix to caller's buffer replacing (D708)
E1B3 index level name length bytes with "/
E1BD ---
E1C1 End it with a "/
E1C5 ---
E1C6 Exit normally

E1C7 *********** VALIDITY CHECK REFERENCE NUMBER ***********
(PASSED BY CALLER)

E1C7 Get Reference Number
E1CB If zero then no good >>E228
E1CF If > 8 then no good >>E228
E1D1 Save Reference Number
E1D2 Multiply by 32
E1D8 Result gives offset into PCB's (FE5A)
E1DC Get back Reference Number
E1DD File Control Block active this Reference? (D808)
E1E0 No - Bad Reference Number >>E223
E1E2 Get Buffer Number (D80B)
E1E5 Find Buffer address in Global Page <PC00>
E1EB No Buffer? >>E214
E1ED Buffer okay, save Page Pointer in $48
E1F1 Second block in $49
E1F3 Set last Device used in Global Page (D801)
E1F9 Finish setting up pointers (FE5A)
E1FC ($4A) -> 1st Block of Buffer (data)

ProDOS MLI -- VI.2 -- 6 SEP 86
NEXT OBJECT ADDR: E1FE

ADDR DESCRIPTION/CONTENTS

E1FE ($48) -> 2nd Block of Buffer (index)
E200 ---
E201 Search all Volume Control Blocks (D910)
E204 for the one which goes with requested unit (D801)
E209 ---
E20F Can't find matching Volume Control Block
E211 So die with error type $A <BF0C>
E214 No Buffer in open File Control Block
E216 So die with error type $0B <BF0C>
E219 Is Volume mounted? (D900)
E21C No, keep looking >>E209
E21E Save Volume Control Block index (FE59)
E222 Exit normally

E223 ---
E225 This looks wrong!!! (FE5A)
E228 Bad Reference Number error
E22B RETURN

E22C ******* MLI ONLINE CALL *******

E22C Set ($4E) -> Data Buffer <F1F5>
E22F Set Length = 0
E239 Get Unit Number
E23B Do all Units? >>E244
E23D No, just one
E23F Set length = 16 (FE52)
E242 Always taken >>E249
E244 If all Units
E246 Set Length = 256 (maximum) (FE53)
E249 Is Buffer in main RAM? <PC46>
E24C No, then exit >>E2B1
E24E Yes, zero out Buffer
E253 ---
E258 Index into Data Buffer = $00 (FE62)
E25D Get Unit Number again
E25F Isolate valid bits
E261 Specific Unit requested? >>E282
E263 No, copy Device List from Global Page <EB47>
E266 Save Device Count (FE85)
E269 Get next Device (FE92)
E26C Generate return data for it <E282>
E26F Bump data buffer index by 16 (FE62)
E278 Get next Device (FE85)
E27C And go do it >>E266
E27E When done, exit
E281 RETURN
E2B2 Save Device Number (BF30)
E2B5 Scan for the Volume Control Block <E859>
E2B8 Error? >>E2C5
E2B9 No
E2BE Read block 2 (Volume Directory) <EBC9>
E2B1 Get Volume Control Block offset (FE59)
E2B4 Volume Directory read OK >>E2A5
E2B6 Bad read, save error number
E2B7 Any file open? (D911)
E2B8 Yes >>E2A2
E2B9 Zero out this VCB entry (D900)
E2A2 Put error number in Accum
E2A3 Always taken >>E2C5
E2A5 Volume name exist? (D900)
E2B8 No >>E2AF
E2AA Yes, Files open? (D911)
E2AD Yes >>E2BB
E2B2 No, set up Volume Control Block for new Vol <E8B4>
E2B2 Error? >>E2C5
E2BB No
E2B6 Was a duplicate Volume Control Block found? (FE7D)
E2B9 Yes, then error >>E2C5
E2BB See if the same Volume is still there (FE59)
E2B1 If not, Disk Switch Error
E2C3 Else, all is well - continue >>E2B3

E2C5 ******* ERROR ****************************

Store code in data buffer entry

E2C5 ---
E2C6 Store Device Number in entry <E2F8>
E2C8 Store error code next
E2CD Duplicate Volume error?
E2CF No - done >>E2E1
E2D2 Store Device Number for duplicate next (FE7E)
E2DA No Duplicate now
E2B1 Exit with error
E2B2 RETURN

E2B3 ******* MAKE ONLINE VOLUME ENTRY *******

E2B3 Get name length for loop index (D900)
E2BC Copy name to Buffer entry (D900)
E2F3 Done yet? (FE80)
E2F6 No, do another >>E2EC
E2F8 Yes, find current Buffer entry (FE80)
E2FB Store Device number (BF30)
E303 Return to caller

E304 ****************************************

***** MLI CREATE CALL *****

********************************

E304 Follow Path to File <E5A6>
E307 Error? - I'm expecting one >>E30D
E309 If File was found - Duplicate error
E308 ---
E30C Return to caller

E30D File not found?
E30F No, then a real error occurred >>E30B
E311 Yes, get requested storage type
E315 Is it Ø0, Ø01, Ø62 or Ø03?
E317 Yes, carry on >>E31D
E319 Is it Ø0D?
E31B No, then exit with error >>E32D
E31D Get status of this device (BF30)
E323 Exit on error >>E338
E325 Is there a free Directory entry? (FE63)
E32B No >>E331
E32A Yes - continue >>E38F

E32D Indicate Bad Storage Type
E330 Return to caller

E331 Is this the Volume Directory? (FB0E)
E337 No, we can extend it >>E33D
E339 Yes, indicate Volume Directory Full error
E33C Return to caller

* EXTEND DIRECTORY FILE *

E330 Save old current Block number
E343 Allocate a Block on Disk <E9A6>
E346 Save the number
E347 Replace BLKNUM
E34D Was there a free Block?
E34A No, then exit >>E330
E350 Yes, set up forward pointer in old one (DC02) to point to it (DC03)
E355 and Write old Directory Block <EBD5>
E359 Error? Yes, then exit >>E330
E35D Set BLKNUM -> new Block number
E362 Loop until done >>E35D
E36C Zero remainder of Block Buffer (DC02)
E36F (including forward pointer) (DB00)
E373 Loop until done >>E36C
E375 Write new Directory Block <EBD5>
ProDOS MLI -- V1.2 -- 6 SEP 86

E370 Error? Yes, then exit >>E330
E37A Set BLKNUM -> Parent Directory block number (FE0E)
E380 Read Block with my entry <E8C0>
E383 Entry number within the Parent Dir. block (FE18)
E386 None relocatable!
E388 Set ($48) -> Buffer
E38A Skip link pointers
E38C --
E38D Count entries
E390 Skip to next (FE11)
E399 Save LSB
E39D Add 1 to Blocks used
E39F and $200 to EOF mark (FD96)
E4A2 in entry
E4A8 Loop until done >>E39D
E3AA Write back Block to Parent Directory <EBD5>
E3AD Error? Then exit >>E3BE
E3AF Start all over now that there's room >>E304

E3B2 ********** ZERO $F600 **********************************************
E3B2 Zero $F600 Block Buffer
E3BE Return to caller

E3BF ********** BUILD NEW FILE ***************************************
E3BF Call Zero $F600 routine <E3B2>
E3C2 Copy Datetime (Creation)
E3C4 to my variables
E3D0 Loop until done >>E3C4
E3D2 Did he give Datetime (Creation)?
E3D3 Yes, carry on >>E3E0
E3D5 No, then use
E3D7 System Datetime instead (BF90)
E3E0 If Storage type is $00, $01, $02 or $03
E3E2 force it to $10
E3E8 else use a $08
E3EA Find File name (FE82)
E3ED OR Storage type to name length (D700)
E3F0 Store Type/Length (FE27)
E3F3 Isolate name length
E3F7 Copy File name to File Entry Buffer (FE82)
E405 Copy caller's Access Byte
   NOTE: This should be validity checked!!!
E40D and copy File type
E412 --
E413 and AUX_TYPE
E41C Copy Version and Min Version ($0, $0) (FD88)
E41F constants to entry (FE43)
E428 Indicate 1 Block used
E42D Copy Directory Header Block number (FE22)

E43C Is this a Seedling file?
E43E Yes >>E475
E440 No, Directory file - Build Header in $F600
E442 Copy completed Directory entry (FE27)
E445 to $F600 buffer first (DC04)
E447 Loop until done >>E442
E44B Make Storage type $E in Header itself
E450 Put "HUSTON" (Author) in Reserved area
E458 and Version, Min_Version, Access, (FD88)
E45B Entry-length, File count and (DC90)
E45E Parent pointer from constants
E45F Loop until done >>E452
E463 EOF = $200 (FE3D)
E466 Copy Parent Block entry number (FE24)
E46D Loop until done >>E466
E46F Copy Parent entry Length (FE19)
E475 Allocate a new disk block <EA9C>
E478 error? >>E4B1
E47A Store it in key pointer of entry (FE38)
E480 and in BLKNUM for I/O
E484 Write zeroed (or DIR HDR) key block <EBD5>
E487 error? >>E4B1
E489 Bump parent's file count (FE18)
E491 Go update directory <E4B2>
E494 error? >>E4B1
E496 Checkpoint Volume Bit Map and exit >>EB76

E499 ********** POINT $48/49 AT DIRECTORY ENTRY ********************
E499 $48/$49 --> Entry
E49D Skip link pointers (+4)
E49F File entry number counter (FE26)
E4A2 --
E4A3 Skip to proper entry
E4A6 Add entry length (FE19)
E4AB (bump MSB)
E4AF (store LSB)
E4B1 RETURN

E4B2 ********** UPDATE DIRECTORY(s) *******************************
E4B2 System date available? (BF90)
E4B5 no, forget it >>E4C2
E4B9 yes, copy to last modified date field (BF90)
E4C2 turn on BUBIT (backup) if appropriate (FE45)
E4CB set DEVNUM of parent (FE21)
E4D1 and BLKNUM (FE24)
E4D7 reread DIR block containing entry <EBC9>
E4DA error? >>E4B1
E4DC Point to proper entry in buffer <E499>
E4E3 Copy constructed entry to buffer (FE27)
E44E Is this block the DIR HDR block?
E44F no, write this modified directory block <EBD5>
E44C error? >>E481
E504 and then read DIR HDR block <EBC9>
E507 error? >>E481
E509 in any case..
E50B copy back update file count to HDR (FE18)
E514 and ACCESS byte (with Backup) (FE18)
E51A re-write the HDR block <EBD5>
E51D error? >>E573
E51F is this the VOL DIR? (DC04)
E526 yes, all done -- exit >>E591
E52B Is subdirectory, get PARENT ENTRY, (DC29)
E52B store in variable area (FE26)
E52E Get PARENT ENTRY_LENGTH, (DC2A)
E531 store in variable area (FE19)
E534 get parent block number (DC27)
E53A read Parent Directory block <EBC9>
E53D error? >>E573
E53F find entry for this subdirectory <E499>
E542 system date available? (BF90)
E545 no >>E554
E547 yes,
E54B copy system date/time to... (BF90)
E54E modified date/time in entry
E544 write it back <EBD5>
E557 error? >>E573
E55B BKLNUM = HDR block number
E564 same block we have now?
E568 yes, go back and date stamp >>E51F
E56A no,
E56E read HDR block <EBD9>
E571 and go back to date stamp parent DIR >>E51F
E573 error? then exit
E574 ********** NOT ProDOS VOLUME ERROR ***************
E574 ---
E577 RETURN

E578 ********** IS THIS ProDOS VOLUME? *********************
E578 Does previous block ptr = 07 (DC04)
E586 no, not a ProDOS volume >>E574
E58B else, (DC04)
E59D does VOL DIR'S STORAGE TYPE = $E or $F?
E59F no, error >>E574
E591 else, ok
E592 RETURN
ProDOS ML1 -- V1.2 -- 6 SEP 96

**NO MORE FILE ENTRIES**

E618 free entry found in directory? (FE63)
E61B yes >>E63B
E61D no, check pointers (DC92)
E620 is there another block after this one? >>E627
E625 no... >>E63B
E627 yes, free entry will be... (FS24)
E630 first in that block
E635 indicate free entry available (FE63)
E638 find next index name <E764>
E63B exiting with error
E63C no more indices in path, file not found >>E641
E63E else, path not found
E640 RETURN

E641 file not found error
E643 RETURN

**FOUND FILE ENTRY**

E644 advance to next sub dir in path <E75D>
E647 end -- save entry no. and exit >>E685
E64B get type of entry
E64F subdir?
E651 no, bad path then >>E63B
E655 copy key block no...
E657 to BLONUM
E65A and to current DIR block no (FE22)
E664 go read key block of subdirectory <E909>
E667 error? >>E68D
E66C new file count (FE80)
E675 check minimum version (DC21)
E678 too new? >>E688
E680 count bits in reserved field of DIR hdr
E681 --- >>E684
E684 ---
E687 there must be 5 bits on (normally $75)
E689 (there are) >>E68F
E68B or else, incompatible file format
E68D ---
E68E RETURN

E68F copy DIR HDR <E695>
E692 and go scan for next level >>E5D4

**COPY DIRECTORY HDR**

E695 Copy,
E697 CREATION, VERSION, MIN_VERS, ACCESS, (DC1C)
E69A ENTRY_LEN, ENTRIES_PER_BLK, FILE_COUNT (FE12)
E6A0 volume directory? (DC94)
E6A7 if so, exit now >>E6B4
E6AB else, copy PARENT_POINTER, (DC27)
E6AE PARENT_ENTRY_NO., and PARENT_ENTRY_LEN (FE0E)
E6B4 RETURN

E6B5 **SAVE DIR ENTRY NO. & BLOCK**

E6B5 compute entry number (FE1A)
E6BE save it (FE26)
E6C3 and the block it's in (FE24)
E6CC exit

**SEARCH ONE DIR BLOCK FOR FILE**

E6CD get entries in this block (FE1A)
E6D3 $48/$49 --> first entry
E6D9 ---
E6DB skip HDR? -->E710
E6D0 no, non empty entry?
E6E1 yes >>E6F0
E6E3 no, do we need one? (FE63)
E6E6 no >>E716
E6E8 yes, remember it <E6B5>
E6E9 don't need another one now (FE63)
E6EE skip to next entry >>E710
E6F0 get length of name
E6F2 count it (FE5F)
E6F5 save it for loop (FE80)
E6FB same len as we are wanting? (D703)
E6FE no, skip it >>E710
E700 ---
E704 compare names (D700)
E70E we found it! exit
E70F RETURN

E710 skip to next entry (FE62)
E714 end of block? if so, exit >>E70F
E71A bump $48/$49 by entry len
E721 and go check next >>E699
E723 ********** GET DIRECTORY DATA ******************************
E723 find base directory <E77C>
E726 error? >>E77B
E72C zero out my variables (FE8E)
E732 set up device number (BF30)
E738 copy DIR HDR to my variables <E695>
E741 copy TOTAL BLOCKS from VCB (D912)
E747 copy BIT MAP Pointer from VCB (D91A)
E74D copy Block No. of this directory (BB46)
E753 make second copy of file count (FE18)
E75D advance to next subdir in path <E764>
E760 and update index (FE82)
E763 RETURN

E764 ********** ADVANCE TO NEXT DIR NAME ******************************
E764 get this DIR's index (FE82)
E768 add len of name to move index to next name (FE82)
E76F still in prefix portion? >> E777
E771 no, now starting caller's path suffix (BF30)
E774 save last DEVNUM accessed (FE67)
E777 return with len of next dir in path (D740)
E77B RETURN

E77C ********** FIND BASE DIRECTORY ******************************
E77C ---
E77E get old PREFIXPTR (BF9A)
E781 fully qualified path name? (FE84)
E784 no >> E77B
E786 yes, no old PREFIXPTR anymore
E787 save old prefix index (FE83)
E78A DEVNUM=0 (BF30)
E78D ---

*** SCAN VCB'S FOR A MOUNTED VOLUME ***
E78F scan (D908)
E792 got one >>E79F
E799 else, bump to next VCB

*** FIND LAST DIR IN PREFIX OR TOL DIR ***
E79F store name length (FE80)
E7A2 same name as in pathname? (D700)
E7A5 no -- skip it >>E794
E7B3 save VCB index (FE59)
E7B6 DEVNUM = VCB's unit no. (D918)
E7BC BLOCK = 2 (read VOLDIR if no old PREFIX)
E859 ********** SCAN VCB’S FOR DEVICE NO. ***************

E859 ---
E85D scan VCB's for a given device number
E864 not it? >>E86B
E866 is it, save VCB index (FE59)
E869 and exit normally
E86A RETURN

E868 else, volume mounted here? (D900)
E86E yes >>E874
E871 no, save VCB index to empty unit (FE59)
E874 ---
E876 bump to next VCB
E878 and go look at it >>E85D
E87A not found...
E87B any free entries? if not, error >>E87E
E87D else, all is well -- return empty VCB
E87E VCB table full error
E880 RETURN

E881 ********** COMPARE DIR NAME WITH PATH LVL ***************

E881 ---
E886 check DIR type (DC04)
E889 VOL DIR or SUB DIR?
E88B neither >>E894
E88D yes,
E88F store len of its name (FE80)
E892 and go on >>E899
E894 error exit
E895 RETURN

E896 compare directory names (DC04)
E89C no match? >>E894
E8A5 they match! exit
E8A6 RETURN

E8A7 ********** MOUNT NEW VOLUME ***********************

E8A7 volume mounted? (FE59)
E8AD no, continue >>E8B4
E8AF yes, same one as one wanted? <E90C>
E8B2 if so exit, else fall thru >>E908

E8B4 ********** SET UP VCB FROM VOLDIR ***********************

E8B4 zero out VCB
E8BF is this a ProDOS volume? <E57B>
E8C2 no -- exit >>E90B
E8C4 duplicate vol in VCB's? <E930>
E8C7 yes -- exit with that one instead >>E90A
E8C9 get new volume's name length (DC04)
E8D0 add to VCB index (FE59)
E8D4 and copy to VCB name field in empty VCB (DC04)
E8DF store in VCB name len field (D900)
E8E2 copy DEVNUM to VCB unit field (BF30)
E8EB copy total blocks to VCB (DC29)
E8F4 copy block no. of vol dir to VCB
E8FE copy bit map block no. to VCB (DC27)
E90A exit
E90B RETURN

E90C ********** COMPARE VOL NAMES TO MAKE ***************

******* SURE THEY MATCH *******

E90C Get length (DC04)
E911 Same in VCB? (D900)
E914 Save VCB offset (FE58)
E917 Different from VCB >>E924
E919 Store len to use as buffer index
E91A Add length to VCB offset to get (FE58)
E91D index into VCB (last char of name)
E921 Compare names (D900)
E924 SEC if no match
E92B CLC if match
E92C Restore VCB offset to X-REG (FE58)
E92F RETURN

E930 ********** LOOK FOR DUPLICATE VOL ********************

E938 start with first VCB
E938 ---
E939 this VCB has same name? <E90C>
E936 no >>E947
E938 yes, files open? (D911)
E93B yes >>E951
E93F no, mark VCB empty (NAME=0) (D900)
E942 (UNIT=0) (D910)
E945 and exit with no error >>E94F
E947 else,
E949 bump to next VCB
E94D and loop >>E932
E94F exit no errors
E950 RETURN
E951 save flag (FE7D)
E954 and VCB index of duplicate vol (FE7E)
E957 exit with error
E958 RETURN

E959 *********** see if a quantity of free ***********
*********** blocks is available on vol ***********

E959 any free blocks counted in VCB? (FE59)
E962 yes >>E966

*** compute VCB free block count ***

E964 no, how many bit map blocks are there? <EA88>
E967 save it (less 1) (FE64)
E96C zero scratch (will count free blocks) (FE4E)
E972 no block found yet
E977 checkpoint bit map buffer <E876>
E97A error? >>E97A
E97F BLKNUM = bit map pointer (D91A)
E989 read block to buffer <EB9D>
E98C error? >>E988
E98E count free blocks marked <E9CB>
E991 drop no. remaining to do (FE64)
E994 none left? >>E994
E996 some, BLKNUM = BLKNUM + 1
E99C go process that >>E989

E99F did we find a free bit? (FE59)
E9A5 no -- volume full >>E9C7
E9A7 save VCB bitmap block offset (D91C)
E9AA save free block count in VCB also (FE4F)
E9B6 are there enough to satisfy request? (D914)
E9C5 yes, exit
E9C6 RETURN

E9C7 volume full error
E9CA RETURN

E9CB *********** scan and count bitmap blocks ***********

E9CB scan through both buffer pages
E9DD counting one bits <E9F8>
E9E0 found free block already? (FE63)
E9E3 if so -- done >>E9F7
E9E5 any blocks found yet? (FE4E)
E9EB no >>E9F7
E9ED yes, compute total no. of bitmap blocks <EA88>
E9F1 less number remaining (FE64)
EA96  exit normally
EA97  RETURN

EA98  bad bitmap error
EA99  RETURN

EA9C  ********** FIND A FREE DISK BLOCK AND **************
      ****** AND ALLOCATE IT  ******

EA9C  go read bitmap <EB43>
EA9F  error? >>EAC4
EAAC  first page 0
EAA6  scan 1st page of bitmap for free block(s) (DA96)
EAAE  bump tm page 1 of buffer (FE6C)
EAB0  bump page offset (FE6B)
EAB4  scan 2nd page too (DB00)
EABC  bump page (FE68)
EABF  get next block <EB28>
EAC2  continue >>EAA1
EAC4  error exit

EAC5  save byte index (FE6A)
EAC8  shift combination of page no. and (FE68)
EACC  byte offset left 3 bits to make (FE4F)
EACE  room for bit position.
EADD  depending on buffer page ... (FE6C)
EAE2  reload bit pattern from page 0 ... (DB00)
EAE7  or page 1 (DA96)
EAEA  shift bit pattern, bumping block no. L5B
EAEB  until a one bit is found >>EAF0
EAF0  then shift it back the way it was
EAF1  (with that bit turned off) >>EAF0
EAF4  store LSB of block no. (FE4E)
EAF6  store updated byte-back in proper page (FE6C)
EB03  indicate bitmap needs checkpoint
EB0D  one less block available in VCB (FE59)
EB0F  ---
EB21  return with new block no. (FE4E)
EB27  RETURN

EB28  ********** GET NEXT BITMAP BLOCK ***************

EB28  use blocks of vol to compute (FE59)
EB2B  number of blocks in bitmap (D913)
EB32  just scanned last block? (D91C)
EB35  yes, no space >>EB72
EB37  no, get next block (D91C)
EB40  checkpoint old one <EB76>

EB43  ******************** READ BITMAP BLOCK ***********************

EB43  have we read bitmap for this unit yet? (FE59)
EB4C  yes >>EB5C
EB4E  no, checkpoint bitmap of some other unit <EB76>
EB51  error? >>EB71
EB56  get new bitmap unit no. (D910)
EB5C  was bitmap modified? (FE6D)
EB5F  yes >>EB66
EB61  no, read it <EB87>
EB64  error? >>EB71
EB66  save bitmap block offset times 2 (FE59)
EB69  (page number) (D91C)
EB70  exit
EB71  RETURN

EB72  disk full error
EB75  RETURN

EB76  **************** CHECKPOINT VOLUME BITMAP *******************

EB76  ---
EB77  needs checkpoint? (FE6D)
EB7A  no >>EB71
EB7C  yes, write it <EB91>
EB7F  error? >>EB71
EB81  doesn't need checkpoint now
EB96  exit

EB87  ******************** READ BITMAP ***********************

EB87  save DEVNUM (FE6E)
EB9A  copy block offset wanted (FE59)
EB94  BITMAP BLOCK = BITMAP PTR + BLOCK OFFSET (D91A)
EBA2  set up read command

*** READ OR WRITE BITMAP ***

EBA4  save I/O command
EBA6  device = bitmap device (FE6E)
EBA8  block = bitmap block (FE6F)
EBA9  point to bitmap buffer (EAB2)
EBBD  do the I/O <<EB9P
EBC2  restore old DEVNUM (BF30)
EBCC  ok? >>EBCB
EBC7  no, error exit
EBC8  RETURN
**EBC9******** READ BLOCK DESIGNATED BY A,X **-----------------------------**
EBC9 Put low byte of block number in BLKNUM
EBC8 and high byte in BLKNUM+1
EBCD Read a block <EBD9>
EBDB RETURN

**EBD1******** WRITE BITMAP ***********************
EBD1 set up write command
EBD3 and go do it >>EBA4

**EBD5******** WRITE BLOCK ***********************
EBD5 set up write command
EBD7 and go do it >>EBDB

**EBD9******** READ BLOCK ***********************
EBD9 set up read command

**EBDB******** READ OR WRITE BLOCK ***********************
EBDB save I/O command
EBDD where is my buffer?
EBDF save flags
EBDB and disable
EBE3 Set low byte of Buffer pointer
EBE5 to zero
EBE7 Initialize Global Page System error to 0 (BP90)
EBEA set I/O transfer occurred flag
EBEF set unit to do I/O on (BF30)
EBF4 do block I/O <DE64>
EBF7 error? >>EBFC
EBF9 no errors, restore things and exit
EBFB RETURN

**EBFC error exit
EBFP RETURN

**EBFP******** MLI GET MARK CALL ***********************
EBFP copy mark to caller's list from FCB (FE5A)
EC3F exit with no errors
EC10 RETURN

**ECl1************ MLI SET MARK CALL **********

**ECl5 set up to...
EClD copy user's mark to temporary
EClF new mark variable (FE79)
ECC4 make sure it will not exceed EOF (D815)
EC29 else, error >>ECl1

** STILL IN SAME DATA BLOCK? **
EC32 get old mark (FE5A)
EC35 find its block no. (*2) (D813)
EC3D compute distance in pages from old mark's (FE73)
EC41 block to new mark (FE42)
EC47 earlier -- need new data block >>EC58
EC48 too far forward -- need new block >>EC58
EC50 MSB's match? (D814)
EC55 then mark is still in this block >>ED79

EC5B check storage type (D807)
EC58 zero? >>EC64
EC5D seedling, sapling or tree?
EC61 no, special handling for DIR files >>EDAB

EC64 This is a bug!!!
The immediate addressing mode was used
where absolute addressing was intended.
EC66 This will stomp on another ECBi (D800)
EC69 return with bad REFNUM error
EC6C RETURN

** NEED DIFFERENT DATA BLOCK **
EC6D copy storage type (D807)
EC73 old data block needs writing? (D808)
EC78 no >>EC7F
EC7A yes, do so <EBE3>
EC7D error? >>EC58
EC7F see if new mark is outside the range of (FE5A)
EC82 the current index block (D814)
EC91 yes >>EC81
EC95 yes >>EC81
EC97 no, same index block (FE5E)
EC9A check storage type
EC9B sapling or tree are ok >>ED16

*** SEEDLING ***

EC9D seedling, check position (FE73)
ECAB if position is outside of block 0..
ECA4 promote to sapling >>ED04
ECA6 else, (D80C)
ECAE go get key block (seedling data block) >>ED6F

*** NEED TO CHANGE DATA BLOCKS ***

ECB1 does old index block need dumping? (D808)
ECB5 no >>ECBD
ECB7 yes, do so <EE97>
ECB8 error? >>ECEB
ECBD check storage type (FE5E)
ECC0 tree file?
ECC2 yes >>ECE9
ECC4 no, sapling (FE74)
ECC9 is position in first index block?
ECCC no, need master index, subindex and data >>ED2F
ECCE yes, first index, reset flags <ED9F>
ECDF is this a seedling?
ECD2 if so, see if in first block >>EC9D

*** SAPLING ***

ECD4 no, sapling, read its only index block <EE2A>
ECDC error? >>ECEB
ECD6 set block no. of index block
ECD8 Always branch >>ED16
ECEB Error exit

*** TREE FILE/NEED ANOTHER INDEX BLOCK ***

EC95 reset flags <ED9F>
EC9C read master index block <EE2A>
EC9E error? >>ECEB
ECF1 make index into block from (FE74)
ECF4 MSB_of_position/2
ECFA is there a subindex there?
ECFC yes! >>ED99
ED02 no, fall thru to make one

*** GET NEW INDEX BLOCK ***

ED04 need an index and data block
ED06 go allocate them >>ED2F
ED09 set up block no. of subindex
ED11 read it <EE0D>
ED14 error? >>ECEB

*** SAPLING/TREE - THIS INDEX BLOCK ***

ED16 make block no. out of position (FE74)
ED1F use as an index to examine index block
ED21 entry
ED27 if its zero...
ED2B need new data block
ED2F set flags for what to allocate (FE5A)
ED38 new index block being created?
ED3A zero data block in any case <ED57>
ED3D if not index block that's it >>ED79
ED3F Zero the Index Block I/O Buffer <ED4S>
ED42 and continue >>ED79

ED45 ******** ZERO INDEX BLOCK I/O BUFFER ***********************

ED45 ---
ED4B Zero first page
ED4F and second page of Index Block I/O buffer
ED54 Restore pointer to beginning of buffer
ED56 RETURN

ED57 ******** ZERO OUT DATA BLK I/O BUFFER ***********************

ED57 ---
ED5A Zero first page
ED61 and second page of data block I/O buffer
ED66 Restore pointer to beginning of buffer
ED68 RETURN

ED69 ******** READ FILE DATA BLOCK ***********************

ED69 set block no. LSB
ED6B copy MSB dram index entry
ED6F ---
ED71 read new data block <EDF4>
ED74 error? >>ED9E
ED76 reset block allocation flags <ED9F>
*** GOT DATA BLOCK WANTED ***

ED79  ---                    
ED80  save previous mark in my variables (D812) 
ED86  set new mark in the PCB (F872) 
ED91  ($4A/$4B --> data block buffer) 
ED93  $4C/$4D --> start of the page in 
ED95  the data block buffer which contains (F873) 
ED98  the mark. 
ED9E  exit

ED9F  ********* RESET BLOCK ALLOC FLAGS **********

ED9F  get flags (F65A) 
EDA5  turn off low 3 bits (allocate no new 
EDA7  blocks to file) (D808) 
EDAA  RETURN

EDAB  ********* SET DIR FILE POSITION **********

EDAB  DIR file? 
EDAD  yes! >>EDB4 
EDAF  no, bad storage type error 
EDB1  go to SYSERR <BF90> 
EDB4  else, get page distance (F64E) 
EDB7  make it into blocks (divide by 2) 
EDBE  new position beyond old? (F873) 
ECD1  yes >>EDD1 
ECD3  else, use previous mark 
ECD5  copy to BLKNUM <EDDF> 
ECD8  error? >>EDEE 
EDEA  count it (F662) 
EDEC  more to skip? >>EDC3 
EDCF  no, got it >>ED79 
EDD1  use next block pointer in DIR block 
EDD3  copy to BLKNUM <EDDF> 
EDD6  error? >>EDEE 
EDDB  count it (F662) 
EDDD  more to skip >>EDD1 
EDDF  got it now! >>ED79

*** COPY LINK TO BLKNUM ***

EDDF  copy block number link 
EDE1  to BLKNUM 
EDE4  if non zero, 
EDEA  then go read block. >>EDF0 
EDEC  else, EOF error 
EDEE  --- 
EDEF  RETURN

ProDOS MLI -- V1.2 -- 6 SEP 86  NEXT OBJECT ADDR: ED76
----------------------------------------
PRODOS MLI -- V1.2 -- 6 SEP 86  NEXT OBJECT ADDR: EDEF
----------------------------------------

EDF0  ********** READ FILE BLOCK **********

EDF0  set block number to read 
EDF4  store read I/O command 
EDF8  read to $48/$49 buffer 
EDFA  read the block <EE50> 
EDFD  error? >>EE0C 
EE02  copy block no. just read to PCB 
EE0C  exit

EE0D  ********** READ SUB-INDEX BLOCK **********

EE0D  set read I/O command 
EE11  read to $48/$49 buffer 
EE13  read the block <EE50> 
EE16  error? >>EE26 
EE1B  save BLKNUM in PCB as current index 
EE1D  block. (D80E) 
EE26  exit

EE27  ********** WRITE KEY INDEX BLOCK **********

EE27  set write I/O command 
EE29  Use bit instruction to skip over two bytes

EE2A  ********** READ KEY INDEX BLOCK **********

EE2A  ---

EE2C  ********** READ OR WRITE KEY INDEX BLOCK **********

EE2C  save command 
EE2F  block no. is key block in PCB (F65A) 
EE34  use $48/$49 buffer 

*** I/O BLOCK ***

EE36  set I/O command 
EE38  and block no. (D800) 
EE42  must be non-zero block number 
EE46  or horrible death! 
EE48  fall through to read/write block (D801)

*** SET UP AWU DO FILE BLOCK I/O ***

EE50  (zreg = buff ptr in zero page) 
EE51  disable 
EE52  set up buffer pointer 
EE5D  get DEVNUM from PCB (D801) 
EE63  set I/O transfer has occurred flag
Beneath Apple ProDOS Supplement

ProDOS MLI — V1.2 — 6 SEP 86

ADR DESCRIPTION/CONTENTS

EE60 set unit no. from DEVNUM (BF38)
EE61 no errors have occurred yet
EE71 do block I/O <DEE4>
EE75 error? >>EE7A
EE77 no, exit normally
EE79 RETURN

EE7A else, exit with error
EE7C RETURN

EE7D ******** CHECKPOINT BITMAP & KEY BLOCK ***********************

EE7D checkpoint bitmap buffer <EE76>
EE80 go write key block for file >>EE27

EE83 ******** CHECKPOINT DATA BLOCK BUFFER *************************

EE83 buffer pointer at $4A/$4B
EE85 point to block no. in PCB
EE8D go write buffer to disk <EE36>
EE90 error? >>EEB4
EE94 go turn off $A0 flag in PCB and exit >>EEAB

EE97 ******** CHECKPOINT INDEX BLOCK BUFFER *************************

EE97 checkpoint volume bitmap <EE76>
EE9A use $48/$49 buffer
EE9C block no. is current index block in PCB
EEA2 set to write
EEA4 go write it to disk <EE36>
EEA7 error? >>EEB4
EEA9 no longer needs checkpoint
EEAB set flags accordingly (FE5A)
EEB4 and exit

EEB5 ************************************************************************

****** MLI OPEN CALL ******
************************************************************************

EEB5 search path for file <E593>
EEB8 found it? >>EEBE
EEBA no, bad path error
EEBC exit >>EEC5
EEBE else, see if FCB already open on file <E9B>
EEC1 for write, if not, continue. >>EEC8
EEC3 else, file already open error
EEC5 no
EEC6 RETURN

EEC7 Error -- unsupported storage type
EECA RETURN

EEC8 get FCB index (FE5A)
EEE1 free FCB found? >>EE77
EEE3 no, all FCB's in use error
EEE6 RETURN

EEE7 zero out unused FCB
EEF2 copy file ID fields to PCB
EEE5 (DEVNUM, DIR HDR BLK, DIR BLK, (FE5A)
EEE8 DIR ENTRY NO.)
EEF3 isolate storage type (FE27)
EEF8 and copy to PCB (D807)
EEFE get access (FE45)
EFP3 DIR file?
EFP5 no >>EFP9
EFP7 yes, we are only reading (I hope)
EFP9 update access flag in PCB (DB99)
EFPB write protected? >>EFP15
EPF10 no, another FCB open on this file? (FE5F)
EPF3 yes, no touchie >>EEC3

EFP5 storage type must be < $4
EFP9 or equal to $D
EFPB else, storage type error >>EEC7
EFPD no
EFPF copy key block, blocks used, and
EF21 EOF mark to PCB (FE5A)
EF31 BLKNUM = key block number
EF36 store REPNUM in PCB (FE62)
EF3C go check and assign 1/O buffer <<FB11
EF3F error? >>EF65
EF41 go find VCB and set buff ptrs <E1E2
EF44 set current level in PCB (BF94)
EF4A seedling, sapling or tree? (DB07)
EF4F no, skip next stuff >>EF7C
EF51 yes, make current mark in PCB outside
EF53 first index block to force a read of all (DB14)
EF56 index blocks and BLOCK 0.
EF5A zero mark wanted, however (FE72)
EF60 go set mark to zero <<EC32
EF61 ok? >>EF61
EF65 no, save the error code
EF69 got and 1/O buffer? (DB08)
EF6C no >>EF74
EF6E yes, free it <FGE>
EF74 mark PCB not in use
EF7A exit with error
EF7B RETURN
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF7C</td>
<td>else, read key block to I/O buffer &lt;EDF4&gt;</td>
</tr>
<tr>
<td>EF7F</td>
<td>error? &gt;&gt;EF65</td>
</tr>
<tr>
<td>EF81</td>
<td>bump open file count in VCB (FE59)</td>
</tr>
<tr>
<td>EF87</td>
<td>indicate files are open in VCB (D911)</td>
</tr>
<tr>
<td>EF8F</td>
<td>put REF NUM in caller's parmlist (FE5A)</td>
</tr>
<tr>
<td>EF99</td>
<td>exit with no errors</td>
</tr>
<tr>
<td>EF9A</td>
<td>RETURN</td>
</tr>
<tr>
<td>EF9B</td>
<td>FIND A FCB</td>
</tr>
<tr>
<td>EF9B</td>
<td>clear flags and index byte</td>
</tr>
<tr>
<td>EF9B</td>
<td>---</td>
</tr>
<tr>
<td>EFAC</td>
<td>no, bump entry count (FE62)</td>
</tr>
<tr>
<td>EFAF</td>
<td>PCB in use? (D880)</td>
</tr>
<tr>
<td>EF21</td>
<td>yes &gt;&gt;EF21</td>
</tr>
<tr>
<td>EF40</td>
<td>no,</td>
</tr>
<tr>
<td>EF47</td>
<td>save index to free PCB (FE5A)</td>
</tr>
<tr>
<td>EF4A</td>
<td>flag that we found one</td>
</tr>
<tr>
<td>EF6F</td>
<td>and skip this PCB &gt;&gt;EFDF</td>
</tr>
<tr>
<td>EF6F</td>
<td>---</td>
</tr>
<tr>
<td>EF7C</td>
<td>compare file ID's to see if this PCB (D880)</td>
</tr>
<tr>
<td>EFCA</td>
<td>is open on the requested file. (FE20)</td>
</tr>
<tr>
<td>EFCD</td>
<td>no match? &gt;&gt;EFDF</td>
</tr>
<tr>
<td>EFDF</td>
<td>indicate PCB already open on file (FE5F)</td>
</tr>
<tr>
<td>EFDF</td>
<td>write enabled? (D889)</td>
</tr>
<tr>
<td>EFDB</td>
<td>if not, allow multiple open access to file &gt;&gt;EFDF</td>
</tr>
<tr>
<td>EFDD</td>
<td>else, error exit</td>
</tr>
<tr>
<td>EFDE</td>
<td>RETURN</td>
</tr>
<tr>
<td>EFDF</td>
<td>return index to start of PCB</td>
</tr>
<tr>
<td>EF61</td>
<td>bump to next PCB</td>
</tr>
<tr>
<td>EF5E</td>
<td>and loop &gt;&gt;EF66</td>
</tr>
<tr>
<td>EF67</td>
<td>when done, exit normally</td>
</tr>
<tr>
<td>EF69</td>
<td>RETURN</td>
</tr>
<tr>
<td>EF9B</td>
<td>************* FIND A FCB ********************</td>
</tr>
<tr>
<td>EF9B</td>
<td>clear flags and index byte</td>
</tr>
<tr>
<td>EFAC</td>
<td>yes &gt;&gt;EFAC</td>
</tr>
<tr>
<td>EF6D</td>
<td>if not, allow multiple open access to file &gt;&gt;EFDF</td>
</tr>
<tr>
<td>EF6D</td>
<td>else, error exit</td>
</tr>
<tr>
<td>EFDE</td>
<td>RETURN</td>
</tr>
<tr>
<td>EF6F</td>
<td>read access permitted?</td>
</tr>
<tr>
<td>EF6G</td>
<td>yes &gt;&gt;EF6F</td>
</tr>
<tr>
<td>EF6H</td>
<td>no, access error</td>
</tr>
<tr>
<td>EF6C</td>
<td>will we read past EOF? &gt;&gt;F02E</td>
</tr>
<tr>
<td>EF7E</td>
<td>yes, (FE5A)</td>
</tr>
</tbody>
</table>
F099 crossed index block? go do set mark >>F067
F09B make index block offset from mark (FE74)
F0A4 BLKSUM = next block in index block
F0A6 zero entry?
F0B2 if so, no direct read can occur until next (FE7A)
F0B5 set-mark/read >>F0BA
F0B7 get MSB of BLKSUM
F0BA (put index ptr back)
F0BE finish setting BLKSUM MSB
F0C0 if no read occurred within setmark, (FE7A)
F0C3 go back to setmark call >>F067
F0C7 disable
F0CB do I/O to caller's buffer directly
F0CC do block I/O directly <DEE4>
F0CF error? >>F0D4
F0D2 go back for more >>F06C

*** ERROR CLEANUP ***

F0D4 ---
F0D5 ---
F0D6 set buffer ptrs/VCB <F195>
F0D8 ---
F0DB finish up I/O <F0E1>
F0DF exit with error
F0E0 RETURN

F0E1 ********* I/O FINISH UP ***********************

F0E1 ---
F0E4 return actual length read in caller's list (FEA2)
F0F5 and exit by setting new mark >>EC32

F0F8 ********* SET UP BUFFER INDEXING ***************

F0F8 ---
F0FC back up pointer to data buffer by an
F0FE amount equal to the LSB of the mark (FE72)
F101 (which makes indexing easier)
F107 newline mode enabled? (D81F)
F10B no, CLC >>F117
F10D yes, SEC
F10E copy newline mask (FE79)
F111 and newline character (D88A)
F117 first char index is LSB of mark in YREG (FE72)
F11A $4C/$4D --- page containing mark
F11E request count LSB in XREG (FE75)
F121 exit

F121 ********** COPY FROM I/O BLOCK BUFF ***************

****** TO DATA BUFFER ******

EXITS IF: LENGTH GOES TO ZERO
NEXT BLOCK IS NEEDED
NEWLINE IS FOUND
ON EXIT: OVERFLOW FLAG SET IF DONE
OVERFLOW ZERO IF NEXT BLOCK NEEDED

F122 ---
F123 partial page to move? >>F12D
F125 no, any full pages left? (FE76)
F12B no, read complete >>F17C
F12A yes, drop MSB of request length (FE76)
F12D ---
F12E copy one byte $4C >> $4E
F132 check for newline if carry set >>F165
F134 ---
F135 end of requested chunk >>F158
F137 ---
F139 more bytes to copy >>F12E
F13B end of page, bump pointers
F13F bump new mark (FE73)
F147 finished first page of block buffer?
F14B if so, continue >>F12E
F14E no, need another block from disk >>F17F
F150 another page in request length? (FE76)
F153 no >>F16F
F156 more in this block-page? >>F15E
F15B no, on last page of block?
F15C no >>F161
F15E yes, drop request len by one page (FE76)
F161 back up to next byte again
F162 go copy next page >>F137

F165 check for newline
F16D not it, never mind >>F134
F16F else, were we done with page?
F170 no >>F17C
F172 yes, bump pointer
F174 and mark (FE73)
F17C set overflow flag (read completed) (F194)

F17F update mark LSB (FE72)
F184 bump request count if necessary
F185 update count LSB (FE75)
F18B point beyond data in caller's buffer
F193 ---
F194 and exit
F195 ********* CLEANUP AFTER DIRECT I/O *************
F195 restore caller's data buffer pointer
F1A8 go set buffers/find VCB and exit >>012
F1A3 ********* DIRECTORY FILE READ ***************
F1A3 set mark/read <EC32>
F1A6 error? >>F1D7
F1A8 set up buffer indexing <F0F8>
F1A8 move data from 1/O buffer <F122>
F1AE need next block? >>F1A3
F1B0 no, finish up I/O <F0E1>
F1B3 ok? exit >>F1D5
F1B5 not ok, EOF error?
F1B8 no, out now >>F1D6
F1BA yes, point beyond EOF anyway? <E079>
F1BD zero out data block I/O buffer <ED57>
F1C5 dummy up an empty DIR block with previous (D810)
F1C8 pointer and no forward pointer in I/O
F1CA buffer.
F1CC zero out current block no. (D810)
F1D5 return to caller
F1D6 RETURN
F1D7 finish up and error exit >>F0DA
F1DA ********* NAME CALLER'S 1/O LENGTH **************
F1DA copy request length to LENGTH and
F1DC a temporary variable
F1ED pick up ACCESS flags for file (FE4A)
F1F3 exit to caller
F1F4 RETURN
F1F5 ********* POINT 4E/4F TO CALLER'S **************
******* DATA BUFFER *******
F1F5 set up pointer
F200 YREG --> PCB (FE5A)
F201 AREG = storage type (D807)
F206 exit
F207 ********* COPY FILE MARK AND COMPUTE ************
******* AND COMPARE END MARK *******
F207 copy file mark (D812)
F213 and set previous mark also (FE55)
F216 add length giving new mark in scratch area (FEA2)
F21D (3 byte addition)
F225 will new mark exceed EOF? (FE4E)
F233 return with carry set accordingly
F234 ********** SET NEW MARK & EOF *******************
F234 set up indexes <F266>
F237 set new EOF in PCB (FE52)
F23D and new mark (FE55)
F243 save new mark in scratch variable too (FE4E)
F24A does mark exceed EOF? <F266>
F24D if so, we must extend EOF <F225>
F253 save old EOF (D815)
F258 set new EOF to mark if necessary (FE4E)
F261 ****
F265 exit
F266 subroutine to set 3 byte indexes
F26D RETURN
F26E *************** MLI WRITE CALL *************
***************
F2A7 count number of blocks needed
F2A8 store number needed (FES5C)
F2B0 see if the blocks are available <E959>
F2B3 no, disk full >>F2C1
F2B5 yes, get PCB flags <<P5D6>
F2BA master index block needed?
F2BC yes, go add it <F381>
F2BF and go on if no errors >>F2D5

F2C1 error,
F2C2 set new mark/E0F <P234>
F2C6 and finish I/O, exit with error >>F0DA
F2C9 check PCB flags again <<P5D6>
F2CC need sub-index block?
F2CE no >>F2D5
F2D0 yes, go do it <P3BD>
F2D3 error? >>F2C1
F2D5 buy a new block for data <F411>
F2D8 error? >>F2C1
F2DA get PCB flags <<P5D6>
F2DD indicate index buffer changed
F2DF no new blocks needed now
F2E1 update PCB flags (D698)
F2E7 make index block offset from mark
F2EF store new block no. in index block (F4E)
F2FC and store it as current data block (F65A)
F306 set up buffer indexing <<P0F8>
F309 start writing <F311>
F30C go see if more blocks are needed >>F298
F30E I/O finish up when done >>F0E1

F311 ********** COPY WRITE DATA TO I/O BLOCK ***************

F311 ---
F314 lower request count by 1 (F7E6)
F31C ---
F31D copy partial page from caller's data
F31F to I/O block buffer
F324 ---
F327 next page in caller's area
F32B bump mark by 8100 (F7E3)
F333 still in same I/O block page?
F337 yes >>F31C
F33A no, clear overflow (I/O incomplete) >>F361

F33C any complete pages left to write? (F7E6)
F33F no >>F351
F341 yes, more in this page?
F342 yes >>F34A
F344 no, first block-page?
F348 no >>F34D
F34A yes, one less complete page to do (F7E6)
F34D readjust index
F34E continue with full page >>F324

F351 ---
F352 a few bytes left to write? >>F35E
F354 no, bump data buffer by 8100
F356 and mark (F7E3)
F35E set overflow (I/O complete) (F194)
F361 store LSB of mark (F7E2)
F364 and of request count (F7E5)
F368 indicate data block modified <<P5D6>
F36B and DIR entry needs update
F371 advance pointer into caller's buffer (F7E72)
F37C no, PCB flag to indicate write occurred <F42C>
F380 exit

F381 ********** ADD NEW MASTER INDEX BLOCK **********************

(MAKE A TREE FILE)

F381 add higher level <F3CA>
F384 error? >>F3C9
F386 get storage type <<F200>
F389 tree?
F38B yes >>F392
F38D no, add another level <F3CA>
F390 error? >>F3C9
F392 buy another block <F411>
F395 error? >>F3C9
F397 make offset into current index block (F7E4)
F39A from current mark
F39C point index to new block (F4E)
F3AB also save as current data block (F65A)
F3B5 checkpoint bitmap & key block <<E7D>
F3B8 error >>F3C9
F3BA zero out new index block >>ED45

F3BD ********** ADD NEW INDEX BLOCK ***********************

F3BD check storage type <<F200>
F3C2 seedling? >>F3CA
F3C4 no, read key index block <<E2A>
F3C7 and go add data block >>F392
F3C9 exit if error occurs
F3CA  buy a block <F411>
F3CD  error? >> F410
F3D2  save old key block number (D80C)
F3DA  make new block the key block (D80C)
F3E7  and current index block in PCB (D80F)
F3F0  store pointer to old key block
F3F3  in first position of new index
F3FA  checkpoint bitmap and new key block <E87D>
F3FD  error? >> F410
F3FF  get storage type <F208>
F404  upgrade it to next higher type (D807)
F407  indicate DIR entry needs update (D88B)
F410  exit

F411  ********* BUY A DISK BLOCK *********************
F411  allocate a disk block <E9A9C>
F414  error? >> F410
F416  get PCB flags <F5D6>
F419  indicate DIR entry needs update
F422  add 1 to blocks in use for file
F42F  ---
F430  exit

F431  ********* DO STATUS IF NO I/O YET *********************
F431  get PCB flags <F5D6>
F434  any buffers in use? (I/O activity)
F436  if so, assume its ok >> F42F
F438  no, (D881)
F438  select new device (BP30)

  *** STATUS CALL ***
F43E  Save Unit Number
F440  Save Block Number on stack
F446  Indicate Status call
F44A  Indicate Block 0
F44E  Go do I/O <D8E4>
F451  Restore Block Number to original value
F459  Exit

F45A  *********** MLI CLOSE CALL ***********

  F45A  check REF NUM
F455  specific close? >> F494

  *** CLOSE ALL OPEN FILES ***
F460  no errors yet (FE89)
F465  store PCB index (FE5D)
F469  get its level (D81B)
F46C  if below system LEVEL, skip it (BP94)
F46F  yes, skip it >> F486
F471  no, active PCB? (D800)
F474  no >> F486
F476  yes, flush it and update directory <F4F2>
F479  error? >> F4C7
F47B  no, close specific PCB <F499>
F480  is this a close-all?
F482  yes, ignore errors >> F486
F484  no, stop on error >> F4C7
F486  bump PCB index to next one (FE5D)
F48C  and continue >> F465
F48E  when all PCBs checked, load error number (FE89)
F491  no error >> F4C5
F493  error exit

  *** CLOSE SPECIFIC FILE ***
F494  flush it <F4FA>
F497  error? >> F4C7
F499  get buffer number (FE5D)
F49F  free its pages <F111>
F4A2  error? >> F4C7
F4A4  release PCB
F4AC  set DEVNUM (D881)
F4B2  find VCB for device <B859>
F4B5  decrement count of open files in VCB (FE5C)
F4BB  some are open... >> F4C5
F4BD  if all are closed, turn off (D911)
F4CE "files open" flag
F4C5  ---
F4C6  exit
F4C7  Branch to handle close error >> F4F7
F4C9 ****************************
***** MLI FLUSH CALL *****
***************************

F4C9 flush specific file?
F4CF yes >>F4FA
F4CF no, clear flush-all error code (FEB9)
F4D2 do all FCBs
F4D4 set FCB index for next FCB (FE5D)
F4DB is this file open? (D800)
F4DB no >>F4E2
F4DD yes, flush it <<F4F2>
F4E0 error? >>F4F7
F4E2 bump to next FCB (FE5D)
F4E8 and go flush it too >>F4D4
F4EA ---
F4E8 return with error code if any (FEB9)
F4F1 RETURN

F4F2 **************************** FLUSH A FILE & UPDATE DIRECTORY ****************************
F4F2 find buffer/VCB <E1E2>
F4F5 no error? >>F504
F4F7 go handle close error >>F5C7
F4FA zero out close-all error
F4FF validity check REF NUM <E1C7>
F502 error? >>F4F7
F504 is write access allowed? (D809)
F509 no, exit >>F4EA
F50B has a write ocurred since last flush? (DB1C)
F50E yes >>F517
F510 no, <<F5D6>
F513 does anything need flushing anyway?
F515 no, then exit now >>F4EA
F517 else, get FCB flags <<F5D6>
F51A has data buffer changed?
F51C no >>F523
F51E yes, checkpoint it <EE83>
F521 error? >>F4F7
F523 get flags again <<F5D6>
F526 has index buffer changed?
F528 no >>F52P
F52A yes, checkpoint it <EE97>
F52D error? >>F4F7
F52F ---
F536 copy file identifier data to my variables (D800)
F540 set DEVNUM (DF30)
F543 BLKNUM = current DIR block (FE25)
F549 read DIR block <EBC9>

F54C error? >>F4F7
F54E copy directory header <E695>
F551 are we in block with this file's entry? (FE27)
F55A no >>F561
F55F yes >>F568
F561 no, set new block number
F565 read it <EBD9>
F568 point at directory entry in block <<E499>
F56B copy file entry from directory <E59B>
F571 copy blocks used count to entry (D818)
F57F copy new EOF (D815)
F580 and new key block no. (D80C)
F593 isolate new storage type (D805)
F59D combine it with name length (FE2A)
F5A5 and update type/len field in entry (FE2A)
F5A8 write entry back to directory <E482>
F5AB error? >>F5C7
F5B0 turn off "write occurred" flag (DB1C)
F5BB same bitmap in memory (FE24)
F5BE no, exit now >>F5C5
F5C0 yes, checkpoint it also <EB76>
F5C5 no errors, exit
F5C6 RETURN

F5C7 **************************** CLOSE ERROR ****************************
F5C7 is this a close or flush all?
F5CC no >>F5D4
F5D0 yes, save error code (FEB9)
F5D3 RETURN
F5D4 else, real error right now
F5D5 RETURN

F5D6 **************************** GET FCB FLAGS ****************************
F5D6 load FCB flags (FE5D)
F5D9 from FCB (D808)
F5DC and exit

F5DD **************************** FILE ACCESS ERROR ****************************
F5DD exit with file access error code
F5D0 RETURN

F5E1 **************************** MLI SET_EOF CALL ****************************
### ProDOS MLI -- V1.2 -- 6 SEP 86

**ADDR** | **DESCRIPTION/CONTENTS**
---|---
F5E1 | get storage type <F200>
F5E4 | if DIR file...
F5E6 | its an access error >>F5DD
F5EB | else, save type for truncate to
F5E9 | mess with.
F5EF | write access permitted? (D809)
F5F4 | no, error >>F5DD
F5F6 | check device status <F431>
F5F9 | error? >>F5DD
F602 | copy EOF from FCB (D81B)
F610 | copy caller's new EOF
F61B | compare old EOF to new (F855)
F621 | if less than or equal to... >>F62B
F623 | if greater... >>F63D
  *** OLD EOF <= NEW EOF ***
  *** NO TRUNCATE NEEDED ***
F628 | new EOF beyond old
F62F | copy caller's EOF to FCB
F63A | exit by indicating flush needed >>F26C
  *** OLD EOF > NEW EOF ***
  *** TRUNCATE FILE ***
F63D | flush first <F40A>
F640 | error? >>F5E0
F642 | $43/$49 -- end of data block I/O buffer
F64C | compare current mark to new EOF (F85D)
F659 | it is prior to EOF >>F672
F661 | if past EOF, force mark back to EOF (F85D)
F672 | construct EOF block number and (F875)
F675 | byte offset into block from new (F891)
F678 | EOF mark. (F876)
F690 | on a block boundary? (F892)
F693 | yes >>F6B2
F695 | no, (F690)
F699 | decrement block by 1
F6A7 | but don't let it fall below 0
F6B2 | copy key block number (F85D)
F6C1 | set blocks freed to zero
F6C9 | truncate file at new EOF <F3E>
F6CC | save status
F6D4 | set new key block in FCB (F8B2)
F6DA | drop PCB block count by number (D818)
F6DD | of blocks freed in truncate routine. (F8BD)
F6EA | copy new storage type (F8BC)
F6F7 | turn off all block allocation flags <ED9F>
F6FA | update VCB free block count <F9BD>
F704 | copy mark (D812)

---

### ProDOS MLI -- V1.2 -- 6 SEP 86

**ADDR** | **DESCRIPTION/CONTENTS**
---|---
F70C | force current mark to infinity (D812)
F713 | go set mark <EC32>
F716 | no error? >>F71F
F718 | if error, indicate in saved status
F71E | but continue
F71F | copy caller's EOF to FCB <F628>
F722 | Flush and update <F40A>
F725 | no errors? >>F72E
F727 | if error, indicate in saved status
F72D | but continue
F72E | ---
F730 | exit

---

F731 | ************************************************************
** MLI GET_EOF CALL ******
************************************************************

---

F731 | ---
F736 | copy EOF to caller's list (D815)
F742 | exit -- no errors

---

F743 | ************************************************************
** MLI NEW LINE CALL ******
************************************************************

---

F743 | ---
F745 | copy newline mask
F74E | and newline character
F754 | return, no errors

---

F755 | ************************************************************
** MLI GET FILE INFO CALL ******
************************************************************

---

F755 | get the file entry <E593>
F758 | ok? >>F79C
F75A | no, bad path?
F75D | no, real error >>F789
F75F | else, make it VOL DIR type
F761 | with name length = 0 (F82A)
F766 | no free blocks needed (F85F)
F76C | go through the motions to update the (F85C)
F76F | VCB block count. <E964>
F775 | copy blocks free from VCB (D815)
F781 | copy total blocks on volume to AUX_ID (D913)
F78F | total - free = blocks used (F85F)
F79C | shift type down from high nibble (F2EA)
F7A8 | copy the data to caller's parm list (FDD7)
F7B9 | and exit
ProDOS MLI -- V1.2 -- 6 SEP 86

ADDRESS DESCRIPTION/CONTENTS

F7BA ****************************

***** MLI SET FILE INFO CALL *****

******************************************************************************

F7BA get the file entry <E593>
F7BD error? >>F7E4
F7BF indicate backup needed now (BF95)
F7CE copy 13 parms from caller's list to (FDD7)
F7D1 file entry staging area >>F7D8
F7D8 ---
F7DD if any spurious access bits are on...
F7E1 access error!
F7E4 RETURN

F7E5 else, anything in his modification date?
F7E9 no >>F7EE
F7EB yes, go update directory >>E4C2
F7EE no, use system date then update directory >>E4B2

F7F1 ****************************

***** MLI RENAME CALL *****

******************************************************************************

F7F1 follow path to file <E5A6>
F7F4 ok? >>F833
F7F6 no, bad name?
F7F8 no, real error >>F812

*** RENAME VOLUME ***

F7FA yes, copy new name <F917>
F7FD error? >>F812
F7FF get first length (D700)
F803 get next (D700)
F806 bad path if more than one name for vol >>F887
F80B files open on volume? (D911)
F80E no, continue >>F814
F810 yes, file open error
F812 ---
F813 RETURN

F814 make type/len for a VOL DIR HDR
F81B write new name to VOL HDR <F908>
F81E error? >>F889
F825 copy new name to device's VCB (D700)
F831 exit, no errors
F832 RETURN

F833 get path index <F925>
F836 copy old name with prefix to my buffer (D700)
F842 copy new name to buffer <F917>
F845 error? >>F889
F847 get path index <F925>
F84D compare all levels of names up to and (DC08)
F850 including the last. Find first which
F851 differ.
F855 save indicies into names which point to (F8E4)
F858 final name. (F8B5)
F85B ---
F865 exit if they match completely
F866 RETURN

F867 index to differing new name (F8E4)
F86A point past it (D700)
F872 must be the last! (D700)
F875 it isn't >>F887
F877 it is, (F8B5)
F87A do the same with the old name (DC08)
F885 difference is only in last index? >>F881
F887 no, bad path error
F889 ---
F88A RETURN

F88B names good. follow path to new file <E5A6>
F88E better get an error >>F894
F890 if found, duplicate name in directory
F893 RETURN

F894 if error, better be file not found
F896 or else its really an error... >>F889
F89B copy old pathname again <E891>
F89B get its file entry <E593>
F89E error? >>F889
F8A0 search PCB's <EF9B>
F8A1 exit if the file is open for write >>F889
F8A4 does ACCESS permit rename?
F8AC yes >>F8B2
F8AE no, access error
F8B0 ---
F8B1 RETURN

F8B2 get type/len from entry (FE2A)
F8B7 DIR file?
F8BD yes, ok >>F8C3
F8BB seedling, sapling or tree?
F8BD yes, ok >>F8C3
ProDOS MLI -- V1.2 -- 6 SEP 86

F8BF  else, compatibility error
F8C3  copy new path again <F917>
F8C6  error? >>F809
F8CB  get length of last name (FEB4)
F8D3  copy it and name to file entry buffer (D700)
F8E3  combine new len with type (D700)
F8E9  DIR file?
F8EB  no, go update entry and exit >>F985
F8ED  yes, (FE3B)
F8F3  read key block of this subdirectory <EBC9>
F8FB  error? >>F889
F8FB  copy new name to DIR HDR (D700)
F900  and update directory's key block <F908>
F903  error? >>F889
F905  go update directory entry and exit >>B4C2

F908  **COPY PATH TO BUFF & WRITE**

F908  copy type/len and path to my buffer
F914  go write the block >>EB05

F917  **POINT TO NEW NAME**

F917  COPY TO BUFFER
F917  $48/$49 --> second pathname
F922  go copy it >>EB0C

F925  **LOAD PATH INDEX**

F925  load pathname index
F92C  (including prefix if any) (BF9A)
F92F  --
F931  RETURN

F932  **MLI DESTROY CALL**

F932  get file entry <E593>
F935  error? >>F97E
F937  find PCB if any <EF9B>
F94A  PCB open? (FE62)
F94D  yes, file open error >>F97C
F94F  no free blocks needed
F947  go compute VCB free block count <E959>
F94A  ok? >>F950
F94C  error, disk full?
F94E  no, real error >>F97E
F950  DESTROY enabled in ACCESS? (FE48)
F955  yes >>F95C
F957  no, access error

F95C  check status of device (BF30)
F962  error? >>F97E
F964  point to key block (FE3B)
F973  DIR file?
F977  no >>F980
F979  yes, handle differently >>F988

F97C  File open error
F97E  --
F97F  RETURN

*** DESTROY NON-DIRECTORY FILE ***

F980  save the storage type (FEB3C)
F987  set EOF to zero (FEB3C)
F98D  byte offset = $200
F992  "truncate" the file at EOF=0 <FA3E>
F995  if error >>F97E
F997  free the key block in volume bitmap (FEB8B)
F9A8  error >>F97E
F9A3  mark the file as deleted in DIR
F9A7  decrement file count in DIR (FEB1E)
F9B2  checkpoint volume bit map <EB7C>
F9B5  error >>F97E
F9B7  update free block count in VCB <F9B0>
F9B8  and go update the directory >>EB42

*** SUBROUTINE TO UPDATE FREE BLOCK ***

*** COUNT IN VCB ***

F9B0  add blocks freed to total free blocks (FEB5C)
F9C0  in VCB. (FEB0D)
F9D2  start next search for free blocks at
F9D4  start of bitmap. (D91C)
F9D7  exit

*** DESTROY DIRECTORY FILE ***

F9D8  DIR file?
F9DA  no, error >>FA27
F9DC  read volume bitmap block <EB43>
F9DF  error? >>FA26
F9E1  BLKNUM = key block pointer (FE3B)
F9EB  read it <EB59>
F9EE  compute VCB? >>FA26
F9F0  if DIR has any files... (DC25)
F9FA  access error
F9FF  write back block marking entry free (DCA4)
FA05  error? >>FA26
FA07  if "next pointer" is zero... (DC02)
FA11  go back and pretend it's a seedling >>F997
ProDOS MLI - V1.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

FA13 else, (DC03)
FA16 free next block <EA1A>
FA19 error? >>FA26
FA1B BLKNUM = next block (DC02)
FA21 read it <EBC9>
FA24 if ok, continue in loop >>FA07
FA26 else, error exit

FA27 incompatible file format error

FA2C ******** SET WRITE OCCURRED FLAG **********************

FA2C save some registers
FA2F indicate write occurred (FE5D)
FA3A restore registers and exit
FA3D RETURN

FA3E ******** TRUNCATE FILE AT EOP **********************

FA3E check storage type*16 (FEBC)
FA41 seedling?
FA43 yes >>FA52
FA45 no, sapling?
FA47 yes >>FA55
FA49 no, tree?
FA4B yes >>FA58
FA4D General error--wrong storage type
FA4F jump to system death <BF0C>

FA52 go to seedling truncate >>FB24

FA55 go to sapling truncate >>FA08

FA58 truncate tree,
FA5A at most 128 blocks in master index (FE93)
FA5D read the master index <FB4F>
FA60 error? >>FABD
FA62 at EOF yet? (FE93)
FA68 yes >>FABE

*** FREE WHOLE INDEX BLOCKS AFTER EOF ***
(free 8 subindex blocks each time the
master index block is read since we must
share its buffer)

FA6A copy up to 8 non-zero index block
FA6C numbers to (DC08)
FA6F a handy table (FE95)
FA80 ***

FA89 if there weren't 8 left to do, zero (FE95)
FA8C remainder of the table (FE9D)

FA92 ---
FA93 update master index counter (FE93)
FA98 for all 8 entries: (FE94)
FA99 set BLKNUM, (FE95)
FAA3 (exit when a 0 entry is found), >>FA5D
FAAA read the sub-index block, <EBD9>
FAAD (quit if error), >>FABD
FAAF zero all its blocks, <FB7D>
FAB2 (quit if error), >>FABD
FABA and loop until all 8 are done. >>FA98
FABA then go back and reread master index >>FA5D
FABC normal exit
FABD RETURN

FABE now go free all the sub-index blocks (FE8F)
FAC2 which follow EOF <FB7F>
FAC5 if error >>FABD
FAC7 write back master index <EBD5>
FACA if error >>FABD
FACC EOF in first subindex? (FE8F)
FACF if so, demote to sapling file >>FAB6
FAD1 else, BLKNUM = subindex block which (DC00)
FAD4 contains the EOF mark
FAD9 (exit if none there) >>FABC
FAB0 else, read the final subindex block <EBD9>
FAB2 and treat it as a sapling file >>FAB0
FAB5 unless there is an error.

FAB6 Demote tree to sapling <FB5B>
FAB9 if error >>FABD

*** TRUNCATE SAPLING FILE ***

FABB read index block <FB4F>
FABE if error >>FABD
FAP0 index of last block in the file (FE90)
FAP3 add one to point past end of file
FAP4 if zero, no blocks to free >>FB00
FAP6 zero blocks past EOF <FB7F>
FAP9 if error >>FABD
FAPB write back modified index block <EBD5>
FAPZ if error >>FABD
FB00 index of last block in file (FE90)
FB03 this index block is empty! >>FB1A
FB05 get BLKNUM of last data block (DC00)
FB0D (no block allocated?) >>FABC
FB14 read in last data block <EBD9>
FB17 and treat it as a sapling file >>FB29
FB19 unless error occurred.
ProDOS MLI -- V1.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

FB1A more index blocks (tree file)? (FE9F)
FB1D yes, must be tree file >>FB85
FB1F no, demote to seedling <FB5B>
FB22 if error >>FB4E

*** TRUNCATE SEEDLING FILE ***

FB24 read key block <FB4F>
FB27 error? >>FB4E
FB29 EOF in first page? (FE92)
FB2C yes >>FB34
FB2E EOF in second page?
FB2F no, exactly 256 bytes >>FB4D
FB31 get byte offset (FE91)
FB34 --
FB36 zero bytes in second page (DD00)
FB3C EOF in first page? (FE92)
FB3F no, we're done. >>FB4A
FB41 yes, zero bytes in first page, too (FE91)
FB4A then write block back and exit >>EBD5

FB4D exit normally
FB4E RETURN

FB4F ********* READ INDEX BLOCK **********************

FB4F Put index block number in A,X (FEBA)
FB55 Go read the block >>BEC9

FB5B ********* DEMOTE FILE TO SMALLER FILE TYPE**********

FB5B get high byte of index block (FE88)
FB58 and low byte (FE8A)
FB61 free the index block in the volume bitmap <EA1A>
FB63 if error >>FB7C
FB66 establish first block of old index block (DC00)
FB69 as new index block. (FEBA)
FB73 reduce storage type by one (FE8C)
FB7B and exit
FB7C RETURN

FB7D ********* FREE ALL BLOCKS IN AN INDEX BLK ***********

FB7D --
FB7F save BLKNUM
FB85 Save Y-register (index within block) (FE68)
FB9B if it is non-zero...
FB97 free the block in the volume bitmap <EA1A>
FB9A if error >>FBAB
FB9C Restore index to Y-reg (FE68)

ProDOS MLI -- V1.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

FB9F zero this entry
FB97 --
FB98 loop through all entries >>FB85
FB9A save error message, if any
FBAD restore old BLKNUM
FBB3 and exit

FB4F ********* ALLOCATE I/O BUFFER **********************

FB4B --
FB4F get I/O buffer page number
FB99 can't be below 8800
FB9B else, error >>FBFF
FB9D can't be above $BC00
FBBF else, error >>FBFF
FB8A $4A/$4B --> I/O buffer
FB88 must be page aligned >>FBFF
FB8E --
FB8F check each page of I/O buffer for <FC3A>
FB82 prior allocation in system bit map (BF5B)
FB8D --
FB8E if ok, mark each page as allocated <FC3A>
FB83 in system memory bit map (BF58)
FB88 assign buffer number (REPNUM*2) in FC8 (D008)
FB89 and save buffer location in buffer list
FB9D exit
FB9E RETURN

FB9F bad I/O buffer error
FB97 exit

FC03 ********* LOCATE I/O BUFFER ***********************

FC03 --
FC04 AREG contains buffer number *2 (BF6E)
FC07 move buffer pointer to NXTBUF variable (FB8B)
FC10 exit

FC11 ********* FREE I/O BUFFER ***********************

FC11 is buffer already free? <FC03>
FC16 yes, exit >>FC3B
FC1A zero its address in system global page (BF6F)
FC27 --
FC2E free each page in buffer <FC3A>
FC23 by marking system bit map
FC30 exit
FC39 RETURN
FC3A ******* LOCATE BIT MAP POSITION ****************************
          (GIVEN PAGE NUMBER)
FC3A XREG contains page number
FC3B compute page number times 8
FC3E use as offset for bitmask (FDCA)
FC45 page number / 8 = byte offset
FC46 into bitmap
FC48 exit

FC49 ******* CHECK BUFFER VALIDITY *******************************
          START > $200   END < $BFO0

FC49 get buffer address (MSB)
FC4D must be >$200 else error >>FBFF
FC4F get length (FEA6)
FC55 compute last page no. of buffer
FC5A ---
FC61 may not extend into $BFO0
FC63 else, error >>FBFF

*** CHECK IF BLOCK OF MEMORY IS FREE ***

FC66 ---
FC67 see if this page is allocated <FC3A>
FC6D if so, error >>FBFF
FC6F else, check other page also
FC73 then exit if both have been checked
FC74 RETURN

PC75 *********************** MLI GET BUFF CALL *********************

PC75 get next available buffer
PC7A put its address in caller's parm list
PC7B and exit
PC78 RETURN

PC84 ******************** MLI SET BUFF CALL ********************

PC84 mark his buffer allocated
PC89 error? >>PCAB
PC8B get old buffer address (FEA9)
PC95 free old buffer's pages in map <PC2D>
PC98 copy old buffer contents
PC9E to new buffer
PCAA then exit

ProDOS MLI -- V1.2 -- 6 SEP 86

----- NEXT OBJECT ADDR: FC39

ADDR DESCRIPTION/CONTENTS

-----

FC3A

----- NEXT OBJECT ADDR: FCAB

ADDR DESCRIPTION/CONTENTS

-----

FCAB

RETUR

FCAC ******* GO TO QUIT CODE HANDLER ******************************

FCAC enable 2nd 4K bank of language card (C081)
FCAP (Quit code lives at $D100-$D3FF) (C083)
FC84 Get first four bytes of page 0, ($000)
FC87 save them on the stack
FC8B Set ($80) -> $D100
FC8D Set ($82) -> $1000
FC89 Set Y = 0
FC8A 3 pages of code to copy
FC8C ---
FC8D copy quit code handler to $1000

The next five lines of code were added for this version (1.2). Fortunately they did not survive the next version (1.3). Let's hope that whoever wrote this "fancy" code is now working for Commodore.

FC8D pull 4 saved bytes off stack
FC8E and restore them to page 0 ($F84)
FC84 enable HIGH RAM BANK1 (C88B)
FC87 (MLI) (C88B)
FC8C point RESET vector at $1000 ($83F2)
FC84 set power-up byte properly
FC89 go to quit code handler at $1000 $1000

FCFC ******* ACCESS RAM-BASED DEVICE DRIVER ********************

This (undocumented?) routine allows a device driver to reside in BANK2 of auxiliary high RAM (they normally reside in slot ROM). When the device driver is set up, the address of this routine, which may be found at $3EA, becomes the address of the device driver. Bytes $3E4 and $3E5 are changed to the address of the real driver in aux high RAM. This routine must call the page 3 routine at $306 because the MLI is in main high RAM and will be swapped out. The page 3 routine calls the real driver and returns here with the error code, if any.

FCFC Get current P-reg in accumulator,
FCFE then save it on the stack
FCFF clear overflow flag
FD00 interrupts disabled?
FD02 not >>FD07
FD04 yes, set overflow flag (FD25)
FD07 disable interrupts
FD08 enable RAM, BANK2 (C883)
FD0E set carry, indicating error
FD8F indicate 6 bytes to move to aux z-page
FD11 Call real driver thru page 3 routine <03D6>
FD14 store error number (BF0F)
FD17 enable RAM, BANK1 (C88B)
ProcDOS MLI -- V1.2 -- 6 SEP 86

---

**FD10**

ADDR DESCRIPTION/CONTENTS

---

FD1D restore original P-reg
FD1F if error number is zero, (BFWF)
FD22 then indicate no error; >>FD25
FD24 otherwise indicate error
FD25 RETURN

---

**FD26** *********** INSTALL A SPECIAL IRQ HANDLER ***************

This routine calls a subroutine located
at $D400 in BANK2 of high RAM. It is called when
an MLI command $42 is executed. Its purpose
is to install a routine that handles unclaimed
interrupts. Apparently the user has to
provide the routine at $D400.

FD26 Switch to BANK2 of high RAM, (C093)
FD29 execute the program there, <D400>
FD2C then back to BANK1 (C09B)
FD2F and return.

---

**FD30**  *******************************************************

DATA AREA

*******************************************************

---

**FD30** *********** MLI COMMAND TABLE ****************************

IN HASH CODE ORDER: IF COMMAND IS...
ABCD EF GH (IN BINARY BITS) INDEX IS COMPUTED AS:
0000 EF GH +0000 ABCD

FD30 GET BU
FD31 UNUSED
FD32 UNUSED
FD33 ALLOC INTERRUPT
FD34 DEALLOC INTERRUPT
FD37 UNUSED
FD38 READ BLOCK
FD39 WRITE BLOCK
FD3A GET TIME
FD3B EXIT
FD3C CREATE
FD3D DESTROY
FD3F RENAME
FD40 SET FILE INFO
FD41 GET FILE INFO
FD42 ON LINE
FD43 SET PREFIX
FD44 OPEN
FD45 NEWLINE
### ProDOS MLI -- VI.2 -- 6 SEP 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD70</td>
<td>CREATE</td>
</tr>
<tr>
<td>FD72</td>
<td>DESTROY</td>
</tr>
<tr>
<td>FD74</td>
<td>RENAME</td>
</tr>
<tr>
<td>FD76</td>
<td>SET FILE INFO</td>
</tr>
<tr>
<td>FD78</td>
<td>GET FILE INFO</td>
</tr>
<tr>
<td>FD7A</td>
<td>ON LINE</td>
</tr>
<tr>
<td>FD7C</td>
<td>SET PREFIX</td>
</tr>
<tr>
<td>FD7E</td>
<td>GET PREFIX</td>
</tr>
<tr>
<td>FD80</td>
<td>OPEN</td>
</tr>
<tr>
<td>FD82</td>
<td>NEWLINE</td>
</tr>
<tr>
<td>FD84</td>
<td>READ</td>
</tr>
<tr>
<td>FD86</td>
<td>WRITE</td>
</tr>
<tr>
<td>FD88</td>
<td>CLOSE</td>
</tr>
<tr>
<td>FD8A</td>
<td>FLUSH</td>
</tr>
<tr>
<td>FD8C</td>
<td>SET MARK</td>
</tr>
<tr>
<td>FD8E</td>
<td>GET MARK</td>
</tr>
<tr>
<td>FD90</td>
<td>SET EOF</td>
</tr>
<tr>
<td>FD92</td>
<td>GET EOF</td>
</tr>
<tr>
<td>FD94</td>
<td>SET BUF</td>
</tr>
<tr>
<td>FD96</td>
<td>GET BUF</td>
</tr>
</tbody>
</table>

### ProDOS MLI -- VI.2 -- 6 SEP 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDAC</td>
<td>BLOCKS USED</td>
</tr>
<tr>
<td>FDAC</td>
<td>END OF FILE</td>
</tr>
<tr>
<td>FDB1</td>
<td>SPECIAL ID (must be 5 bits on)</td>
</tr>
<tr>
<td>FDB2</td>
<td>'HUSTON1' Author's name</td>
</tr>
<tr>
<td>FDB9</td>
<td>PREVIOUS BLOCK OF VOL DIR KEY BLOCK</td>
</tr>
<tr>
<td></td>
<td>THE FOLLOWING IS COPIED TO SUBDIR HDR+$20</td>
</tr>
<tr>
<td>FDBB</td>
<td>VERSION OF PRODOS</td>
</tr>
<tr>
<td>FDBC</td>
<td>MINIMUM VERSION</td>
</tr>
<tr>
<td>FDBD</td>
<td>ACCESS BYTE (D</td>
</tr>
<tr>
<td>FDBE</td>
<td>ENTRY LENGTH</td>
</tr>
<tr>
<td>FDBF</td>
<td>ENTRIES PER BLOCK</td>
</tr>
<tr>
<td>FDC0</td>
<td>FILE COUNT</td>
</tr>
<tr>
<td>FDC2</td>
<td>PARENT LSB (copied to SUBDIR HDR +$20)</td>
</tr>
<tr>
<td>FDC3</td>
<td>FILE TYPE (directory)</td>
</tr>
<tr>
<td>FDC4</td>
<td>BLOCK NUMBER</td>
</tr>
<tr>
<td>FDC6</td>
<td>NUMBER OF BLOCKS</td>
</tr>
<tr>
<td>FDC8</td>
<td>END OF FILE</td>
</tr>
</tbody>
</table>

### FDCB

<table>
<thead>
<tr>
<th>ADDR</th>
<th>BMASK TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDCC</td>
<td>10000000</td>
</tr>
<tr>
<td>FDDC</td>
<td>01000000</td>
</tr>
<tr>
<td>FDDD</td>
<td>00100000</td>
</tr>
<tr>
<td>FDDC</td>
<td>00010000</td>
</tr>
<tr>
<td>FDDC</td>
<td>00001000</td>
</tr>
<tr>
<td>FDDD</td>
<td>00000100</td>
</tr>
<tr>
<td>FDDC</td>
<td>00000010</td>
</tr>
<tr>
<td>FDDC</td>
<td>00000001</td>
</tr>
</tbody>
</table>

### FDD3

<table>
<thead>
<tr>
<th>ADDR</th>
<th>OFFSETS INTO FILE CONTROL BLOCKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDC3</td>
<td>(FCB's are at $D983-$DDFF)</td>
</tr>
</tbody>
</table>

### FDD3

<table>
<thead>
<tr>
<th>ADDR</th>
<th>KEY BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDD5</td>
<td># BLOCKS USED</td>
</tr>
<tr>
<td>FDD7</td>
<td>END OF FILE</td>
</tr>
</tbody>
</table>

### FDDA

<table>
<thead>
<tr>
<th>ADDR</th>
<th>SET/GET FILE_INFO OFFSETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDDA</td>
<td>ACCESS</td>
</tr>
<tr>
<td>FDDB</td>
<td>FILE TYPE</td>
</tr>
<tr>
<td>FDDC</td>
<td>AUX TYPE</td>
</tr>
<tr>
<td>FDDD</td>
<td>STORAGE TYPE</td>
</tr>
<tr>
<td>FDDF</td>
<td>BLOCKS USED (MSB on means GET only no SET</td>
</tr>
</tbody>
</table>

### FD98

<table>
<thead>
<tr>
<th>ADDR</th>
<th>COMMAND INFO BYTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD98</td>
<td>PATHNAME FLAG</td>
</tr>
<tr>
<td></td>
<td>DATE/TIME STAMP FLAG</td>
</tr>
<tr>
<td></td>
<td>COMMAND NUMBER</td>
</tr>
<tr>
<td>FD99</td>
<td>REFERENCE NUMBER FLAG</td>
</tr>
<tr>
<td>FD9A</td>
<td>1 0 1 - 00</td>
</tr>
<tr>
<td>FD9B</td>
<td>1 0 1 - 01</td>
</tr>
<tr>
<td>FD9C</td>
<td>1 0 1 - 02</td>
</tr>
<tr>
<td>FD9D</td>
<td>1 0 1 - 03</td>
</tr>
<tr>
<td>FD9E</td>
<td>1 0 1 - 04</td>
</tr>
<tr>
<td>FD9F</td>
<td>1 0 1 - 05</td>
</tr>
<tr>
<td>FD9G</td>
<td>1 0 1 - 06</td>
</tr>
<tr>
<td>FDA0</td>
<td>1 0 1 - 07</td>
</tr>
<tr>
<td>FDA1</td>
<td>1 0 1 - 08</td>
</tr>
<tr>
<td>FDA2</td>
<td>1 0 1 - 09</td>
</tr>
<tr>
<td>FDA3</td>
<td>1 0 1 - 0A</td>
</tr>
<tr>
<td>FDA4</td>
<td>1 0 1 - 0B</td>
</tr>
<tr>
<td>FDA5</td>
<td>1 0 1 - 0C</td>
</tr>
<tr>
<td>FDA6</td>
<td>1 0 1 - 0D</td>
</tr>
<tr>
<td>FDA7</td>
<td>1 0 1 - 0E</td>
</tr>
<tr>
<td>FDA8</td>
<td>1 0 1 - 0F</td>
</tr>
<tr>
<td>FDA9</td>
<td>1 0 1 - 10</td>
</tr>
<tr>
<td>FDAA</td>
<td>1 0 1 - 11</td>
</tr>
<tr>
<td>FDB</td>
<td>1 0 1 - 12</td>
</tr>
<tr>
<td>FDBA</td>
<td>1 0 1 - 13</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

ProDOS MLI -- 6 SEP 86
ProDOS MLI -- 6 SEP 86

FDE1 Datetime (Last Mod)
FDE5 Datetime (Creation)
FDE9 *********** FATAL ERROR MESSAGE ***********
FDB9 ' INSERT SYSTEM DISK AND RESTART -ERR 0 '
FE11 ---
FE11 ********** VARIABLES - DATA AREA ***********
FE11 Parent Pointer Block
FE13 Parent Entry Number
FE14 Parent Entry Length
FE15 Datetime (Creation)
FE19 Version
FE1A Min Version
FE1B Access Byte
FE1C Entry Length
FE1D Entries per Block
FE1E File Count
FE20 Bit Map Pointer
FE22 Total Blocks
  THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY
  A FILE:
FE24 Device Number
FE25 Current Directory Block Number (HDR)
FE27 Block Number of File Entry in Directory
FE29 File Entry Number in Directory
FE2A ******** FILE ENTRY BUFFER ***********
FE2A Type/Length (TTTTLLLL)
FE2B File Name (Max 15) >>000F
FE3A File Type
FE3B Key Pointer
FE3D Blocks Used
FE3F End of File
FE42 Datetime (Creation)
FE46 Version
FE47 Min Version
FE48 Access Attribute
FE49 Aux Type (Load Address/Record Length)
FE4B Datetime (Last Mod)

FE4F Header Pointer
FE51 ******** Variable Work Area ***********
FE51 3 Byte Scratch
FE54 ---
FE55 End of File
FE58 Previous Mark
FE5B Compare Vol Name Scratch
FE5C Offset into VCB Table ($D000)
FE5D Offset into FCB Table ($D000)
FE5E Free FCB found Flag
FE5F Number of Free Blocks needed
FE61 Storage Type
FE61 Number of Entries Examined or...
FE62 FCB already open flag
FE63 File Count
FE65 Entries/Block Loop Count/Free FCB's refnum
FE65 Free Entry Found Flag (if > 0) or...
FE66 bit for free
FE67 # Blocks in Bitmap left to search
FE68 Y Register temp
FE69 Pathname Length
FE6A Devnum for Prefix Directory Header
FE6B Block of Prefix Directory Header
FE6D Bitmap Byte Offset in Page
FE6E Bitmap Page Offset
FE6F Bitmap Buffer Page (0 or 1)
FE70 Bitmap Flag (if $00, needs writing)
FE71 Bitmap DEVENUM
FE72 Bitmap Block Number
FE74 Bitmap Block offset for Multiblock Bitmaps
  New Mark to be Positioned to for Set Mark
  or New Moving Mark (for READ)
FE75 or New EOF for SET_EOF
FE78 Request Count (Read/Write etc.)
FE7A Multi-Block I/O count
FE7B Newline character
FE7C Newline mask
FE7D I/O Transfer occurred flag
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE7E</td>
<td>MLI Command * 2</td>
</tr>
<tr>
<td>FE7F</td>
<td>ORed into Access Flags ($20 - Backup)</td>
</tr>
<tr>
<td>FE80</td>
<td>Duplicate Volume Flag (if $PF)</td>
</tr>
<tr>
<td>FE81</td>
<td>Duplicate Volume's VCB index</td>
</tr>
<tr>
<td>FE82</td>
<td>MLI function code (low 5 bits)</td>
</tr>
<tr>
<td>FE83</td>
<td>ONLINE: volname len - loop index</td>
</tr>
<tr>
<td>FE84</td>
<td>new pathname; index to last name</td>
</tr>
<tr>
<td></td>
<td>old pathname; index to last name or..</td>
</tr>
<tr>
<td>FE85</td>
<td>ONLINE: index to data buffer</td>
</tr>
<tr>
<td>FE86</td>
<td>Old PREFIXPTR value</td>
</tr>
<tr>
<td>FE87</td>
<td>Pathname fully qualified flag (if $PF)</td>
</tr>
<tr>
<td></td>
<td>Pathname: temp save area for index or..</td>
</tr>
<tr>
<td>FE88</td>
<td>ONLINE: DEVCNT</td>
</tr>
<tr>
<td>FE89</td>
<td>close-all error code</td>
</tr>
<tr>
<td>FE8A</td>
<td>Set EOF: new Key Block pointer</td>
</tr>
<tr>
<td>FE8C</td>
<td>New storage type (SET_EOF)</td>
</tr>
<tr>
<td>FE8D</td>
<td>Freed Blocks count</td>
</tr>
<tr>
<td>FE8F</td>
<td>EOF Block number (MSB then LSB)</td>
</tr>
<tr>
<td>FE91</td>
<td>EOF byte offset into Block</td>
</tr>
<tr>
<td>FE93</td>
<td>EOF - Master index counter</td>
</tr>
<tr>
<td>FE94</td>
<td>Save area for index into table below</td>
</tr>
<tr>
<td>FE95</td>
<td>********** DEVICE TABLE BUILT BY ONLINE **********</td>
</tr>
<tr>
<td></td>
<td>(also used by SET_EOF to keep track of 8 blocks to be freed at a time)</td>
</tr>
<tr>
<td>FE95</td>
<td>device table part one</td>
</tr>
<tr>
<td>FE9D</td>
<td>device table part two</td>
</tr>
<tr>
<td>FE9A</td>
<td>length of path, etc.</td>
</tr>
<tr>
<td>FE98</td>
<td>next buffer address</td>
</tr>
<tr>
<td>FE9A</td>
<td>16 byte stack save area</td>
</tr>
<tr>
<td>FE9B</td>
<td>6 byte zero page save area</td>
</tr>
<tr>
<td>FEC0</td>
<td>Jump Vector, used for indirect jumps</td>
</tr>
<tr>
<td>FEC2</td>
<td>********** $PEBF-$PEFF NOT USED **********</td>
</tr>
<tr>
<td>FEC2</td>
<td>not used</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.3 -- 2 DEC 86
NEXT OBJECT ADDR: D700

ADDR DESCRIPTION/CONTENTS

D700 MODULE STARTING ADDRESS

********************************************************************
* * PRODOS MACHINE LANGUAGE INTERFACE * *
* * VERSION 1.3 -- 2 DEC 86 * *
* *
********************************************************************

THE 1.3 VERSION OF THE PRODOS & MLI IS THE SAME AS VERSION 1.2,
INSTRUCTION FOR INSTRUCTION,
FROM THE START ($DE00) TO ADDRESS $F992. SOME
BYTES BEFORE $F992 CHANGE BECAUSE THEY REFER TO
ADDRESSES GREATER THAN $F992.

ONLY THE PART OF THE MLI FROM $F980 TO $FEFF IS
DOCUMENTED HERE FOR VERSION 1.3. REFER TO THE
1.2 VERSION FOR THE FIRST PART OF THE MLI.

F980 ---

*** DESTROY NON-DIRECTORY FILE ***

F980 save the storage type (FE85)
F987 set EOF to zero (FE85)
F988 byte offset = $280
F992 turn on destroy flag (FE88)
F998 "truncate" the file at EOF=0 <FA44>
F999 turn off the destroy flag (FE88)
F99A error during truncation >>F97E
F9AA error >>F97E
F9AB mark the file as deleted in DIR
F9AD decrement file count in DIR (FE47)
F9BC checkpoint volume bitmap <EB76>
F9BD update free block count in VCB <F9C3>
F9C0 and go update the directory >>E4B2

*** SUBROUTINE TO UPDATE FREE BLOCK ***
*** COUNT IN VOLUME CONTROL BLOCK ***

F9C3 add blocks freed to total free blocks (FE85)
F9C5 in VCB. (FE86)
F9DA start next search for free blocks at
F9DD exit

F9DE DIR file?
F9EA no, error >>FA2D
F9EC read volume bitmap block <EB43>
F9ED error? >>FA2C
F9EE BLKNUM = key block pointer (FE64)
F9EF read it <EBD9>
F9F0 errors? >>FA2C
F9F0 if DIR has any files... (DC25)
FA00 access error
FA05 write back block marking entry free (DC04)
FA08 error? >>FA2C
FA0D if "next_pointer" is zero... (DC02)
FA17 go back and pretend it's a seedling >>F99D
FA19 else, (DC03)
FA1C free next block <EALA>
FA1F error? >>FA2C
FA21 BLKNUM = next block (DC02)
FA27 read it <BBC9>
FA2A if ok, continue in loop >>FA0D
FA2C else, error exit

FA2D incompatable file format error

FA32 ****** SET WRITE OCCURRED FLAG ******

FA32 save some registers
FA35 indicate write occurred (FE86)
FA40 restore registers and exit
FA43 RETURN

FA44 ****** TRUNCATE FILE AT EOF ******

FA44 check storage type*16 (FE85)
FA47 seedling?
FA49 yes >>PA5B
FA4B no, mapping?
FA4D yes >>PA5B
FA4F no, tree?
FA51 yes >>PA58
PA53 General error--wrong storage type
PA55 jump to system death (BF0C)
PA58 go to seedling truncate >>FE2F
FA5B  go to sapling truncate >>FAF6
FA5E  truncate tree,
FA50  at most 128 blocks in master index (FEBC)
FA63  read the master index <FB5A>
FA66  error? >>PAC8
FA68  at EOF yet? (FEBC)
FA6E  yes >>FA9

*** FREE WHOLE INDEX BLOCKS AFTER EOF ***
(free 8 subindex blocks each time the master index block is read since we must share its buffer)

FA70  copy up to 8 non-zero index block
FA72  numbers to (DCB0)
FA75  a handy table (FEBE)
FA86  ---
FA8F  if there weren't 8 left to do, zero (FEBE)
FA92  remainder of the table (FEBC)
FA98  ---
FA99  update master index counter (FEBC)
FA9E  for all 8 entries: (FEBD)
FAA1  set BLKNUM, (FEBE)
FAA9  (exit when a 0 entry is found), >>FA63
FAB0  read the sub-index block, <EBD9>
FAB3  (quit if error), >>PAC8
FAB5  zero all its blocks (if truncating) <FB95>
FAB8  or swap pages (if destroying),
FA8B  (quit if error), >>PAC8
FA8A  write the former index block back out, <EBD5>
FAA3  and loop until all 8 are done. >>FA9E
FA9C  then go back and reread master index >>FA63
FA97  normal exit
FA98  RETURN

FA9C  now go free all the sub-index blocks (FEB8)
FACD  which follow EOF <FB97>
FA8B  if error >>PAC8
FA92  write back master index <EBD5>
FA95  if error >>PAC8
FA97  EOF in first subindex? (FEB0)
FADA  if so, demote to sapling file >>FAF1
FAD3  else, BLKNUM = subindex block which (DCB0)
FADF  contains the EOF mark
FAE4  (exit if none there) >>FA7
FAE8  else, read the final subindex block <EBD9>
FAEE  and treat it as a sapling file >>FAF8
FAF0  unless there is an error.

FAF1  Demote tree to sapling <FB63>
FAF4  if error >>PAC8

*** TRUNCATE SAPLING FILE ***
FAF6  read index block <FB5A>
FAF9  if error >>PAC8
FAF8  index of last block in the file (FEB9)
FAFE  add one to point past end of file
FAFF  if zero, no blocks to free >>FB66
FB81  zero blocks past EOF (when truncating) <FB97>
or swap bytes for all but first block (when destroying).
FB84  if error >>PAC8
FB86  write back modified index block <EBD5>
FB89  if error >>PAC8
FB88  index of last block in file (FEB9)
FB8E  this index block is empty!! >>FB25
FB10  Get BLKNUM of last data block (DCB0)
FB18  (no block allocated?) >>FA79
FB1F  read in last data block <EBD9>
FB22  and treat it as a seedling file >>FB34
FB24  unless error occurred.
FB25  more index blocks (tree file)? (FEB8)
FB2B  yes, must be tree file >>FB10
FB2A  no, demote to seedling <FB63>
FB2D  if error >>PB9

*** TRUNCATE SEEDLING FILE ***
FB2F  read key block <FB5A>
FB32  error? >>PB8
FB34  EOF in first page? (FEB8)
FB37  yes >>FB3F
FB39  EOF in second page?
FB3A  no, exactly 256 bytes >>PB58
FB3C  get byte offset (FEBA)
FB3F  ---
FB41  zero bytes in second page (DCB0)
FB47  EOF in first page? (FEB8)
FB4A  no, we're done. >>PB55
FB4C  yes, zero bytes in first page, too (FEBA)
FB55  then write back block and exit >>EBD5
FB58  exit normally
FB59  RETURN
ProDOS MLI -- V1.3 -- 2 DEC 86

ADDR DESCRIPTION/CONTENTS

FB59A ********* READ INDEX BLOCK ***************

FB5A Put index block number in A.X (FE83)
FB60 Go read the block >>FB9C

FB63 ******** DEMOTE FILE TO SMALLER FILE TYPE**************

FB63 get high byte of index block (FB4)
FB66 save it on stack
FB68 get low byte (FEB3)
FB6B save it, too
FB6C free the index block in the volume bitmap <EAA>
FB6F restore the block number of the index block
FB70 to zero page.
FB75 if error writing bitmap >>FB94
FB77 New index block is first block (DC00)
FB7A from old index block. (FEB3)
FB83 For first entry in old index block,
FB85 zero the block number (if truncating) <FB7C>
FB89 reduce storage type by one (FEB5)
FB91 Write the deleted index block back out. <EB05
FB94 RETURN

FB95 ********* FREE ALL BLOCK NUMBERS IN ***************

*** AN INDEX BLOCK ***

FB95 ---

FB97 ******** FREE BLOCK NUMBERS BEYOND EOF **************

FB97 save BLKNUM
FB9D Save Y-register (index within block) (FE91)
FBAB if it is non-zero....
FBAB free the block in the volume bitmap <EAA>
FBB2 if error >>FBBE
FBB4 Restore index to Y-reg (F901)
FBB7 zero this entry (when truncating) or <FB7C>
FB0 swap the two bytes (when destroying).
FBBA ---
FBBB loop through all entries >>FB9D
FBBE save error message, if any
FBC0 restore old BLKNUM
FBC6 and exit
**ProDOS MLI -- V1.3 -- 2 DEC 86**  
**NEXT OBJECT ADDR: FC2C**

**ADDR**  | **DESCRIPTION/CONTENTS**
--- | ---
**FC2C**  | **LOCATE I/O BUFFER**
--- | ---
**FC2C** | **--**
**FC2D** | AREG contains buffer number *2 (BBF6E)
**FC30** | move buffer pointer to NXXBUF variable (FED1)
**FC39** | exit

**FC3A**  | **FREE I/O BUFFER**
--- | ---
**FC3A** | is buffer already free? <FC2C>
**FC3F** | yes, exit >>PC61
**FC43** | zero its address in system global page (BBF6F)
**FC58** | **--**
**FC51** | free each page in buffer <FC63>
**FC54** | by marking system bit map
**FC61** | exit
**FC62** | RETURN

**FC63**  | **LOCATE BIT MAP POSITION**
--- | ---
**FC63** | XREG contains page number
**FC64** | compute page number times 9
**FC67** | use as offset for bitmask (FDF4)
**FC6F** | page number / 8 = byte offset
**FC6F** | into bitmap
**FC71** | exit

**FC72**  | **CHECK BUFFER VALIDITY**
--- | ---
**FC72** | START > $200 END < $BF00
**FC76** | get buffer address (MSB)
**FC78** | must be >$200 else error >>PC28
**FC7E** | get length (FECF)
**FC81** | compute last page no. of buffer
**FC83** | may not extend into $BF00
**FC8C** | else, error >>FC28

*** CHECK IF BLOCK OF MEMORY IS FREE ***

**FC8F** | **--**
**FC90** | see if this page is allocated <FC63>
**FC96** | if so, error >>FC28
**FC98** | else, check other page also
**FC9C** | then exit if both have been checked
**FC9D** | RETURN

**ProDOS MLI -- V1.3 -- 2 DEC 86**  
**NEXT OBJECT ADDR: FC9D**

**ADDR**  | **DESCRIPTION/CONTENTS**
--- | ---
**FC9E**  | **MLI GET BUFF CALL**
--- | ---
**FC9E** | get next available buffer
**FC9E** | FCA3 put its address in caller's parmlist
**FCAB** | and exit
**FCAC** | RETURN

**FCAD**  | **MLI SET BUFF CALL**
--- | ---
**FCAD** | mark his buffer allocated
**FCB2** | error? >>FCD4
**FCB4** | get old buffer address (FED2)
**FCBE** | free old buffer's pages in map <FC49>
**FCCE** | copy old buffer contents
**FCC7** | to new buffer
**FCD3** | then exit
**FCD4** | RETURN

**FCD5**  | **GO TO QUIT CODE HANDLER**
--- | ---
**FCD5** | enable 2nd 4K bank of language card (C093)
**FCD8** | (Quit code lives at $D10W-$DF3F) (C093)
**FCDG** | get first four bytes of page 0, (0000)
**FCE0** | save them on the stack
**FCE4** | Set ($80) -> $D100
**FCE6** | Set ($82) -> $1000
**FCF2** | Set Y = 0
**FCF3** | 3 pages of code to copy
**FCF5** | ---
**FCF6** | copy quit code handler to $1000
**FDF4** | pull 4 saved bytes off stack
**FDFD** | copy HIGH RAM BANK1 (C0B8)
**FD18** | (MLI) (C0B8)
**FD15** | point RESET vector at $1000 (03F2)
**FD1D** | set power-up byte properly
**FD22** | go to quit code handler at $1000 >>1000

**FD25**  | **ACCESS RAM-BASED DEVICE DRIVER**
--- | ---
This (undocumented?) routine allows a device driver to reside in BANK2 of auxiliary high RAM (they normally reside in slot ROM). When the device driver is set up, the address of this routine, which may be found at $3E4, becomes the address of the device driver. Bytes $3E4 and $3E5 are changed to the address of the real driver in aux high RAM. This routine must call the page 3 routine at $3D6 because the MLI is in main high RAM and will be
swapped out. The page 3 routine calls the real driver
and returns here with the error code, if any.

FD25 Get current P-reg in accumulator,
FD27 then save it on the stack
FD28 clear overflow flag
FD29 interrupts disabled?
FD2B no >>FD30
FD2D yes, set overflow flag (FD4E)
FD30 disable interrupts
FD31 enable RAM, BANK2 (C803)
FD37 set carry, indicating error
FD38 indicate 6 bytes to move to aux z-page
FD3A Call real driver thru page 3 routine 03D6>
FD3D store error number (BF8F)
FD40 enable RAM, BANK1 (C888)
FD46 restore original P-reg
FD4B if error number is zero, (BF8F)
FD4D otherwise indicate error
FD4E RETURN

FD4F **************** INSTALL UNCLAIMED IRQ HANDLER ***********************
This routine calls a subroutine located
at $D400 in BANK2 of high RAM. It is called
when MLI command $42 is executed. Its purpose
is to install a routine that handles unclaimed
interrupts. Apparently the user must supply
the routine at $D400.

FD4F Switch to BANK2 of high RAM, (C803)
FD52 execute the program there, <D400>
FD55 then back to BANK1 (C888)
FD58 and return.

FD59 ******************** DATA AREA ********************

FD59 ******************** MLI COMMAND TABLE ********************
IN HASH CODE ORDER: IF COMMAND IS... 
ABCD EFGH (IN BINARY Bits)
INDEX IS COMPUTED AS: 
SHS H S EFGH
+0000 ABCD

FD59 GET BUF
FD5A UNUSED
FD5B UNUSED
FD5C UNUSED
FD5D ALLOC INTERRUPT
**ProDOS MLI -- V1.3 -- 2 DEC 86**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
FD90 | WRITE
FD91 | CLOSE
FD92 | FLUSH
FD93 | SET MARK
FD94 | GET MARK
FD95 | UNUSED
FD96 | SET EOF
FD97 | GET EOF
FD98 | SET BUF
FD99 | ********** MLI COMMAND ADDRESS TABLE **********
FD99 | CREATE
FD9B | DESTROY
FD9D | RENAME
FD9F | SET FILE INFO
FDB1 | OPEN
FDB2 | NEWLINE
FDB4 | READ
FDB5 | WRITE
FDB6 | CLOSE
FDB8 | FLUSH
FDB9 | SET MARK
FDBA | GET MARK
FDBC | SET EOF
FDBD | GET EOF
FDBF | SET BUF
FDC0 | GET BUF
FDC1 | ********** MLI COMMAND INFO BYTE **********

<table>
<thead>
<tr>
<th>PATHNAME FLAG</th>
<th>REFERENCE NUMBER FLAG</th>
<th>DATETIME STAMP FLAG</th>
<th>COMMAND NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDCA</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FDCC</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FDCE</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FD8F</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FD8D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FD8D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FD8D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FD8D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FD8D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FD8D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**ProDOS MLI -- V1.3 -- 2 DEC 86**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
FDCC | 0 | 1 | 0 | 0B |
FDCC | 0 | 1 | 0 | 0C |
FDCC | 0 | 1 | 0 | 0D |
FDCC | 0 | 1 | 0 | 0E |
FDCC | 0 | 1 | 0 | 0F |
FDCC | 0 | 1 | 0 | 10 |
FDCC | 0 | 1 | 0 | 11 |
FDCC | 0 | 1 | 0 | 12 |
FDCC | 0 | 1 | 0 | 13 |

**FD55** | ********** CONSTANTS - DATA AREA **********

<table>
<thead>
<tr>
<th>FD55</th>
<th>Blocks Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD57</td>
<td>End of File</td>
</tr>
<tr>
<td>FD8A</td>
<td>Special ID (Must be 5 bits on)</td>
</tr>
<tr>
<td>FD8B</td>
<td>'HUSTOD'</td>
</tr>
<tr>
<td>FD8E</td>
<td>Previous Block of Vol Dir Key Block</td>
</tr>
</tbody>
</table>

---

**FD54** | Version of ProDOS
**FD55** | Minimum Version
**FD56** | Access Byte (D|Rn|B|000|W|R)
**FD57** | Entry Length
**FD58** | Blocks per Block
**FD59** | File Count
**FD5A** | Parent LSB (copied to SUBDIR HDR+20)

---

**FD5C** | File Type (Directory)
**FD5D** | Block Number
**FD5F** | Number of Blocks
**FD61** | End of File

**FD64** | ********** BITMASK TABLE **********

<table>
<thead>
<tr>
<th>FD64</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD65</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FD66</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FD67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FD68</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FD69</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FD6A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FD6B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**FD5C** | ********** OPSET INTO FILE CONTROL BLOCKS **********
**FD5C** | ***(FCB's are at $D800-$D8FF)**********

**FD5C** | Key Block
ProDOS MLI -- vl.3 -- 2 DEC 86

FDFE  # Blocks Used
FE30  End of File
FE83  ******* SET/GET FILE_INFO OFFSETS  ****************************

FE83  Access
FE84  File Type
FE85  Aux Type
FE87  Storage Type
FE88  Blocks Used (MSB on means GET only no SET
FE8A  Datetime (Last Mod)
FE8E  Datetime (Creation)

FE12  ******* PATAL ERROR MESSAGE  ****************************

FE12  ' INSERT SYSTEM DISK AND RESTART -ERR 0 '

FE3A  ---

FE3A  ******* VARIABLES - DATA AREA  ****************************

FE3A  Parent Pointer Block
FE3C  Parent Entry Number
FE3D  Parent Entry Length
FE3E  Datetime (Creation)
FE42  Version
FE43  Min Version
FE44  Access Byte
FE45  Entry Length
FE46  Entries per Block
FE47  File Count
FE49  Bit Map Pointer
FE4B  Total Blocks

THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY
A FILE:

FE4D  Device Number
FE4E  Current Directory Block Number (HDR)
FE50  Block Number of File Entry in Directory
FE52  File Entry Number in Directory

FE53  ******* FILE ENTRY BUFFER  ****************************

FE53  Type/Length (TTTLLLLL)
FE54  File Name (Max 15) >>0000F
FE63  File Type
FE64  Key Pointer
FE66  Blocks Used

FE68  End of File
FE6B  Datetime (Creation)
FE6F  Version
FE70  Min Version
FE71  Access Attribute
FE72  Aux Type (Load Address/Record Length)
FE74  Datetime (Last Mod)
FE78  Header Pointer

FE7A  ******* Variable Work Area  ****************************

FE7A  3 Byte Scratch
FE7D  ---
FE7E  End of File
FE81  Previous Mark
FE84  Compare Vol Name Scratch
FE85  Offset into VCB Table ($D900)
FE86  Offset into FCB Table ($D800)
FE87  Free FCB Found Flag
FE88  Number of Free Blocks needed
FE8A  Storage Type
      Number of Entries Examined or...
FE8B  FCB already open flag
FE8C  File Count

FE8E  Entries/block Loop Count/Free FCB's refnum
      Free Entry Found Flag (if > 0) or...
      # of 1st bitmap block with free bit on or...
FE8F  bit for free
FE90  # Blocks in Bitmap left to search
FE91  Y Register temp
FE92  Pathname Length
FE93  Devnum for Prefix Directory Header
FE94  Block of Prefix Directory Header
FE96  Bitmap Byte Offset in Page
FE97  Bitmap Page Offset
FE98  Bitmap Buffer Page (0 or 1)
FE99  Bitmap Flag (if $80, needs writing)
FE9A  Bitmap DEVNUM
FE9B  Bitmap Block Number
FE9D  Bitmap Block offset for Multiblock Bitmaps
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.3 -- 2 DEC 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE9E</td>
<td>New Mark to be Positioned to for Set Mark or New Moving Mark (for READ)</td>
</tr>
<tr>
<td>FE9E</td>
<td>or New EOF for SET_EOF</td>
</tr>
<tr>
<td>FEA1</td>
<td>Request Count (Read/Write etc.)</td>
</tr>
<tr>
<td>FEA3</td>
<td>Multi-Block I/O count</td>
</tr>
<tr>
<td>FEA4</td>
<td>Newline character</td>
</tr>
<tr>
<td>FEA5</td>
<td>Newline mask</td>
</tr>
<tr>
<td>FEA6</td>
<td>I/O Transfer occurred flag</td>
</tr>
<tr>
<td>FEA7</td>
<td>MLI Command * 2</td>
</tr>
<tr>
<td>FEA8</td>
<td>ORed into Access Flags ($20 - Backup)</td>
</tr>
<tr>
<td>FEA9</td>
<td>Duplicate Volume Flag (if $FF)</td>
</tr>
<tr>
<td>FEA9</td>
<td>Duplicate Volume's VCB index</td>
</tr>
<tr>
<td>FEA9</td>
<td>MLI function code (low 5 bits)</td>
</tr>
<tr>
<td>FEA9</td>
<td>Characters in current Pathname index lvl or old pathname: index to last name</td>
</tr>
<tr>
<td>FEA9</td>
<td>new pathname: index to last name</td>
</tr>
<tr>
<td>FEA9</td>
<td>FEA9 ONLINE: index to data buffer</td>
</tr>
<tr>
<td>FEA9</td>
<td>Old PFIXPTR value</td>
</tr>
<tr>
<td>FEA9</td>
<td>FEB0 Pathname fully qualified flag (if $FF)</td>
</tr>
<tr>
<td>FEB0</td>
<td>Pathname: temp save area for index or..</td>
</tr>
<tr>
<td>FEB1</td>
<td>ONLINE: DEVCNT</td>
</tr>
<tr>
<td>FEB2</td>
<td>close-all error code</td>
</tr>
<tr>
<td>FEB3</td>
<td>Set EOF: new Key Block pointer</td>
</tr>
<tr>
<td>FEB5</td>
<td>New storage type (SET_EOF)</td>
</tr>
<tr>
<td>FEB6</td>
<td>Freed Blocks count</td>
</tr>
<tr>
<td>FEB8</td>
<td>EOF Block number (MSB then LSB)</td>
</tr>
<tr>
<td>FEB9</td>
<td>EOF byte offset into Block</td>
</tr>
<tr>
<td>FEB9</td>
<td>EOF - Master index counter</td>
</tr>
<tr>
<td>FEB9</td>
<td>Save area for index into table below</td>
</tr>
</tbody>
</table>

FEBC *********** $FEBC-$FEFF NOT USED ****************************************

FEBC not used

FEBC *********** DEVICE TABLE BUILT BY ONLINE ***********************

(also used by SET_EOF to keep track of 8 blocks to be freed at a time)

| FEBC | device table part one |
| FEBC | device table part two |
| FEBC | length of path, etc. |
| FEBC | next buffer address |
| FEBC | 16 byte stack save area |
| FEBC | 6 byte zero page save area |
| FF9E | --- |
| FF9E | Jump Vector, used for indirect jumps |
| FF9E | Destroy flag (1 = destroy operation) |
ProDOS System Global Page -- V1.2

**MODULE STARTING ADDRESS**

* ProDOS System Global Page
* Version 1.2 -- 6 Sep 86
* Version 1.3 -- 2 Dec 86

DE00 Main MLI entry point.
DEAC Address for no device connected.
DF4E IRQ handler within MLI.
DFFF System error handler.
E009 System death handler.
F0D8 Patch in ProDOS IRQ Handler.

**JUMP VECTORS**

BF00 ENTRY JMP to MLI. >>BF4B
BF03 JSHARE System death address. >>BF03
BF06 DATETIME JMP To Date/Time routine (RTS if no clock).
BF07 Normal clock code address.
BF09 SYSEX JMP to system error handler. >>DFFF
BF0C SYSCMD JMP to system death handler. >>E009
BF0F SERR System error number.

**DEVICE INFORMATION**

BF10 DEVAD1 Slot 0 reserved.
BF12 DEVAD11 Slot 1, drive 1 device driver address.
BF14 DEVAD2 Slot 2, drive 1 device driver address.
BF16 DEVAD3 Slot 3, drive 1 device driver address.
BF18 DEVAD4 Slot 4, drive 1 device driver address.
BF1A DEVAD5 Slot 5, drive 1 device driver address.
BF1C DEVAD6 Slot 6, drive 1 device driver address.
BF1E DEVAD7 Slot 7, drive 1 device driver address.
BF20 DEVADR0 Slot 0 reserved.
BF22 DEVADR1 Slot 1, drive 2 device driver address.
BF24 DEVADR2 Slot 2, drive 2 device driver address.
BF26 DEVADR3 Slot 3, drive 2 device driver address.
BF28 DEVADR4 Slot 4, drive 2 device driver address.
BF2A DEVADR5 Slot 5, drive 2 device driver address.
BF2C DEVADR6 Slot 6, drive 2 device driver address.
BF2E DEVADR7 Slot 7, drive 2 device driver address.

**INTERRUPT INFORMATION**

BF90 INT 1 Interrupt handler address (highest priority).
BF92 INTERRUPT 2 Interrupt handler address.
BF94 INTERRUPT 3 Interrupt handler address.
BF96 INTERRUPT 4 Interrupt handler address (lowest priority).
BF98 INT A Register save area.
BF9A INT X Register save area.
BF9C INT Y Register save area.
BF9E INT S Register save area.
BF9C INT F Register save area.
BF9D INTBANKID Bank ID byte (ROM, RAM1, or RAM2).
BF9E INTADDR Interrupt return address.

**GENERAL SYSTEM INFO**

BF90 DATE YYYYMMDDDD.
BF92 TIME HHHHHHHM.
BF94 LEVEL Current file level.
BF95 BUBIT Backup bit.
BF96 SPARE1 Currently unused.
BF98 MACHID Machine ID byte.

Note: The text is a detailed description of the ProDOS System Global Page, including the starting address, jump vectors, device information, interrupt information, and general system information.
ProDOS System Global Page -- V1.2

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>01.. 0</td>
<td>II+</td>
</tr>
<tr>
<td>10.. 0</td>
<td>Ile or IIGS</td>
</tr>
<tr>
<td>11.. 0</td>
<td>Future expansion</td>
</tr>
<tr>
<td>00.. 1</td>
<td>Future expansion</td>
</tr>
<tr>
<td>01.. 1</td>
<td>Future expansion</td>
</tr>
<tr>
<td>10.. 1</td>
<td>Ile</td>
</tr>
<tr>
<td>11.. 1</td>
<td>Future expansion</td>
</tr>
<tr>
<td>....0</td>
<td>Unused</td>
</tr>
<tr>
<td>....1</td>
<td>48K</td>
</tr>
<tr>
<td>....10</td>
<td>64K</td>
</tr>
<tr>
<td>....11</td>
<td>128K</td>
</tr>
<tr>
<td>....X</td>
<td>Reserved</td>
</tr>
<tr>
<td>....0</td>
<td>No 80-column display</td>
</tr>
<tr>
<td>....00</td>
<td>80-column display</td>
</tr>
<tr>
<td>....01</td>
<td>No compatible clock</td>
</tr>
<tr>
<td>....1</td>
<td>Compatible clock present</td>
</tr>
</tbody>
</table>

BF99 SLTBYT Slot ROM map (bit on indicates ROM present)
BF9A PPFXPR Prefix flag (0 indicates no active prefix).
BF9B MLIACVT MLI active flag (1... indicates active.
BF9C CMDDAR Last MLI call return address.
BF9E SAVEX X-register savearea for MLI calls.
BF9F SAVEX Y-register savearea for MLI calls.

BFA0 HANDLE BANK SWITCHING AFTER IRQ ********************
(Enter reading high RAM, BANK1)

BFA0 EXIT $E900 same as save byte? ($E900)
BFA1 Yes, check for BANK1/BANK2 >>BFAA
BFA3 No, enable ROM ($E002)
BFA5 and exit now. >>BFA8
BFA8 EXIT1 Get RAM save byte for $D000. (BF5)
BFA9 Is it the same as BANK1? ($D000)
BFAA Yes, exit now. >>BF5
BFAE No, switch to BANK2. (C083)
BF56 EXIT2 Restore A-Register and
BF66 return from the interrupt.

BF7 MLI ENTRY, CONTINUED ****************************

BF77 MLICONT Carry set will
BF78 roll flag bit into MLIACTV. (BF9B)
BF7B Save high memory configuration ($E000)
BF7E by storing $E000 in BKBYT1 (BF74)
BF81 and $D000 (D000)
BF84 in BKBYT2. (BF5)
BF87 Then enable high RAM, BANK1 (C08B)
BF8A for read and write. (C08B)
BFCD Jump to actual MLI. >>DE00

ProDOS System Global Page -- V1.2

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
</table>
| BFD0 INTERRUPT ROUTINES ***********************
| BFD0 IRQXIT Determine state of high memory. (BF8D)
| BFD3 BANK1 enabled. >>BFE2
| BFD5 BANK2 enabled. ||BFDF
| BFD7 System have only 48K |
| BFD8 Yes, only ROM in high memory. >>BFE7
| BFD9 Enable ROM. (C0B1)
| BFD1 Always branch. >>BFE7
| BFD2 IRQXIT1 Switch to BANK2. (C0B3)
| BFD2 IRQXIT2 Preset BANKID for ROM.
| BFE4 (Later reset if high RAM interrupt). (BF8D)
| BFE7 ROMKIX Restore accumulator and (BF88)
| BFEA exit. |
| BFE8 IRQENT Enable high RAM, BANK1 (C0B8)
| BFF1 for read and write. (C0B8)
| BFF1 Jump into MLI. >>DFAE

BFF4 DATA *****************************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
</table>
| BF43 BKNBYT1 Storage for byte at $E000.
| BF55 BKNBYT2 Storage for byte at $D000.
| BF56 BFF4-8BF5 currently not used.

BFFC VERSION INFORMATION *****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
</table>
| BFFC IBKVER Minimum version of Kernel needed for this interpreter.
| BFFD IVERSION Version number of this in interpreter.
| BFFE KBKVER Currently undefined. Reserved for future use.
| BFPP KVVERSION MLI Version number: 02 in Version 1.2
| 03 in Version 1.3
### ProDOS QUIT Code -- V1.2 -- 6 SEP 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td><strong>MODULE STARTING ADDRESS</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FF3A</td>
<td>Sound Bell</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>********** INITIALIZATION ***********************</td>
</tr>
<tr>
<td>1000</td>
<td>Select ROM (C082)</td>
</tr>
<tr>
<td>1001</td>
<td>Disable 80 column card (CWAC)</td>
</tr>
<tr>
<td>1006</td>
<td>Select standard character set (C082)</td>
</tr>
<tr>
<td>1009</td>
<td>Clear 80-column store (C080)</td>
</tr>
<tr>
<td>106C</td>
<td>Set Normal display (white on black) (FE84)</td>
</tr>
<tr>
<td>106F</td>
<td>Initialize 40-column display (FE84)</td>
</tr>
<tr>
<td>1062</td>
<td>Set Video (FE93)</td>
</tr>
<tr>
<td>1015</td>
<td>Set Keyboard (FE89)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1018</td>
<td>********** INITIALIZE MEMORY BITMAP *******</td>
</tr>
<tr>
<td>1018</td>
<td>Mark pages $0, $1, $4 through $7</td>
</tr>
<tr>
<td>101A</td>
<td>and $BF as in use</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>102D</td>
<td>********** DISPLAY CURRENT PREFIX ******************</td>
</tr>
<tr>
<td>102D</td>
<td>Clear Screen and Home cursor (FC58)</td>
</tr>
<tr>
<td>1030</td>
<td>Go down 1 line (FD8E)</td>
</tr>
<tr>
<td>1033</td>
<td>Point to prompt number 1</td>
</tr>
<tr>
<td>1035</td>
<td>Print it out (1166)</td>
</tr>
<tr>
<td>1038</td>
<td>Position to line 3</td>
</tr>
<tr>
<td>103F</td>
<td>Call MLI (GET PREFIX) (BF80)</td>
</tr>
<tr>
<td>1042</td>
<td>Data: GET_PREFIX command number</td>
</tr>
<tr>
<td>1043</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>1045</td>
<td>Get length of Prefix (.encoder)</td>
</tr>
<tr>
<td>1048</td>
<td>Put a 0</td>
</tr>
<tr>
<td>104A</td>
<td>at the end of the prefix ( encoder)</td>
</tr>
<tr>
<td>104D</td>
<td>Check prefix length... (encoder)</td>
</tr>
<tr>
<td>1050</td>
<td>If length=0, there is no current Prefix &gt;&gt;105D</td>
</tr>
<tr>
<td>1052</td>
<td>If non-zero, display the current Prefix (encoder)</td>
</tr>
<tr>
<td>1057</td>
<td>on the video screen (05FF)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>105D</td>
<td>********** GET PREFIX NAME **********************</td>
</tr>
<tr>
<td>105D</td>
<td>Initialize counter</td>
</tr>
<tr>
<td>1064</td>
<td>Read a key (FD0C)</td>
</tr>
<tr>
<td>1067</td>
<td>Is it CARRIAGE RETURN?</td>
</tr>
<tr>
<td>1069</td>
<td>Yes, then accept Prefix &gt;&gt;10BD</td>
</tr>
<tr>
<td>1068</td>
<td>No, then save character</td>
</tr>
<tr>
<td>106C</td>
<td>Clear to end of line (FC9C)</td>
</tr>
<tr>
<td>106F</td>
<td>Retrieve character</td>
</tr>
<tr>
<td>1070</td>
<td>Is it ESCAPE?</td>
</tr>
<tr>
<td>1072</td>
<td>Yes, start all over again &gt;&gt;102D</td>
</tr>
<tr>
<td>1074</td>
<td>Is it CANCEL?</td>
</tr>
<tr>
<td>1076</td>
<td>Yes, start all over again &gt;&gt;102D</td>
</tr>
<tr>
<td>1078</td>
<td>Is it TAB?</td>
</tr>
<tr>
<td>107A</td>
<td>Yes, sound Bell, get another character &gt;&gt;1093</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>1087C</td>
<td>Is it CANCEL? Yes, go back and try again &gt; 1099</td>
</tr>
<tr>
<td>1087D</td>
<td>Yes, go back and try again &gt; 1099</td>
</tr>
<tr>
<td>1088A</td>
<td>No, keep checking &gt; 1099</td>
</tr>
<tr>
<td>1088B</td>
<td>No, don't try &gt; 1099</td>
</tr>
<tr>
<td>1088C</td>
<td>No, keep checking &gt; 1099</td>
</tr>
<tr>
<td>1088D</td>
<td>No, don't try &gt; 1099</td>
</tr>
<tr>
<td>1089A</td>
<td>Yes, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089B</td>
<td>No, keep checking &gt; 1100</td>
</tr>
<tr>
<td>1089C</td>
<td>No, keep checking &gt; 1100</td>
</tr>
<tr>
<td>1089D</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089E</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089F</td>
<td>No, keep checking &gt; 1100</td>
</tr>
<tr>
<td>1089G</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089H</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089I</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089J</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089K</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089L</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089M</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089N</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089O</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089P</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089Q</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089R</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089S</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089T</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089U</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089V</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089W</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089X</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>1089Y</td>
<td>No, sound Bell - try again &gt; 1099</td>
</tr>
<tr>
<td>1089Z</td>
<td>Yes, then handle it &gt; 1100</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>114A</td>
<td>Continue if no error &gt;&gt;114F</td>
</tr>
<tr>
<td>114C</td>
<td>Else, go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>114F</td>
<td>Get File Type (12A5)</td>
</tr>
<tr>
<td>1152</td>
<td>Is it ProDOS System file?</td>
</tr>
<tr>
<td>1154</td>
<td>Yes, continue &gt;&gt;115B</td>
</tr>
<tr>
<td>1156</td>
<td>No, indicate Error $01</td>
</tr>
<tr>
<td>1156</td>
<td>Go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>115B</td>
<td>Set Reference number to 0</td>
</tr>
<tr>
<td>1160</td>
<td>Call MLI (CLOSE) &lt;BF00&gt;</td>
</tr>
<tr>
<td>1163</td>
<td>Data: CLOSE command number</td>
</tr>
<tr>
<td>1164</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>1166</td>
<td>Continue if no error &gt;&gt;1168</td>
</tr>
<tr>
<td>1168</td>
<td>Else, go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>116B</td>
<td>Get Access byte (12A4)</td>
</tr>
<tr>
<td>1170</td>
<td>Yes, &gt;&gt;1177</td>
</tr>
<tr>
<td>1172</td>
<td>No, Indicate Error $27</td>
</tr>
<tr>
<td>1174</td>
<td>Go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>1177</td>
<td>Call MLI (OPEN) &lt;BF00&gt;</td>
</tr>
<tr>
<td>117A</td>
<td>Data: OPEN command number</td>
</tr>
<tr>
<td>117B</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>117D</td>
<td>Continue if no error &gt;&gt;1182</td>
</tr>
<tr>
<td>117F</td>
<td>Else, go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>1182</td>
<td>Get Reference Number (12B8)</td>
</tr>
<tr>
<td>1185</td>
<td>and update READ and (12BC)</td>
</tr>
<tr>
<td>118B</td>
<td>GET EOF parameter lists (12C4)</td>
</tr>
<tr>
<td>118B</td>
<td>Call MLI (GET EOF) &lt;BF00&gt;</td>
</tr>
<tr>
<td>118E</td>
<td>Data: GET_EOF command number</td>
</tr>
<tr>
<td>118F</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>1191</td>
<td>if error, handle it &gt;&gt;11E2</td>
</tr>
<tr>
<td>1193</td>
<td>Is EOF mark less than $10000 (12C7)</td>
</tr>
<tr>
<td>1196</td>
<td>Yes, continue on &gt;&gt;119C</td>
</tr>
<tr>
<td>1198</td>
<td>No, Indicate Error $27</td>
</tr>
<tr>
<td>119A</td>
<td>Go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>119C</td>
<td>Transfer EOF to Request count (12C5)</td>
</tr>
<tr>
<td>119F</td>
<td>in READ parameter list (12BF)</td>
</tr>
<tr>
<td>11A0</td>
<td>Call MLI (READ) &lt;BF00&gt;</td>
</tr>
<tr>
<td>11A3</td>
<td>Data: READ command number</td>
</tr>
<tr>
<td>11AC</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>11AE</td>
<td>Save status of READ</td>
</tr>
<tr>
<td>11AF</td>
<td>Call MLI (CLOSE) &lt;BF00&gt;</td>
</tr>
<tr>
<td>11B2</td>
<td>Data: Get Prefix command number</td>
</tr>
<tr>
<td>11B3</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>11B5</td>
<td>Continue if no error &gt;&gt;11B8</td>
</tr>
<tr>
<td>11B7</td>
<td>Else, retrieve status</td>
</tr>
<tr>
<td>11B8</td>
<td>and go to Error Handler &gt;&gt;11E2</td>
</tr>
</tbody>
</table>

This area of the code was modified for Version 1.2, and a bug was created. We are not sure it is safe to assume that the P-register is non-zero; that is, a BNS may not force the required branch. Also, there is a misplaced label here that will cause read errors to be ignored.

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>118A</td>
<td>Was READ good?</td>
</tr>
<tr>
<td>118B</td>
<td>No, go to Error Handler &gt;&gt;11B7</td>
</tr>
<tr>
<td>118D</td>
<td>Yes, execute application &gt;&gt;2000</td>
</tr>
<tr>
<td>11C0</td>
<td>************ BACKSPACE ROUTINE *************</td>
</tr>
<tr>
<td>11C0</td>
<td>Get cursor position horizontal</td>
</tr>
<tr>
<td>11C2</td>
<td>If 0 exit routine &gt;&gt;11D3</td>
</tr>
<tr>
<td>11C4</td>
<td>Decrement counter</td>
</tr>
<tr>
<td>11C5</td>
<td>Output a space</td>
</tr>
<tr>
<td>11CA</td>
<td>Move cursor back 2 spaces</td>
</tr>
<tr>
<td>11C8</td>
<td>Output a space &lt;FED&gt;</td>
</tr>
<tr>
<td>11D1</td>
<td>Move cursor back 1 space</td>
</tr>
<tr>
<td>11D3</td>
<td>Return to get another character &gt;&gt;10E7</td>
</tr>
<tr>
<td>11D6</td>
<td>************ PRINT TEXT ROUTINE *************</td>
</tr>
<tr>
<td>11D6</td>
<td>Get a character (1211)</td>
</tr>
<tr>
<td>11D9</td>
<td>If it is 0 then exit &gt;&gt;11E1</td>
</tr>
<tr>
<td>11DB</td>
<td>Output it &lt;FUED&gt;</td>
</tr>
<tr>
<td>11DE</td>
<td>Increment offset</td>
</tr>
<tr>
<td>11DF</td>
<td>Get another character &gt;&gt;11D6</td>
</tr>
<tr>
<td>11E1</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>11E2</td>
<td>************ PRINT ERROR MESSAGE *************</td>
</tr>
<tr>
<td>11E2</td>
<td>Save Accumulator (Error Number)</td>
</tr>
<tr>
<td>11E4</td>
<td>Position to line 12</td>
</tr>
<tr>
<td>11EB</td>
<td>Get Error number</td>
</tr>
<tr>
<td>11ED</td>
<td>Is it 1?</td>
</tr>
<tr>
<td>11EF</td>
<td>No, keep checking &gt;&gt;11F5</td>
</tr>
<tr>
<td>11F1</td>
<td>Yes, point to error message 1</td>
</tr>
<tr>
<td>11F3</td>
<td>and go print it &gt;&gt;1200</td>
</tr>
<tr>
<td>11F5</td>
<td>Is it $40?</td>
</tr>
<tr>
<td>11F7</td>
<td>Yes, print error message 3 &gt;&gt;1209</td>
</tr>
<tr>
<td>11F9</td>
<td>Is it $44?</td>
</tr>
<tr>
<td>11FB</td>
<td>Yes, print error message 3 &gt;&gt;1209</td>
</tr>
<tr>
<td>11FD</td>
<td>Is it $45?</td>
</tr>
<tr>
<td>11FF</td>
<td>Yes, print error message 3 &gt;&gt;1209</td>
</tr>
<tr>
<td>1201</td>
<td>Is it $46?</td>
</tr>
<tr>
<td>1203</td>
<td>Yes, print error message 3 &gt;&gt;1209</td>
</tr>
<tr>
<td>1205</td>
<td>Point to error message 2</td>
</tr>
<tr>
<td>1207</td>
<td>and go print it &gt;&gt;1208</td>
</tr>
</tbody>
</table>
ProDOS QUIT Code -- V1.2 -- 6 SEP 86

NEXT OBJECT ADDR: 1209

ADDR DESCRIPTION/CONTENTS

1209 Point to error message 3
120B Print Error message <1D6>
120E Get application name again >>10DE

1211 ********** ASCII TEXT ***********************************************

Prompt1
1211 'ENTER PREFIX (PRESS "RETURN" TO ACCEPT)'

Prompt2
1239 'ENTER PATHNAME OF NEXT APPLICATION'

Error1
125C Ring Bell
125D 'NOT A TYPE "SYS" FILE'

Error2
1273 Ring Bell
1274 'I/O ERROR'

Error3
128A Ring Bell
128B 'FILE/PATH NOT FOUND'

12A1 ********** PARAMETER LISTS ***************************************

GET FILE INFO Parmlist
12A1 Parmcount
12A2 Pathname
12A4 Access
12A5 File Type
12A6 Aux Type
12A8 Storage Type
12A9 Blocks Used
12AB Datetime (modified)
12AF Datetime (creation)

OPEN Parmlist
12B3 Parmcount
12B4 Pathname
12B6 I/O Buffer
12B8 Reference Number

CLOSE Parmlist
12B9 Parmcount
12BA Reference Number

ProDOS QUIT Code -- V1.2 -- 6 SEP 86

NEXT OBJECT ADDR: 12BA

ADDR DESCRIPTION/CONTENTS

READ Parmlist
12BB Parmcount
12BC Reference Number
12BD Data Buffer
12BF Request Count
12CI Transfer Count

GET_EOF Parmlist
12C3 Parmcount
12C4 Reference Number
12C5 EOF Mark

GET/SET_PREFIX Parmlist
12C8 Parmcount
12C9 Pathname

12CB *********** $12CB-$12FF UNUSED ***************************************

12CB These unused bytes are $D3CB-$D3FF in high RAM
12FF and $5B3B-$5BFF when loaded as part of "PRODOS" file.
**Disk II Device Driver -- V1.2 -- 6 SEP 86**

**ADDR** DESCRIPTION/CONTENTS

---

**D000** MODULE STARTING ADDRESS

******************************************************************************
* 5.25" DISK DEVICE DRIVER
* RESIDES AT $D000-$D6FF
* VERSION 1.2 -- 6 SEP 86
* (SAME AS IN VERSION 1.1.1)
******************************************************************************

**D000** ******** ZERO PAGE EQUATES ****************************

D03A Checksum
D03B Workbyte
D03E Slot (Temporary)
D042 Command
D043 Unit Number
D044 I/O Buffer Pointer (low)
D045 I/O Buffer Pointer (high)
D046 Block Number (low)
D047 Block Number (high)

**D000** ******** INTERNAL EQUATES ****************************

1000 Dummy Block Buffer (1st half)
1100 Dummy Block Buffer (2nd half)

**D000** ******** EXTERNAL EQUATES ****************************

C000 Phase Zero Off
C009 Motor On
C00A Drive Select
C00C Read Data Register
C00D Write Data Register
C00E Set Read Mode
C00F Set Write Mode
C00EC Read Data Register (slot 6)

**D000** ****** 5.25" DISK DRIVER ENTRY ******

D000 Clear decimal mode
D001 Clear phases in case IWM device in this slot <D6BE>
D004 Five NOP's so code below will
D005 Set up against Table at $D196
D009 Check validity of calling parameters <D6D0>
D00C If not valid exit with error >>D34
D00D Convert Block Number to a Track and Sector

---

**Disk II Device Driver -- V1.2 -- 6 SEP 86**

**ADDR** DESCRIPTION/CONTENTS

---

D010 ---
D014 $003000T TTITTABC
D015 . . . >>D010
D017 . . . >>D01C
D01A WOTTTTTT $000BC9A
D01C ---
D028 Preserve Sector Number
D02A Execute command <D03B>
D02C Restore Sector Number - Was prior action ok?
D02D No, then exit >>D03B
D027 Increment Buffer Pointer
D029 Increment Sector Number by 2 for rest of block
D02B Execute command <D03B>
D02E Decrement Buffer Pointer (to start of block)
D030 Get error number (if any - 0 indicates no error) (D35B)
D033 Return to caller

**D034** ********* I/O ERROR ROUTINE ****************************

D034 Indicate "I/O Error"
D036 Set Carry flag
D037 Return to caller

**D038** ******** MAIN CODE ****************************

D03B Set recalibration count to 1
D03D Preserve sector number (D357)
D03A Get "Unitnum" DSS0000
D032 Strip out Drive DSS0000
D034 Preserve slot number
D036 Check for slot change, turn off motor if so <<D6B>
D039 See if motor is on <D4A>
D03C Save test results
D03F Initialize counter for delay routine (D370)
D034 See if slot or drive has changed (D359)
D035 Update "current" unit number (D359)
D03A Save test results
D03B Put drive number in Carry flag
D03C Turn motor on (C009)
D032 Select appropriate drive (C008)
D036 Check test results - Same slot/drive?
D036 Yes, then skip delay >>D072
D069 Wait for new Drive
D06B to come up to speed <D385>
D072 Is command a status request?
D074 Yes, then do not move disk arm >>D07C
D076 Get track number for current request (D356)
D079 And go there <D1BC>
D07C Check test results - Was motor on?
D07D Yes, then skip delay >>D08B
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D07F</td>
<td>Wait for Drive to</td>
</tr>
<tr>
<td>D081</td>
<td>come up to speed &lt;D385&gt;</td>
</tr>
<tr>
<td>D089</td>
<td>Is motor on yet? &lt;D4DA&gt;</td>
</tr>
<tr>
<td>D08C</td>
<td>No, then exit with error &gt;&gt;D0EA</td>
</tr>
<tr>
<td>D08E</td>
<td>Is command a &quot;status&quot; request?</td>
</tr>
<tr>
<td>D09B</td>
<td>Yes, then determine status &gt;&gt;D0FD</td>
</tr>
<tr>
<td>D09D</td>
<td>Is command a &quot;read&quot; request?</td>
</tr>
<tr>
<td>D098</td>
<td>Yes, then continue on &gt;&gt;D09B</td>
</tr>
<tr>
<td>D099</td>
<td>Prepare data for write (pre nibblize) &lt;D5F6&gt;</td>
</tr>
<tr>
<td>D099</td>
<td>---</td>
</tr>
<tr>
<td>D09A</td>
<td>Initialize &quot;retry&quot; count at 64 (D369)</td>
</tr>
<tr>
<td>D09D</td>
<td>---</td>
</tr>
<tr>
<td>D09F</td>
<td>Read an address field - Good read? &lt;D39B&gt;</td>
</tr>
<tr>
<td>D0A2</td>
<td>Yes, then continue on &gt;&gt;D0BE</td>
</tr>
<tr>
<td>D0A4</td>
<td>Decrement &quot;retry&quot; count - More to try? (D369)</td>
</tr>
<tr>
<td>D0A7</td>
<td>Yes, then try again &gt;&gt;D09D</td>
</tr>
<tr>
<td>D0A9</td>
<td>No, just in case indicate &quot;I/O Error&quot;</td>
</tr>
<tr>
<td>D0AB</td>
<td>Decrement &quot;recalibration&quot; count - More to try? (D36A)</td>
</tr>
<tr>
<td>D0AE</td>
<td>No, then exit with error &gt;&gt;D0EA</td>
</tr>
<tr>
<td>D0B0</td>
<td>Get &quot;current&quot; track (D35A)</td>
</tr>
<tr>
<td>D0B3</td>
<td>Preserve it</td>
</tr>
<tr>
<td>D0B4</td>
<td>Double it and</td>
</tr>
<tr>
<td>D0B5</td>
<td>add 16 to it for recalibration</td>
</tr>
<tr>
<td>D0B7</td>
<td>Reinitialize Retry Count</td>
</tr>
<tr>
<td>D0BC</td>
<td>Branch always taken &gt;&gt;D0CC</td>
</tr>
<tr>
<td>D0C1</td>
<td>Was the right track found? (D35A)</td>
</tr>
<tr>
<td>D0C4</td>
<td>Yes, then continue on &gt;&gt;D0D5</td>
</tr>
<tr>
<td>D0C6</td>
<td>Get &quot;current&quot; track (D35A)</td>
</tr>
<tr>
<td>D0C9</td>
<td>Preserve it</td>
</tr>
<tr>
<td>D0CA</td>
<td>Get track we found</td>
</tr>
<tr>
<td>D0CD</td>
<td>Double it</td>
</tr>
<tr>
<td>D0CC</td>
<td>Put new value in Device Track Table &lt;D4D3&gt;</td>
</tr>
<tr>
<td>D0CF</td>
<td>Get track we want</td>
</tr>
<tr>
<td>D0D0</td>
<td>And go there &lt;D10C&gt;</td>
</tr>
<tr>
<td>D0D3</td>
<td>Branch always taken &gt;&gt;D09D</td>
</tr>
<tr>
<td>D0D8</td>
<td>Was the right sector found? (D357)</td>
</tr>
<tr>
<td>D0DB</td>
<td>No, then try again &gt;&gt;D0A4</td>
</tr>
<tr>
<td>D0DF</td>
<td>Is command a &quot;write&quot; request?</td>
</tr>
<tr>
<td>D0E9</td>
<td>Yes, then go do it &gt;&gt;D0F4</td>
</tr>
<tr>
<td>D0E2</td>
<td>Read the data - Good read? &lt;D3FD&gt;</td>
</tr>
<tr>
<td>D0E5</td>
<td>No, then try again &gt;&gt;D0A4</td>
</tr>
<tr>
<td>D0E7</td>
<td>Indicate no errors</td>
</tr>
<tr>
<td>D0E9</td>
<td>BNE instruction, never taken</td>
</tr>
<tr>
<td>D0EA</td>
<td>Indicate error</td>
</tr>
<tr>
<td>D0EB</td>
<td>Preserve error number (D35B)</td>
</tr>
<tr>
<td>D0EE</td>
<td>Get Slot</td>
</tr>
<tr>
<td>D0F0</td>
<td>Turn motor off (C088)</td>
</tr>
<tr>
<td>D0F3</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D0F4</td>
<td>********** HANDLE WRITE REQUEST ***********************</td>
</tr>
<tr>
<td>D0F4</td>
<td>Write data - Good write? &lt;D500&gt;</td>
</tr>
<tr>
<td>D0F7</td>
<td>Yes, then exit &gt;&gt;D0E7</td>
</tr>
<tr>
<td>D0F9</td>
<td>Indicate &quot;write-protect error&quot;</td>
</tr>
<tr>
<td>D0FB</td>
<td>Branch always taken &gt;&gt;D0EA</td>
</tr>
<tr>
<td>D0FD</td>
<td>********** GET STATUS ***********************</td>
</tr>
<tr>
<td>D0FD</td>
<td>Get Slot number</td>
</tr>
<tr>
<td>D102</td>
<td>Check &quot;write-protect&quot; status (C08E)</td>
</tr>
<tr>
<td>D105</td>
<td>Put result in Carry flag</td>
</tr>
<tr>
<td>D106</td>
<td>Select read mode (C08C)</td>
</tr>
<tr>
<td>D109</td>
<td>Exit with appropriate status &gt;&gt;D0F7</td>
</tr>
<tr>
<td>D10C</td>
<td>********** LOCATE DESIRED TRACK ***********************</td>
</tr>
<tr>
<td>D10C</td>
<td>Double the track number for proper phase</td>
</tr>
<tr>
<td>D10D</td>
<td>Preserve destination track * 2 (D36F)</td>
</tr>
<tr>
<td>D110</td>
<td>Turn all phases off &lt;D125&gt;</td>
</tr>
<tr>
<td>D113</td>
<td>Get offset into Device Track Table &lt;D4F1&gt;</td>
</tr>
<tr>
<td>D116</td>
<td>Get track (D359)</td>
</tr>
<tr>
<td>D119</td>
<td>Update &quot;current&quot; track (D35A)</td>
</tr>
<tr>
<td>D11C</td>
<td>Get destination track (D36F)</td>
</tr>
<tr>
<td>D11F</td>
<td>Update Device Track Table (D359)</td>
</tr>
<tr>
<td>D122</td>
<td>Move arm to desired track &lt;D133&gt;</td>
</tr>
<tr>
<td>D125</td>
<td>Initial phase number, starting with 3</td>
</tr>
<tr>
<td>D127</td>
<td>---</td>
</tr>
<tr>
<td>D128</td>
<td>Clear a phase &lt;D18A&gt;</td>
</tr>
<tr>
<td>D12B</td>
<td>Decrement phase number - More to do?</td>
</tr>
<tr>
<td>D12C</td>
<td>Yes, then continue until all phases done &gt;&gt;D127</td>
</tr>
<tr>
<td>D12E</td>
<td>Divide track number by 2 (D35A)</td>
</tr>
<tr>
<td>D132</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D133</td>
<td>********** ARM MOVE ROUTINE ***********************</td>
</tr>
<tr>
<td>D133</td>
<td>Preserve track to find (D372)</td>
</tr>
<tr>
<td>D136</td>
<td>Are we already there? (D35A)</td>
</tr>
<tr>
<td>D139</td>
<td>Yes, then set appropriate phase and exit &gt;&gt;D187</td>
</tr>
<tr>
<td>D13D</td>
<td>Initialize phase count (halftracks) (D36B)</td>
</tr>
<tr>
<td>D143</td>
<td>Prepare &quot;current&quot; track for comparisons (D371)</td>
</tr>
<tr>
<td>D146</td>
<td>Subtract track to find to compute delta-tracks</td>
</tr>
<tr>
<td>D147</td>
<td>Are we already there? (D372)</td>
</tr>
<tr>
<td>D14A</td>
<td>Yes, then clear prior phase and exit &gt;&gt;D103</td>
</tr>
<tr>
<td>D14C</td>
<td>Positive delta-tracks - go move arm out &gt;&gt;D155</td>
</tr>
<tr>
<td>D14E</td>
<td>Negative delta-tracks - Get absolute value delta-tracks less 1</td>
</tr>
<tr>
<td>D150</td>
<td>Increment current phase to move in (D35A)</td>
</tr>
<tr>
<td>D153</td>
<td>Branch always taken &gt;&gt;D15A</td>
</tr>
<tr>
<td>D155</td>
<td>Compute absolute value delta-tracks less 1</td>
</tr>
<tr>
<td>D157</td>
<td>Decrement current phase to move out (D35A)</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>D15A</td>
<td>Compare delta-tracks with phases moved (D36B)</td>
</tr>
<tr>
<td>D15D</td>
<td>Use smaller value for offset to delay tables &gt;&gt;D162</td>
</tr>
<tr>
<td>D162</td>
<td>Are we pointing at last table value yet?</td>
</tr>
<tr>
<td>D164</td>
<td>Yes, then continue to use current offset &gt;&gt;D168</td>
</tr>
<tr>
<td>D166</td>
<td>Else, use new offset</td>
</tr>
<tr>
<td>D167</td>
<td>Set Carry flag for set phase operation</td>
</tr>
<tr>
<td>D168</td>
<td>Set a phase &lt;D187&gt;</td>
</tr>
<tr>
<td>D16B</td>
<td>Get delay value from table (D373)</td>
</tr>
<tr>
<td>D16E</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D171</td>
<td>Get prior phase number (D371)</td>
</tr>
<tr>
<td>D174</td>
<td>Clear Carry flag for clear phase operation</td>
</tr>
<tr>
<td>D175</td>
<td>Clear a phase &lt;D18A&gt;</td>
</tr>
<tr>
<td>D178</td>
<td>Get delay value from table (D37C)</td>
</tr>
<tr>
<td>D17B</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D17E</td>
<td>Increment phases moved (D36B)</td>
</tr>
<tr>
<td>D183</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D187</td>
<td>Get &quot;current&quot; phase number (D35A)</td>
</tr>
<tr>
<td>D18A</td>
<td>Use low two bits only, zero to three - 000000FF</td>
</tr>
<tr>
<td>D18C</td>
<td>Multiply by two and bring in Carry - 000000FPC</td>
</tr>
<tr>
<td>D18D</td>
<td>Merge in slot number - 000000FPC</td>
</tr>
<tr>
<td>D18F</td>
<td>Put in X-reg for following operation</td>
</tr>
<tr>
<td>D190</td>
<td>Toggle appropriate phase (C000)</td>
</tr>
<tr>
<td>D193</td>
<td>Restore slot number to X-reg</td>
</tr>
<tr>
<td>D195</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

**TABLE 1**

Read Translate Table with Preniblize
Bit mask Tables and Epilog Table in unused areas

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1A8</td>
<td>00000000</td>
</tr>
<tr>
<td>D1A1</td>
<td>10000000</td>
</tr>
<tr>
<td>D1A2</td>
<td>01000000</td>
</tr>
<tr>
<td>D1A3</td>
<td>11000000</td>
</tr>
<tr>
<td>D1A4</td>
<td>Read Translate</td>
</tr>
<tr>
<td>D1C0</td>
<td>00000000</td>
</tr>
<tr>
<td>D1C1</td>
<td>01000000</td>
</tr>
<tr>
<td>D1C2</td>
<td>00100000</td>
</tr>
<tr>
<td>D1C3</td>
<td>01100000</td>
</tr>
<tr>
<td>D1C4</td>
<td>Epilog Table ($DE,$AA,$EB)</td>
</tr>
</tbody>
</table>

**TABLE 2**

Write Translate Table
Every 4th byte starting at $D203

Put nibbleize Bit mask Tables
Bit mask 1 (Every 4th byte starting at $D200
Bit mask 2 (Every 4th byte starting at $D201
Bit mask 3 (Every 4th byte starting at $D202

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D200</td>
<td>Entry for Bit Mask 1</td>
</tr>
<tr>
<td>D201</td>
<td>Entry for Bit Mask 2</td>
</tr>
<tr>
<td>D202</td>
<td>Entry for Bit Mask 3</td>
</tr>
<tr>
<td>D203</td>
<td>Entry for Write Translate</td>
</tr>
</tbody>
</table>

**AUXILIARY BUFFER**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D300</td>
<td>Auxiliary Buffer ($56 bytes) &gt;&gt;0056</td>
</tr>
<tr>
<td>D356</td>
<td>Track number</td>
</tr>
<tr>
<td>D357</td>
<td>Sector number</td>
</tr>
<tr>
<td>D358</td>
<td>Error number</td>
</tr>
</tbody>
</table>

**DEVICE TRACK TABLE**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D359</td>
<td>Table Entry</td>
</tr>
<tr>
<td>D359</td>
<td>Current Unit</td>
</tr>
<tr>
<td>D35A</td>
<td>Current Track</td>
</tr>
<tr>
<td>D35B</td>
<td>Slot 1, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D35D</td>
<td>Slot 2, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D35F</td>
<td>Slot 3, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D361</td>
<td>Slot 4, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D363</td>
<td>Slot 5, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D365</td>
<td>Slot 6, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D367</td>
<td>Slot 7, Devices 1 &amp; 2</td>
</tr>
</tbody>
</table>
D398 ** READ ADDRESS FIELD ******************
D398 Initialize "must find" count at $FCFC
D399 Increment count (low order byte) - Zero yet?
D39B No, skip ahead >>D3AB
D3AC Yes, exit and indicate Read Error >>D3FB
D3AD Read data register (C08C)
D3AF Loop until data valid >>D3AF
D3AA Is it first address mark ($05)?
D3AC No, then increment "must find" count >>D39D
D3AD Delay for data latch to clear
D3AF Read data register (C08C)
D3B2 Loop until data valid >>D3B2
D3B4 Is it second address mark ($A9)?
D3B6 No, then see if it's first address mark >>D3AA
D3AA Initialize count for four byte read
D3AB Read data register (C08C)
D3BF Loop until valid >>D3B2
D3C1 No, then see if it's first address mark >>D3AD
D3C3 Set interrupt flag
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D443</td>
<td>Loop until data valid &gt;&gt;D440</td>
</tr>
<tr>
<td>D445</td>
<td>Is it 2nd header mark ($AA)?</td>
</tr>
<tr>
<td>D447</td>
<td>No, then see if it is 1st header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D449</td>
<td>Delay for register to clear</td>
</tr>
<tr>
<td>D44A</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D44D</td>
<td>Loop until data valid &gt;&gt;D44A</td>
</tr>
<tr>
<td>D44F</td>
<td>Is it 3rd header mark ($AD)?</td>
</tr>
<tr>
<td>D451</td>
<td>No, then see if it is 1st header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D453</td>
<td>Initialize offset into data buffer</td>
</tr>
<tr>
<td>D457</td>
<td>Initialize checksum</td>
</tr>
<tr>
<td>D459</td>
<td>Read a data byte (C08C)</td>
</tr>
<tr>
<td>D45E</td>
<td>Translate it (D100)</td>
</tr>
<tr>
<td>D461</td>
<td>Store it in Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D464</td>
<td>Compute running checksum</td>
</tr>
<tr>
<td>D466</td>
<td>Increment offset - More to do?</td>
</tr>
<tr>
<td>D467</td>
<td>Yes, then continue &gt;&gt;D457</td>
</tr>
<tr>
<td>D469</td>
<td>Reinitialize offset into data buffer</td>
</tr>
<tr>
<td>D46B</td>
<td>Branch always taken &gt;&gt;D472</td>
</tr>
<tr>
<td>D46D</td>
<td>Set carry flag indicating error</td>
</tr>
<tr>
<td>D46E</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D46F</td>
<td>Store byte in Primary buffer (bottom third) (1000)</td>
</tr>
<tr>
<td>D472</td>
<td>Read a data byte (C08C).</td>
</tr>
<tr>
<td>D477</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D47A</td>
<td>Bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D480</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D481</td>
<td>No, then do another &gt;&gt;D46F</td>
</tr>
<tr>
<td>D483</td>
<td>Save last byte for later, no time now</td>
</tr>
<tr>
<td>D484</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D486</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D488</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D48D</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D490</td>
<td>Bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D496</td>
<td>Store byte in Primary buffer (middle third) (1000)</td>
</tr>
<tr>
<td>D499</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D49A</td>
<td>No, then do another &gt;&gt;D48B</td>
</tr>
<tr>
<td>D49C</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4A1</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D4A3</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D4A5</td>
<td>Translate byte and merge in (D100)</td>
</tr>
<tr>
<td>D4A8</td>
<td>Bits from Auxiliary buffer (D254)</td>
</tr>
<tr>
<td>D4A3</td>
<td>Store byte in Primary buffer (top third) (1000)</td>
</tr>
<tr>
<td>D4B1</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4B6</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D4B7</td>
<td>No, then do another &gt;&gt;D4A5</td>
</tr>
<tr>
<td>D4B9</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D4BB</td>
<td>Is checksum valid? (D100)</td>
</tr>
<tr>
<td>D4BE</td>
<td>No, then exit with error &gt;&gt;D4CC</td>
</tr>
<tr>
<td>D4C0</td>
<td>Get slot number</td>
</tr>
<tr>
<td>D4C2</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D4C5</td>
<td>Loop until data valid &gt;&gt;D4C2</td>
</tr>
<tr>
<td>D4C7</td>
<td>Is it last trailing mark ($DE)?</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D443</td>
<td>Loop until data valid &gt;&gt;D440</td>
</tr>
<tr>
<td>D445</td>
<td>Is it 2nd header mark ($AA)?</td>
</tr>
<tr>
<td>D447</td>
<td>No, then see if it is 1st header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D449</td>
<td>Delay for register to clear</td>
</tr>
<tr>
<td>D44A</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D44D</td>
<td>Loop until data valid &gt;&gt;D44A</td>
</tr>
<tr>
<td>D44F</td>
<td>Is it 3rd header mark ($AD)?</td>
</tr>
<tr>
<td>D451</td>
<td>No, then see if it is 1st header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D453</td>
<td>Initialize offset into data buffer</td>
</tr>
<tr>
<td>D457</td>
<td>Initialize checksum</td>
</tr>
<tr>
<td>D459</td>
<td>Read a data byte (C08C)</td>
</tr>
<tr>
<td>D45E</td>
<td>Translate it (D100)</td>
</tr>
<tr>
<td>D461</td>
<td>Store it in Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D464</td>
<td>Compute running checksum</td>
</tr>
<tr>
<td>D466</td>
<td>Increment offset - More to do?</td>
</tr>
<tr>
<td>D467</td>
<td>Yes, then continue &gt;&gt;D457</td>
</tr>
<tr>
<td>D469</td>
<td>Reinitialize offset into data buffer</td>
</tr>
<tr>
<td>D46B</td>
<td>Branch always taken &gt;&gt;D472</td>
</tr>
<tr>
<td>D46D</td>
<td>Set carry flag indicating error</td>
</tr>
<tr>
<td>D46E</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D46F</td>
<td>Store byte in Primary buffer (bottom third) (1000)</td>
</tr>
<tr>
<td>D472</td>
<td>Read a data byte (C08C).</td>
</tr>
<tr>
<td>D477</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D47A</td>
<td>Bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D480</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D481</td>
<td>No, then do another &gt;&gt;D46F</td>
</tr>
<tr>
<td>D483</td>
<td>Save last byte for later, no time now</td>
</tr>
<tr>
<td>D484</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D486</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D488</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D48D</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D490</td>
<td>Bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D496</td>
<td>Store byte in Primary buffer (middle third) (1000)</td>
</tr>
<tr>
<td>D499</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D49A</td>
<td>No, then do another &gt;&gt;D48B</td>
</tr>
<tr>
<td>D49C</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4A1</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D4A3</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D4A5</td>
<td>Translate byte and merge in (D100)</td>
</tr>
<tr>
<td>D4A8</td>
<td>Bits from Auxiliary buffer (D254)</td>
</tr>
<tr>
<td>D4A3</td>
<td>Store byte in Primary buffer (top third) (1000)</td>
</tr>
<tr>
<td>D4B1</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4B6</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D4B7</td>
<td>No, then do another &gt;&gt;D4A5</td>
</tr>
<tr>
<td>D4B9</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D4BB</td>
<td>Is checksum valid? (D100)</td>
</tr>
<tr>
<td>D4BE</td>
<td>No, then exit with error &gt;&gt;D4CC</td>
</tr>
<tr>
<td>D4C0</td>
<td>Get slot number</td>
</tr>
<tr>
<td>D4C2</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D4C5</td>
<td>Loop until data valid &gt;&gt;D4C2</td>
</tr>
<tr>
<td>D4C7</td>
<td>Is it last trailing mark ($DE)?</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D44A</td>
<td>Yes, then continue with carry clear &gt;&gt;D4CD</td>
</tr>
<tr>
<td>D44C</td>
<td>Set Carry flag indicating error</td>
</tr>
<tr>
<td>D44D</td>
<td>Get byte we stored away, we have time now</td>
</tr>
<tr>
<td>D44E</td>
<td>Set proper offset</td>
</tr>
<tr>
<td>D450</td>
<td>Store byte in Primary buffer (offset $55)</td>
</tr>
<tr>
<td>D452</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D453</td>
<td>Update DEVICE TRACK TABLE ***************</td>
</tr>
<tr>
<td>D45D</td>
<td>Get offset into Device Track Table &lt;D4F1&gt;</td>
</tr>
<tr>
<td>D45E</td>
<td>Update Device Track Table (D359)</td>
</tr>
<tr>
<td>D45F</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D46A</td>
<td>Determine if drive is on (DATA CHANGING) ***************</td>
</tr>
<tr>
<td>D47A</td>
<td>Get slot number</td>
</tr>
<tr>
<td>D47C</td>
<td>Initialize counter</td>
</tr>
<tr>
<td>D47E</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D481</td>
<td>Delay 25 cycles &lt;D4F0&gt;</td>
</tr>
<tr>
<td>D485</td>
<td>Has data register changed? (C08C)</td>
</tr>
<tr>
<td>D489</td>
<td>Yes, then exit &gt;&gt;D4F0</td>
</tr>
<tr>
<td>D48B</td>
<td>Just in case indicate No Device Connected Error</td>
</tr>
<tr>
<td>D49D</td>
<td>Decrement count = 256 tries yet?</td>
</tr>
<tr>
<td>D4EE</td>
<td>No, try again &gt;&gt;D4DE</td>
</tr>
<tr>
<td>D4F0</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D4F1</td>
<td>CONVERT SLOT/DRIVE TO TABLE OFFSET ***************</td>
</tr>
<tr>
<td>D4F2</td>
<td>Preserve A-register</td>
</tr>
<tr>
<td>D4F4</td>
<td>Get Unit number DSSS0000</td>
</tr>
<tr>
<td>D4F6</td>
<td>Divide by 16 00000000D</td>
</tr>
<tr>
<td>D4FA</td>
<td>Strip out Drive 00000000D</td>
</tr>
<tr>
<td>D4FC</td>
<td>Roll left 00000000D</td>
</tr>
<tr>
<td>D4FD</td>
<td>Put result in X-register</td>
</tr>
<tr>
<td>D4FE</td>
<td>Restore A-register</td>
</tr>
<tr>
<td>D4FF</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D500</td>
<td>WRITE DATA ROUTINE ***************</td>
</tr>
<tr>
<td>D500</td>
<td>Set Carry flag (anticipate error)</td>
</tr>
<tr>
<td>D504</td>
<td>Is diskette &quot;write-protected&quot;? (C08E)</td>
</tr>
<tr>
<td>D507</td>
<td>No, then continue on &gt;&gt;D50C</td>
</tr>
<tr>
<td>D509</td>
<td>Go to error routine &gt;&gt;D5DF</td>
</tr>
<tr>
<td>D50C</td>
<td>Put transition byte from secondary buffer (D300)</td>
</tr>
<tr>
<td>D50F</td>
<td>Into zero page for timing</td>
</tr>
<tr>
<td>D511</td>
<td>Use $FF for &quot;sync&quot; byte</td>
</tr>
<tr>
<td>D513</td>
<td>Write first &quot;sync&quot; byte (C08F)</td>
</tr>
<tr>
<td>D519</td>
<td>Set counter for four more</td>
</tr>
<tr>
<td>D51C</td>
<td>Delay so that writes occur</td>
</tr>
<tr>
<td>D51D</td>
<td>Exactly on 40 cycle loops</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
</tr>
<tr>
<td>D51E</td>
<td>---</td>
</tr>
<tr>
<td>D520</td>
<td>Write &quot;sync&quot; byte &lt;D5E7&gt;</td>
</tr>
<tr>
<td>D523</td>
<td>Decrement counter, done yet?</td>
</tr>
<tr>
<td>D524</td>
<td>No, then do another &gt;&gt;D51E</td>
</tr>
<tr>
<td>D526</td>
<td>Write first data mark ($D5)</td>
</tr>
<tr>
<td>D52B</td>
<td>Write second data mark ($AA)</td>
</tr>
<tr>
<td>D530</td>
<td>Write third data mark ($AD)</td>
</tr>
<tr>
<td>D535</td>
<td>Initialize checksum</td>
</tr>
<tr>
<td>D536</td>
<td>Initialize index into Auxiliary buffer</td>
</tr>
<tr>
<td>D538</td>
<td>Branch always taken &gt;&gt;D53D</td>
</tr>
<tr>
<td>D53A</td>
<td>Get data byte (Auxiliary buffer) (D300)</td>
</tr>
<tr>
<td>D53D</td>
<td>Exclusive-or with previous data byte (D2FF)</td>
</tr>
<tr>
<td>D540</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D541</td>
<td>Lookup &quot;disk byte&quot; in table (D2B3)</td>
</tr>
<tr>
<td>D544</td>
<td>Get slot</td>
</tr>
<tr>
<td>D546</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>D54C</td>
<td>Decrement index - Done with Auxiliary buffer?</td>
</tr>
<tr>
<td>D54D</td>
<td>No, then another byte &gt;&gt;D53A</td>
</tr>
<tr>
<td>D54F</td>
<td>Get last byte of Auxiliary buffer</td>
</tr>
<tr>
<td>D551</td>
<td>Initialize index into Primary buffer</td>
</tr>
<tr>
<td>D553</td>
<td>Exclusive-or with next data byte (1000)</td>
</tr>
<tr>
<td>D556</td>
<td>Strip out last two bits XXXXXXX0</td>
</tr>
<tr>
<td>D558</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D559</td>
<td>Lookup &quot;disk byte&quot; in table (D2B3)</td>
</tr>
<tr>
<td>D55C</td>
<td>Get slot</td>
</tr>
<tr>
<td>D55E</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>D564</td>
<td>Get data byte (Primary buffer) (1000)</td>
</tr>
<tr>
<td>D567</td>
<td>Increment offset, end of this page?</td>
</tr>
<tr>
<td>D568</td>
<td>No, then continue on &gt;&gt;D553</td>
</tr>
<tr>
<td>D56A</td>
<td>Did buffer start on page boundary?</td>
</tr>
<tr>
<td>D56C</td>
<td>Yes, then go write checksum &gt;&gt;D5C0</td>
</tr>
<tr>
<td>D56E</td>
<td>Did buffer start one past page boundary?</td>
</tr>
<tr>
<td>D570</td>
<td>Yes, then go write last byte &gt;&gt;D5B3</td>
</tr>
<tr>
<td>D572</td>
<td>Carry indicates odd or even buffer end</td>
</tr>
<tr>
<td>D573</td>
<td>Get transition byte</td>
</tr>
<tr>
<td>D575</td>
<td>Write it (C08D)</td>
</tr>
<tr>
<td>D57B</td>
<td>Get second transition byte</td>
</tr>
<tr>
<td>D57D</td>
<td>Delay 2 cycles for correct timing</td>
</tr>
<tr>
<td>D57E</td>
<td>Increment offset, buffer end on odd byte?</td>
</tr>
<tr>
<td>D57F</td>
<td>Yes, go see if we're done then &gt;&gt;D599</td>
</tr>
<tr>
<td>D581</td>
<td>Exclusive-or with next data byte (1100)</td>
</tr>
<tr>
<td>D584</td>
<td>Strip out last two bits XXXXXXX0</td>
</tr>
<tr>
<td>D586</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D587</td>
<td>Lookup &quot;disk byte&quot; in table (D2B3)</td>
</tr>
<tr>
<td>D58A</td>
<td>Get slot</td>
</tr>
<tr>
<td>D58C</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>D592</td>
<td>Get data byte (Primary buffer - page 2) (1100)</td>
</tr>
<tr>
<td>D595</td>
<td>Increment offset</td>
</tr>
<tr>
<td>D596</td>
<td>Exclusive-or with next data byte (1100)</td>
</tr>
<tr>
<td>D599</td>
<td>End of buffer? - Put result in carry</td>
</tr>
<tr>
<td>D59B</td>
<td>Strip out last two bits XXXXXXX0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D59D</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D59E</td>
<td>Get slot</td>
</tr>
<tr>
<td>D59F</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>D5A0</td>
<td>Get data byte (Primary buffer - page 2) (1100)</td>
</tr>
<tr>
<td>D5AC</td>
<td>Increment offset - Done yet?</td>
</tr>
<tr>
<td>D5AD</td>
<td>No, then do another &gt;&gt;D5B1</td>
</tr>
<tr>
<td>D5AF</td>
<td>Yes, then go write checksum &gt;&gt;D5B1</td>
</tr>
<tr>
<td>D5B1</td>
<td>--- &gt;&gt;D5C0</td>
</tr>
<tr>
<td>D5B3</td>
<td>Get last byte</td>
</tr>
<tr>
<td>D5B6</td>
<td>Write it (C08D)</td>
</tr>
<tr>
<td>D5B8</td>
<td>Delay 14 cycles for correct timing</td>
</tr>
<tr>
<td>D5BC</td>
<td>Use last byte in Primary buffer as checksum</td>
</tr>
<tr>
<td>D5C2</td>
<td>Lookup &quot;disk byte&quot; (D2B3)</td>
</tr>
<tr>
<td>D5C5</td>
<td>Get slot</td>
</tr>
<tr>
<td>D5C7</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>D5C9</td>
<td>Initialize offset into &quot;epilog&quot; table</td>
</tr>
<tr>
<td>D5C6</td>
<td>Delay 11 cycles for correct timing</td>
</tr>
<tr>
<td>D5D3</td>
<td>Load &quot;epilog&quot; from table ($DE,$AA,$EB,$FF) (D1C4)</td>
</tr>
<tr>
<td>D5D6</td>
<td>Go write it &lt;D5E9&gt;</td>
</tr>
<tr>
<td>D5D9</td>
<td>Increment offset</td>
</tr>
<tr>
<td>D5DA</td>
<td>Done all four yet?</td>
</tr>
<tr>
<td>D5DC</td>
<td>No, then do another &gt;&gt;D5D3</td>
</tr>
<tr>
<td>D5DE</td>
<td>Clear Carry flag (no error)</td>
</tr>
<tr>
<td>D5DF</td>
<td>Select read mode (C08B)</td>
</tr>
<tr>
<td>D5E5</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

---

**WRITE A BYTE SUBROUTINE**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D5E6</td>
<td>Wait 9 cycles before write</td>
</tr>
<tr>
<td>D5E7</td>
<td>Wait 7 cycles before write</td>
</tr>
<tr>
<td>D5E9</td>
<td>Put A-register in data register (C08D)</td>
</tr>
<tr>
<td>D5EC</td>
<td>And write data register (C0BC)</td>
</tr>
<tr>
<td>D5EF</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

---

**PREINITIALIZE BLOCK ROUTINE**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D5F0</td>
<td>Get buffer pointer</td>
</tr>
<tr>
<td>D5F5</td>
<td>Add $2 to buffer address</td>
</tr>
<tr>
<td>D5F7</td>
<td>To access top third of buffer &gt;&gt;D5FA</td>
</tr>
<tr>
<td>D5FA</td>
<td>Store result in code below (D630)</td>
</tr>
<tr>
<td>D601</td>
<td>Subtract $34 from buffer address</td>
</tr>
<tr>
<td>D603</td>
<td>To access middle third of buffer &gt;&gt;D606</td>
</tr>
<tr>
<td>D606</td>
<td>Store result in code below (D625)</td>
</tr>
<tr>
<td>D60D</td>
<td>Subtract $9A from buffer address</td>
</tr>
<tr>
<td>D61F</td>
<td>To access bottom third of buffer &gt;&gt;D612</td>
</tr>
<tr>
<td>D612</td>
<td>Store result in code below (D61B)</td>
</tr>
<tr>
<td>D618</td>
<td>Initialize offset</td>
</tr>
<tr>
<td>D61A</td>
<td>Get data byte (bottom third) XXXXXXX (1000)</td>
</tr>
<tr>
<td>D61D</td>
<td>Get last two bits 000080AB</td>
</tr>
<tr>
<td>D61F</td>
<td>Put in X-reg for table lookup</td>
</tr>
</tbody>
</table>
Disk II Device Driver -- V1.2 -- 6 SEP 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D620</td>
<td>Use lookup to reposition bits</td>
</tr>
<tr>
<td>D623</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D624</td>
<td>Get data byte (middle third)</td>
</tr>
<tr>
<td>D627</td>
<td>Get last two bits</td>
</tr>
<tr>
<td>D629</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D62A</td>
<td>Get current value from stack</td>
</tr>
<tr>
<td>D62B</td>
<td>Merge in new bits using table</td>
</tr>
<tr>
<td>D62E</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D62F</td>
<td>Get data byte (top third)</td>
</tr>
<tr>
<td>D632</td>
<td>Get last two bits</td>
</tr>
<tr>
<td>D634</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D635</td>
<td>Get current value from stack</td>
</tr>
<tr>
<td>D636</td>
<td>Merge in new bits using table</td>
</tr>
<tr>
<td>D63A</td>
<td>Get offset into primary buffer</td>
</tr>
<tr>
<td>D63B</td>
<td>Compute offset into Auxiliary buffer</td>
</tr>
<tr>
<td>D63D</td>
<td>Put in X-reg</td>
</tr>
<tr>
<td>D63E</td>
<td>Get data byte just created</td>
</tr>
<tr>
<td>D63F</td>
<td>Store it in Auxiliary buffer (D300)</td>
</tr>
<tr>
<td>D642</td>
<td>Increment offset primary buffer, done yet?</td>
</tr>
<tr>
<td>D643</td>
<td>NO, then do another &gt;&gt;D6A</td>
</tr>
<tr>
<td>D645</td>
<td>Get low order byte of buffer</td>
</tr>
<tr>
<td>D647</td>
<td>Subtract 1 (offset to last byte in buffer)</td>
</tr>
<tr>
<td>D648</td>
<td>Save it for later</td>
</tr>
<tr>
<td>D64A</td>
<td>Get low order byte of buffer</td>
</tr>
<tr>
<td>D64C</td>
<td>Modify code in Write Data Routine (offset) (D552)</td>
</tr>
<tr>
<td>D64F</td>
<td>Buffer on page boundary? - Yes, skip ahead &gt;&gt;D65F</td>
</tr>
<tr>
<td>D651</td>
<td>Else, compute offset to last byte</td>
</tr>
<tr>
<td>D653</td>
<td>Before page boundary</td>
</tr>
<tr>
<td>D654</td>
<td>Get byte (page boundary -1)</td>
</tr>
<tr>
<td>D656</td>
<td>Point at next byte (page boundary)</td>
</tr>
<tr>
<td>D657</td>
<td>Exclusive-or them together</td>
</tr>
<tr>
<td>D659</td>
<td>Strip off last two bits</td>
</tr>
<tr>
<td>D65B</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D65C</td>
<td>Get &quot;disk byte&quot; from table (transition byte) (D203)</td>
</tr>
<tr>
<td>D65F</td>
<td>Save result (0 indicates page boundary)</td>
</tr>
<tr>
<td>D661</td>
<td>Buffer on page boundary? - Yes, skip ahead &gt;&gt;D66F</td>
</tr>
<tr>
<td>D663</td>
<td>Get offset to last byte in buffer</td>
</tr>
<tr>
<td>D665</td>
<td>Carry indicates odd or even buffer start</td>
</tr>
<tr>
<td>D666</td>
<td>Get byte (page boundary)</td>
</tr>
<tr>
<td>D66B</td>
<td>Did buffer start on odd byte? - Yes skip &gt;&gt;D66D</td>
</tr>
<tr>
<td>D66A</td>
<td>Point at next byte (page boundary +1)</td>
</tr>
<tr>
<td>D66B</td>
<td>Exclusive-or them together</td>
</tr>
<tr>
<td>D66D</td>
<td>Save result</td>
</tr>
<tr>
<td>D66F</td>
<td>Point at last byte in buffer</td>
</tr>
<tr>
<td>D671</td>
<td>Get last byte in buffer</td>
</tr>
<tr>
<td>D673</td>
<td>Strip off last two bits</td>
</tr>
<tr>
<td>D675</td>
<td>Save result (&quot;checksum byte&quot;)</td>
</tr>
<tr>
<td>D677</td>
<td>Get high order byte of buffer</td>
</tr>
<tr>
<td>D679</td>
<td>Modify code in Write Data Routine (D555)</td>
</tr>
<tr>
<td>D68C</td>
<td>Get slot number for this operation</td>
</tr>
</tbody>
</table>

Disk II Device Driver -- V1.2 -- 6 SEP 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D68E</td>
<td>Modify code in Write Data Routine (D555)</td>
</tr>
<tr>
<td>D69A</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D69B</td>
<td>Determine if Slot/Drive Has Changed</td>
</tr>
<tr>
<td>D69E</td>
<td>Compare unit number with &quot;current&quot; unit number (D359)</td>
</tr>
<tr>
<td>D69F</td>
<td>Has slot changed? - No, then exit &gt;&gt;D6BD</td>
</tr>
<tr>
<td>D6A9</td>
<td>Get &quot;current&quot; slot</td>
</tr>
<tr>
<td>D6AB</td>
<td>Put in X-register</td>
</tr>
<tr>
<td>D6AC</td>
<td>Exit if Slot Ø &gt;&gt;D6BD</td>
</tr>
<tr>
<td>D6AE</td>
<td>Is &quot;current&quot; motor is on? &lt;D4DC&gt;</td>
</tr>
<tr>
<td>D6B1</td>
<td>No, then exit &gt;&gt;D6BD</td>
</tr>
<tr>
<td>D6BB</td>
<td>Wait until &quot;current&quot; motor is off (D78)</td>
</tr>
<tr>
<td>D6BD</td>
<td>Or else timeout &gt;&gt;D6A6</td>
</tr>
<tr>
<td>D6BE</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6BE</td>
<td>Clear ISM Phases</td>
</tr>
<tr>
<td>D6C0</td>
<td>Strip drive bit</td>
</tr>
<tr>
<td>D6C2</td>
<td>Put slot 16 in X-Register</td>
</tr>
<tr>
<td>D6C3</td>
<td>Clear phases in case there is (C080)</td>
</tr>
<tr>
<td>D6C6</td>
<td>One of them new-fangled storage (C082)</td>
</tr>
<tr>
<td>D6C9</td>
<td>Devices sharing this slot (C084)</td>
</tr>
<tr>
<td>D6CC</td>
<td>Modify code with my (trust) old Disk II. (C086)</td>
</tr>
<tr>
<td>D6CF</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6D0</td>
<td>Check Calling Parameters</td>
</tr>
<tr>
<td>D6D0</td>
<td>Note: For 80-track drives, change byte at $D6E3 from 18 to 40</td>
</tr>
<tr>
<td>D6D8</td>
<td>Check command code</td>
</tr>
<tr>
<td>D6D9</td>
<td>Is it greater or equal to 4?</td>
</tr>
<tr>
<td>D6DA</td>
<td>Yes, indicate error &gt;&gt;D6E6</td>
</tr>
<tr>
<td>D6D6</td>
<td>Get Block Number</td>
</tr>
<tr>
<td>D6DA</td>
<td>Is Block Number good? (D356)</td>
</tr>
<tr>
<td>D6DD</td>
<td>Yes, if less than $100 &gt;&gt;D6E8</td>
</tr>
<tr>
<td>D6DE</td>
<td>No, if greater than or equal to $200 &gt;&gt;D6E6</td>
</tr>
<tr>
<td>D6DF</td>
<td>No, if greater than or equal to $118 &gt;&gt;D6E8</td>
</tr>
<tr>
<td>D6E1</td>
<td>Indicate error</td>
</tr>
<tr>
<td>D6E7</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6E8</td>
<td>All is well</td>
</tr>
<tr>
<td>D6E9</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6EA</td>
<td>$D6EA-$D6FF NOT USED</td>
</tr>
<tr>
<td>D6EA</td>
<td>Not used</td>
</tr>
</tbody>
</table>
Disk II Device Driver -- V1.3 -- 2 DEC 86

NEXT OBJECT ADDR: D6BE

ADDR DESCRIPTION/CONTENTS

*********************************************************
* 5.25" DISK DEVICE DRIVER
* RESIDES AT $D000-$D6FF
* VERSION 1.3 -- 2 DEC 86
*********************************************************

The Disk II Device Driver for Version 1.3 changes only
in one routine--the "clear phases" subroutine.
Phases are now cleared with a "LDA" instead of
a "STA" to eliminate bus fights that potentially cause
unwanted writing to the 5.25" disk.

D6BE ****** CLEAR IWF PHASES ****************************

D6BE  Get unit number
D6C9  Strip drive bit
D6C2  Put slot*16 in X-Register
D6C3  Turn off 8 phases
D6CF  Return to caller

D6D0 ****** CHECK CALLING PARAMETERS ********************

Note: For 40-track drives, change byte at $D6E3
from $18 to $40.

D6D0  Check command code
D6D2  Is it greater or equal to 4?
D6D4  Yes, indicate error >>D6E6
D6D6  Get Block Number
D6DA  Is Block Number good? (D356)
D6DD  Yes, if less than $1W8 >>D6E8
D6E0  No, if greater than or equal to $200 >>D6E6
D6E4  No, if greater than or equal to $118 >>D6E8
D6E6  Indicate error
D6E7  Return to caller
D6E8  All is well
D6E9  Return to caller

D6EA ****** $D6EA-$D6FF NOT USED ****************************

D6EA  Not used
**Beneath Apple ProDOS Supplement**

**IRQ Handler -- V1.2 -- 6 SEP 86**

**NEXT OBJECT ADDR: FP98**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF9B</td>
<td><strong>MODULE STARTING ADDRESS</strong></td>
</tr>
<tr>
<td></td>
<td>***************************</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>* IRQ Handler</td>
</tr>
<tr>
<td></td>
<td>* Resides at $FF9B. Put</td>
</tr>
<tr>
<td></td>
<td>* there by ProDOS Relocator.</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.2 -- 6 SEP 86</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.3 -- 2 DEC 86</td>
</tr>
<tr>
<td></td>
<td>* (The IRQ Handler is still the</td>
</tr>
<tr>
<td></td>
<td>* same as it was in Version 1.0.1)</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

**GLOBAL PAGE EQUATES**

| BF56 | Temporary storage 1 |
| BF57 | Temporary storage 2 |
| BF88 | A register savearea |
| BF8D | Bank ID byte         |
| BFD3 | IRQ exit code        |

**EXTERNAL EQUATES**

| D000 | RAM/ROM test byte |
| C082 | ROM Select        |
| C08B | BANK1 Select      |

**IRQ CODE**

| FF9B | Put A-Register on stack |
| FF9C | Get Accumulator value from $45 |
| FF9E | and save it (BF56) |
| FF9F | Replace $45 with A-Register |
| FF9G | since it may have been destroyed |
| FF9H | Load Status register |
| FF9I | Restore onto stack |
| FF9J | Isolate B flag - Was it a BRK? |
| FF9K | Yes, skip Interrupt stuff >>FFC2 |
| FF9L | Else, Check location $D000 (D000) |
| FF9M | DO we have RAM active |
| FF9N | Yes, indicate so >>F9B3 |
| FF9O | Else, indicate ROM |
| FF9P | Update Bank ID byte (BF8D) |
| FF9Q | Also save temporarily (BF57) |
| FF9R | Push ($BF50) address of |
| FF9S | routine to bank in Ram and |
| FF9T | call IRQ on the stack |
| FF9U | Push a new P-Register on stack with |
| FF9V | the Interrupt Disable flag set |
| FF9W | Push ($F4A1) address less 1 of |
ThunderClock Code -- V1.2 -- 6 SEP 86

ADDRES DESCRIPTION/CONTENTS

D742 MODULE STARTING ADDRESS

*****************************************************************************
* CLOCK Code (for ThunderClock)
* If a ThunderClock or its equivalent is located, then
* this code is loaded into the
* ProDOS data area at $D742.
* 
* VERSION 1.2 -- 6 SEP 86, and
* VERSION 1.3 -- 2 DEC 86
*
*****************************************************************************

D742 ********* ZERO PAGE EQUATES ******************************

@03A Binary month (1=JAN, 2=FEB, etc.)
@03B Binary day of week (0=Sunday, 1=Monday, etc.)
@03C Binary day of the month (1-31)
@03D Binary hour of the day (0-23)
@03E Binary minute of the hour (0-59)

D742 ********* EXTERNAL EQUATES ******************************

@0200 Input Buffer
@BF90 Global page year-month-day
@BF92 Global page hour-minutes

D742 ********* CLOCK CODE ENTRY POINT ******************************

D742 Get slot ROM high byte (D750)
D745 Get a screen hole byte for this slot ($53B)
D74B and save it on the stack
The two JSR addresses that follow will be
modified by ProDOS Relocator so that they
will access the correct slot ROM.

D748 Write an $A3 to the clock (consult your ThunderClock manual) <C10B>
D74E Read the clock. <C108>

Reading the clock results in an ASCII string
being placed in the input buffer. A sample
string might be "07,06,04,22,46,57", which is
July (month 07) Saturday (day-of-week 6)
the 4th (day of month 4) 10 PM (hour 22)
46 minutes and 57 seconds after the hour.

ThunderClock Code -- V1.2 -- 6 SEP 86

ADDRES DESCRIPTION/CONTENTS

D751 ********* CONVERT ASCII TO BINARY ******************************

D751 ---
D752 Five values to convert (ProDOS ignores seconds)
D754 Y-reg is index into
D756 the input buffer ($0200)
D759 Strip ASCII from ten's digit
D75D by 10
D762 Add in one's digit ($0201)
D766 and subtract off ASCII,
D768 then store as binary in 2-page
D76A Skip over comma
D76B and two digits
D76E More values to convert >>D756

D770 ********* NOW CONVERT TO PRODOS DATE, TIME ******************************

D770 Save month in Y-reg
D771 three low bits of month
D774 three high bits of acum,
D775 combine with day of month,
D777 and store in low byte of DATE (MMDDDDD) (BF9B)
D77A Save carry (high bit of month)
D77B Add day of the month
D77D to table value to get Julian date (D7AB)
D780 Late September? >>D784
D782 Yes, add 3 and carry (see notes below)
D784 Compute Julian date MOD 7
D785 ---
D78B Subtract day-of-week to get year index
D78D Index positive? >>D791
D78F No, make it positive
D791 Index to Y-Reg
D792 Get year from year table (D788)
D795 Get high bit of month in carry
D796 roll it into accumulator
D797 to get high byte of DATE (YYYYYYMM) (BF91)
D799 Put hour
D79C In high byte of TIME (BF93)
D79F Put minutes
D7A1 in low byte of TIME (BF92)
D7A6 Restore saved screen hole value ($038)
D7AB RETURN

D7AC ********* JULIAN TABLE ******************************

D7AC January
D7AD February
D7AE March
D7AF April
D7BB May
ThunderClock Code -- V1.2 -- 6 SEP 86
NEXT OBJECT ADDR: D781

ADDR DESCRIPTION/CONTENTS

D781 June
D782 July
   If month>7, value in table is one
   less than Julian, because a carry
   is added along with the day of the month
D783 August
D784 September
   Note: For Julian dates>255, the table value
   is three more than the low byte should be,
   so that it will properly divide by 7.
D785 October
D786 November
D787 December

D788 ***** YEAR TABLE ********************************************
   This table is good for the years 1986-91
D788 1990
D789 1989
D78A 1988 (March to December)
D78B 1988 (January and February)
D78C 1987
D78D 1986
D78E 1991
IIGS Clock Code -- V1.2 -- 6 SEP 86  
NEXT OBJECT ADDR: D742

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D742</td>
<td>Module starting address</td>
</tr>
</tbody>
</table>

* * * * * * * * * * * * *
* IIGS CLOCK Code * * * *
* If ProDOS is booted on * an Apple IIGS, then this * code is loaded into the * ProDOS data area at $D742. * * VERSION 1.2 -- 6 SEP 86, and * VERSION 1.3 -- 2 DEC 86 * *
* * * * * * * * * * * * *

D742 ***** GLOBAL PAGE VALUES ***********************

BF90 ProDOS DATE word, YYYYMMDDDDDDDD.
BF92 ProDOS TIME word, 00000000000000.

D742 ***** IIGS SOFT SWITCH ***********************

C868 One byte sets 8 soft switches.

D742 ***** CLOCK CODE ENTRY POINT ***********************

D742 8-bit memory and index operations.
D744 Get IIGS STATREG Status Byte, (C868)
D747 save it temporarily. (D790)
D74A Make sure we're in bank 0.
D750 Get out of emulation mode.
D751 16-bit memory and index operations.
D756 Push 4 null words on the stack.
D75A Tool = ReadTimeChar. (0D03)
D75D Jump to the tool dispatcher. <E10000>
D761 8-bit memory operations.
D763 Get pre-call Status Byte (D790)
D766 and restore all soft switches. (C868)
D769 Throw away seconds.
D76B Store minutes in ProDOS Global Page. (BF92)
D76F Store hours in ProDOS Global Page. (BF93)
D772 Get year
D773 and store it temporarily. (BF91)
D776 Get day of month
D777 in range 1-31
D778 and save it temporarily. (BF90)
D778 Get month
D77C in range 1-12
D77D and shift it left five bits.
D782 Combine with the date (BF90)
D785 and store in Global Page as MMMDDDDDD. (BF90)
HOW "BASIC.SYSTEM" IS LOADED AND RELOCATED

1. The "BASIC.SYSTEM" file is loaded to memory address $2000 by the SYSTEM file loader (or a "-" command) which then jumps to $2000 (the BI Relocator).

2. The BI Relocator moves the Interpreter to $9A00-$BCFF, and the BI Global Page to $BE00-$BEFF.

3. The BI Relocator searches for a "STARTUP" file in the same directory as "BASIC.SYSTEM". If found, it loads and executes the "STARTUP" program. Otherwise, it prints out a greeting and cold starts BASIC by jumping to the BASIC entry point at $BE00.
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>&quot;PRM&quot; POINT FOR COPY</td>
</tr>
<tr>
<td>0001</td>
<td>&quot;TO&quot; POINT FOR COPY</td>
</tr>
<tr>
<td>0003</td>
<td>CWARE VICTOR</td>
</tr>
<tr>
<td>0036</td>
<td>KWARE VICTOR</td>
</tr>
<tr>
<td>006F</td>
<td>APPLESOFT START OF STRINGS</td>
</tr>
<tr>
<td>0073</td>
<td>APPLESOFT HIMEM</td>
</tr>
<tr>
<td>00F2</td>
<td>APPLESOFT TRACER</td>
</tr>
<tr>
<td>0200</td>
<td>PATHNAME BUFFER</td>
</tr>
<tr>
<td>020F</td>
<td>PREFIX BUFFER</td>
</tr>
<tr>
<td>0261</td>
<td>START OF PREFIX NAME</td>
</tr>
<tr>
<td>0300</td>
<td>WARMSTART VICTOR</td>
</tr>
<tr>
<td>0303</td>
<td>COLDSTART VICTOR</td>
</tr>
<tr>
<td>03F0</td>
<td>BRK HANDLER ADDRESS</td>
</tr>
<tr>
<td>03F1</td>
<td>RESET HANDLER ADDRESS</td>
</tr>
<tr>
<td>03F3</td>
<td>POWER-UP BYTE</td>
</tr>
<tr>
<td>03F5</td>
<td>APPLESOFT &amp; VECTOR</td>
</tr>
<tr>
<td>03FB</td>
<td>CTL-Y VICTOR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0060</td>
<td>FIRST SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>0480</td>
<td>SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>0628</td>
<td>SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>BC7A</td>
<td>BASIC INTERPRETER Version NUMBER</td>
</tr>
<tr>
<td>BE90</td>
<td>BASIC INTERPRETER ENTRY POINT</td>
</tr>
<tr>
<td>BE00</td>
<td>81 COMMAND SCANNER (SYNTAX)</td>
</tr>
<tr>
<td>BE10</td>
<td>COUT VECTORS FOR EACH SLOT</td>
</tr>
<tr>
<td>BE20</td>
<td>KSWL VECTORS FOR EACH SLOT</td>
</tr>
<tr>
<td>BE3C</td>
<td>DEFAULT SLOT NO.</td>
</tr>
<tr>
<td>BE3D</td>
<td>DEFAULT DRIVE NO.</td>
</tr>
<tr>
<td>BFFA</td>
<td>HIMEM</td>
</tr>
<tr>
<td>BF00</td>
<td>MACHINE LANGUAGE INTERFACE ENTRY</td>
</tr>
<tr>
<td>BF30</td>
<td>LAST DEVICE USED</td>
</tr>
<tr>
<td>BF58</td>
<td>MEMORY MAP</td>
</tr>
<tr>
<td>BF98</td>
<td>MACHINE TYPE FLAGS</td>
</tr>
<tr>
<td>BF99</td>
<td>SLOTS WHICH CONTAINS CARDS WITH ROM</td>
</tr>
<tr>
<td>BF9A</td>
<td>IF 6, NO PREFIX ACTIVE</td>
</tr>
<tr>
<td>BFBD</td>
<td>INTERPRETER VERSION NUMBER</td>
</tr>
<tr>
<td>B200</td>
<td>APPLESOFT ENTRY POINT</td>
</tr>
<tr>
<td>FA59</td>
<td>BRK HANDLER</td>
</tr>
<tr>
<td>FA2F</td>
<td>INIT SCREEN, MONITOR, ETC.</td>
</tr>
<tr>
<td>FC58</td>
<td>CLEAR SCREEN, HOME CURSOR</td>
</tr>
<tr>
<td>FEDD</td>
<td>STANDARD CHARACTER OUT</td>
</tr>
<tr>
<td>FD80</td>
<td>CHARACTER OUTPUT TO SCREEN</td>
</tr>
<tr>
<td>FE84</td>
<td>SET NORMAL CHARACTER ATTRIBUTE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2047</td>
<td>JUMP OVER STARTUP FILENAME &gt;&gt;2048</td>
</tr>
<tr>
<td>2056</td>
<td>STARTUP FILENAME LENGTH (7)</td>
</tr>
<tr>
<td>2007</td>
<td>'STARTUP'</td>
</tr>
<tr>
<td>2008</td>
<td>ALLOW FOR 64 CHAR FILENAME</td>
</tr>
<tr>
<td>2047</td>
<td>$00 --&gt; $2400</td>
</tr>
<tr>
<td>2048</td>
<td>$02 --&gt; $9A00</td>
</tr>
<tr>
<td>2055</td>
<td>COPY 35 PAGES</td>
</tr>
<tr>
<td>2058</td>
<td>COPY INTERP TO HIGH MEMORY AT $9A00 &lt;20C4&gt;</td>
</tr>
<tr>
<td>205D</td>
<td>PAGE FOLLOWING INTERP IMAGE IS...</td>
</tr>
<tr>
<td>205F</td>
<td>BASIC GLOBAL PAGE IMAGE</td>
</tr>
<tr>
<td>2061</td>
<td>COPY THAT TO $8E00 &lt;20C4&gt;</td>
</tr>
<tr>
<td>2064</td>
<td>TO GET 40-COL DISPLAY, SEND A CTRL-U</td>
</tr>
</tbody>
</table>
2066 OUT THE NORMAL OUTPUT VECTOR. <FED6>
2069 SET NORMAL CHARACTER ATTRIBUTE <FB84>
206C INITIALIZE SCREEN/WINDOW <FB2F>
206F CLEAR SCREEN/HOME CURSOR <FC58>
2076 SET BITMAP TO MARK LOWER 40K FREE (BF58)
207C EXCEPT PAGES 0 AND 1 AND
207E TEXT PAGES 4 THROUGH 7 (BF8B)
2086 MARK $9000-$BFFF IN USE...
2091 EXCEPT FOR $8A00-$BFFF ARE FREE
2096 LOOK AT LANGUAGE IN ROM ($B800)
2099 IS IT APPLESOFT?
209B NO, THEN CAN'T RUN INTERP >>20B1
20A0 GET AT LEAST 64K?
20A2 NO, THIS ONLY WORKS IN 64K >>20B1
20A6 SET MY C SwL/KS wL FOR INTERP INIT (221A)
20AC COPY ALL 4 BYTES >>20A6
20AE THEN GO TO BASIC COLDSTART >>B000

(WE WILL GET CONTROL AT $2004 AGAIN)

20B1**********ERROR EXIT***********************

20B1---
20B3 PRINT "UNABLE TO EXECUTE BASIC SYSTEM" (223F)
20BC ALLOW REBOOT IF RESET PRESSED (03F4)
20BC GO TO SLEEP FOREVER >>20C2

20C4**********COPY PAGES ($8/1--$2/3)***************

20C4---
20C5 COPY FROM $0/1
20C7 TO $2/3
20CA A PAGE AT A TIME >>20C4
20D0 COUNT PAGES
20D3 RETURN

20D4**********C SwL INTERCEPT / CONTINUE***************

20D4 "J" APPLESOFT PROMPT?
20D6 NO...DON'T PRINT WHATEVER IT IS >>20D3
20D8 YES, APPLESOFT DONE SETTING UP (BE10)
20DB POINT C SwL TO STANDARD OUTPUT
20DB CHECK LAST DEVICE USED (BF30)
20E5 SET ONLINE PARAMETER TO THIS (223B)
20EB DRIVE ONE OR TWO? >>20EE
20EE STORE DEFAULT DRIVE (D) (BE3D)
20F2 ISOLATE SLOT FROM DEVICE NO.
20F7 AND STORE DEFAULT S LOP (S) (BE3C)
20FF GET SLOT BYTE SHOWING CARDS PRESENT? (BF99)
2102 PICK OFF ITS BITS ONE BY ONE
2108 SET OUTVSC AND INVSC TO $C800 (BE18)
210B FOR ALL SLOTS WITH ROMS IN THEM (BE20)

2115---
211B SET HIMEM TO $9600
211D IN VARIOUS PLACES
2124 GOT A DEFAULT PREFIX? (BF9A)
2127 NO >>214E
2129 YES, AND GET PREFIX <BF00>
212F ERROR? >>218B
2136 BACKSCAN PREFIX FOR "/" $ (0280)
213B AND COUNT THEM IN $223E ($23E)
213E---
213F FOR A COUNT OF SUBLEVELS >>2136
2146 MORE THAN JUST VOLUME NAME? >>216F
2148 NO, MLI: SET PREFIX <BF00>
214E MLI: ONLINE <BF00>
2154 ERROR? >>218B
2156 GET VOL NAME LENGTH ($281)
215B NONE THERE? >>218B
215F ADD ONE TO NAME LENGTH ($280)
2164 AND PREFIX IT WITH A "/" ($281)
2167 MLI: SET PREFIX <BF00>
216D ERROR? >>218B

*******FIND STARTUP FILE***********

216F MLI: GET FILE INFO <BF00>
2172 FIND "STARTUP" FILE
2175 ERROR? >>218B
217A SAVE LENGTH OF STARTUP FILE NAME (2236)
217D COPY NAME TO $200 (206E)
2186 FIRST COMMAND WILL BE "STARTUP"
218B CHECK NUMBER OF SUBLEVELS (223E)
2190 MORE THAN JUST VOL? >>2198
2192 MLI: SET PREFIX <BF00>
2198 ANY STARTUP FILE NAME? (2236)
219B YES, SKIP MESSAGE >>21C1
219D SET TRUE KS wL <2209>
21A2 PRINT 'PRODOS BASIC 1.1' (2267)
21A5 PRINT 'COPYRIGHT ...' (2283)
21B6 SKIP THREE LINES

*******FINISH UP AND GO TO 81**********

21C1---
21C3 COPY WARNSTART JMP TO PAGE 3 (21FF)
21C9 AND COLDSTART (U3D3)
21CC AND C TL+y (03F6)
21CF FIND & VECTOR (2246)
21D2 TO $8E0J (CMD SCANNER) (03F5)
21D6 COPY BRK HANDLER JMP ALSO (2202)
21E7 AND RESET JMP (83F2)
21F2 SET POWER-UP BYTE ACCORDINGLY (03F4)
BI Relocator -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 21F7

ADDR DESCRIPTION/CONTENTS

21F7 SET APPLESOFI IN NON-TRACE MODE
21F9 GET INTERPRETER VERSION NUMBER, (BC7A)
21PC PUT IT, IN SYSTEM GLOBAL PAGE. (3FFD)
21FF GO TO INTERPRETER >>BE00

******** VECTOR ADDRESSES ************

22A2 BREAK HANDLER ADDRESS FOR PAGE 3
22A4 RESET HANDLER IS BASIC INTERP
22A6 APPLESOFI & GOES TO BI CMD SCANNER >>BE03

2209 ********** FIRST KSWL INTERCEPT **********************

2209 SET KSWL TO CURRENT DEVICE HANDLER (BE20)
2213 RETURN LENGTH OF FIRST COMMAND (2006)
2217 FOLLOWED BY A RETURN
2219 RETURN

221A ******** DATA **********************

221A CSWL (2004) INTERCEPT ADDR
221C KSWL (2209) INTERCEPT ADDR

221E GET FILE INFO PARM LIST
221F FILE NAME IS AT $2006
2221 15 BYTES RESERVED FOR OTHER GET_FILE PARM (NOT USED)
2230 THIS BYTE NOT USED

2231 SET PREFIX PARM LIST
2232 FOR PREFIX AT $2234

2234 NULL PREFIX
2235 "/

2236 SAVED LENGTH OF STARTUP FILE NAME

2237 ONLINE PARM LIST
2239 PUT VOLUME NAME AT $281

223B SET PREFIX PARM LIST
223C PREFIX IS AT $290

223E NUMBER OF SUBL LEVEIS IN PREFIX +1

223F *** UNABLE TO EXECUTE BASIC SYSTEM ***
2267 ' PRODOS BASIC 1.1'
2283 ' COPYRIGHT APPLE, 1983-84'
9A00  MODULE STARTING ADDRESS

* PRODOS BASIC INTERPRETER (BI)  
* THIS CODE STARTS IN THE THIRD 
* BLOCK OF THE FILE BASIC.SYS.  
* IT PERFORMS COMMAND HANDLING 
* FOR ALL BUILT-IN PRODOS COM- 
* MANDS AND SUPPORTS BASIC’S FILE 
* HANDLING.  
*  
* VERSION 1.1 -- 18 JUN 84 
*  
* DISTRIBUTED WITH PRODOS VERSIONS 
* 1.1.1, 1.2, AND 1.3.  

********* ZERO PAGE ADDRESSES *********

0024  CURSOR HORIZONTAL 
0028  SCREEN LINE BASE ADDR 
0029  
0033  MONITOR PROMPT CHARACTER 
0036  CRT DISPLAY VECTOR (CSWL) 
0037  
0038  KEYBOARD INPUT VECTOR (KSWL) 
0039  
003A  SCRATCH POINTER AND LOOP COUNTER 
003B  
003C  SCRATCH POINTER AND LOOP COUNTER 
003D  
003E  POINTER TO APPLESOF VARIABLES 
003F  
0050  APPLESOF: LINE NUMBER 
0051  
0067  APPLESOF: START OF PROGRAM PTR 
0068  APPLESOF: LOMEM (START OF VARS) 
0069  APPLESOF: START OF ARRAY VARS PTR 
006A  
006B  APPLESOF: START OF FREEAREA PTR 
006C  
006D  APPLESOF: START OF STRINGS PTR 
006E  
0070  APPLESOF: HIMEM (END OF STRINGS) 
0073  
0074  APPLESOF: CURRENT LINE BEING EXECUTED 
0075  
0076  
009B  APPLESOF: ADDR OF LINE AFTER FINDLINE 
009C  
009F  APPLESOF: END OF PROGRAM PTR 
00B0  
00BB  APPLESOF: START OF PROGRAM PTR 
00BD  
00BE  APPLESOF: ONERR CODE 
00F2  APPLESOF: TRACE ACTIVE FLAG 
00FB  APPLESOF: INTERNAL STACK 
0100  START OF 6502 STACK 
0200  KEYBOARD INPUT LINE BUFFER 
03F4  POWERON RESET FLAG 

********* BI GLOBAL PAGE **********

AE06  EXTERNAL COMMAND ENTRY TO BI 
BE0C  PRINT ERROR MESSAGE ENTRY TO BI 
BE0F  PRODOS ERROR CODE 
BE10  OUTPUT VECTORS FOR ALL SLOTS 
BE10  CURRENT OUTPUT VECTOR 
BE12  CURRENT INPUT VECTOR 
BE14  PRODOS INTERCEPT VECTORS (INPUT/OUTPUT) 
BE18  BI’S INTERNAL REDIRECTION VECTORS 
BE1C  DEFAULT SLOT 
BE1D  DEFAULT DRIVE 
BE1E  A REGISTER SAVE AREA 
BE1F  X REGISTER SAVE AREA 
BE40  Y REGISTER SAVE AREA 
BE41  TRACE FLAG (APPLESOF TRACE ON/OFF) 
BE42  IMMEDIATE COMMANDS=04, DEFERRED=1 
BE43  EXE5 FILE ACTIVE=5050 
BE44  READ FILE ACTIVE=5050 
BE45  WRITE FILE ACTIVE=5050 
BE46  READING PREFIX ACTIVE=5050 
BE47  DIRECTORY FILE BEING ACCESSED 
BE49  FREE STRING SPACE DURING GARBAGE COLLECT 
BE4A  BUFFERED I/O BYTE COUNT 
BE4B  INEX INTI INPUT COMMAND LINE 
BE4C  LAST OUTPUT CHAR TO PREVENT RECURSION 
BE4D  NUMBER OF OPEN NON-EXEC FILES 
BE4E  EXEC FILE BEING CLOSED FLAG 
BE4F  READ FILE IS TRANSLATED DIRECTORY 
BE50  VECTOR TO EXTERNAL COMMAND HANDLER 
BE52  LENGTH-1 OF EXTERNAL COMMAND STRING 
BE53  COMMAND NUMBER 
BE54  PARAMETERS ALLOWED FOR THIS COMMAND
BASIC Interpreter (81) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9A00

ADDR DESCRIPTION/CONTENTS

(SEE BIT DEFINITIONS IN TABLE LATER)

BE56 PARAMETERS FOUND WITH THIS COMMAND
(SAME BIT DEFINITIONS AS FOR PBITS)

BE58 A KEYWORD VALUE
BE5A B KEYWORD VALUE
BE5C D KEYWORD VALUE
BE5E E KEYWORD VALUE
BE60 L KEYWORD VALUE
BE62 S KEYWORD VALUE
BE64 D KEYWORD VALUE
BE66 F KEYWORD VALUE
BE68 R KEYWORD VALUE
BE6A T KEYWORD VALUE
BE6C SLOT NUMBER FROM IN# OR PR#
BE70 ISSUE MLI CALL AND XLIST ERROR CODES
MLI PAR M LIST FIELDS

BEA3 CREATE: ACCESS CODE
BEA5 CREATE: AUX ID
BEA7 CREATE: FILE KIND
BEA9 SET/GET FILE INFO: PARM COUNT
BEAC SET/GET FILE INFO: ACCESS CODE
BEAF SET/GET FILE INFO: FILE ID
BEB0 SET/GET FILE INFO: AUX ID
BEB3 SET/GET FILE INFO: FILE KIND
BEBC SET/GET FILE INFO: BLOCKS USED
BEBE SET/GET FILE INFO: MODIFY DATE/TIME
BECE ONLINE/GET/SET MARK/EOF/BUF: REP NUM
BECE ONLINE/GET/SET MARK/EOF/BUF: MARK/BUF

9A00 ************ BASIC INTERPRETER LOAD POINT ************
(ENTRY POINT IS AT $ABE1, WARMDO)

9A08 ************ REMOVE KSWL/CSSLW INTERCEPTS ************

9A17 ************ RESET MODE/SET BI INTERCEPTS ************

BF03 QUIT VECTOR
BF30 LAST DEVICE USED
BF50 MEMORY UTILIZATION BIT MAP
BF94 OPEN FILE LEVEL
BF9A PREFIX ACTIVE FLAG (IF NONZERO)
9AA3 ********** SET CSWL/KS WL INTERCEPTS ***************

9AA3 ---
9AA4 COPY VDOSIO VECTORS (BE34)
9AA7 TO CSWL
9A8 AND KSWL
9A89 EXIT TO CALLER

9ABA ********** INPUT INTERCEPT: MODE = 0 ***************
(Immediate Mode)

9ABA IS EXEC FILE ACTIVE? (BE43)
9ABD NO >> 9AC5
9ABF YES, SAVE REGISTERS <9F62>
9AC2 AND GO READ EXEC FILE FOR INPUT COMMANDS >> 9ABF
9AC5 NO EXEC FILE, RESTORE REAL CSWL/KS WL <9A98>
9AC8 NO, READ A KEY FROM KEYBOARD <FD10>
9ACB RETURN?
9ACD NO, EXIT >> 9AB5
9ACF YES, SAVE REGISTERS <9F62>
9AD2 STORE IT IN LINE BUFFER (0200) -- This entry called by exec to process

A command string stored at $200
9AD5 GO PROCESS THE COMMAND STRING <A677>
9A92 CHECK COMMAND NUMBER RETURNED FROM PARSE (BE53)
9ABD EXIT BI RIGHT NOW >> 9A89
9ADD NO, COMMAND RETURNED WITH ERROR CODE? >> 9AFD
9A9F NO, RESTORE Y REG (BE46)
9AEB RETURN A BACKSPACE TO CALLER OF KEYBOARD
9AE4 AND A LINE INDEX OF 200
9AEB EXIT THE BI >> 9A89

9A88 RESTORE CALLER'S REGISTERS <9F6C>
9A8E AND EXIT BI BY INSTALLING INTERCEPTS >> 9A3D

9AEE ********** ERROR HANDLER ***************

9AEE ERROR=3, "NO DEVICE CONNECTED"
9AP0 MAIN ENTRY: STORE ERROR CODE (BE0F)
9AP3 AND IN APPLESOF TRACER
9AP5 CHECK BI STATE (BE42)
9AP8 MEMORIZE WHETHER IT'S IMMEDIATE MODE
9APD SET A HIGH FILE LEVEL FOR NON-EXEC FILES (B94)
9B02 NO ACTIRE READ/WR ITE FILES OR PREFIX READ (BE44)
9B08 CLOSE ALL OPEN FILES AT OR ABOVE (BEDE)
9B0E FILE LEVEL = $9F
9B10 MLI: CLOSE (ALL) <BE70>
9B13 ERROR? >>9B27
9B15 WRITE ANY DATA I HAVE BUFFERED <A0W>
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
ADDRESS DESCRIPTION/CONTENTS

9B10 ERROR? >>9B27
9B1A PUT FILE LEVEL BACK TO ZERO
9B22 NOW FLUSH ALL OPEN FILES
9B24 MLIU: FLUSH (ALL) <BE70>
9B27 ---
9B28 ASSUME MODE WILL BE 4 (DEPRECATED)
9B2A MEMORIZE WHETHER BASIC ON DISPLAY ACTIVE
9B2C DEFERRED MODE CURRENTLY? >>9B30
9B2E NO, STILL IMMEDIATE MODE (MODE=0)
9B30 ---
9B31 SET MODE AS DEFINED ABOVE <9F76>
9B34 RESTORE B1'S CSWL/KSRL INTERCEPTS <9AA3>
9B37 GET ERROR CODE <BE0F>
9B3B BASIC ONERR ACTIVATE THEN GO HANDLE IT >>9B4D
9B3E NO, JUST PRINT ERROR MESSAGE <8B8C>
9B41 CLOSE EXEC FILE IF ONE IS OPEN <B2FB>
9B45 DEFERRED MODE? >>9B53
9B47 IMMED. MODE, PRINT RETURN AND... <9FAB>
9B4A WARMSTART APPLESOFT >>D43F
9B4D RESTORE STACK FOR BASIC
9B52 PASS ERROR CODE TO BASIC
9B53 ---
9B55 JUMP INTO APPLESOFT ERROR HANDLER >>D865
9B58 ********** RETURN TO IMMEDIATE MODE ***********************
9B59 CLEAR APPLESOFT ERRNUM
9B5C WILL LOOK FOR "$" FROM APPLESOFT
9B61 SET NORMAL VIDEO IN APPLESOFT <F273>
9B64 RESTORE TRUE CSWL/KSRL <9A06>
9B67 TRY TO WRITE BUFFERED DATA <<9F64>
9B6A RESTORE MODE/SET UP BI'S INTERCEPTS <9A17>
9B6D RESTORE REGISTERS <9F6C>
9B70 GO TO PROCESS IMMED. INPUT REQUEST >>9BAA
9B73 ********** INPUT INTERCEPT: MODE=4 or 8 *******************
9B75 SAVE REGISTERS <9F62>
9B76 PREFIX INPUT ACTIVE? (BE46)
9B79 NO >>9B7E
9B7B YES, GO DO SPECIAL HANDLING >>9D67
9B7E ELSE, IS READ FILE ACTIVE? (BE44)
9B81 NO >>9B86
9B83 YES, GO DO SPECIAL HANDLING FOR THAT >>9C16
9B86 ELSE, IS EXEC FILE ACTIVE? (BE43)
9B89 NO >>9B8F
9B8B YES, GET PROMPT CHARACTER
9B8D IT BETTER NOT BE A """"
9B8F IT IS, RETURN TO IMMEDIATE MODE >>9B58
9B91 ELSE, SET TRUE CSWL/KSRL <9A00>

BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
ADDRESS DESCRIPTION/CONTENTS

9B94 AND PASS CALLEER'S AREG TO REMOVE CURSOR (BE3E)
9B97 RESTORE Y-REGISTER (BE46)
9B9A RESTORE CURSOR AND GET A KEYPRESS <FD10>
9B9D BACKSPACE?
9B9F NO, EXIT BI >>9BAC
9BA1 YES, CHECK PROMPT
9BA3 IF ITS A "",...
9BA5 THEN EXIT WITH THE BACKSPACE >>9BAA
9BA8 ELSE, IF AT START OF LINE, REPROMPT >>9B94
9BAA MIDDLE OF LINE, RETURN A BACKSPACE
9BAC EXIT BI TO CALLER >>9ABD

9BAF ********** READ EXEC FILE **********************
9BAP REMOVE CURSOR FROM SCREEN
9BB1 CHECK PROMPT CHARACTER
9BB3 IF ITS A "",...
9BB5 DO THINGS DIFFERENTLY >>9BF2
9BB7 CHECK KEYBOARD (C080)
9BBA NO KEY READY? >>9B0C
9BBC GOT A KEY, IS IT CONTROL-C?
9BBD NO, IGNORE IT >>9B0C
9BC0 YES, CLOSE EXEC FILE <B2FB>
9BC3 IMMEDIATE MODE? (BE42)
9BC6 NO >>9C01
9BC8 YES, CLEAR KEYBOARD STROBE (C610)
9BCB AND GO START NEW LINE >>9C01
9BCC SET UP FOR EXEC LINE READ <9D6A>
9BD0 READ A LINE TO $200 <9C6C>
9BD3 ERROR? >>9BFA
9BD5 SAVE REGISTERS <9F62>
9BDB HOP INTO LOOP >>9BDE
9BDA ---
9BDB BACKSCANNING $200 BUFFER (0200)
9BDE FORCING THE MSG ON
9BEE RESTORE TRUE CSWL/KSRL <9A06>
9BE9 GO PROCESS COMMAND LINE <9AD5>
9BEC CHECK COMMAND NUMBER (BE53)
9BEF IMMEDIATE EXIT? IF NOT, GET NEXT LINE >>9BCD
9BF1 RETURN

********** HANDLE EXEC PROMPT > **********
9BF2 GET SET TO READ EXEC LINE <9D6A>
9BF5 READ SINGLE CHARACTER PER CALL <9C40>
9BF8 NO ERRORS, EXIT TO CALLER NOW >>9BF1
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9BF8

******* EXEC ERROR RECOVERY ************

9BF8  CLOSE EXEC FILE <B245>
9BF9  WAS ERROR "END OF DATA"?
9BFF  NO, REAL ERROR THEN >> 9C13
9C01  ELSE, OK -- JUST STOP EXECING
9C03  GET CURSOR HORIZONTAL POSITION
9C05  IF IN MID LINE, PASS SCREEN CHAR BACK >> 9C0E
9C07  ELSE, CHANGE PROMPT TO "|
9C0B  AND RETURN WITH A BACKSPACE
9C0D  RETURN
9C0E  GET SCREEN CHARACTER UNDER CURSOR
9C10  AND EXIT THRU KSML TO GET REAL KEYPRESS >> 9038
9C13  REAL ERROR, GO TO BI'S MAIN ERROR HANDLER >> 9AF0

9C16  INPUT FILE ACTIVE **********************

9C16  GET PROMPT
9C18  IF ITS A "|
9C1C  THEN RESET TO IMMEDIATE MODE >> 9B5B
9C1F  ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
9C24  CHECK KEYBOARD (C006)
9C27  NO KEYPRESS? >> 9C31
9C29  GOT A KEY, IS IT CONTROL-C?
9C2B  NO, IGNORE IT >> 9C31
9C2D  CLEAR STROBE AND EXIT TO CALLER (C01B)
9C30  RETURN

9C31  GET PROMPT AGAIN
9C33  IS THIS A DIRECTORY FILE? (BE47)
9C36  YES >> 9C95
9C38  NO, IS PROMPT = "|
9C3A  YES, READ A SINGLE BYTE AT A TIME >> 9C42
9C3C  ELSE, READ ENTIRE LINE << 9C67
9C3F  ERROR? >> 9C13
9C41  RETURN

9C42  READ SINGLE BYTE FROM INPUT FILE << 9C48
9C45  ERROR? >> 9C13
9C47  RETURN

9C48  READ NEXT BYTE OF FILE ******************

9C48  SAVE CURRENT READ/WRITE COUNT (B0E9)
9C48  IN L KEYWORD VALUE (BE5F)
9C58  SET UP TO READ ONE BYTE (B0E9)
9C55  ML1: READ <BE70>
9C58  ERROR? >> 9C66
9C5A  PUT COUNT BACK TO MAXIMUM AGAIN (BE5F)
9C60  GET FIRST CHARACTER ON $200 LINE (BED7)

9C63  AND RETURN THAT TO CALLER (0200)
9C66  RETURN

9C67  READ NEXT LINE OF FILE ******************

9C67  REMOVE CURSOR FROM SCREEN (BE3E)
9C6C  ---
9C6E  ML1: READ <BE70>
9C71  ERROR? >> 9C66
9C73  GET LENGTH ACTUALLY TRANSMITTED (BED8)
9C76  NOTHING? >> 9C8E
9C79  GOT SOMETHING, FIND END OF DATA (BED7)
9C7D  FETCH LAST BYTE OF LINE ($1PF)
9C82  IS IT A RETURN CHARACTER?
9C84  NO, LEAVE LINE ALONE >> 9C8E
9C86  YES, WAS L KEYWORD GIVEN? (BE57)
9C88  YES, LEAVE IT BE >> 9C8E
9C8D  ELSE, CHOP OFF THE RETURN ITSELF
9C8E  AND EXIT WITH A RETURN
9C90  RESTORING Y REG AS YOU GO (BE40)
9C94  RETURN

9C95  READING DIR FILE **********************

9C95  ".
9C97  PROMPT?
9C99  YES, EXIT RIGHT NOW >> 9C8E
9C99  ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
9C9E  SET 80 COLUMNS
9CA5  ML1: GET NAME <BE70>
9CAB  ERROR? >> 9D1F
9CAA  ARE WE AT BEGINNING OF THIS FILE? (BE8)
9CB0  NO, CONTINUE >> 9CDF
9CB2  YES, CAT FLAG = 2
9CBB  THE OPEN FILE LIST TO THE ENTRY LENGTH (8CB8)
9CCD  AND THE NUMBER OF ENTRIES PER BLOCK (B000)

********* FORMAT DIRECTORY NAME **********

9CD0  GJ: FORMAT NAME OF DIRECTORY <BBB8>
9CD3  STORE THE LENGTH OF LINE AT $200
9CD8  PUT A RETURN CHAR AT END OF LINE
9CDF  AND EXIT TO CALLER
9CDE  RETURN
### BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84

**NEXT OBJECT ADDR: 9CDE**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
9CDE | GET CAT FLAG (BE4F)
9CE4 | IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D22
9CF4 | IF MINUS, GO DO SUMMARY LINE OR EXIT >>9C9F
9C6E | POSITIVE, ASSUME NULL LINE WANTED
9C68 | DROP CAT FLAG BY ONE (BE4F)
9C88 | IF ZERO, JUST GO PRINT A BLANK LINE >>9CD3

**FORMAT TITLE LINE ************

9C8D | ELSE, BLANK OUT 280 AND <A6C>
9CF2 | UNPACK "NAME TYPE BLOCKS ETC... <9FB0>
9CF5 | LINE LENGTH IS 80
9CF7 | GO RETURN IT TO CALLER >>9CD3

**FORMAT SUMMARY LINE ************

9CF9 | DO SUMMARY LINE?
9CFA | NO, JUST EXIT (ALL DONE) >>9D1C
9CFD | YES, DROP CAT FLAG SO EXIT NEXT TIME (BE4F)
9DF8 | CLEAR READ/WRITE COUNT (BED9)
9DBA | WLI: READ <BE74>
9DBD | FORMAT BLOCKS FREE AND INUSE SUMMARY LINE <BE79>
9DL1 | GET REF NUM (BED6)
9D14 | AND COPY TO GET/SET LIST (BEC7)
9D1B | NO ERRORS, EXIT >>9CF5
9DBA | ERROR, JUMP TO BI ERROR EXIT >>9D1F
9D1C | "END OF DATA" ERROR
9DBF | GO TO BI ERROR EXIT >>9APS

**FORMAT FILE/DIR ENTRIES ************

9D22 | SET DIR ENTRY NUM COUNTER TO -1
9D27 | GET REF NUM (BED6)
9D2A | *32
9D2F | USE AS INDEX TO GET ENTRY LENGTH (BCFF)
9D35 | AND ENTRIES PER BLOCK FROM OPEN FILE LIST (BD80)
9D38 | POSITION ON EVEN BLOCK BOUNDARY (BEC9)
9D41 | AND GET SECTOR OFFSET (BEC8)
9D45 | SKIP FILE/DIR ENTRIES UNTIL POSITIONED TO (BCBB)
9D48 | CURRENT POSITION IN THIS BLOCK (BCB7)
9D50 | READ NEXT DIR ENTRY FROM FILE <BID1>
9D53 | NO ERROR? >>9D61
9D55 | ERROR, IF RANGE ERROR...
9D57 | NO, TRUE ERROR >>9D1F
9D59 | RANGE ERROR, READY FOR SUMMARY LINE NEXT (BE4F)
9D5E | RETURN A BLANK LINE THIS TIME >>9CD3

**NEXT OBJECT ADDR: 9D5E**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
9D61 | FORMAT FILE/DIR ENTRY INTO $201 <A4C4>
9D64 | AND RETURN IT TO CALLER >>9CF5

**PREFIX INPUT ACTIVE ************

9D67 | PROMPT = "J"?
9D69 | NO, ALL IS WELL >>9D6E
9D6B | YES, RETURN TO IMMEDIATE MODE NOW >>9B58
9D6E | REMOVE CURSOR FROM SCREEN (BE3E)
9D75 | PREFIX NO LONGER ACTIVE AFTER THIS (BE46)
9D7B | COPY PATHNAME BUFFER (PREFIX) (BCBC)
9D7E | TO $200 (81FF)
9D84 | RETURN WITH IT TO BASIC (BC3C)
9D87 | RETURN

**SETUP TO READ LINE FROM EXEC ************

9D8A | SET READ REF NUM FOR EXEC FILE (BCA3)
9D90 | READ TO $200
9D95 | FOR $EF BYTES OF LENGTH
9D9A | (OR UNTIL A RETURN CHAR)
9D9B | RETURN

**OUTPUT INTERCEPT: MODE = C ************

(LOOK FOR CONTROL-D)

9D93 | SAVE REGISTERS <9F62>
9D96 | PRINTING A CONTROL-D?
9D98 | NO >>9D10
9DAA | YES, WRITE OUT ANY BUFFERED DATA <9F4A>
9DAD | NOTHING IN COMMAND LINE (BE4B)
9DB0 | READ FILE INACTIVE (BE44)
9DB3 | WRITE FILE INACTIVE (BE45)
9DB6 | PREFIX READ INACTIVE (BE46)
9DBB | SET MODE = 8 FROM NOW ON <9F76>
9DBE | RESTORE REGS AND EXIT >>9F6C

**GOT A CONTROL-D...**

9DC1 | SET MODE = 4 FROM NOW ON <9F76>
9DCC | RESTORE REGISTERS <9F6C>
9DCC | OUTPUT CHARACTER AND EXIT >>B7F1

**OUTPUT INTERCEPT: MODE = 8 ************

(ASSEMBLE COMMAND LINE)

9DCC | SAVE REGISTERS <9F62>
9D02 | SAVE CHAR IN COMMAND LINE (0200)
9D05 | WAS IT A RETURN?
9D07 | YES, READY TO ROLL >>9DE7
9D09 | NO, BUMP CHARACTER COUNTER (BE4B)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9DE3</td>
<td>ELSE, RESTORE X REG AND EXIT (BE3F)</td>
</tr>
<tr>
<td>9DE6</td>
<td>RETURN</td>
</tr>
<tr>
<td>9DE8</td>
<td>---</td>
</tr>
<tr>
<td>9DE9</td>
<td>NULL LINE? &gt;&gt;9DF6</td>
</tr>
<tr>
<td>9DEB</td>
<td>NO, PUT BACK TRUE CSWL/KSWL &lt;9A00&gt;</td>
</tr>
<tr>
<td>9DEE</td>
<td>SYNTAX SCAN CMD LINE &lt;A677&gt;</td>
</tr>
<tr>
<td>9DF1</td>
<td>ERROR? &gt;&gt;9DE8</td>
</tr>
<tr>
<td>9DF3</td>
<td>NO, PUT BACK BI'S INTERCEPTS &lt;9ABD&gt;</td>
</tr>
<tr>
<td>9DF6</td>
<td>---</td>
</tr>
<tr>
<td>9DF8</td>
<td>MODE = 4 NOW &lt;9F76&gt;</td>
</tr>
<tr>
<td>9DFB</td>
<td>RESTORE REGS AND EXIT &gt;&gt;9F6C</td>
</tr>
<tr>
<td>9DFE</td>
<td>************ WRITE BUFFERED CHARACTER *************************************************</td>
</tr>
<tr>
<td>9E01</td>
<td>CHECK PROMPT</td>
</tr>
<tr>
<td>9E03</td>
<td>CHECK TO SEE IF WE ARE IN &quot;IP&quot;, &gt;&gt;9E11</td>
</tr>
<tr>
<td>9E06</td>
<td>&quot;PRINT&quot;, &quot;LIST&quot;, OR &quot;CALL&quot; STATEMENTS &gt;&gt;9E11</td>
</tr>
<tr>
<td>9E09</td>
<td>OF AN APPLESOFT PROGRAM &gt;&gt;9E6</td>
</tr>
<tr>
<td>9E0B</td>
<td>IF NOT, EXIT TO CALLER... (BE40)</td>
</tr>
<tr>
<td>9E0E</td>
<td>WITH CHARACTER ECHOED TO SCREEN &gt;&gt;9A74</td>
</tr>
<tr>
<td>9E11</td>
<td>GET INDEX TO TEMPORARILY BUFFERED CHAR (BE4A)</td>
</tr>
<tr>
<td>9E16</td>
<td>STORE INTO BUFFER JUST ABOVE HINEM</td>
</tr>
<tr>
<td>9E1B</td>
<td>BUMP INDEX (BE4A)</td>
</tr>
<tr>
<td>9E1E</td>
<td>OK &gt;&gt;9E2B</td>
</tr>
<tr>
<td>9E20</td>
<td>BUFFER FULL, SAVE REGISTERS &lt;&lt;9F62</td>
</tr>
<tr>
<td>9E23</td>
<td>WRITE BUFFER OUT TO DISK &lt;&lt;9EE</td>
</tr>
<tr>
<td>9E26</td>
<td>ERROR? &gt;&gt;9DE0</td>
</tr>
<tr>
<td>9E28</td>
<td>RESTORE REGISTERS &lt;&lt;9F6C</td>
</tr>
<tr>
<td>9E2B</td>
<td>AND EXIT ANYWAY</td>
</tr>
<tr>
<td>9E2C</td>
<td>************ OUTPUT INTERCEPT: MODE = 4 ************************************************</td>
</tr>
<tr>
<td></td>
<td>(INITIAL ENTRY FOR A RUNNING PROGRAM)</td>
</tr>
<tr>
<td></td>
<td>(FLUSH OUT NON COMMAND LINES)</td>
</tr>
<tr>
<td>9E2C</td>
<td>PRINTING A &quot;#&quot;? (9F61)</td>
</tr>
<tr>
<td>9E2F</td>
<td>NO &gt;&gt;9E49</td>
</tr>
<tr>
<td>9E31</td>
<td>YES, SAVE X REGISTER (BE3F)</td>
</tr>
<tr>
<td>9E35</td>
<td>RETURN ADDR IS IN APPLESOFT... (0103)</td>
</tr>
<tr>
<td>9E3B</td>
<td>TRACE ROUTINE...</td>
</tr>
<tr>
<td>9E3C</td>
<td>AT $D8127 (0104)</td>
</tr>
<tr>
<td>9E41</td>
<td>YES &gt;&gt;9EB6</td>
</tr>
<tr>
<td>9E43</td>
<td>NO, RESTORE REGISTERS (9F61)</td>
</tr>
<tr>
<td>9E49</td>
<td>IS WRITE FILE ACTIVE? (BE45)</td>
</tr>
<tr>
<td>9E4C</td>
<td>NOPE &gt;&gt;9E6C</td>
</tr>
<tr>
<td>9E4E</td>
<td>YES, PRINTING A &quot;</td>
</tr>
<tr>
<td>9E58</td>
<td>NO &gt;&gt;9E56</td>
</tr>
<tr>
<td>9E52</td>
<td>YES, SAME AS PROMPT CHARACTER?</td>
</tr>
<tr>
<td>9E54</td>
<td>YES &gt;&gt;9E6</td>
</tr>
<tr>
<td>9E56</td>
<td>NO, PRINTING A RETURN CHAR?</td>
</tr>
<tr>
<td>9E58</td>
<td>NO &gt;&gt;9DFE</td>
</tr>
<tr>
<td>9E5A</td>
<td>YES, GET PROMPT</td>
</tr>
<tr>
<td>9E60</td>
<td>DOES IT INDICATE RECURSION? &gt;&gt;9DFE</td>
</tr>
<tr>
<td>9E62</td>
<td>YES, WRITE BUFFER OUT &lt;9FF4&gt;</td>
</tr>
<tr>
<td>9E65</td>
<td>OUTPUT FILE INACTIVE NOW (BE45)</td>
</tr>
<tr>
<td>9E6A</td>
<td>EXIT WITH RETURN CHAR &gt;&gt;9E9F</td>
</tr>
<tr>
<td>9E6C</td>
<td>---</td>
</tr>
<tr>
<td>9E6D</td>
<td>INPUT FILE ACTIVE? (BE44)</td>
</tr>
<tr>
<td>9E73</td>
<td>NO &gt;&gt;9E7D</td>
</tr>
<tr>
<td>9E75</td>
<td>YES, CHECK PROMPT</td>
</tr>
<tr>
<td>9E77</td>
<td>OR IN $94</td>
</tr>
<tr>
<td>9E79</td>
<td>CONTROL-D?</td>
</tr>
<tr>
<td>9E7B</td>
<td>YES &gt;&gt;9E2A</td>
</tr>
<tr>
<td>9E7D</td>
<td>--</td>
</tr>
<tr>
<td>9E7E</td>
<td>NO, HOW BOUT &quot;]&quot;?</td>
</tr>
<tr>
<td>9E80</td>
<td>NO, EXIT WITH ECHO THEN &gt;&gt;9E9F</td>
</tr>
<tr>
<td>9E82</td>
<td>YES, IS THIS THE PROMPT CHAR?</td>
</tr>
<tr>
<td>9E84</td>
<td>NO, EXIT WITH ECHO &gt;&gt;9E9F</td>
</tr>
<tr>
<td>9E86</td>
<td>YES, SAVE REGISTERS &gt;&gt;9F62</td>
</tr>
<tr>
<td>9E89</td>
<td>CHECK OPEN FILE COUNT (BE4D)</td>
</tr>
<tr>
<td>9E8C</td>
<td>NONE OPEN? &gt;&gt;9F9C</td>
</tr>
<tr>
<td>9E8E</td>
<td>SOME OPEN, WRITE BUFFER OUT &lt;&lt;9F4</td>
</tr>
<tr>
<td>9E91</td>
<td>INDICATE WRITE FILE INACTIVE NOW (BE45)</td>
</tr>
<tr>
<td>9E94</td>
<td>SET TRUE CSWL/KSWL &lt;9A00&gt;</td>
</tr>
<tr>
<td>9E99</td>
<td>PRINT &quot;FILE(S) STILL OPEN&quot; &lt;BE8C&gt;</td>
</tr>
<tr>
<td>9E9C</td>
<td>RESTORE REGS &lt;&lt;9F6C</td>
</tr>
<tr>
<td>9E9F</td>
<td>AND ECHO EXIT &gt;&gt;9A74</td>
</tr>
<tr>
<td>9EA2</td>
<td>---</td>
</tr>
<tr>
<td>9EAA</td>
<td>CHAR IS A RETURN?</td>
</tr>
<tr>
<td>9E95</td>
<td>NO &gt;&gt;9EAA</td>
</tr>
<tr>
<td>9E97</td>
<td>YES, SAME AS LAST CHAR OUTPUT? (BE4C)</td>
</tr>
<tr>
<td>9EAA</td>
<td>(SAVE IT FOR THIS TEST NEXT TIME) (BE4C)</td>
</tr>
<tr>
<td>9EAD</td>
<td>NOT SAME, NO PROBLEM THEN &gt;&gt;9EB1</td>
</tr>
<tr>
<td>9EAF</td>
<td>SAME, MARK PROMPT FOR RECURSION</td>
</tr>
<tr>
<td>9EB1</td>
<td>RETURN</td>
</tr>
<tr>
<td>9EB2</td>
<td>************ APPLESOFT TRACE INTERCEPT ************************************************</td>
</tr>
<tr>
<td></td>
<td>(CONTROL PASSES HERE FOR EVERY STATEMENT)</td>
</tr>
<tr>
<td></td>
<td>(EXECUTED WHILE PRODOS IS ACTIVE)</td>
</tr>
<tr>
<td>9EB2</td>
<td>BUMP APPLESOFT LINE POINTER</td>
</tr>
<tr>
<td>9EB6</td>
<td>---</td>
</tr>
<tr>
<td>9E9A</td>
<td>MARK PROMPT FOR RECURSION</td>
</tr>
<tr>
<td>9EBC</td>
<td>JUST IN CASE WE DIE IN HERE</td>
</tr>
<tr>
<td>9EBE</td>
<td>RESTORE APPLESOFT'S STACK</td>
</tr>
</tbody>
</table>
**BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84**

**NEXT OBJECT ADDR: 9EC1**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9EC1</td>
<td>DOES BI KNOW WE ARE TRACING? (BE41)</td>
</tr>
<tr>
<td>9EC4</td>
<td>YES, REAL LIVE TRACE THEN &gt;&gt;9F39</td>
</tr>
<tr>
<td>9EC6</td>
<td>ELSE, PICK UP NEXT TOKEN ON LINE</td>
</tr>
<tr>
<td>9EC8</td>
<td>IS IT A TOKEN? &gt;&gt;9F1</td>
</tr>
<tr>
<td>9ECC</td>
<td>OR END OF LINE? &gt;&gt;9EEE</td>
</tr>
<tr>
<td>9ECF</td>
<td>NEITHER, DECUREMENT STRING SPACE CTR (BE49)</td>
</tr>
<tr>
<td>9ED0</td>
<td>OK &gt;&gt;9EEC</td>
</tr>
<tr>
<td>9ED2</td>
<td>COMPUTE SIZE OF FREESPACE IN PAGES</td>
</tr>
<tr>
<td>9ED5</td>
<td>AT LEAST 3 PAGES AVAILABLE?</td>
</tr>
<tr>
<td>9ED8</td>
<td>YES &gt;&gt;9EE5</td>
</tr>
<tr>
<td>9EDB</td>
<td>NO, WRITE BUFFERED DATA &lt;&lt;9F4</td>
</tr>
<tr>
<td>9EE2</td>
<td>AND THEN GARBAGE COLLECT &lt;&lt;9B4</td>
</tr>
<tr>
<td>9EE3</td>
<td>COMPUTE FREE SPACE NOW</td>
</tr>
<tr>
<td>9EE5</td>
<td>AND SAVE IN STRING SPACE CTR (BE49)</td>
</tr>
<tr>
<td>9EEA</td>
<td>GET NEXT TOKEN</td>
</tr>
<tr>
<td>9EEC</td>
<td>---</td>
</tr>
<tr>
<td>9EEE</td>
<td>JUMP BACK INTO APPLESOFT TO EXECUTE IT &gt;&gt;D820</td>
</tr>
<tr>
<td>9EF1</td>
<td>STORE TOKEN IN PROMPT</td>
</tr>
<tr>
<td>9EF4</td>
<td>LOOK UP TOKEN IN BI'S TOKEN TABLE (B799)</td>
</tr>
<tr>
<td>9EF7</td>
<td>ITS NOT ONE BI IS INTERESTED IN &gt;&gt;9EEE</td>
</tr>
<tr>
<td>9EF9</td>
<td>IT IS INTERESTING, CHANGE BRANCH (9EFD)</td>
</tr>
<tr>
<td>9EFc</td>
<td>AND JUMP TO ONE OF THE FOLLOWING: &gt;&gt;9Ff</td>
</tr>
<tr>
<td>9F11</td>
<td>IF OR PRINT: PROMPT = 0</td>
</tr>
<tr>
<td>9F13</td>
<td>CLEAR OUT LAST CHAR SAVEDREA (BE4C)</td>
</tr>
<tr>
<td>9F15</td>
<td>GO TO MODE = C NEXT TIME THRU (BE03)</td>
</tr>
<tr>
<td>9F16</td>
<td>(BEGIN LOOKING FOR COMMANDS) (BE38)</td>
</tr>
<tr>
<td>9F18</td>
<td>NOW GO PROCESS THE IF OR PRINT &gt;&gt;9F2</td>
</tr>
<tr>
<td>9F1I</td>
<td>LIST: PROMPT = 1</td>
</tr>
<tr>
<td>9F13</td>
<td>(DON'T LOOK FOR COMMANDS NOW)</td>
</tr>
<tr>
<td>9F15</td>
<td>GO DO IT &gt;&gt;9F2</td>
</tr>
<tr>
<td>9F17</td>
<td>CALL: PROMPT = 2</td>
</tr>
<tr>
<td>9F19</td>
<td>(DON'T LOOK FOR COMMANDS NOW)</td>
</tr>
<tr>
<td>9F18</td>
<td>GO DO IT &gt;&gt;9F2</td>
</tr>
<tr>
<td>9FID</td>
<td>LET: DECREMENT STRING CTR</td>
</tr>
<tr>
<td>9F1E</td>
<td>AND GO BACK FOR NEXT TOKEN &gt;&gt;9E</td>
</tr>
<tr>
<td>9F21</td>
<td>TRACE: TURN TRACE ON (BE41)</td>
</tr>
<tr>
<td>9F24</td>
<td>THEN CONTINUE BELOW &gt;&gt;9F</td>
</tr>
<tr>
<td>9F26</td>
<td>NOTRACE: DROP INTO BACKGROUND TRACE (BE41)</td>
</tr>
<tr>
<td>9F29</td>
<td>CHANGE TOKEN TO &quot;TRACE&quot;</td>
</tr>
<tr>
<td>9F2A</td>
<td>FORCE ON APPLESOFT TRACE</td>
</tr>
<tr>
<td>9F2B</td>
<td>---</td>
</tr>
<tr>
<td>9F2C</td>
<td>GO BACK TO APPLESOFT TO PERFORM IT &gt;&gt;D820</td>
</tr>
</tbody>
</table>

**BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84**

**NEXT OBJECT ADDR: 9F2F**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9F32</td>
<td>RESUME: CLEAR ONERR CODE</td>
</tr>
<tr>
<td>9F37</td>
<td>GO TO APPLESOFT TO PROCESS IT &gt;&gt;9E</td>
</tr>
</tbody>
</table>

********* REAL TRACE ACTIVE *****************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9F39</td>
<td>RESTORE TRUE CSWL/KSWL &lt;9A06&gt;</td>
</tr>
<tr>
<td>9F3E</td>
<td>PRINT &quot;#&quot; &lt;FED&gt;</td>
</tr>
<tr>
<td>9F45</td>
<td>USE APPLESOFT TO PRINT CURRENT LINE NO. &lt;ED2&gt;</td>
</tr>
<tr>
<td>9F4A</td>
<td>PRINT A BLANK SPACE &lt;FED&gt;</td>
</tr>
<tr>
<td>9F4D</td>
<td>PUT BI'S CSWL/KSWL INTERCEPTS BACK &lt;9AC&gt;</td>
</tr>
<tr>
<td>9F51</td>
<td>THEN GO BACK AND HANDLE AS USUAL &gt;&gt;9E6</td>
</tr>
<tr>
<td>9F54</td>
<td>LOOKING FOR A LOWER CASE &quot;c&quot;</td>
</tr>
<tr>
<td>9F58</td>
<td>LOOKING FOR A &quot;#&quot;</td>
</tr>
<tr>
<td>9F5A</td>
<td>STORE CHAR TO SEARCH FOR (9F61)</td>
</tr>
<tr>
<td>9F5E</td>
<td>BRANCH BACK INTO APPLESOFT &gt;&gt;9EC</td>
</tr>
<tr>
<td>9F60</td>
<td>BREAK IF Y IS ZERO:11</td>
</tr>
<tr>
<td>9F61</td>
<td>&quot;#&quot; CHARACTER (ASOFT TRACE CHAR)</td>
</tr>
<tr>
<td>9F62</td>
<td>************ SAVE CALLER'S REGISTERS ************</td>
</tr>
<tr>
<td>9F62</td>
<td>SAVE A,X AND Y REGS (BE3E)</td>
</tr>
<tr>
<td>9F6B</td>
<td>RETURN</td>
</tr>
<tr>
<td>9F6C</td>
<td>************ RESTORE CALLERS REGISTERS ************</td>
</tr>
<tr>
<td>9F6C</td>
<td>RESTORE A,X AND Y REGS (BE3E)</td>
</tr>
<tr>
<td>9F75</td>
<td>RETURN</td>
</tr>
<tr>
<td>9F76</td>
<td>************ SET MODE AND CSWL/KSWL ************</td>
</tr>
<tr>
<td>9F76</td>
<td>STORE &quot;STATE&quot; MODE FROM X REGISTER (BE42)</td>
</tr>
<tr>
<td>9F7B</td>
<td>COPY PROPER CSWL/KSWL VALUES TO REDIRECT... (B7F7)</td>
</tr>
<tr>
<td>9F7E</td>
<td>VECTOR DEPENDING ON CURRENT MODE (BE3E)</td>
</tr>
<tr>
<td>9F87</td>
<td>RETURN</td>
</tr>
<tr>
<td>9F88</td>
<td>************ PRINTER: PRINT ERROR MSG ************</td>
</tr>
<tr>
<td>9F88</td>
<td>---</td>
</tr>
<tr>
<td>9F89</td>
<td>GET INDEX INTO PACKED MESSAGE TEXTS (BA13)</td>
</tr>
<tr>
<td>9F8C</td>
<td>UNPACK MESSAGE INTO $201 &lt;9FB6&gt;</td>
</tr>
<tr>
<td>9F92</td>
<td>SAVE THE LENGTH (B9C6)</td>
</tr>
<tr>
<td>9F95</td>
<td>SKIP A LINE &lt;9FAB&gt;</td>
</tr>
<tr>
<td>9F9A</td>
<td>PRINT A BELL &lt;FAD&gt;</td>
</tr>
<tr>
<td>9F9D</td>
<td>---</td>
</tr>
<tr>
<td>9F9F</td>
<td>PRINT CONTENTS OF $201 MSG BUFFER ($201)</td>
</tr>
<tr>
<td>9FAB</td>
<td>PRINT A RETURN CHARACTER</td>
</tr>
<tr>
<td>9FAD</td>
<td>AND EXIT &gt;&gt;FED</td>
</tr>
</tbody>
</table>
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 9FD

ADDR DESCRIPTION/CONTENTS

9FB0 ******* UNPACK ERROR MESSAGE ****************************

9FB0 NOTHING IN BUFFER AT FIRST
9FB6 GET A NIBBLE FROM PACKED MSG <9FD2
9FB9 NON-ZERO, COMMON CHARACTER >>9FC8
9FBB IF ZERO, GET NEXT NIBBLE <9FD2
9FBD AND CONVERT TO UNCOMMON CHAR INDEX
9FC0 ---
9FC1 GET THE LETTER THIS NIBBLE REPRESENTS (BA28)
9FC4 ZERO? THEN END OF MESSAGE >>9FD1
9FC6 GET INDEX INTO OUTPUT BUFFER (8E4B)
9FC9 AND STORE THE CHARACTER THERE (8201)
9FCC BUMP INDEX (8E4B)
9FCF AND CONTINUE >>9FB6
9FD1 RETURN

9FD2 ******* UNPACK MESSAGE BYTE ****************************

9FD2 GET NEXT MSG BYTE (BA48)
9FD5 WORKING ON SECOND NIBBLE? >>9FE9
9FDE NO, TAB INDICATOR? >>9DFD
9FDF NO, ISOLATE HIGH NIBBLE
9FDE NEXT TIME GET LOW NIBBLE
9FDD RETURN

9DFD ---
9DFB GET TAB POSITION (BA48)
9DF3 AND BUMP OUTPUT PTR ACCORDINGLY (BE4B)
9FEB THEN GO BACK FOR NEXT NIBBLE >>9FD2
9FE3 BUMP BYTE PTR FOR NEXT TIME
9FEA ISOLATE LOW NIBBLE
9FEC NEXT TIME GET HIGH NIBBLE
9FED RETURN

9FE6 ******* WRITE ONE BUFFERED BYTE ****************************

9FE6 SET UP COUNT OF 0001
9F72 AND JUMP INTO ROUTINE BELOW >>A007

9FF4 ******* WRITE BUFFERED DATA/TEST ERROR ***********************

9FF4 WRITE BUFFERED DATA <A000>
9FF7 OK? THEN EXIT >>A01C
9FFA ERROR, POP OUT OF THIS SUBROUTINE
9FFD AND GO TO ERROR HANDLER >>9AF8

BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 9FDD

ADDR DESCRIPTION/CONTENTS

A000 ******* WRITE ALL BUFFERED DATA ****************************

A000 ---
A002 GET BUFFERED DATA COUNT (BE4A)
A005 NONE BUFFERED? >>A01B
A007 STORE BUFFERED DATA COUNT IN RW PARGS (BED9)
A009 MLT: WRITE <BE78>
A015 NOTHING BUFFERED NOW, COUNT=0 (BE4A)
A019 ERROR? >>A01C
A01B NO, EXIT
A01C RETURN

A01D ******* SPECIAL GARBAGE COLLECT *******************************
(PULL OUT STRING CONSTANTS ALSO)

A01D DO GARBAGE COLLECTION NORMALLY FIRST <A044>
A020 ERROR? >>A043
A024 START OF STRING AREA = PROGRAM START PTR (BC84)
A02C USE GENERAL PURPOSE BUFFER (ABOVE HIMEM)
A02E FOR A GARBAGE COLLECT WORKAREA (BC7D)
A033 IT IS 3+1 PAGES IN LENGTH (BC7E)
A036 END OF STRING AREA IS AT END OF FREEAREA (BC86)
A048 GO COLLECT CONSTANT STRINGS NOW <A085>
A043 THEN EXIT

A044 ******* "PRE" COMMAND *************************************
(PAST APPLESOF SOFTWARE GARBAGE COLLECTION)

---------
| GENERAL PURPOSE BUFFER | -----
| (TOP OF OLD STRINGS) | -----
---------
| HIMEM =--|--
| NEW STRINGS BUILDING | ---
| DOWN | ---
---------
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD STRINGS</td>
<td>----</td>
</tr>
<tr>
<td>FREE AREA</td>
<td>----</td>
</tr>
</tbody>
</table>

TOP PART OF OLD STRINGS IS SAVED IN THE GENERAL PURPOSE BUFFER OR IN THE FREE AREA (WHICHEVER IS LARGER) AND A NEW COPY OF THE STRINGS IS BUILT JUST BELOW HIMEM.
A044 STRING AREA START IS ON PAGE BOUNDARY
A04B ASSUME 4 PAGE WORKAREA (BC7E)
A050 IN GENERAL PURPOSE BUFFER ABOVE HIMEM (BC7D)
A055 STRING START PTR IS START OF STRING AREA (BC84)
A059 COMPUTE NUMBER OF FREE PAGES
A05B AT LEAST 7?
A05D IF NOT, USE G.P. WORKAREA INSTEAD >> A079
A05F DON'T USE ALL OF FREE AREA (LEAVE $300)
A061 NEW WORKAREA SIZE IS FREE AREA SIZE-$300 (BC7E)
A065 SET PTR TO WORKAREA AT FIRST FREE PAGE
A06D COMPUTE NUMBER OF STRING PAGES
A071 USE SMALLER OF STRING PAGES OR WORKAREA SIZE (BC7E)
A075 AS NEW WORKAREA SIZE (BC7E)
A079 END OF STRING AREA IS HIMEM
A085 RECORD WHETHER LAST PAGE IS PARTIAL
A089 STRING START MSB IS HIMEM INITIALLY (BC86)
A08E ADJUST LORANGE AND HIRANGE MSB'S
A090 FOR PARTIAL PAGES AT EITHER END, (BC7F)
A093 SETTING THEM AT HIMEM FOR NOW.
A09C SET UP ARRAY END MSB +1 FOR COMPARES (BC82)
A09F $3E/$3F -- FIRST VARIABLE (LESS 7 BYTES)
A0A0 (EACH VARIABLE IS 7 BYTES)
A0A8 SET UP ARRAY START LSB FOR COMPARES
A0B0 GET LORANGE VALUE (BC7F)
A0B3 PRIOR TO STRING AREA? (BC84)
A0B6 YES, THEN DONEL >> A0F6
A0B8 ELSE, DROP LORANGE BY WORKAREA SIZE (BC7E)
A0BB AND SAVE THIS VALUE (BC7C)
A0BE NO DROP IT ALSO BY THE DISTANCE BETWEEN
A0C0 ... THE OLD LORANGE AND THE STRING START PTR (BC7F)
A0CA USE THE LOWER OF THE TWO VALUES (BC7C)
A0CF TO PRODUCE THE MAXIMUM SIZED RANGE (BC7C)
A0D2 IS THIS BELOW THE BOTTOM OF THE STRINGS? (BC84)
A0D5 NO >> A0DC
A0D7 YES, USE THE BOTTOM POINTER INSTEAD (BC84)
A0DA (ADJUSTING FOR PARTIAL PAGE)
A0DC STORE FINAL LORANGE VALUE (BC7F)
A0DF COPY SOME PAGES BELOW HIRANGE TO WORKAREA <A195>
A0E2 (TO MAKE ROOM FOR NEW STRINGS)
A0F4 COLLECT SIMPLE STRING VARS FOR THIS RANGE <A0F7>
A0F7 ERROR? >> A0F4
A0F9 THEN COLLECT STRING ARRAYS <A12D>
A0FC NEW HIRANGE = OLD LORANGE (BC7F)
A0F2 CONTINUE LOOPING >> A0F9
A0F4 IF ERROR, "RAM TOO LARGE"
A0F6 EXIT TO CALLER

A0B7 *************** COLLECT SIMPLE STRINGS ***************

A0B7 ---
A0B8 ADD 7 BYTES TO $3E/$3F PTR FOR NEXT VAR
A102 PTR AT ARRAYS NOW?
A108 IF SO, WE ARE DONE >> A12B
A10A IS THIS A STRING VARIABLE?
A111 NO >> A0B7
A113 MAKE ABSOLUTELY SURE
A117 GET MSB OF STRING POINTER
A11B IS IT WITHIN MY RANGE? (BC7F)
A11E NO >> A0F6
A123 NO >> A0F7
A125 YES, PULL IT OUT AND TACK IT TO HIMEM <A1B8>
A128 ALL WENT WELL, GET NEXT VARIABLE >> A0F8
A12A IF ERROR, EXIT NOW
A12B NORMAL EXIT TO CALLER
A12C RETURN

A12D *************** COLLECT STRING ARRAYS ***************

A12D FIND THE NEXT ARRAY <A1C5>
A130 NO MORE? >> A12B
A132 GOM ONE, GET MSB OF ITS STRING PTR
A136 WITHIN MY RANGE? (BC7F)
A139 NO >> A146
A13E NO >> A146
A140 YES, PULL IT OUT AND TACK IT TO HIMEM <A1B8>
A143 AND CONTINUE WITH NEXT ARRAY ELEMENT >> A147
A145 ERROR EXIT
A146 ---
A147 BUMP POINTER TO NEXT ARRAY MEMBER
A151 POINTER NOW AT NEXT ARRAY? (BC81)
A154 NO, DO THIS ELEMENT >> A132
A158 NO >> A132
A15A YES, SET UP TO PROCESS THAT ONE THEN >> A12D

A15C *************** FIND NEXT STRING ARRAY ***************

A15C ---
A15D $3E --> ARRAY VARIABLES (BC81)
A164 AT END OF ARRAY VARS
A166 NO, CONTINUE >> A16C
A16A YES, OUT (CARRY SET, NO MORE ARRAYS) >> A194
A16C  POINT TO ARRAY FOLLOWING THIS (LSB AND...)
A176  MSB TO X REGISTER
A17D  CHECK TYPE OF VARIABLE
A182  SKIP INTEGER AND REAL ARRAYS >>A15C
A186  GET NUMBER OF DIMENSIONS
A188  *2 TO SKIP SIZES
A189  *5 TO SKIP FIXED STUFF AT BEGINNING
A18D  POINT TO FIRST ARRAY MEMBER
A191  READY TO ROLL, $3E POINTS TO IT
A194  RETURN

A195  ************ COPY PAGES TO WORKAREA **************

A195  $3A/$3B  --> FIRST PAGE TO SAVE (BC7C)
A19A  $3C/$3D  --> WORKAREA (BC7D)
A195  $3E  --> SIZE OF WORKAREA (BC7E)
A199  ---
A1B7  EXIT WHEN FINISHED

A1B8  ************ PULL STRING OUT **********************

A1B8  IS STRING BELOW SAVED AREA? (BC7C)
A1BB  YES, ITS STILL THERE THEN >>A1C4
A1BD  ELSE, POINT TO SAVED STRING IN WORKAREA (BC7C)
A1C4  $3A/$3B  --> STRING
A1CF  DROP STRING START PTR BY LEN OF THIS STRING
A1D4  UPDATE STRING's LSB IN VARIABLE PTR
A1DB  FIX UP MSB OF STRING START PTR ALSO
A1DD  AND OF VARIABLE PTR
A1E1  IS THIS A NULL LENGTH STRING?
A1E3  YES, NO MOVE TO DO >>A1BE
A1BE  ---
A1E7  ELSE, COPY STRING OUT
A1EE  ---
A1F0  OUT OF FREESPACE? (BC82)
A1F4  RETURN TO CALLER WITH INDICATION

A1F5  ************ ALLOCATE BUFFER **********************

A1F5  NEED 4 PAGES
<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: A2B4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADDR</strong></td>
<td><strong>DESCRIPTION/CONTENTS</strong></td>
</tr>
<tr>
<td>A2B5</td>
<td>************ GETBUF:R; GET A BUFFER ***********************</td>
</tr>
<tr>
<td></td>
<td>THIS ROUTINE IS CALLED THROUGH AN EXTERNAL</td>
</tr>
<tr>
<td></td>
<td>ENTRY POINT IN THE GLOBAL PAGE. IT ALLO-</td>
</tr>
<tr>
<td></td>
<td>CATES A FIXED LOCATION BUFFER BETWEEN THE</td>
</tr>
<tr>
<td></td>
<td>BI AND ITS BUFFERS.</td>
</tr>
<tr>
<td>A2B5</td>
<td>ALLOCATE A BUFFER OF ANY SIZE (A= PAGES) &lt;A1P7&gt;</td>
</tr>
<tr>
<td>A2B8</td>
<td>ERROR? &gt;&gt;A300</td>
</tr>
<tr>
<td>A2BD</td>
<td>FIND FIRST PAGE OF BUFFER (BB4A)</td>
</tr>
<tr>
<td>A2C4</td>
<td>GET FILE OPEN COUNT (&lt;B34D)</td>
</tr>
<tr>
<td>A2C7</td>
<td>WORE OPEN? &gt;&gt;A2EA</td>
</tr>
<tr>
<td>A2C9</td>
<td>BUMP BUFFER PAGE PTR BY $400 (BB49)</td>
</tr>
<tr>
<td>A2CD</td>
<td>TO POINT TO PREVIOUSLY ALLOCATED</td>
</tr>
<tr>
<td>A2CF</td>
<td>BUFFER. (BB49)</td>
</tr>
<tr>
<td>A2D2</td>
<td>FIND OPEN FILE WITH THIS BUFFER (BC93)</td>
</tr>
<tr>
<td>A2D7</td>
<td>GOT IT, (BC39)</td>
</tr>
<tr>
<td>A2DA</td>
<td>SET FILE BUFFER REAL LOW IN MEMORY &lt;A352&gt;</td>
</tr>
<tr>
<td>A2DD</td>
<td>THEN SET IT TO NEW BUFFER LOCATION &lt;A29B&gt;</td>
</tr>
<tr>
<td>A2E0</td>
<td>BELOW ALL OTHERS (BC93)</td>
</tr>
<tr>
<td>A2E7</td>
<td>DO THIS FOR EACH OPEN FILE...</td>
</tr>
<tr>
<td>A2EB</td>
<td>THEREBY INSERTING A BLANK BUFFER &gt;&gt;A2D2</td>
</tr>
<tr>
<td>A2ED</td>
<td>IS EXEC FILE ACTIVE? (BE43)</td>
</tr>
<tr>
<td>A2F0</td>
<td>NO, DONE &gt;&gt;A2FF</td>
</tr>
<tr>
<td>A2F2</td>
<td>YES,</td>
</tr>
<tr>
<td>A2F4</td>
<td>MOVE EXEC BUFFER DOWN ALSO &lt;A352&gt;</td>
</tr>
<tr>
<td>A2FD</td>
<td>AND BUMP UP ABOVE IT</td>
</tr>
<tr>
<td>A2FF</td>
<td>EXIT TO CALLER</td>
</tr>
<tr>
<td>A300</td>
<td>RETURN</td>
</tr>
<tr>
<td>A301</td>
<td>************ FREEBUF:R; FREE BUFFER ***********************</td>
</tr>
<tr>
<td></td>
<td>THIS ROUTINE IS CALLED THROUGH AN EXTERNAL</td>
</tr>
<tr>
<td></td>
<td>ENTRY POINT IN THE GLOBAL PAGE. IT FREES</td>
</tr>
<tr>
<td></td>
<td>A FIXED LOCATION BUFFER PREVIOUSLY ALLO-</td>
</tr>
<tr>
<td></td>
<td>CATED BY GETBUF:R.</td>
</tr>
<tr>
<td>A301</td>
<td>GET COUNT OF OPEN FILES (BE4D)</td>
</tr>
<tr>
<td>A305</td>
<td>INDEX THIS BY 4 PAGES PER FILE</td>
</tr>
<tr>
<td>A306</td>
<td>ADD TO HIMEM MSB</td>
</tr>
<tr>
<td>A308</td>
<td>SAVE THIS AS TOP OF BUFFERS (BB49)</td>
</tr>
<tr>
<td>A30D</td>
<td>THEN SET UP BOTTOM AS HIMEM MSB (BB4A)</td>
</tr>
<tr>
<td>A310</td>
<td>GET OLD ORIGINAL HIMEM (BEFORE ANY BUFFERS) (BE98)</td>
</tr>
<tr>
<td>A313</td>
<td>SAME AS THIS ONE?</td>
</tr>
<tr>
<td>A315</td>
<td>THEN NOTHING ELSE TO DO &gt;&gt;A350</td>
</tr>
<tr>
<td>A317</td>
<td>ASSUME NO BUFFERS BY REPLACING OLD HIMEM</td>
</tr>
<tr>
<td>A319</td>
<td>ANY EXEC FILE OPEN? (BE43)</td>
</tr>
<tr>
<td>A31C</td>
<td>NO, CONTINUE &gt;&gt;A323</td>
</tr>
<tr>
<td>A31E</td>
<td>YES, MOVE EXEC BUFFER TO OLD HIMEM &lt;A2F2&gt;</td>
</tr>
<tr>
<td>A321</td>
<td>AND GO MOVE HIMEM DOWN BY $400 &gt;&gt;A341</td>
</tr>
<tr>
<td>A323</td>
<td>ELSE, START WITH TOP BUFFER (BB49)</td>
</tr>
<tr>
<td>A326</td>
<td>ANY OPEN FILES? (BE4D)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: A329</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADDR</strong></td>
<td><strong>DESCRIPTION/CONTENTS</strong></td>
</tr>
<tr>
<td>A329</td>
<td>IF NOT, WE ARE DONE &gt;&gt;A340</td>
</tr>
<tr>
<td>A32B</td>
<td>SEARCH FOR OPEN FILE WITH THIS BUFFER (BC93)</td>
</tr>
<tr>
<td>A32E</td>
<td>NOT IT? &gt;&gt;A34A</td>
</tr>
<tr>
<td>A330</td>
<td>GOT IT, GIVE IT NEW HOME AT HIMEM</td>
</tr>
<tr>
<td>A332</td>
<td>AND SET BUFFER LOW &lt;A352&gt;</td>
</tr>
<tr>
<td>A335</td>
<td>THEN TO NEW LOC 'A29B'</td>
</tr>
<tr>
<td>A339</td>
<td>DROP TOP BUFFER PTR BY $400 (BB49)</td>
</tr>
<tr>
<td>A341</td>
<td>AND DROP HIMEM BY $400</td>
</tr>
<tr>
<td>A348</td>
<td>AND GO DO NEXT BUFFER &gt;&gt;A323</td>
</tr>
<tr>
<td>A34A</td>
<td>---</td>
</tr>
<tr>
<td>A34B</td>
<td>(LOOP TO SEARCH FOR OPEN FILES) &gt;&gt;A32B</td>
</tr>
<tr>
<td>A34D</td>
<td>WHEN FINISHED, GARBAGE COLLECT &lt;A944&gt;</td>
</tr>
<tr>
<td>A350</td>
<td>---</td>
</tr>
<tr>
<td>A351</td>
<td>THEN EXIT NORMALLY TO CALLER</td>
</tr>
<tr>
<td></td>
<td>******** SET BUFFER BELOW ALL OTHERS ***</td>
</tr>
<tr>
<td>A352</td>
<td>---</td>
</tr>
<tr>
<td>A353</td>
<td>USE BOTTOM BUFFER PTR (BB4A)</td>
</tr>
<tr>
<td>A356</td>
<td>SET FILE BUFFER &lt;A29B&gt;</td>
</tr>
<tr>
<td>A35A</td>
<td>AND EXIT</td>
</tr>
<tr>
<td>A35B</td>
<td>*********** COPY BLOCK DOWN IN MEMORY ***********************</td>
</tr>
<tr>
<td>A35B</td>
<td>COPY ALL FULL PAGES DOWN TO THEIR NEW HOME</td>
</tr>
<tr>
<td>A362</td>
<td>COPYING $3A---$3C</td>
</tr>
<tr>
<td>A369</td>
<td>BUMP BOTH MSB'S</td>
</tr>
<tr>
<td>A36D</td>
<td>DROP PAGE COUNTER (BC93)</td>
</tr>
<tr>
<td>A370</td>
<td>AND CONTINUE &gt;&gt;A362</td>
</tr>
<tr>
<td>A372</td>
<td>NO SHORT LAST PAGE? (BC92)</td>
</tr>
<tr>
<td>A375</td>
<td>THEN EXIT NOW &gt;&gt;A37E</td>
</tr>
<tr>
<td>A377</td>
<td>ELSE, COPY PARTIAL PAGE</td>
</tr>
<tr>
<td>A37E</td>
<td>THEN EXIT</td>
</tr>
<tr>
<td>A37F</td>
<td>*********** COPY BLOCK UP IN MEMORY ***********************</td>
</tr>
<tr>
<td>A37F</td>
<td>PARTIAL PAGE? (BC92)</td>
</tr>
<tr>
<td>A380</td>
<td>NO, JUST COPY FULL PAGES NOW &gt;&gt;A38B</td>
</tr>
<tr>
<td>A384</td>
<td>YES, COPY SHORT PAGE FIRST 'A996'</td>
</tr>
<tr>
<td>A387</td>
<td>DROP BOTH MSB'S</td>
</tr>
<tr>
<td>A38B</td>
<td>PAGE COUNT GONE TO ZERO? (BC93)</td>
</tr>
<tr>
<td>A38E</td>
<td>YES, DONE &gt;&gt;A39E</td>
</tr>
<tr>
<td>A390</td>
<td>ELSE, DROP PAGE COUNT (BC93)</td>
</tr>
<tr>
<td>A393</td>
<td>AND GO COPY A FULL PAGE UP &gt;&gt;A384</td>
</tr>
<tr>
<td>A396</td>
<td>---</td>
</tr>
<tr>
<td>A397</td>
<td>COPY REMAINDER OF PAGE UP (BACKWARDS)</td>
</tr>
<tr>
<td>A39E</td>
<td>RETURN</td>
</tr>
</tbody>
</table>
A39F ******* ADJUST ALL STRING ADDRS ***********************
(BC87 HAS ADDITIVE ADJUSTMENT FACTOR)

A39F USE LOMEM PAGE AS MSF FOR $3E/3F
A3A3 GET LOMEM LSB
A3A5 AND END OF SIMPLE VARS PAGE
A3AB JUMP INTO THE LOOP >>A3AF
A3AA ---
A3AB SKIP ONE SIMPLE VARIABLE
A3AF ---
A3B1 OVERFLOW? >>A3B5
A3B3 YES, BUMP MSB
A3B5 FINISHED WITH SIMPLE VARS?
A3B9 (CHECK BOTH MSB AND LSB OF PTR)
A3BB ---
A3BC YES... >>A3D2
A3BE NO,
A3C0 LOOK AT A SIMPLE VARIABLE
A3C5 SKIP INTEGER AND REAL VARS >>A3AA
A3C7 (DOUBLE CHECK MSB)
A3CB ITS A STRING, POINT TO ITS LEN/ADDR
A3CC ADJUST IT IF NECESSARY <A3F9>
A3CF THEN SKIP OVER IT >>A3AA

A3D2 COPY ARRAYS STARTING LSB
A3D4 (MSB IS IN X REGISTER NOW) (BC81)
A3D7 ---
A3DB FIND A STRING ARRAY <A15C>
A3DB NO MORE? THEN DONE... >>A40C
A3DD ---
A3E0 ADJUST ITS ADDRESS IF NEED BE <A3F9>
A3E6 SKIP TO NEXT STRING ELEMENT OF ARRAY
A3EE AT END OF THIS ARRAY YET? (BC81)
A3F1 NO... >>A3DD
A3F3 (CHECK MSB ALSO)
A3F7 YES..., GO GET NEXT ARRAY >>A3D7

A3F9 ******* ADJUST A STRING ADDRESS ***********************

A3F9 GET STRING LENGTH
A3FB IGNORE NULL STRINGS >>A40C
A3FD POINT TO MSF OF ADDRESS
A3FF IS STRING STORED OUTSIDE OF PROGRAM?
A40J NO, LEAVE IT ALONE >>A40C
A405 STORE ABOVE LOMEM, ADD FACTOR TO MSB
A40C THEN EXIT

A46D ********** COMPRESS ALL ASSOF VARS ******************
THIS ROUTINE SQUEEZES ALL APPLESOFT VARS
UP AGAINST THE BOTTOM OF THE STRINGS
HIMEM -->

<table>
<thead>
<tr>
<th>STRINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY VARS</td>
</tr>
<tr>
<td>SIMPLE VARS</td>
</tr>
</tbody>
</table>

A48D GARBAGE COLLECT FIRST <A81D>
A470 ERROR? >>A471
A412 COMPUTE LENGTH OF SIMPLE AND ARRAY VARS
A417 AND SAVE IT (BC89)
A427 NEXT, COMPUTE LENGTH OF SIMPLE VARS ONLY
A42B AND SAVE IT (BC89)
A435 SUBTRACT VAR LENGTH FROM STRING START
A437 TO FIND A PLACE TO PUT THE VARS UNDER (BC92)
A43A THE STRINGS (START ON AN EVEN PAGE BOUND)
A440 $3C/$3D --> PLACE TO PUT VARS
A447 $3A/$3B --> START OF VARS (ROUNDED TO EVEN
PAGE ALIGNMENT)
A44F COPY VARS UP AGAINST STRINGS <A37F>
A454 STORE START OF VARS PTR (BC88)
A457 BUMPING PAGE NUMBER BY ONE
A463 SUBTRACT THIS PTR FROM HIMEM TO COMPUTE (BC90)
A466 TOTAL LENGTH OF COMBINED VARS/STRINGS
A46B AND SAVE THIS TOO (BC89)
A46D ALSO, SAVE HIMEM MSB IN CASE THEY ARE MOVED
A471 DONE, EXIT

A472 ********** REEXPAND COMPRESSED VARS *************
THIS ROUTINE MOVES SIMPLE AND ARRAY VARS
BACK DOWN TO LOMEM.
HIMEM -->

<table>
<thead>
<tr>
<th>STRINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREE SPACE</td>
</tr>
<tr>
<td>-------</td>
</tr>
</tbody>
</table>
BASIC Interpreter (Bl) -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 4127

ADDR DESCRIPTION/CONTENTS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ARRAY VARS</td>
<td>SIMPLE VARS</td>
<td></td>
</tr>
<tr>
<td>LOMEM --</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A472 SAVE LENGTH OF SIMPLE AND ARRAY VARS (BC89)
A479 ADD THIS TO START OF COMPRESSED VARS PTR
A487 $3C/$3D --- LOMEM (WHERE TO PUT SIMPLE VARS)
A488 $68/$6C --- WHERE TO PUT ARRAY VARS
A499 $3A/$3B --- START OF COMPRESSED VARS (BC88)
A4A3 COPY SIMPLE/ARRAY VARS DOWN TO LOMEM <A55B>
A4A9 COMPUTE START OF STRINGS BY ADDING VARS
A4AC LENGTH TO VARS ORIGIN
A4B5 DID HIMEM MOVE SINCE VARS WERE COMPRESSED?
A4BA NO... >>A4C2
A4BC YES, ADJUST BY DIFFERENCE IN HIMEM'S (BC87)
A4BF GO ADJUST ALL STRING POINTERS <A39F>
A4C2 THEN EXIT
A4C3 RETURN

A4C4 ********** FORMAT CATALOG ENTRY LINE ***********************
A4C4 PUT OUT A BLANK LINE <A66C>
A4C7 DOUBLE QUOTE TO $29D
A4CC GET LENGTH OF NAME ($259)
A4D2 COPY NAME TO LINE ($259)
A4D3 ZERO ACCUMULATOR FOR LATER (BCB1)
A4E0 GET FILE TYPE ($269)
A4E3 I KNOW OF ONLY 13
A4E5 ---
A4E7 LOOK UP FILE TYPE IN TABLE (BC99)
A4EA FOUND IT? >>A4F7
A4FB FILE TYPE NOT IN MY TABLE
A4F2 PRINT IT IN HEXADECIMAL <A612>
A4F5 AND CONTINUE BELOW >>A53B
A4F7 ELSE, FOR KNOWN TYPES
A4FA COPY NAME OF TYPE TO THE LINE (BC97)
A505 @0 COLUMNS PER LINE? (BC96)
A508 YES... >>A553
A50A NO,
A50C BIN FILE?
A50E YES... >>A525
A510 TXT FILE?

BASIC Interpreter (Bl) -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 4512

ADDR DESCRIPTION/CONTENTS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A512 NO... &gt;&gt;A53B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A514 YES, R VALUE GIVEN AS SUBTYPE
A51F CONVERT R VALUE TO DECIMAL <A62P>
A522 SKIP OVER BIN CODE >>A536 |
A525 BIN FILE, USE AD VALUE AS SUBTYPE
A52D CONVERT IT TO TWO HEX DIGITS <A612>
A536 ADD AN "=" SIGN
A538 COPY MSB OF END OF FILE MARK ($27B)
A549 CONVERT LOW TWO BYTES OF EOF <A62F>
A550 DO CREATION DATE/TIME <A570>
A552 ---
A55B CONVERT BLOCKS USED <A62P>
A563 CHECK FOR WRITE ACCESS
A565 UNLOCKED? >>A56C
A567 NO, ADD A ""
A56C FALL THRU TO DO LAST MODIFIED DATE/TIME
A56E AND THEN EXIT TO CALLER
A576 ********** FORMAT A DATE/TIME *******************************
X = OFFSET FROM $259 TO FIELD
Y = $261 OFFSET TO DATE/TIME VALUE
A579 ISOLATE YEAR ($25A)
A57A AND STORE IT (BCB5)
A57B ISOLATE DAY
A57D AND STORE IT (BCB4)
A581 ISOLATE MONTH
A587 (MONTH = 0 IS NO GOOD) >>A5A3
A58B (MONTH > 12 IS ALSO BAD) >>A5A3
A58D STORE MONTH (BCB3)
A591 MULTIPLY MONTH INDEX BY 3 (BCB3)
A594 AND SAVE IT INSTEAD (BCB3)
A59A (DAY = 0 IS NO GOOD) >>A5A3
A5A1 (YEAR MUST BE < 99) >>A5B5
A5A3 OTHERWISE, BAD DATE1
A5A5 BACK UP 6 CHARACTERS ON LINE
A5AA AND PRINT "NO DATE" (B9E5)
A5B4 THEN EXIT RIGHT AWAY
A5B5 DATE OK, GET HOUR ($25C)
A5B9 AND MINUTES ($25B)
A5BE MINUTES > 60?
A5C0 NO... >>A5C3
A5C2 YES, USE ZERO MINUTES
A5C3 CONVERT MINUTES (LEFT ZERO FILL) <A6A0>
A5CB THEN PRINT A ";" ($201)
A5CC GET HOUR AGAIN
A5CF GREATER THAN 24 HOURS?
A5D1 NOPE >>A5D4
A5D3 YES, USE ZERO
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A5D4</td>
<td>10 OR MORE HOURS (TWO DIGITS?)</td>
</tr>
<tr>
<td>A5D7</td>
<td>IN ANY CASE, CONVERT HOURS &lt;A62F&gt;</td>
</tr>
<tr>
<td>A5DB</td>
<td>IF TWO DIGITS... &gt;A5DE</td>
</tr>
<tr>
<td>A5DD</td>
<td>IF ONE, ADJUST LINE PTR</td>
</tr>
<tr>
<td>A5DE</td>
<td>---</td>
</tr>
<tr>
<td>A5E2</td>
<td>CONVERT YEAR (LEFT ZERO FILL) &lt;A60A&gt;</td>
</tr>
<tr>
<td>A5E6</td>
<td>GET MONTH INDEX (*3) [BCB3]</td>
</tr>
<tr>
<td>A5E9</td>
<td>POINT TO LAST CHARACTER</td>
</tr>
<tr>
<td>A5EC</td>
<td>COPY MONTH NAME FROM TABLE (B9BD)</td>
</tr>
<tr>
<td>A5EF</td>
<td>TO LINE (0201)</td>
</tr>
<tr>
<td>A5F7</td>
<td>BACKWARDS... &gt;&gt;A5EC</td>
</tr>
<tr>
<td>A5F8</td>
<td>PUT A &quot;-&quot; IN (0201)</td>
</tr>
<tr>
<td>A5FE</td>
<td>TWO PLACES (0205)</td>
</tr>
<tr>
<td>A607</td>
<td>EXIT BY CONVERTING DAY &gt;&gt;A62F</td>
</tr>
</tbody>
</table>

A60A ************ CONVERT 2 DIGIT NUMBER *********************

(FORCE LEFT ZERO FILL)

A60B ---
A60D ADD 100 TO FORCE SIGNIFICANCE IN TENS
A610 IGNOR 100'S PLACE
A611 RETURN

A612 ************ CONVERT TO HEX ***********************

A613 ---
A617 ISOLATE LOW NIBBLE
A618 AND GO CONVERT IT FIRST <A61D>
A619 NOW ISOLATE HIGH NIBBLE
A61C AND FALL THRU TO CONVERT IT ALSO
A61D CONVERT NIBBLE TO NUMERIC ASCII
A61F >97
A621 NO >>A625
A623 YES, CONVERT $BA-$BF TO $C1-$C6
A625 AND STORE THE RESULT (0201)
A628 BUMP LINE INDEX BACK
A629 PRECEED WITH A '/' SIGN
A62E RETURN

A62F ************ CONVERT TO DECIMAL ***********************

A62F A,X = NUMBER Y=INDEX TO LAST FIELD DIGIT (BCB0)
A632 STORE NUMBER IN ACCUMULATOR (BCAF)
A635 DIVIDE BY 10 <A64D>
A638 GET DIGIT AND CONVERT IT (BCB2)
A63D STORE IN LINE (0201)
A640 AND DROP LINE INDEX BY ONE
A641 IS QUOTIENT NOW ZERO? (BCAF)
A64A NO, CONTINUE UNTIL IT IS >>A635

A64C ELSE, EXIT

********** DIVIDE ACCUMULATOR BY 10 **********

A64D 24 BIT SHIFT (3 BYTES)
A651 CLEAR SUM (BCB2)
A654 GO ROL ACCUMULATOR LEFT ONE BIT <A677>
A657 ALSO ROL 4TH BYTE OF ACCUM (BCB2)
A65B IF MSB > 1B... (BCB2)
A665 THEN ADD ONE TO ACCUMULATIVE SUM (BCAF)
A668 ---
A669 SHIFT 24 TIMES >>A654
A66B RETURN
A66C ---
A67C RETURN

A677 ********** SYNTAX: PARSE COMMAND LINE **********

(ALSO EXTERNAL ENTRY FOR COMMAND STRINGS)

A677 INIT COMMAND NUMBER TO -1
A678 A BLANK ENDS EACH STRING (BCA9)
A683 AT MUST 9 CHARACTERS IN A COMMAND (BCAA)
A686 PARSE COMMAND ITSELF <A1B>
A689 GET FIRST LETTER (BCBD)
A68C MUST BE ALPHABETIC
A68E IT IS... >>A697
A690 IT'S NOT, IS IT A "-"?
A692 YES, OK THEN... >>A697
A694 ELSE, ITS BAD - SYNTAX ERROR >>A639
A697 SCAN FOR COMMAND IN TABLES <AA1>
A699 BAD COMMAND >>A694
A69C NO, IMMEDIATE COMMAND MODE? (BE42)
A69F NO, DEFERRED... >>A6AC
A6A1 IMMEDIATE, EXEC ACTIVE? (BE43)
A6A4 YES, NEVER MIND >>A6AC
A6A6 ERASE TO END OF LINE <FC9C>
A6A9 AND GO TO A NEW LINE ON SCREEN <FAB>
A6AC_ASSUME NO PARMS AT ALL
A6B4 NO PATH NAME YET (BCBD)
A6B7 NO SECONDARY PATH NAME EITHER (B280)
A6BD CURRENT SLOT = DEFAULT SLOT (BE61)
A6C3 CURRENT DRIVE = DEFAULT DRIVE (BE62)
A6CB BUFFER ALLOCATION = HIMEM (BCHB)
A6CB GET LENGTH OF COMMAND NAME (BE52)
A6D0 ALLOW 2 MORE CHARACTERS FOR NOW (BCAA)
A6D3 ARE ANY PARAMETERS PERMITTED? (BE54)
A6D6 NO... MUST BE NON OR NONmö <<A736
A6DB YES, INH OR PRF?
A6D9 YES... >>A739
A6DB ELSE, REPARSE THE COMMAND <A1B>
A6DB FOR THIS COMMAND... (BE54)
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A810

--- DESCRIPTION/CONTENTS ---

A810 EXTERNAL COMMAND? IF SO GO NOW >>A836
A812 MY OWN COMMAND, "PREFIX"?
A814 YES, GO NOW >>A836
A816 S OR D VALID KEYWORDS FOR THIS CMD?
A818 NO, GO NOW >>A836
A820 PATHNAME1 GIVEN WITH THIS COMMAND?
A822 NO, GO NOW >>A836
A823 YES, GET FILE INFO FOR PATHNAME1 <B7D0>
A825 NO ERRORS I HOPE >>A836
A827 ERROR WAS PATH NOT FOUND?
A829 NO, REAL ERROR - SAY SO >>A838
A82B CAN WE CREATE PATHNAME1?
A830 YES, OK THEN >>A836
A832 ELSE, "PATH NOT FOUND"
A834 RETURN
A836 GO TO COMMAND HANDLING ROUTINE >>BCAB

A839 ********** SYNTAX ERROR **********
A83B AND RETURN WITH ERROR CONDITION
A83C RETURN

A83D ********** ADD PREFIX TO PATHNAMES **********
A844 PUT SLOT IN HIGH 3 BITS
A846 ADD DRIVE TO TOP BIT AND SHIFT SLOT DOWN (B607)
A848 ...TO FORM THE UNIT NUMBER (BEC7)
A850 READ THE PATHNAME PREFIX TO $201 (BEC8)
A852 MLI: ONLINE (B678)
A854 ERROR? >>A83B
A855 DEFAULT DRIVE = Parsed drive (B630)
A856 DEFAULT SLOTS = Parsed slot (B63C)
A858 PATHNAME1 STARTS WITH "/"?
A859 THEN ITS ALREADY GOT A PREFIX >>A866
A85B ELSE, GET LENGTH OF PATHNAME
A85C BUMP IT BY 2 (TO ALLOW FOR "/")
A85E WITH PREFIX WILL IT EXCEED 64 CHARs?
A860 YES, "SYNTAX ERROR" >>A867
A862 NO, UPDATE LENGTH TO INCLUDE PREFIX (BCBC)
A866 ---
A868 AND COPY PATHNAME1 FORWARD TO MAKE NOO (BCBD)
A86A PUT A "/" AT THE BEGINNING
A86C AND AT THE END (BCBD)
A86E COPY PREFIX JUST READ TO START OF PATHNAME1 (0208)
A870 GET COMMAND NUMBER (0253)
A872 "OPEN"?
A874 YES, DONE NOW! >>A866
A876 "APPEND"?
A878 YES, DONE NOW! >>A866

--- DESCRIPTION/CONTENTS ---

A865 "EXEC"?
A867 YES, DONE NOW! >>A886
A869 NO, GET LENGTH OF PATHNAME2 (0208)
A86B MORE THAN 64 CHARs?
A86C IF SO, "SYNTAX ERROR" >>A887
A86E COPY PATHNAME2 FORWARD TO MAKE ROOM (0281)
A88B PUT A "/" IN FIRST
A88D THEN THE PREFIX AND ANOTHER SLASH (0281)
A88F ---
A891 DONE!

A88B ********** KEYWORD LOOKUP **********
A88D ZERO THE ACCUMULATOR <A837>
A88E NINE POSSIBLE KEYWORDS IN TABLE
A88F COMPARE AGAINST EACH (B6BE)
A890 FOUND IT? >>A927
A892 NO, "I"? (FILE TYPE)
A894 YES, OK THEN >>A89C
A896 ELSE, BAD KEYWORD >>A882
A89C IT'S "I", IS IT PERMITTED ON THIS CMD?
A89E NO, ERROR >>A923
A896 ELSE, MARK WE HAVE "I" (B656)
A898 START WITH TYPE INDEX OF 0 (BCAC)
A89A INDICATE WHERE T VALUE IS TO GO (BCAC)
A89C AND GO PARSER ONE CHAR <AA5A>
A89E NOTHING THERE?? >>A89F
A89A IS IT A S?  
A89C YES, HE GAVE TYPE IN HEX >>A976
A89E IS IT ALPHABETIC?
A89E ELSE, CONVERT DECIMAL TYPE >>A960
A89F ELSE, GO LOOKUP TYPE NAME IN TABLE >>A9B6
A892 ---
A89B "INVALID PARAMETER"
A89C RETURN

A895 GET BIT POSITION OF THIS KEYWORD (B975)
A89A IGNORE "Y" >>A947
A89C IS THIS KEYWORD PERMITTED? (B655)
A89F NO, NOT WITH THIS COMMAND ANYWAY >>A935
A892 S OR D?
A893 NO >>A941
A895 YES, ALREADY FOUND IT ON THIS LINE? (B57)
A897 YES, DON'T CHANGE DRIVE DEFAULT >>A947
A89C ELSE, ASSUME DRIVE = 1
A89D MARK WE HAVE SLOT/DRIVE (B57)
A897 GET SIZE-1 IN BYTES OF VALUE (B97F)
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A954

ADDR  DESCRIPTION/CONTENTS

A954  AND OFFSET TO VALUE IN STORAGE AREA (BCAE)
A957  FLUSH TO NON-BLANK <AA3A>
A95A  NOTHING ELSE THERE? >>A9B0
A95C  IS NEXT CHAR A "$"?
A95E  YES, GO CONVERT HEX - ELSE, FALL THRU >>A976

A960  ************* CONVERT DECIMAL NUMBER **************

A960  SAVE LINE INDEX (BE48)
A963  CONVERT/ADD ONE DECIMAL DIGIT TO ACCUM <AA5C>
A966  OK... >>A96C
A968  OVERFLOW? THEN "RANGE ERROR" >>A9B3
A96A  BAD DIGIT? THEN "SYNTAX ERROR" >>A9B0
A96C  RESTORE LINE INDEX (BE48)
A96F  FLUSH TO NEXT NON-BLANK <AA3A>
A972  AND GO BACK TO CONVERT NEXT DIGIT >>A960
A974  ALL DONE, END OF LINE OR COMMA >>A98F

A976  ************* CONVERT HEX NUMBER **********************

A976  FLUSH TO NEXT NON-BLANK (SKIP "$") <AA3A>
A979  NOTHING LEFT? >>A9B0
A97B  SAVE LINE INDEX (BE48)
A97E  CONVERT HEX DIGIT <AAA>
A981  OK... >>A987
A983  OVERFLOW? THEN "RANGE ERROR" >>A9B3
A985  BAD DIGIT? THEN "SYNTAX ERROR" >>A9B0
A987  RESTORE LINE INDEX (BE48)
A98A  FLUSH TO NEXT NON-BLANK <AA3A>
A98D  AND GO CONVERT NEXT DIGIT >>A97B

A98F  ************* STORE KEYWORD VALUE **********************

A98F  HOW MANY BYTES TO CHECK?
A994  ALL HAVE BEEN CHECKED? >>A99E
A996  NO, INSURE MSB'S OF ACCUM ARE ZERO (BCAP)
A999  IF NUMBER IS A SHORT INTEGER >>A9B3
A9A1  COPY ACCUM TO PROPER PARM STORAGE CELL (BCAP)
A9A4  RESTORE LINE INDEX (BE4B)
A9AF  AND EXIT

A9B0  "SYNTAX ERROR" JUMP >>A839
A9B3  "RANGE ERROR" JUMP >>A75E

A9B6  ************* STORE KEYWORD VALUE **************

A9B6  ----
A9B8  COPY 3 CHARACTER TYPE TO ACCUM (BCAP)
A9BE  (COPYED ALL 3?) >>A9C7
A9C0  (GET NEXT CHAR IGNORING BLANKS) <AA3A>
A9C5  MUST HAVE 3 CHARACTERS! >>A9B0

BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A9C7

ADDR  DESCRIPTION/CONTENTS

A9C7  SAVE LINE INDEX (BE48)
A9C8  INITIALIZE NAME INDEX TO ZERO
A9CF  HAVE ALL 13 BEEN CHECKED?
A9D1  YES, NO MATCH >>A9B0
A9D4  ELSE, INDEX* 3 (BCAD)
A9DB  COMPARE TYPE GIVEN (BCAF)
A9DB  TO TYPES IN TABLE (B997)
A9DE  (IGNORE MSB'S)
A9DF  NO MATCH ALREADY... >>A9E9
A9E3  ELSE,
A9E5  CHECK ALL THREE CHARS >>A9D8
A9E7  THEY ALL MATCH WE FOUND IT >>A9EE
A9F9  NOT THE RIGHT ONE, (BCAU)
A9EC  GO TRY THE NEXT ONE >>A9CA
A9EE  REVERSE NAME INDEX
A9F5  AND GET TYPE VALUE FROM TABLE (B989)
A9F8  STORE IT IN TYPE VALUE STORAGE AREA (BE6A)
A9FF  RESTORE LINE INDEX (BE48)

AA00  ************* COPY PATHNAME2 **********************

AA00  GET NEXT CHARACTER <AA4A>
AA03  AND STORE IT INDEXED OFF $280 (B280)
AA07  COMMA?
AA09  YES, DONE >>AA37
AA0B  BLANK?
AA0D  YES, DONE >>AA37
AA0F  RETURN?
AA11  YES, OUT NOW >>AA48
AA13  PATHNAME TOO LONG? (BCAA)
AA16  NO, CONTINUE COPYING >>AA00
AA18  ELSE, SET NOT-EQUAL CONDITION
AA1A  AND EXIT

AA1B  ************* COPY COMMAND NAME INTO TXTBUF **********************

AA1B  SET INDICES
AA1F  GET NEXT NON-BLANK <AA4A>
AA22  COPY TO TXTBUF (BCBD)
AA26  COMMA?
AA29  YES, DONE >>AA37
AA2A  BLANK?
AA2C  YES, DONE >>AA37
AA2E  RETURN?
AA30  YES, DONE >>AA48
AA32  AT MAX LENGTH (8)? (BCAA)
AA35  NO, CONTINUE >>AA1F
AA37  ELSE, SET NOT-EQUAL CONDITION
AA39  AND EXIT
**BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84**

**ADDR**  **DESCRIPTION/CONTENTS**

**AA3A ******* FLUSH TO NON-BLANK **************************
Z-FLAG SET IF COMMA OR RETURN FOUND
C-FLAG SET IF COMMA

AA3A IGNORE BLANKS
AA3F GET NEXT NON-BLANK <AA4A>
AA42 COMMA?
AA44 YES, OUT >>AA49
AA46 RETURN?
AA4B EXIT INDICATING WHAT WE FOUND
AA49 RETURN

**AA4A ******* GET NEXT CHARACTER **************************

AA4A GET NEXT CHAR IN INPUT LINE (0200)
AA4D FORCE OFF MSB
AA4F LOWER CASE?
AA51 NO >>AA55
AA52 YES, FORCE UPPER CASE
AA55 BUMP LINE INDEX
AA56 IS THIS A FLUSH CHARACTER (LIKE BLANK)? (BCA9)
AA59 YES, GO GET NEXT ONE >>AA4A
AA5B ELSE, RETURN WITH IT

**AA5C ******* CONVERT DIGIT AND ADD TO ACCUM ***************

AA5C NUMERIC?
AA5E NO >>AA64
AA62 YES >>AA68
AA64 NOT NUMERIC, EXIT WITH CARRY SET
AA65 AND Z-FLAG RESET
AA67 RETURN
AA68 ISOLATE DECIMAL PORTION OF DIGIT
AA6B CURRENT VALUE OF ACCUM... (BCB1)
AA6E >1,703,936?
AA70 YES, OVERFLOW >>AA94
AA74 PUSH ENTIRE ACCUM ONTO STACK (BCAF)
AA77 ACCUM*2 (ROL IT ONCE) <AAD7>
AA7E ACCUM*4 (AND AGAIN) <AAD7>
AA84 ---
AA85 ACCUM*4+ACCUM -- ACCUM*5 (BCAF)
AA91 FINALLY, ACCUM*5+2 -- ACCUM*10 <AAD7>
AA94 ---
AA95 ACCUM OVERFLOW? >>AAAA
AA97 NO, ADD NEW DIGIT TO ACCUM (BCAF)
AA99 ADD STORED IT (BCAF)
AA9D NO CARRY? >>AAAD
AAAD GOT CARRY, PROPAGATE IT THRU ACCUM (BCB0)
AAA0 OVERFLOW ERROR
AAA9 NORMAL EXIT
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: ABB0

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A25</td>
<td>NOT THE ONE, SKIP TO NEXT (BE52)</td>
</tr>
<tr>
<td>A2E</td>
<td>IF THERE ARE ANY MORE &gt;&gt;A9FA</td>
</tr>
<tr>
<td>A30</td>
<td>ELSE, NO SUCH COMMAND (BE53)</td>
</tr>
<tr>
<td>A34</td>
<td>RETURN THRU $BE06 VECTOR &gt;&gt;BE06</td>
</tr>
<tr>
<td>A37</td>
<td>************ ZERO THREE BYTE ACCUM ***********************</td>
</tr>
<tr>
<td>A37</td>
<td>ZERO THE THREE BYTE WORK</td>
</tr>
<tr>
<td>A39</td>
<td>...ACCUMULATOR (SCAF)</td>
</tr>
<tr>
<td>A41</td>
<td>RETURN</td>
</tr>
<tr>
<td>A43</td>
<td>************ &quot;=&quot; COMMAND *********************************</td>
</tr>
<tr>
<td>A43</td>
<td>CHECK FILE TYPE (BE88)</td>
</tr>
<tr>
<td>A46</td>
<td>APPLESOFT PROGRAM?</td>
</tr>
<tr>
<td>A48</td>
<td>YES, &quot;RUN&quot; IT &gt;&gt;ABB2</td>
</tr>
<tr>
<td>A4A</td>
<td>BINARY FILE?</td>
</tr>
<tr>
<td>A4C</td>
<td>YES, &quot;BRUN&quot; IT &gt;&gt;ABB8</td>
</tr>
<tr>
<td>A4E</td>
<td>TEXT FILE?</td>
</tr>
<tr>
<td>A50</td>
<td>NO &gt;&gt;AB55</td>
</tr>
<tr>
<td>A52</td>
<td>YES, &quot;EXEC&quot; IT &gt;&gt;B221</td>
</tr>
<tr>
<td>A55</td>
<td>SYS FILE?</td>
</tr>
<tr>
<td>A57</td>
<td>YES, GO RUN IT &gt;&gt;AB5D</td>
</tr>
<tr>
<td>A59</td>
<td>ELSE, &quot;FILE TYPE MISMATCH&quot;</td>
</tr>
<tr>
<td>A5C</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

******** RUN "SYS" FILE *************

| A5D  | CLOSE ALL OPEN FILES <B4F2> |
| A60  | CLOSE EXEC <B2FB> |
| A65  | LSB OF A$ IS $0 (BE69) |
| A68  | FREE UP ALL OF $I$ MEMORY (BF6B) |
| A7B  | A28000 IS WHERE IT WILL LOAD (BE59) |
| A80  | TYPE IS "SYS" (BE6A) |
| A8A  | FORCE. T, PATHNAME1, AD PARMS (BE56) |
| A8D  | GO DO A STANDARD BRUN >>AE16 |
| A90  | ************ "CHAIN" COMMAND ****************************|

| A90  | SQUASH VARIABLES UP AGAINST HIMEM <A40D> |
| A95  | SAVE HIMEM (BC7B) |
| A9C  | SET NEW HIMEM BELOW COMBINED VARS |
| A9E  | LOAD FILE (LEAVE OTHERS OPEN) <AC0J> |
| A9A  | RESTORE OLD HIMEM |
| A9E  | ERROR? >>AC14 |
| AAB  | NO, CLEAR VARIABLES <D665> |
| ABB  | THEN GO "RUN" PROGRAM >>ABC7 |
| ABB0 | LOAD A PROGRAM *********************** |

ABB0   ************ "RUN" COMMAND ***********************

| ABB2 | NO INPUT FILE ACTIVE NOW |
| ABB7 | NO APPLESOFT ERROR NUMBER |
| ABB8 | GET PATHNAME1 |
| ABBD | NO, ERROR >>ABD5 |
| ABBF | YES, LOAD PROGRAM <ABFE> |
| ABC2 | ERROR? >>AC14 |
| ABC4 | NO, CLEAR VARIABLES <D665> |
| ABC7 | CLEAR ERROR FLAG |
| ABC9 | POSITION TO LINE NUMBER IF GIVEN <AC97> |
| ABCC | RESTORE MY INTERCEPTS <9A8D> |
| ABCE | CLEAR COMMAND NUMBER ETC., MODE = 4 <ABD5> |
| ABD2 | JUMP INTO APPLESOFT TO RUN PROGRAM >>D7D2 |
| ABD5 | ********** CLEAR COMMAND NUMBER ETC. *************** |

| ABD5 | SET NORMAL (NON-INVERSE OR FLASH) <F73> |
| ABD8 | SEARCH CHARACTER FOR TRACe IS "#" (9F61) |
| ABD8 | NO COMMAND NUMBER NOW (BE53) |
| ABD0 | PROMPT |
| ABD6 | SET MODE=4 (DEPRECATED) <9F76> |
| ABE9 | "SYNTAX ERROR" IF THINGS GO WRONG >>AB39 |
| ABCE | ********* "LOAD" COMMAND ************** |

| ABCE | LOAD PROGRAM <ABFE> |
| ABEF | ERROR? IF NOT, FALL THRU TO WARMSTART >>AC14 |

| ABF1 | ********* WARMSTO: WARMSTART BI ***************** |

| ABF1 | CLEAR APPLESOFT, RESET POINTERS <D665> |
| ABF4 | RESET MODE/SET INTERCEPTS <9A17> |
| ABF9 | CURSOR HORIZ. = 0 (START OF LINE) |
| ABF8 | GO WARMSTART APPLESOFT >>D43F |
| ABFE | LOAD A PROGRAM *********************** |

| ABFE | CLOSE ALL OPEN FILES <B4F2> |
| AC01 | ERROR? >>AC14 |
| AC03 | GO LOAD FILE <AC15> |
| AC06 | ERROR? >>AC14 |
| AC0B | SET LOMEM = ARRAYS = FREESTART |
| AC0A | ALL TO END OF PROGRAM LOADED |
| AC14 | RETURN |
AC15 ********** READ A PROGRAM FROM A FILE **************

AC15 READ REQUESTED
AC17 TYPE = BAS ASSUMED
AC19 OPEN THE FILE <B194>
AC1C ERROR? >>AC14
AC20 MLI: GET EOF <BE70>
AC23 ERROR? >>AC14
AC27 APPLESOFT PROGRAM START --> READ DATA (BED7)
AC2A ADD TO THAT THE EOF MARK TO ... (BEC8)
AC2D SET AD PARM --> END OF PROGRAM IMAGE (BE58)
AC38 OVERFLOW? >>AC3F
AC3D NO, WOULD PROGRAM EXCEED HIMEM?
AC3F IF SO... AC41 "PROGRAM TOO LARGE" >>AC14
AC43 ELSE, PICK UP LENGTH AGAIN (BEC8)
AC49 AND GO READ IT IN <AF98>
AC4C ERROR? >>AC14
AC4E CLOSE FILE <AF94>
AC51 ERROR? >>AC14
AC53 RELOCATE PROGRAM IF NECESSARY <AC61>
AC5C COPY AD PARM TO APPLESOFT PGM END PTR
AC60 RETURN

AC61 ********** RELOCATE APPLESOFT PROGRAM ***************

AC61 ---
AC62 WAS APPLESOFT PROGRAM SAVED FROM SAME
AC64 MEMORY LOCATION? (BE59)
AC73 YES, NOTHING TO DO THEN >>ACBA
AC79 ELSE, LOOP THROUGH PROGRAM
AC7B ADJUSTING ALL ADDRESSES TO
AC7D THE NEW LOAD LOCATION

AC97 ********** POSITION TO LINE NUMBER ***************

AC97 WAS A LINE NUMBER PARM GIVEN? (BE57)
AC9D NO, NEVER MIND >>ACBA
AC9F COPY L KEYWORD VALUE TO APPLESOFT'S LINE # (BE68)
ACA9 THEN CALL APPLESOFT TO FIND THE LINE <D61A>
ACA9F SUBTRACT ONE FROM THE ADDRESS
ACB1 AND POINT APPLESOFT'S GETCHR SUBROUTINE
ACB3 AT IT (SO NEXT CHAR READ WILL BE FIRST
ACB5 CHARACTER ON THE LINE).
ACBA RETURN

ACBB ********** "SAVE" COMMAND ***************

ACBB DOES FILE EXIST ALREADY? >>ACDF
ACBD NO, TYPE = BAS
ACBF IN T KEYWORD VALUE (BE6A)
ACCC AND MLI LIST (BE59)
ACCC ALLOW ALL ACCESSES (READ/WRIT/ETC.) (BE77)
ACCF SAVE PROGRAM START ADDRESS IN (BE58)
ACCF AUXID'S (BE59)
ACDA GO CREATE A NEW FILE <AD46>
ACDD ERROR? >>AD28

ACDF WRITE ACCESS REQUESTED
ACED BAS TYPE FILE
ACED OPEN IT <B194>
ACED ERROR? >>AD28
ACEE SUBTRACT APPLESOFT PRTS TO COMPUTE
ACED LENGTH OF PROGRAM.
ACCE STORE THIS IN EOF MARK LIST (BEC8)
ACCF MSB OF EOF MARK IS 0 (64K PGM) (BECA)
AD00 POINT LIST TO PROGRAM AS DATA TO WRITE (BED7)
AD02 WRITE A RANGE TO DISK FILE <AF9C>
AD0B ERROR? >>AD28
AD0F MLI: SET EOF (TO TRUNCATE OLD LONGER FILE) <BE78>
AD12 ERROR? >>AD28
AD14 CLOSE THE FILE <AF94>
AD17 ERROR? >>AD28
AD1B DOES PROGRAM START MATCH AUXID IN FILE INFO?
AD20 NO, CHANGE IT >>AD29
AD29 ELSE, EXIT
AD29 TO CHANGE IT, (BE59)
AD2F EXIT THRU SET FILE INFO ROUTINE >>BE79

ACD2 ********** "CREATE" COMMAND ***************

ACD2 AUXID = 0 (AS OR RECLN)
ACD3 TYPE KEYWORD GIVEN?
ACD3F YES >>AD46
ACD4 NO, ASSUME TYPE = DIR (BE6A)
AD46 *** CREATE FILE ENTRY *** (BE43)
AD49 EXEC FILE ACTIVE?
AD4C HOW MANY FILES ARE OPEN INCLUDING EXEC? (BE4D)
AD4F 8 OR MORE7
AD51 YES, ERROR >>AD60
AD56 ELSE, SET TYPE IN MLI LIST (BEA4)
AD5B FULL ACCESS (READ/WRIT/ETC.)
AD5D KIND = STANDARD FILE
AD5D DIR FILE WANTED?
BASIC Interpreter (BI) -- V1.1  --8 JUN 84  NEXT OBJECT ADDR: AD55

ADDR DESCRIPTION/CONTENTS

AD55  NO >>AD63
AD61  YES, KIND = DIR FILE
AD63  SET ACCESS (BEA1)
AD66  AND KIND (BEA7)
AD6B  ML1: CREATS (DON'T COME BACK HERE) >>BE70

AD6E  "RAM TOO LARGE" ERROR
AD70  RETURN

AD71  ********** "RENAME" COMMAND  **********************

AD71  ---
AD75  SECOND PATHNAME GIVEN?
AD78  IF SO, GO ML1: RENAME >>AD7F
AD7A  "SYNTAX ERROR" OTHERWISE >>AD99

AD7D  ********** "DELETE" COMMAND  **********************

AD7D  SETUP ML1: DELETE CALL TYPE
AD7F  EXIT THRU ML1 CALL >>BE70

AD82  ********** "LOCK" COMMAND  **********************

AD82  GET FILE INFO FOR PATHNAME1 <B7D0>
AD85  GET ACCESS CODES (BE97)
AD88  TURN OFF ALL...
AD93  BUT READ
AD8F  THEN GO SET UPDATED FILE INFO >>B7E7

AD92  ********** "UNLOCK" COMMAND  **********************

AD92  GET FILE INFO FOR PATHNAME1 <B7D0>
AD95  TURN ON ALL FILE ACCESES
AD9D  THEN GO SET UPDATED FILE INFO >>B7E7

ADA0  ********** "PREFIX" COMMAND  **********************

ADA0  SLOT/DRIVE GIVEN ON COMMAND? (BE97)
ADA6  IF SO, GOT OPERAND ALREADY >>ADAC
ADA8  ELSE, (BE96)
ADA8  CHECK FOR PATHNAME1
ADAC  AND GO TO ML1: GET PREFIX ...
ADAE  IF IT'S THERE >>AD7F
ADB0  ELSE, IS BASIC PROGRAM RUNNING?
AD82  IF SO, SET PREFIX ACTIVE FLAG >>ADD1
ADB4  NO, NEW LINE <9FA8>
ADBC  END OF NAME YET? >>ADC9
ADB4  NO, COPY NAME IN PATHNAME1 BUFFER (BCBD)
ADCB  TO OUTPUT DEVICE <9FA8>
ADCD  AND SKIP A BLANK LINE <9FA8>
ADDE  DONE

BASIC Interpreter (BI) -- V1.1  --8 JUN 84  NEXT OBJECT ADDR: ADD0

ADDR DESCRIPTION/CONTENTS

ADD1  SET PREFIX ACTIVE FLAG
ADD3  SO BASIC CAN READ THE PREFIX (BE46)
ADD7  RETURN

ADD8  ********** "SAVE" COMMAND  **********************

ADD8  PATHNAME1 FOUND? >>AEB0
ADD9  NO, NEW FILE (BE97)
ADD0  AD, L, AND E POSSIBLE
ADDF  AD AND EITHER L OR E REQUIRED
ADE1  OR ELSE ERROR >>AE12
ADE6  PUT AD IN CREATE PARAMETER LIST (BEA5)
ADE9  AND IN GET FILE INFO LIST (BE99)
ADFE  TYPE = BIN ASSUED (BE6A)
AE00  T KEYWORD GIVEN?
AE02  IF SO, ERROR >>AE12
AE04  GO CREATE THE FILE <AD46>
AE07  ERROR? >>AE14
AE09  GET FILE INFO <B7D0>
AE0C  ERROR? >>AE14
AE0E  WRITING...
AE10  GO PROCESS LIKE A BLOAD OTHERWISE >>AE25
AE12  "PATH NOT FOUND" ERROR
AE14  ---
AE15  RETURN

AE16  ********** "RUN" COMMAND  **********************

(DOES NOT SET MODE=4 SO DOS COMMANDS MAY NOT BE ISSUED AS WITH A BASIC PROGRAM)

AE16  BLOAD IT FIRST <AE23>
AE19  ERROR? >>AE14
AE1B  THEN CALL IT <AE20>
AE1E  THEN EXIT
AE1F  RETURN
AE20  INDIRECT JMP TO BINARY PROGRAM >>BED7

AE23  ********** "BLOAD" COMMAND  **********************

AE23  READING...
AE25  TYPE = BIN
AE27  OPEN THE FILE <B194>
AE2A  ERROR? >>AE14
AE2C  ASSEMBLE USER SPECIFIED AD KEYWORD (BE59)
AE35  IF SO, USE HIS ADDRESS >>AE47
AE37  ELSE, USE AD IN FILE INFO AUXID (BE99)
AE40  WAS T KEYWORD GIVEN?
AE42  YES, INVALID PATH (ONLY BIN IS LEGAL) >>AE78
AE47  POINT READ/WRITE RAM TO DATA (BED7)
<table>
<thead>
<tr>
<th>BASIC Interpreter (B1) -- V1.1 --18 JUN 84</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>Description/Contents</td>
</tr>
<tr>
<td>AE4D</td>
<td>PICK UP LENGTH FROM L KEYWORD VALUE (BE5F)</td>
</tr>
<tr>
<td>AE53</td>
<td>WAS L OR E GIVEN?</td>
</tr>
<tr>
<td>AE55</td>
<td>NEITHER &gt;&gt;AE7C</td>
</tr>
<tr>
<td>AE57</td>
<td>BOTH?</td>
</tr>
<tr>
<td>AE59</td>
<td>YES...NAUGHTY! &gt;&gt;AE78</td>
</tr>
<tr>
<td>AE5B</td>
<td>E GIVEN?</td>
</tr>
<tr>
<td>AE5D</td>
<td>NO, MUST BE L &gt;&gt;AE92</td>
</tr>
<tr>
<td>AE5F</td>
<td>YES... (BE5D)</td>
</tr>
<tr>
<td>AE63</td>
<td>COMPUTE L = (E - AD) (BE5B)</td>
</tr>
<tr>
<td>AE6F</td>
<td>PLUS ONE FOR INCLUSIVE RANGE &gt;&gt;AE72</td>
</tr>
<tr>
<td>AE72</td>
<td>MAKE SURE NO BORROW OCCURRED &gt;&gt;AE92</td>
</tr>
<tr>
<td>AE74</td>
<td>OR ELSE, &quot;RANGE ERROR&quot;</td>
</tr>
<tr>
<td>AE77</td>
<td>RETURN</td>
</tr>
<tr>
<td>AE78</td>
<td>&quot;INVALID FARM&quot; ERROR</td>
</tr>
<tr>
<td>AE79</td>
<td>RETURN</td>
</tr>
<tr>
<td>AE7C</td>
<td>---</td>
</tr>
<tr>
<td>AE7E</td>
<td>MLI: GET EOF &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE81</td>
<td>ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE83</td>
<td>GET L (EOF MARK) (BECA)</td>
</tr>
<tr>
<td>AE89</td>
<td>BETTER NOT EXCEED 64K (BECA)</td>
</tr>
<tr>
<td>AE8C</td>
<td>NO.. &gt;&gt;AE92</td>
</tr>
<tr>
<td>AE8E</td>
<td>YES, &quot;PROGRAM TOO LARGE&quot;</td>
</tr>
<tr>
<td>AE90</td>
<td>---</td>
</tr>
<tr>
<td>AE91</td>
<td>RETURN</td>
</tr>
<tr>
<td>AE92</td>
<td>STORE LENGTH TO READ OR WRITE (BE99)</td>
</tr>
<tr>
<td>AE9B</td>
<td>B KEYWORD GIVEN?</td>
</tr>
<tr>
<td>AE9D</td>
<td>NO &gt;&gt;AE99</td>
</tr>
<tr>
<td>AE9A</td>
<td>YES, COPY B VALUE TO SET MARK LIST (BE5A)</td>
</tr>
<tr>
<td>AE9C</td>
<td>---</td>
</tr>
<tr>
<td>AE9A</td>
<td>MLI: SET MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9B</td>
<td>NO ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE94</td>
<td>ERROR, RANGE ERROR?</td>
</tr>
<tr>
<td>AE96</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE98</td>
<td>BSAVING (NOT BLOAD/BRUNING)?</td>
</tr>
<tr>
<td>AE9A</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE9B</td>
<td>MLI: FORCE EOF FORWARD TO MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9C</td>
<td>AND TRY SET MARK AGAIN &gt;&gt;AE25</td>
</tr>
<tr>
<td>AE92</td>
<td>RETURN</td>
</tr>
<tr>
<td>AECC</td>
<td>GET COMMAND NUMBER (BE53)</td>
</tr>
<tr>
<td>AECD</td>
<td>ASSUME READ</td>
</tr>
<tr>
<td>AECA</td>
<td>BSAVE?</td>
</tr>
<tr>
<td>AECD</td>
<td>NO, READ IS CORRECT &gt;&gt;AE7F</td>
</tr>
<tr>
<td>AECE</td>
<td>WRITING</td>
</tr>
<tr>
<td>AECC</td>
<td>MLI: READ OR WRITE &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE92</td>
<td>ERROR? &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE4D</td>
<td>THEN EXIT THRU CLOSE &gt;&gt;AF94</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>BASIC Interpreter (B1) -- V1.1 --18 JUN 84</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>Description/Contents</td>
</tr>
<tr>
<td>AE7A</td>
<td>&quot;STORE&quot; COMMAND</td>
</tr>
<tr>
<td>AE77</td>
<td>---</td>
</tr>
<tr>
<td>AE78</td>
<td>---</td>
</tr>
<tr>
<td>AE79</td>
<td>---</td>
</tr>
<tr>
<td>AE7C</td>
<td>---</td>
</tr>
<tr>
<td>AE7E</td>
<td>MLI: GET EOF &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE81</td>
<td>ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE83</td>
<td>GET L (EOF MARK) (BECA)</td>
</tr>
<tr>
<td>AE89</td>
<td>BETTER NOT EXCEED 64K (BECA)</td>
</tr>
<tr>
<td>AE8C</td>
<td>NO.. &gt;&gt;AE92</td>
</tr>
<tr>
<td>AE8E</td>
<td>YES, &quot;PROGRAM TOO LARGE&quot;</td>
</tr>
<tr>
<td>AE90</td>
<td>---</td>
</tr>
<tr>
<td>AE91</td>
<td>RETURN</td>
</tr>
<tr>
<td>AE92</td>
<td>STORE LENGTH TO READ OR WRITE (BE99)</td>
</tr>
<tr>
<td>AE9B</td>
<td>B KEYWORD GIVEN?</td>
</tr>
<tr>
<td>AE9D</td>
<td>NO &gt;&gt;AE99</td>
</tr>
<tr>
<td>AE9A</td>
<td>YES, COPY B VALUE TO SET MARK LIST (BE5A)</td>
</tr>
<tr>
<td>AE9C</td>
<td>---</td>
</tr>
<tr>
<td>AE9A</td>
<td>MLI: SET MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9B</td>
<td>NO ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE94</td>
<td>ERROR, RANGE ERROR?</td>
</tr>
<tr>
<td>AE96</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE98</td>
<td>BSAVING (NOT BLOAD/BRUNING)?</td>
</tr>
<tr>
<td>AE9A</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE9B</td>
<td>MLI: FORCE EOF FORWARD TO MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9C</td>
<td>AND TRY SET MARK AGAIN &gt;&gt;AE25</td>
</tr>
<tr>
<td>AE92</td>
<td>RETURN</td>
</tr>
<tr>
<td>AECC</td>
<td>GET COMMAND NUMBER (BE53)</td>
</tr>
<tr>
<td>AECD</td>
<td>ASSUME READ</td>
</tr>
<tr>
<td>AECA</td>
<td>BSAVE?</td>
</tr>
<tr>
<td>AECD</td>
<td>NO, READ IS CORRECT &gt;&gt;AE7F</td>
</tr>
<tr>
<td>AECE</td>
<td>WRITING</td>
</tr>
<tr>
<td>AECC</td>
<td>MLI: READ OR WRITE &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE92</td>
<td>ERROR? &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE4D</td>
<td>THEN EXIT THRU CLOSE &gt;&gt;AF94</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>BASIC Interpreter (B1) -- V1.1 --18 JUN 84</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>Description/Contents</td>
</tr>
<tr>
<td>AE7A</td>
<td>&quot;STORE&quot; COMMAND</td>
</tr>
<tr>
<td>AE77</td>
<td>---</td>
</tr>
<tr>
<td>AE78</td>
<td>---</td>
</tr>
<tr>
<td>AE79</td>
<td>---</td>
</tr>
<tr>
<td>AE7C</td>
<td>---</td>
</tr>
<tr>
<td>AE7E</td>
<td>MLI: GET EOF &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE81</td>
<td>ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE83</td>
<td>GET L (EOF MARK) (BECA)</td>
</tr>
<tr>
<td>AE89</td>
<td>BETTER NOT EXCEED 64K (BECA)</td>
</tr>
<tr>
<td>AE8C</td>
<td>NO.. &gt;&gt;AE92</td>
</tr>
<tr>
<td>AE8E</td>
<td>YES, &quot;PROGRAM TOO LARGE&quot;</td>
</tr>
<tr>
<td>AE90</td>
<td>---</td>
</tr>
<tr>
<td>AE91</td>
<td>RETURN</td>
</tr>
<tr>
<td>AE92</td>
<td>STORE LENGTH TO READ OR WRITE (BE99)</td>
</tr>
<tr>
<td>AE9B</td>
<td>B KEYWORD GIVEN?</td>
</tr>
<tr>
<td>AE9D</td>
<td>NO &gt;&gt;AE99</td>
</tr>
<tr>
<td>AE9A</td>
<td>YES, COPY B VALUE TO SET MARK LIST (BE5A)</td>
</tr>
<tr>
<td>AE9C</td>
<td>---</td>
</tr>
<tr>
<td>AE9A</td>
<td>MLI: SET MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9B</td>
<td>NO ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE94</td>
<td>ERROR, RANGE ERROR?</td>
</tr>
<tr>
<td>AE96</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE98</td>
<td>BSAVING (NOT BLOAD/BRUNING)?</td>
</tr>
<tr>
<td>AE9A</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE9B</td>
<td>MLI: FORCE EOF FORWARD TO MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9C</td>
<td>AND TRY SET MARK AGAIN &gt;&gt;AE25</td>
</tr>
<tr>
<td>AE92</td>
<td>RETURN</td>
</tr>
<tr>
<td>AECC</td>
<td>GET COMMAND NUMBER (BE53)</td>
</tr>
<tr>
<td>AECD</td>
<td>ASSUME READ</td>
</tr>
<tr>
<td>AECA</td>
<td>BSAVE?</td>
</tr>
<tr>
<td>AECD</td>
<td>NO, READ IS CORRECT &gt;&gt;AE7F</td>
</tr>
<tr>
<td>AECE</td>
<td>WRITING</td>
</tr>
<tr>
<td>AECC</td>
<td>MLI: READ OR WRITE &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE92</td>
<td>ERROR? &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE4D</td>
<td>THEN EXIT THRU CLOSE &gt;&gt;AF94</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>BASIC Interpreter (B1) -- V1.1 --18 JUN 84</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>Description/Contents</td>
</tr>
<tr>
<td>AE7A</td>
<td>&quot;STORE&quot; COMMAND</td>
</tr>
<tr>
<td>AE77</td>
<td>---</td>
</tr>
<tr>
<td>AE78</td>
<td>---</td>
</tr>
<tr>
<td>AE79</td>
<td>---</td>
</tr>
<tr>
<td>AE7C</td>
<td>---</td>
</tr>
<tr>
<td>AE7E</td>
<td>MLI: GET EOF &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE81</td>
<td>ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE83</td>
<td>GET L (EOF MARK) (BECA)</td>
</tr>
<tr>
<td>AE89</td>
<td>BETTER NOT EXCEED 64K (BECA)</td>
</tr>
<tr>
<td>AE8C</td>
<td>NO.. &gt;&gt;AE92</td>
</tr>
<tr>
<td>AE8E</td>
<td>YES, &quot;PROGRAM TOO LARGE&quot;</td>
</tr>
<tr>
<td>AE90</td>
<td>---</td>
</tr>
<tr>
<td>AE91</td>
<td>RETURN</td>
</tr>
<tr>
<td>AE92</td>
<td>STORE LENGTH TO READ OR WRITE (BE99)</td>
</tr>
<tr>
<td>AE9B</td>
<td>B KEYWORD GIVEN?</td>
</tr>
<tr>
<td>AE9D</td>
<td>NO &gt;&gt;AE99</td>
</tr>
<tr>
<td>AE9A</td>
<td>YES, COPY B VALUE TO SET MARK LIST (BE5A)</td>
</tr>
<tr>
<td>AE9C</td>
<td>---</td>
</tr>
<tr>
<td>AE9A</td>
<td>MLI: SET MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9B</td>
<td>NO ERROR? &gt;&gt;AE94</td>
</tr>
<tr>
<td>AE94</td>
<td>ERROR, RANGE ERROR?</td>
</tr>
<tr>
<td>AE96</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE98</td>
<td>BSAVING (NOT BLOAD/BRUNING)?</td>
</tr>
<tr>
<td>AE9A</td>
<td>NO &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE9B</td>
<td>MLI: FORCE EOF FORWARD TO MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE9C</td>
<td>AND TRY SET MARK AGAIN &gt;&gt;AE25</td>
</tr>
<tr>
<td>AE92</td>
<td>RETURN</td>
</tr>
<tr>
<td>AECC</td>
<td>GET COMMAND NUMBER (BE53)</td>
</tr>
<tr>
<td>AECD</td>
<td>ASSUME READ</td>
</tr>
<tr>
<td>AECA</td>
<td>BSAVE?</td>
</tr>
<tr>
<td>AECD</td>
<td>NO, READ IS CORRECT &gt;&gt;AE7F</td>
</tr>
<tr>
<td>AECE</td>
<td>WRITING</td>
</tr>
<tr>
<td>AECC</td>
<td>MLI: READ OR WRITE &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE92</td>
<td>ERROR? &gt;&gt;AE90</td>
</tr>
<tr>
<td>AE4D</td>
<td>THEN EXIT THRU CLOSE &gt;&gt;AF94</td>
</tr>
</tbody>
</table>
BASIC Interpreter (BI) -- V1.1 --18 JUN 84  NEXT OBJECT ADDR: AF5D

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF5D</td>
<td>FROM AUXID (BC8E)</td>
</tr>
<tr>
<td>AF63</td>
<td>ADJUST MSB OF THIS BY THE DIFFERENCE</td>
</tr>
<tr>
<td>AF66</td>
<td>BETWEEN HIMEM'S (NOW AND WHEN STORED) (BCBD)</td>
</tr>
<tr>
<td>AF73</td>
<td>MAKE SURE VARS WON'T OVERLAY PROGRAM</td>
</tr>
<tr>
<td>AF75</td>
<td>IF SO, ERROR &gt;&gt;AF90</td>
</tr>
<tr>
<td>AF76</td>
<td>COMPUTE LENGTH OF ALL VARS/STRINGS</td>
</tr>
<tr>
<td>AF80</td>
<td>(HIMEM-START) (BCBF)</td>
</tr>
<tr>
<td>AF85</td>
<td>GO READ COMBINED VARS INTO MEMORY &lt;AF90&gt;</td>
</tr>
<tr>
<td>AF88</td>
<td>ERROR? &gt;&gt;AF39</td>
</tr>
<tr>
<td>AF8A</td>
<td>CLOSE THE FILE &lt;AF94&gt;</td>
</tr>
<tr>
<td>AF8D</td>
<td>EXIT BY REEXPANDING THE VARS DOWN &gt;&gt;AF32</td>
</tr>
<tr>
<td>AF90</td>
<td>&quot;PROGRAM TOO LARGE&quot; ERROR</td>
</tr>
<tr>
<td>AF93</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

APF9 ********** CLOSE FILE ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF94</td>
<td>SET MLI CLOSE OPCODE</td>
</tr>
<tr>
<td>AF96</td>
<td>AND GO TO MLI &gt;&gt;AF9A</td>
</tr>
</tbody>
</table>

APF98 ********** READ/WRITE A RANGE *******************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF98</td>
<td>READ MLI OPCODE</td>
</tr>
<tr>
<td>AF9A</td>
<td>JUMP IN &gt;&gt;AF9E</td>
</tr>
<tr>
<td>AF9C</td>
<td>WRITE MLI OPCODE</td>
</tr>
<tr>
<td>AF9E</td>
<td>STORE LENGTH (BE0A)</td>
</tr>
<tr>
<td>AFA4</td>
<td>EXIT THRU MLI:READ OR WRITE &gt;&gt;BE70</td>
</tr>
</tbody>
</table>

APF7 ********** "PR#" COMMAND ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFA7</td>
<td>USE CSWL AND OUTVEC</td>
</tr>
<tr>
<td>AFAC</td>
<td>JUMP TO COMMON CODE &gt;&gt;AF85</td>
</tr>
</tbody>
</table>

APAE ********** "IN#" COMMAND *********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFAE</td>
<td>USE KSWL</td>
</tr>
<tr>
<td>AF83</td>
<td>AND INVEC</td>
</tr>
</tbody>
</table>

AP85 | OR IN SLOT GIVEN BY USER (BE6A) |
| AF88 | *2 FOR USE AS INDEX INTO TABLE |
| AF8D | WAS SLOT PARAMETER GIVEN? |
| AF8F | NO... >>AFD2 |
| AFC1 | YES, (BE57) |
| AFC4 | AD GIVEN? >>AFE7 |
| AFC6 | NO, GET INVEC OR OUTVEC FOR THIS SLOT (BE10) |
| AFC9 | AND STORE ON AD KWADCARD VALUE (BE56) |
| AF02 | VALIDITY CHECK I/O DRIVER <AF9> |
| AF05 | NO GOOD? >>AF6 |
| AF07 | GET INDEX TO CSWL OR KSWL (BCA9) |
| AF0D | AND REPLACE ONE OR THE OTHER WITH <B036> |
| AF88 | HIS ADDRESS (BE59) |

APF6 RETURN

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF7</td>
<td>VALIDITY CHECK AD KWADCARD VALUE &lt;AF9&gt;</td>
</tr>
<tr>
<td>AF8A</td>
<td>NO GOOD? &gt;&gt;AF8</td>
</tr>
<tr>
<td>AF8C</td>
<td>GOOD, COPY VALUE TO INVEC OR OUTVEC (BE59)</td>
</tr>
<tr>
<td>AF8F</td>
<td>EXIT BUT DON'T REDIRECT I/O NOW</td>
</tr>
</tbody>
</table>

APF9 ********** VALIDITY CHECK I/O DRIVER ********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFA</td>
<td>$3A/3B -- NEW HANDLER (FROM AD PARM) (BE58)</td>
</tr>
<tr>
<td>AFB5</td>
<td>IS DRIVER IN MAIN RAM (BE00)?</td>
</tr>
<tr>
<td>AFO7</td>
<td>YES &gt;&gt;B01E</td>
</tr>
<tr>
<td>AFO9</td>
<td>NO, RESET I/O CARD ROMS (CFPF)</td>
</tr>
<tr>
<td>AFOC</td>
<td>USE $3C TO COUNT ITERATIONS</td>
</tr>
<tr>
<td>AFOE</td>
<td>TEST ROM AT USER'S ADDRESS</td>
</tr>
<tr>
<td>B014</td>
<td>FOR STABILITY</td>
</tr>
<tr>
<td>B018</td>
<td>256 TIMES</td>
</tr>
<tr>
<td>B01C</td>
<td>MUST BE OK</td>
</tr>
<tr>
<td>B01D</td>
<td>RETURN</td>
</tr>
<tr>
<td>B01E</td>
<td>MAIN RAM I/O DRIVER</td>
</tr>
<tr>
<td>B020</td>
<td>MUST START WITH A &quot;CLD&quot; INSTRUCTION</td>
</tr>
<tr>
<td>B022</td>
<td>OK... &gt;&gt;B01C</td>
</tr>
<tr>
<td>B024</td>
<td>ELSE, &quot;NO DEVICE CONNECTED&quot;</td>
</tr>
<tr>
<td>B027</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

B028 ********** "BYE" COMMAND ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B02B</td>
<td>CLOSE ANY OPEN FILES &lt;B4F2&gt;</td>
</tr>
<tr>
<td>B02B</td>
<td>CLOSE EXEC FILE, IF ANY &lt;B2FB&gt;</td>
</tr>
<tr>
<td>B036</td>
<td>MLI CALL: &lt;BF80&gt;</td>
</tr>
<tr>
<td>B033</td>
<td>QUIT</td>
</tr>
<tr>
<td>B034</td>
<td>USE READ PARMLIST BECAUSE QUIT DOESN'T NEED PARMS.</td>
</tr>
</tbody>
</table>

B036 ********** "CAT" COMMAND ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B036</td>
<td>39 CHARACTERS PER LINE</td>
</tr>
<tr>
<td>B038</td>
<td>THEN PROCESS LIKE &quot;CATALOG&quot; &gt;&gt;B03C</td>
</tr>
</tbody>
</table>

B03A ********** "CATALOG" COMMAND ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B03A</td>
<td>79 CHARACTERS PER LINE</td>
</tr>
<tr>
<td>B03C</td>
<td>STORE LINE LENGTH (BCB6)</td>
</tr>
<tr>
<td>B042</td>
<td>TEST FOR T AND</td>
</tr>
<tr>
<td>B044</td>
<td>...PATHNAME1 GIVEN</td>
</tr>
<tr>
<td>B045</td>
<td>GOT T &gt;&gt;B04A</td>
</tr>
<tr>
<td>B047</td>
<td>K T, T (ANY TYPE WILL DO) (BE6A)</td>
</tr>
<tr>
<td>B04A</td>
<td>GOT PATHNAME1 &gt;&gt;B051</td>
</tr>
<tr>
<td>B04C</td>
<td>NO PATHNAME1, GET FILE INFO FOR PREFIX &lt;B7D0&gt;</td>
</tr>
<tr>
<td>B04F</td>
<td>ERROR? &gt;&gt;B057</td>
</tr>
<tr>
<td>B051</td>
<td>OPEN/READ DIRECTORY HEADER &lt;B14A&gt;</td>
</tr>
</tbody>
</table>
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84       NEXT OBJECT ADDR: B054

ADDR    DESCRIPTION/CONTENTS

B054    ERROR? >>B057
B056    SKIP TO A NEW LINE <9FAB>
B059    FORMAT DIRECTORY'S NAME TO $201 <B0B8>
B05C    PRINT $201 <9F9D>
B05F    SKIP TO A NEW LINE <9FAB>
B062    BLANK $201 BUFFER <A66C>
B067    UNPACK HEADING MESSAGE LINE <9FB8>
B06A    PRINT IT (40 OR 80 COLUMNS) <9F9D>
B06D    SKIP TO A NEW LINE <9FAB>
B073    ANY FILES IN THIS DIRECTORY? (BCBA)
B076    NO >>B0A3
B079    YES, READ NEXT ENTRY <B1D1>
B07B    ERROR? >>B0B7
B07D    GET TYPE REQUESTED FOR SEARCH (B06A)
B080    ANY TYPE WILL DO? >>B087
B082    NO, CHECK TYPE AGAINST THIS ENTRY (B029)
B085    NOT IT, SKIP IT >>B00D
B087    ELSE, FORMAT ENTRY TO $201 <A4C4>
B08A    AND PRINT $201 <9F9D>
B08D    CHECK KEYBOARD (C498)
B090    FOR A CONTROL-C
B092    IGNORE ANYTHING ELSE >>B09E
B094    CONTROL-C, WHAT STATE ARE WE IN? (BE42)
B097    DEFERRED >>B0A3
B099    NO, IMMEDIATE, RESET KEYBOARD STROBE (C010)
B09C    AND EXIT RIGHT NOW >>B0A3
B09E    ELSE, ANY FILES LEFT IN COUNT? (BCBA)
B0A1    YES, CONTINUE >>B078
B0A3    ELSE, CLOSE DIRECTORY <AF94>
B0A6    ERROR? >>B0B7
B0A8    SKIP TO A NEW LINE <9FAB>
B0AB    FORMAT BLOCKS FREE AND IN USE TO $201 <BOE7>
B0AE    ERROR? >>B0B7
B0B0    PRINT $201 <9F9D>
B0B3    SKIP A LINE <9FAB>
B0B7    DONE

B0B8    FORMAT NAME OF DIRECTORY

B0B8    BLANK $201 BUFFER <A66C>
B0B8    FILE NAME IS AT +1 INTO DIR ENTRY
B0BD    GET NAME LENGTH/TYPE (025D)
B0C2    VOLUME DIRECTORY HEADER?
B0C4    NO >>B0AC
B0C6    YES, START NAME WITH "/" (0200)
B0CA    ---
B0CB    ISOLATE NAME LENGTH FROM TYPE
B0CD    AND SET UP LENGTH TO COPY (B200)
B0D2    COPY DIRECTORY NAME TO (0259)

B0E1    SET $200 TO MAXIMUM LENGTH
B0E6    RETURN

B0E7    FORMAT BLOCKS FREE/INUSE

B0E7    POINT ML1:ONLINE PARMLIST
B0E9    TO TXTBAF (PATNAME=) (BECB)
B0F1    COPY DEVICE NUMBER (UNIT) (BF30)
B0F9    ML1: ONLINE <BE70>
B0FC    ERROR? >>B0B7
B101    ISOLATE NAME LENGTH FROM BUFFER
B104    BUMP BY ONE TO INCLUDE "/
B105    AND STORE IT AS A PREFIX (BCBC)
B10A    STORE "/" AS FIRST CHARACTER (BCBD)
B10D    GET FILE INFO FOR PREFIX <B7D9>
B110    ERROR? >>B0B7
B112    BLANK $201 BUFFER <A66C>
B117    UNPACK "BLOCKS FREE: BLOCKS USED..." <9FB8>
B11A    COPY INTO THREE BYTE ACCUM <AB37>
B125    CONVERT ACCID (TOTAL BLOCKS) <A62F>
B130    CONVERT BLOCKS USED <A62F>
B137    BLOCKS_FREE = TOTAL BLOCKS (BEBC)
B13E    ... - BLOCKS USED (BEBD)
B145    CONVERT BLOCKS FREE <A62F>
B149    DONE!

B14A    OPEN/READ DIRECTORY HDR

B14A    READ ONLY
B14E    CHECK FILE KIND (BEBB)
B151    DIRECT DIRECTORY?
B153    NO >>B158
B155    YES, TYPE = DIR (BEBB)
B15B    OPEN THE FILE <B1A0>
B15B    ERROR? IF NOT, FALL THRU >>B193

B15D    READ DIRECTORY HDR

B15D    BUFFER IS $259
B169    LENGTH IS $2B (ONE ENTRY) (BEB9)
B173    ML1: READ <BE70>
B176    ERROR? >>B193
B17A    COPY ENTRY LENGTH, ENTRIES PER BLOCK, (027C)
B17D    AND FILE COUNT FROM DIR HDR (BC7)
B183    STORE ENTRY LENGTH IN READ LENGTH NOW (BEB9)
B188    SET COUNTER TO FIRST ENTRY IN BLOCK (BCBB)
B18D    MARK $8 (START OF FILE) (BEC9)
B193    RETURN
B194 ********** OPEN FILE ********************
A REGISTER = ACCESS BITS
X REGISTER = DEFAULT TYPE

B194 ---
B190 T KEYWORD GIVEN?
B19A NO >>B19F
B19C YES, USE KEYWORD VALUE INSTEAD (BEEE)
B19F ---
B1AF EXISTING FILE OF THIS TYPE? (BE88)
B1A3 NO, ERROR >>B1C9
B1A5 CHECK ACCESS REQUESTED (BEB7)
B1A8 REQUESTED ACCESS NOT PERMITTED >>B1CD
B1AA SET SYSTEM BUFFER IN OPEN PARM LIST (BC88)
B1B2 LEVEL = $0F (BF94)
B1B7 MLI: OPEN <BE70>
B1BA ERROR? >>B1CB
B1BF SAVE REFPNUM IN READ/WRITE PARMLIST (BEB6)
B1C2 AND CLOSE PARMLIST (BEDE)
B1C5 AND GET/SET EOF/MARK LIST (BEC7)
B1C8 AND EXIT

B1C9 "FILE TYPE MISMATCH"
B1CC RETURN
B1CD "FILE LOCKED"
B1DD RETURN

B1D1 ********** READ NEXT DIRECTORY ENTRY **************

B1D1 FORC MARK TO START OF THIS BLOCK (BEC9)
B1D9 CHECK ENTRY NUMBER (BCB8)
B1DE LAST ENTRY IN THIS BLOCK? (BCB8)
B1E1 NO >>B1ED
B1E4 YES, ENTRY 0 NEXT TIME (BCBB)
B1E7 BUMP MARK TO NEXT BLOCK (BEC9)
B1ED ---
B1EF MARK POSITIONED TO PROPER ENTRY YET? >>B1F8
B1F1 NO, BUMP POINTER TO NEXT ENTRY (BCB7)
B1F4 AND CONTINUE IF STILL FIRST PAGE >>B1ED
B1F6 JUST ENTERED SECOND PAGE >>B1BA
B1F8 ADD 4 TO PTR TO ADJUST FOR BLOCK PREFIX
B1FF MLI: SET MARK <BE70>
B202 ERROR? >>B21D
B206 MLI: READ <BE70>
B209 ERROR? >>B21D
B20B BUMP ENTRY COUNTER (BCBB)
B211 IS THIS ENTRY VALID?
B213 NO, SKIP OVER IT >>B1D1
B215 DECREMENT FILE COUNT (BCB9)
B21D AND RETURN TO CALLER
**BASIC Interpreter (Bl) --- V1.1 --- 18 JUN 84 **
**NEXT OBJECT ADDR: B278**

**ADDR**

**DESCRIPTION/CONTENTS**

********** MAKE EXEC TOPMOST BUFFER **********

B27A
OTHERS ARE OPEN...
B27C
OPENCOUNT*4 (4 PAGES PER BUFFER)
B27E
ADD THIS TO MY BUFFER TO FIND TOP BUFFER (BC88)
B282
SEARCH OPEN FILES TO FIND THE FILE WHICH (BC93)
B285
IS USING THIS BUFFER... >>B28B
B28A
IF IT IS NOT FOUND, BREAK!
B28B
---
B28C
MOVE THAT FILE TO THE NEW BUFFER INSTEAD (BC93)
B28F
GET THAT FILE'S REFNUM ALSO (BC9B)
B297
MLI: SET BUFF <BE78>
B29A
NO ERRORS? >>B29D
B29C
IF ERROR, BREAK!
B29D
---

********** OPEN NEW EXEC FILE **********

B29E
SET NEW BUFFER ALLOCATION PAGE (BC88)
B2A1
SET UP OPEN LIST FOR EXEC TOO (BECF)
B2A6
LEVEL = 0 (BF94)
B2AB
MLI: OPEN (EXEC FILE) <BE78>
B2AE
NO ERROR? >>B2B7
B2B0
---
B2B1
IF ERROR, FREE BUFFER FIRST <A24C>
B2B6
THEN EXIT WITH ERROR
B2B7
SAVE BUFSIZE FOR EXEC (BECF)
B2BD
AND REFNUM TOO (BED0)

********** COMPLETE EXEC COMMAND **********

B2CJ
SAVE READ REFNUM (BED6)
B2C6
AND GET/SET REFNUM (BEC7)
B2C9
AND NEWLINE REFNUM (BED2)
B2CF
SET "L" VALUE FROM AUXID (BE5F)
B2D8
SAVE PATHNAME/AUXID IN OPEN FILE TABLE <B3EB>
B2DD
IGNORE MSH FOR END OF LINE CHARS (BED3)
B2E2
MLI: SET NEWLINE <BE78>
B2E8
WAS "P" OR "R" GIVEN ON COMMAND LINE?
B2EA
NO >>B2F4
B2EC
YES, POSITION TO SPECIFIED STARTING PT <B522>
B2EF
NO ERRORS? >>B2F4
B2F1
IF ERROR, GO CLOSE EXEC >>B245
B2F4
MARK EXEC ACTIVE
B2FA
AND RETURN TO CALLER

**BASIC Interpreter (Bl) --- V1.1 --- 18 JUN 84 **
**NEXT OBJECT ADDR: B2FA**

**ADDR**

**DESCRIPTION/CONTENTS**

********** CLOSE EXEC FILE **********

B2FB
EXEC ACT? (BE43)
B2FE
NO, SKIP IT >>B30B
B300
INDICATE EXEC FILE CLOSING (BE48)
B305
PICK UP REFNUM FOR EXEC (BC9B)
B308
AND GO CLOSE IT <B4A5>
B30B
RETURN

B30C
********** "VERIFY" COMMAND **********

B30C
FILE NOT FOUND? >>B347
B311
FILE FOUND, WAS A PATHNAME GIVEN?
B313
YES >>B31D
B315
NO,
B317
PRINT "(C) APPLE COMPUTER..." <9FSC
B31A
AND A NEW LINE <9FAB>
B31D
THEN EXIT
B31E
RETURN

B31F
********** FLUSH ALL OPEN FILES **********

B31F
REFNUM = 0 (ALL FILES)
B321
JUMP INTO FLUSH >>B32F

B323
********** "FLUSH" COMMAND **********

B323
---
B326
WAS PATHNAME GIVEN?
B328
NO, FLUSH ALL FILES >>B32F
B32A
ELSE, LOOK UP NAME IN OPEN FILE LISTS <B41F>
B32D
NOT AN OPEN FILE >>B37?
B32F
SAVE REFNUM IN FARM LIST (BEDE)
B334
MLI: FLUSH <BE78>
B337
EXIT

B338
********** "OPEN" COMMAND **********

B338
---
B339
LOOK UP NAME IN OPEN FILE LIST <B41F>
B33C
NOT CURRENTLY OPEN? >>B34B
B33E
---
B33F
IT IS OPEN, "FILE BUSY" ERROR
B342
RETURN
BASIC Interpreter (B1) -- V1.1 --18 JUN 84

ADDR   DESCRIPTION/CONTENTS
------- -------------------------------
B343   "FILE TYPE MISMATCH" ERROR
B346   RETURN
B347   "PATH NOT FOUND" ERROR
B349   ---
B34A   RETURN
B34B   ---
B34C   ASSUME "L" IS ZERO
B353   WAS "L" KEYWORD GIVEN?
B355   YES, USE HIS VALUE >>B35D
B357   NO, SET "L" TO ZERO (BE68)
B363   WAS "T" GIVEN?
B364   YES, USE HIS TYPE >>B36B
B365   ELSE, DEFAULT TO "TXT"
B366   DOES THE FILE ALREADY EXIST? >>B3BE
B36D   NO, "T" GIVEN? IF NO, ERROR >>B347
B36F   FORCE TYPE = "TXT" (BE68)
B374   FULL ACCESS (BE67)
B37A   COPY "L" KEYWORD VALUE (BE5F)
B37D   TO CREATE (BEA6)
B380   AND SET FILE INFO LISTS (BEBA)
B389   GO CREATE THE FILE <AD46>
B38C   ERROR? >>B349
B38E   CHECK FILE TYPE (BE8B)
B391   AGAINST HIS "T" VALUE (BEA6)
B394   MISMATCH? >>B343
B396   NO, TYPE = TXT?
B398   NO >>B3AD
B39A   YES, GET RECORD LENGTH FROM AUID (BEBA)
B3AD   WA S"L" KEYWORD VALUE GIVEN?
B3AF   YES, USE THAT INSTEAD >>B3AD
B3A7   OTHERWISE, SAVE AUID RECORD LEN (BE69)
B3A8   ALLOCATE A NEW FILE BUFFER <A1F5>
B3B0   ERROR? >>B349
B3B2   GET BUFFER PAGE NO. (BC98)
B3B5   AND STORE IN OPEN LIST (BCF2)
B3BA   LEVEL = 7 (BF94)
B3BF   MLI: OPEN <BE78>
B3C2   NO ERRORS? >>BJCB
B3C4   ---
B3C5   ERROR, FREE BUFFER FIRST <<A24C
B3CA   THEN EXIT WITH ERROR CODE
B3CB   CHECK FILE TYPE AGAIN (BE8B)
B3Cf   "DIR" FILE?
B3D0   YES >>B3D3
B3D2   NO
B3D3   ---

B3D6   SET DIR FLAG ACCORDINGLY (BE47)
B3D9   USING OPEN COUNT AS AN INDEX (BE4D)
B3DF   STORE BUFFER LOCATION IN OPEN FILE LIST (BC94)
B3F7   ALSO, THE REPNUM (BC9C)
B3F8   AND BUMP OPEN FILE COUNT AND FALL THRU (BE4D)
B3EB   ************ SAVE FILE NAME/RECLEN IN TABLE *************
B3EB   MAKE INDEX FROM REPNUM*32 BYTES
B3F1   GET NAME LENGTH (B2B0)
B3F4   OR IN DIR FLAG (BE47)
B3F7   AND STORE IN OPEN FILE NAME LIST (BCFE)
B3FD   NAME > OR = TO 30 BYTES?
B3FF   NO... >>B403
B401   YES, USE 29
B403   STORE THAT AS A LOOP COUNTER
B408   COPY "L" KEYWORD VALUE TO NAME LIST TOO (BCFF)
B411   ---
B412   COPY FILE NAME TO NAME LIST (B2B0)
B415   COPY ALL OF NAME, THEN FALL THRU TO EXIT >>B411
B41D   ************ "MON" AND "NOMON" COMMANDS ***************
B41D   IGNORE THESE COMMANDS AND
B41E   RETURN TO CALLER
B41F   ********** LOOKUP OPEN FILENAME *******************
B41F   (RETURNS REPNUM OF OPEN FILE)
B41F   ---
B422   WAS PATHNAME1 GIVEN?
B424   YES >>B42A
B426   NO, "SYNTAX ERROR"
B429   EXIT WITH ERROR
B42A   ANY FILES CURRENTLY OPEN? (BE4D)
B42D   NO, CAN'T FIND IT THEN >>B448
B42F   YES, CLEAR EXEC FILE CLOSING FLAG (BE4E)
B432   STORE FILE COUNT AS LOOP COUNTER
B43D   GET NEXT REPNUM (BC9B)
B437   COMPARE FILENAMES <B462>
B44A   NOT THE ONE? >>B44J
B44C   ELSE, WE'VE GOT IT!
B44E   PICK UP APPROPRIATE REPNUM (BC9B)
B441   ---
B442   AND RETURN WITH IT
B443   ELSE, NOT IT, TRY NEXT ONE
B446   AND CONTINUE LOoping >>B4J2
<table>
<thead>
<tr>
<th>BASIC Interpreter (B1) -- V1.1  --18 JUN 84</th>
<th>NEXT OBJECT ADDR: B446</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>B446</td>
<td>CAN'T FIND IT, IS EXEC ACTIVE? (BE43)</td>
</tr>
<tr>
<td>B448</td>
<td>NO, THEN WE MUST GIVE UP &gt;&gt;B45E</td>
</tr>
<tr>
<td>B450</td>
<td>IS HE LOOKING FOR EXEC FILE? &lt;B462&gt;</td>
</tr>
<tr>
<td>B452</td>
<td>NO, GIVE UP &gt;&gt;B45E</td>
</tr>
<tr>
<td>B457</td>
<td>YES, EXEC FILE CLOSING (BE4E)</td>
</tr>
<tr>
<td>B45C</td>
<td>AND RETURN WITH EXEC'S REFNUM &gt;&gt;B43E</td>
</tr>
<tr>
<td>B45E</td>
<td>&quot;FILE NOT OPEN&quot; ERROR</td>
</tr>
<tr>
<td>B461</td>
<td>RETURN WITH ERROR CODE</td>
</tr>
<tr>
<td>B462</td>
<td>***************** COMPAR FILENAMES **********************</td>
</tr>
<tr>
<td>B462</td>
<td>REFNUM*32 FOR FILENAME INDEX</td>
</tr>
<tr>
<td>B468</td>
<td>PICK UP DIR FLAG FROM THIS ENTRY (BCFE)</td>
</tr>
<tr>
<td>B470</td>
<td>SAME LENGTH AS HIS FILENAME? (B280)</td>
</tr>
<tr>
<td>B473</td>
<td>NO, CAN'T BE IT THEN &gt;&gt;B49B</td>
</tr>
<tr>
<td>B476</td>
<td>MAKE SURE LENGTH DOES NOT EXCEED 29</td>
</tr>
<tr>
<td>B47A</td>
<td>IF IT DOES, ONLY LOOK AT FIRST 29</td>
</tr>
<tr>
<td>B47C</td>
<td>USE 3A AS LOOP COUNTER</td>
</tr>
<tr>
<td>B481</td>
<td>COPY &quot;L&quot; OF THIS FILE TO KEYWORD (BCA4)</td>
</tr>
<tr>
<td>B48A</td>
<td>---</td>
</tr>
<tr>
<td>B48B</td>
<td>COMPARE NAMES (B280)</td>
</tr>
<tr>
<td>B491</td>
<td>NO MATCH? EXIT WITH Z FLAG CLEAR &gt;&gt;B498</td>
</tr>
<tr>
<td>B498</td>
<td>MATCH, EXIT WITH Z FLAG SET</td>
</tr>
<tr>
<td>B499</td>
<td>********** &quot;CLOSE&quot; COMMAND *******************</td>
</tr>
<tr>
<td>B499</td>
<td>---</td>
</tr>
<tr>
<td>B49C</td>
<td>PATHNAME1 GIVEN?</td>
</tr>
<tr>
<td>B49E</td>
<td>NO, CLOSE ALL FILES &gt;&gt;B4F2</td>
</tr>
<tr>
<td>B4A0</td>
<td>YES, LOOK IT UP IN OPEN FILE TABLES &lt;B41F&gt;</td>
</tr>
<tr>
<td>B4A3</td>
<td>NOT FOUND? &gt;&gt;B441</td>
</tr>
<tr>
<td>B4AA</td>
<td>FOUND IT, STORE REFNUM IN CLOSE LIST (BEDE)</td>
</tr>
<tr>
<td>B4AB</td>
<td>MARK BUFFER PAGE FREE (BC88)</td>
</tr>
<tr>
<td>B4AE</td>
<td>EXEC CLOSING? (BE4E)</td>
</tr>
<tr>
<td>B4B1</td>
<td>YES...NO NEED TO COMPRESS LISTS &gt;&gt;B4CF</td>
</tr>
<tr>
<td>B4B3</td>
<td>GET OPEN COUNT (LAST OPENED FILE NO.) (BE4D)</td>
</tr>
<tr>
<td>B4B7</td>
<td>SWAP BUFFERS (BC93)</td>
</tr>
<tr>
<td>B4C5</td>
<td>AND REFNUMS WITH THE LAST OPENED FILE (BC9B)</td>
</tr>
<tr>
<td>B4CF</td>
<td>---</td>
</tr>
<tr>
<td>B4D1</td>
<td>LEVEL = 0 (BE94)</td>
</tr>
<tr>
<td>B4DF</td>
<td>MLI: CLOSE &lt;BE78&gt;</td>
</tr>
<tr>
<td>B4D9</td>
<td>ERROR? &gt;&gt;B502</td>
</tr>
<tr>
<td>B4DB</td>
<td>RELEASE THE BUFFER &lt;A24C&gt;</td>
</tr>
<tr>
<td>B4DE</td>
<td>EXEC FILE CLOSING? (BE4E)</td>
</tr>
<tr>
<td>B4E1</td>
<td>NO &gt;&gt;B4EE</td>
</tr>
<tr>
<td>B4E6</td>
<td>YES, EXEC NO LONGER ACTIVE (BE43)</td>
</tr>
<tr>
<td>B4E9</td>
<td>AND NO LONGER CLOSING (BE4E)</td>
</tr>
<tr>
<td>B4ED</td>
<td>RETURN TO CALLER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BASIC Interpreter (B1) -- V1.1  --18 JUN 84</th>
<th>NEXT OBJECT ADDR: B4ED</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>B4EE</td>
<td>DROP OPEN FILE COUNT (BE4D)</td>
</tr>
<tr>
<td>B4F1</td>
<td>AND EXIT</td>
</tr>
<tr>
<td>B4F2</td>
<td>****************** CLOSE ALL OPEN FILES ***************</td>
</tr>
<tr>
<td>B4F2</td>
<td>---</td>
</tr>
<tr>
<td>B4F5</td>
<td>ANY FILES OPEN? (BE4D)</td>
</tr>
<tr>
<td>B4F7</td>
<td>YES, EXEC NOT CLOSING (BE4E)</td>
</tr>
<tr>
<td>B4FD</td>
<td>CLOSE LAST FILE OPENED &lt;B4A5&gt;</td>
</tr>
<tr>
<td>B500</td>
<td>IF THAT WORKS, START ALL OVER AGAIN &gt;&gt;B4F2</td>
</tr>
<tr>
<td>B502</td>
<td>EXIT WHEN ALL ARE CLOSED</td>
</tr>
<tr>
<td>B503</td>
<td>---</td>
</tr>
<tr>
<td>B505</td>
<td>SET CLOSE REFNUM TO ZERO (ALL FILES) (BEDE)</td>
</tr>
<tr>
<td>B50A</td>
<td>LEVEL = 7 (LEVEL 0 FILES ALREADY CLOSED) (BF94)</td>
</tr>
<tr>
<td>B50F</td>
<td>EXIT THRU MLI: CLOSE &gt;&gt;B70</td>
</tr>
<tr>
<td>B512</td>
<td>********** &quot;POSITION&quot; COMMAND ***********************</td>
</tr>
<tr>
<td>B512</td>
<td>LOOKUP NAME OF FILE &lt;B41F&gt;</td>
</tr>
<tr>
<td>B515</td>
<td>NOT OPEN? &gt;&gt;B57F</td>
</tr>
<tr>
<td>B517</td>
<td>SET REFNUM IN READ/WRITE PARMLIST (BE69)</td>
</tr>
<tr>
<td>B51A</td>
<td>AND SET NEWLINES LIST (BED2)</td>
</tr>
<tr>
<td>B51D</td>
<td>DIR FILE? (BE47)</td>
</tr>
<tr>
<td>B520</td>
<td>YES, GET OUT RIGHT NOW! &gt;&gt;B580</td>
</tr>
<tr>
<td>B522</td>
<td>&quot;F&quot; OR &quot;R&quot; GIVEN? (BE57)</td>
</tr>
<tr>
<td>B527</td>
<td>NO, INVALID PARM &gt;&gt;B57D</td>
</tr>
<tr>
<td>B529</td>
<td>BOTH GIVEN?</td>
</tr>
<tr>
<td>B52B</td>
<td>YES, INVALID PARM &gt;&gt;B57D</td>
</tr>
<tr>
<td>B52D</td>
<td>JUST &quot;R&quot; GIVEN?</td>
</tr>
<tr>
<td>B52F</td>
<td>NO, JUST &quot;F&quot; &gt;&gt;B53D</td>
</tr>
<tr>
<td>B531</td>
<td>JUST &quot;R&quot;, COPY &quot;R&quot; VALUE TO &quot;F&quot; (BE65)</td>
</tr>
<tr>
<td>B534</td>
<td>&quot;R&quot; AND &quot;F&quot; ARE ALIASES (BE63)</td>
</tr>
<tr>
<td>B53D</td>
<td>SET COUNT TO 239. (MAXIMUM LINE LEN)</td>
</tr>
<tr>
<td>B54C</td>
<td>BUFFER IS AT $200 (BED9)</td>
</tr>
<tr>
<td>B54F</td>
<td>---</td>
</tr>
<tr>
<td>B551</td>
<td>NEW LINE CHAR IS EITHER $BD OR $BD (BED3)</td>
</tr>
<tr>
<td>B556</td>
<td>MLI: SET NEWLINE &lt;BE70&gt;</td>
</tr>
<tr>
<td>B559</td>
<td>ERROR? &gt;&gt;B57F</td>
</tr>
<tr>
<td>B55B</td>
<td>---</td>
</tr>
<tr>
<td>B55E</td>
<td>&quot;F&quot; = u? (BE64)</td>
</tr>
<tr>
<td>B562</td>
<td>YES, DONE &gt;&gt;B50d</td>
</tr>
<tr>
<td>B564</td>
<td>ELSE...</td>
</tr>
<tr>
<td>B566</td>
<td>MLI: READ NEXT FIELD (LINE) &lt;BE70&gt;</td>
</tr>
<tr>
<td>B569</td>
<td>ERROR? &gt;&gt;B57F</td>
</tr>
<tr>
<td>B56E</td>
<td>DECREMENT &quot;F&quot; VALUE BY ONE</td>
</tr>
</tbody>
</table>
BASICK Interpreter (BI) -- V1.1 -- 18 JUN 84

BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84

B57B AND GO CHECK IT AGAIN >> B55B

B57D "INVALID PARAMETER" ERROR
B57F ---
B580 EXIT TO CALLER

B581 ************ COMPUTE NEW FILE POSITION ***********************
(COMPUTES ABSOLUTE FILE POSITION MARK)

B581 ACCUM = CURRENT RECORD LENGTH (BCA4)
B595 MARK = 0 (BEC8)

******** MARK = "R" * RECLEN ************

B59E SHIFT "R" VALUE RIGHT (BEB6)
B5A6 IF LOW BIT OFF, NO ADD >> B5AF
B5A9 ADD ONE INSTANCE OF RECLEN TO MARK (BCAF)
B5B0 OVERFLOW? >> B5D2
B5BD ACCUM OVERFLOW? >> B5D2
B5BF SCALE ACCUM (MULTIPLER) UP BY 2 (BCAF)
B5C3 IF "R" NON ZERO... (BEX3)
B5CE CONTINUE LOOPING >> B59E
B5D1 ELSE, EXIT TO CALLER

B5D2 "RANGE ERROR"
B5D5 RETURN

B5D6 ************ "READ" COMMAND **********************

B5D6 LOOK UP FILE NAME <B41F>
B5D9 NOT OPEN? >> B62B
B5DB ITS OPEN, STORE REFNUM IN READ/WRITE... (BED6)
B5DE GET/SET... (BEC7)
B5E1 AND SET NEWLINE PARMLISTS (BED2)
B5E4 DIR FILE? (BE47)
B5E7 YES, SPECIAL HANDLING REQUIRED >> B62C
B5E9 NO, PRE-POSITION FOR "R", "F", OR "R" <B666>
B5EC ERROR POSITIONING? >> B62B
B5EE ASSUME "L" = 239.
B5F5 "L" GIVEN?
B5F7 NO >> B64C
B5F9 YES, USE HIS "L" VALUE (B5EF)
B5FF UNLESS ITS > 256 >> B661
B603 OR > 239. >> B601
B607 DOUBLE QOUTE IT SO COMMAS COME THRU (B20U)
B60A READ INTO 2201
B66C IF NO "L", READ TO $200 (BED7)
B612 NL CHARS = $00/$8D (OR NONE IF "L") (BED3)
B621 ML:1 SET NEWLINE <B70>
B624 ERROR? >> B62B
B626 ---

B628 MARK INPUT "READ" FILE ACTIVE (BE44)
B62B AND RETURN

*********** READ DIR FILE ***********

B62C SET/READ/WRITE LIST REFNUM (BED6)
B62F AND GET/SET LIST REFNUM (BEC7)
B634 READING TO $299 (BED7)
B63E INIT CAT FLAG TO FIRST LINE VALUE (BE4F)
B644 "R" GIVEN?
B647 NO, DONE >> B626
B64B YES, ZERO OUT MARK (BEC8)
B656 ML:1 REWIND FILE <B70>
B659 ERROR? >> B660
B65D MARK INPUT FILE ACTIVE (BE44)
B660 AND EXIT

B661 *********** "RANGE ERROR" ***********

B661 "RANGE ERROR" CODE
B665 EXIT TO CALLER

B666 ************ PRE-POSITION FOR I/O ********************

B666 ---
B669 "B", "F", OR "R" GIVEN?
B66B NO, EXIT >> B6AF
B66D "R"
B66F NO >> B67B
B671 YES, COMPUTE ABSOLUTE POSITION <B581>
B674 ERROR? >> B661
B676 NO, SET MARK TO NEW POSITION <B6A8>
B679 ERROR? >> B660
B67B "F" GIVEN? (B57)
B680 NO >> B667
B682 SKIP LINES UNTIL "F" = 0 <B53D>
B685 ERROR? >> B660
B687 "B" GIVEN? (B57)
B68C NO >> B6AF
B690 ML:1 GET MARK <B70>
B693 ERROR? >> B660
B699 ADD "L" VALUE TO CURRENT MARK (BE5A)
B69C (3 BYTE ADD) (BEC8)
B6A6 OVERFLOW? >> B661
B6A8 ML:1 SET MARK <B70>
B6AA ERROR? >> B680
B6AF ---
B6B0 ---
B6B2 EXIT TO CALLER
**B6B3**  "WRITE" COMMAND

- B6B3  LOOKUP OPEN FILE NAME <B41F>
- B6B6  NOT AN OPEN FILE? >>B6C8
- B6B8  STORE READ/WRITE RENUM (BED6)
- B6B9  AND GET/SET RENUM (BEC7)
- B6BE  AND NEWLINE RENUM IN PARM LISTS (BED2)
- B6C1  DIR FILE? (BE47)
- B6C4  NO, OK >>B6CA

- B6C6  YES, "FILE LOCKED" ERROR
- B6C8  ---
- B6C9  EXIT TO CALLER

- B6CA  DATA BUFFER AT $200
- B6D4  PRE-POSITION FOR "R", "F", AND "R" <B666>
- B6D7  NO ERRORS? >>B6ED
- B6D9  WAS ERROR A RANGE ERROR?
- B6DB  NO, REAL ERROR >>B6C8
- B6DD  YES, MY RANGE ERROR OR MLI'S?
- B6DF  MINE... >>B6C8
- B6E1  MLI'S... SET EOF FARTHER INTO FILE
- B6E3  MLI: SET EOF <BE70>
- B6E5  ERROR? >>B6C8
- B6EB  AND THEN TRY AGAIN TO SET MARK <B676>
- B6EB  ERROR? THEN I GIVE UP >>B6C8
- B6ED  BUFFER IS AT HIMEM
- B6F9  INDICATE OUTPUT "WRITE" FILE ACTIVE (BE45)
- B6FD  RETURN TO CALLER

**B6FE**  "APPEND" COMMAND

- B6FE  ---
- B6FF  LOOK UP NAME IN OPEN FILE LIST <B41F>
- B702  FOUND IT? >>B710
- B705  NO, OPEN IT FIRST <B338>
- B70B  ERROR? >>B71E
- B70A  NO, RENUM NON-ZERO? (BED0)
- B70D  YES, OK >>B711
- B70F  ELSE, BREAK111
- B710  ---
- B711  RENUM TO READ/WRITE PARM LIST (BED6)
- B714  AND GET/SET LIST (BEC7)
- B717  DIR FILE? (BE47)
- B71A  NO >>B720

**B71C**  "FILE LOCKED"
- B71E  ---
- B71F  EXIT TO CALLER

- B720  PICK UP "L" VALUE (BE5F)
- B729  DID USER SPECIFY ONE?
- B72B  YES... >>B733
- B72D  NO, USE FILE'S CURRENT "L" VALUE (BE89)
- B733  ---
- B73B  COMPUTE RENUM*32 FOR INDEX INTO
- B739  FILE NAME TABLE
- B73E  SAVE CURRENT "L" VALUE IN OPEN FILE (BCFF)
- B741  NAME TABLE AND IN CURRENT RECLLEN (BCA4)
- B74D  MLI: GET EOF <BE70>
- B750  ERROR? >>B71E
- B752  IS "L" VALUE < 27 (NO SPECIFIC "L") (BCA5)
- B755  NO >>B75E
- B75C  YES >>B763
- B75E  NO, FORCE TO RECORD BOUNDARY <B766>
- B761  ERROR? >>B71E
- B763  ELSE, GO SET EOF=MARK/OUTPUT FILE ACTIVE >>B6E1

**B766**  "FORCE TO EVEN RECORD BOUNDARY"

(FIND RECORD NUMBER OF THIS POSITION)

- B766  ---
- B768  COPY EOF TO ACCUM (BEC7)
- B771  CLEAR MSB'S (8CB2)
- B777  GET READY FOR A 24 BIT DIVIDE
- B779  DIVIDE EOF BY... <AAD7>
- B786  RECORD LENGTH (BCA4)
- B79B  ---
- B7A1  WAS THERE A REMAINDER? (BC83)
- B7A5  NO, OK... >>B7CF
- B7AB  YES, CURRENT RECORD LEN LESS REMAINDER (BCB2)
- B7BB  PLUS OLD EOF MARK (BEC8)
- B7C2  GIVES NEW EOF ON AN EVEN RECORD BOUNDARY (BEC9)
- B7CD  "RANGE ERROR" POSSIBLE IF OVERFLOW OCCURS
- B7CF  RETURN TO CALLER

**B7D0**  "GET FILE INFO"

- B7D0  SET NUMBER OF PARMS (10)
- B7D5  MLI CODE FOR GET FILE INFO
- B7D7  GO DO IT >>B7EE
B7D9 ********** SET FILE INFO *************************************************

B7D9 MODIFIED TIME/DATE = @
B7E7 SET NUMBER OF PARM'S (7)
B7EC MLI CODE FOR SET FILE INFO
B7EE EXIT THRU MLI: GET/SET FILE INFO >>BE7B

B7F1 ********** BI I/O INDIRECTION VECTORS ***************

B7F1 DOSOUT VECTOR >>BE38
B7F4 DOSIN VECTOR >>BE3A

B7F7 ********** STATE I/O VECTORS TABLE ***************

B7F7 IMMEDIATE MODE (STATE=8) CSWL/CSWL
B7FB DEFERRED MODE (STATE=4) CSWL/CSWL
B7FF (STATE=8) CSWL/CSWL
B803 (STATE=C) CSWL

B805 ********** SYSCTRL **************

LSB'S OF MLI CALL PARAMETER LISTS IN THE BI GLOBAL PAGE ($BEXX)

B805 CREATE: $A0 DESTROY: $AC RENAME: $AF
B806 SP#: $B4 GFI: $B4 ONLINE: $CB
B80B SPFX: $AC GPFX: $AC OPEN: $CB
B80E NEWLINE: $D1 READ: $D5 WRITE: $D5
B811 CLOSE: $D3 FLUSH: $D3 MARK: $D6
B814 GMARK: $C6 SEOF: $C6 GEOF: $C6
B817 SBUF: $C6 GSBUF: $C6

B819 ********** APPLESOF'TOKENS ***********************

TOKENS REQUIRING SPECIAL ATTENTION HAVE THEIR MSB OFF AND ARE AN OFFSET FROM A JMP IN THE TRACE HANDLER IN THE BI

B819 FIRST IS $80 (END)
B823 CALL
B833 TRACE, NOTRACE, NORMAL
B837 INVERSE, FLASH
B83F RESUME
B844 LET, IF
B853 PRINT, LIST

B859 ********** COMMAND NAME TABLES ***************

OFFSETS TO LAST CHARACTER OF EACH COMMAND NAME IN THE COMMAND NAME TABLE BELOW. COMMANDS ARE ARRANGED ACCORDING TO LENGTH WITH THREE BYTE NAMES FIRST. IF THE MSB OF AN INDEX IS ON, THEN THIS IS THE LAST
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: B91F

B91F UNLOCK
B921 VERIFY
B923 CATALOG
B925 *RESTORE
B927 POSITION
B929 "-" COMMAND

B92B ************ PERMITTED KEYWORDS FOR CMDS ***************

Two bytes per command in the order above.
Each entry has 16 bit settings for the
parameters permitted on that command.

B930F = FSTK PREFIX, PATHNAME OPTIONAL
B9310 = SLOT (FOR PR# OR IN#)
B9318 = DEFERRED COMMAND ONLY
B9319 = FSTK IS OPTIONAL
B931F = IF FILE NOT FOUND, CREATE IT
B9320 = "T" (FILE TYPE) PERMITTED
B9321 = PATHNAME2 (RENAMe) PERMITTED
B9322 = PATHNAME1 EXPECTED
B9323 = "A" (ADDRESS) PERMITTED
B9324 = "B" (BYTE) PERMITTED
B9325 = "E" (END ADDRESS) PERMITTED
B9326 = "L" (LENGTH) PERMITTED
B9327 = "R" (LINE NO.) PERMITTED
B9328 = "S" (AND/OR "D" (SLOT/DRIVE)
B9329 = "Z" (FILE) PERMITTED
B932A = "F" (RECORD) PERMITTED
B932B = "Z" (RECORD) PERMITTED

('V' IS IGNORED)

B932C

C  P S D F N T P A B E L  S  F  R
O  F E L N E  A  A  I  D
M  O  F  C  W  T  T  I  D
M  T  E  P  S  H  H
N  I  R  T  I  E  I
D  I  I  I  L  I  I

B92B IN#  X  X  .
B92D FR#  X  X  X  X
B92F CAT  X  X  X  X
B931 FRE  .
B933 BYE  .
B935 RUN  .
B937 BRUN  .
B939 EXEC  .
B93B LOAD  .
B93D LOCK  .
B93F OPEN  .
B941 READ  .
B943 SAVE  .
B945 BLOAD  .
B947 BSAVE  .

B93E50

B93F A: 2 BYTES AT +1
B93F B: 3 BYTES AT +4
B93E C: 2 BYTES AT +6
B93E D: 2 BYTES AT +8
B93E L: 1 BYTE AT +9
B93E S: 1 BYTE AT +A
B93E T: 2 BYTES AT +C
B93E R: 2 BYTES AT +E
B93E V: 1 BYTE AT +10 (IGNORED)
B93E Q: 2 BYTES AT +11

B93E59

B999 ************ FILE TYPES TABLES ***************

FILE TYPE CODES, GIVEN IN INVERSE ORDER
TO FILE TYPE NAMES WHICH FOLLOW.
BASIC Interpreter (BI) -- V1.1 --18 JUN 84  NEXT OBJECT ADDR: B989

ADDR  DESCRIPTION/CONTENTS

B989  $FF = "SYS"
B98A  $FE = "REL"
B98B  $FD = "VAR"
B98C  $FC = "BAS"
B98D  $FB = "JVR"
B98E  $FA = "INT"
B98F  $F0 = "CMD"
B990  $F8 = "DIR"
B991  $E6 = "BIN"
B992  $E4 = "TXT"
B993  $E2 = "PSG"
B994  $1A = "AFP"
B995  $18 = "ASP"
B996  $16 = "ADB"
B997  ""

B997 'ADBASPAWPASTXTBINDIRCMDINTIVRBSVARRELSYS'

B9C1  *********** MONTH TABLE **********************************************

B9C1  'JANFEBMARAPRMAYJUNJULAUGSEPONOVDEC'
B9E5  'NO DATE'

B9EE  *********** MLIERTBL ***********************************************
MLI ERROR CODES WHICH HAVE BI EQUIVALENTS

B9EE  ""

BA01  *********** BIERTBL ***********************************************
BI EQUIVALENTS TO MLI ERROR CODES ABOVE
IF MLI CODE NOT FOUND, MAPS TO LAST CODE
IN THIS TABLE, $08 "I/O ERROR"

BA01  ""

BA15  *********** INDEXES TO PACKED MESSAGES ****************************
BY BI ERROR NUMBER

BA15  ""

BA29  *********** COMMON LETTERS IN MESSAGES ****************************

BA29  ""

BA29 'ACDEFMNORTU '

BASIC Interpreter (BI) -- V1.1 --18 JUN 84  NEXT OBJECT ADDR: BA38

ADDR  DESCRIPTION/CONTENTS

BA38  *********** LESS COMMON LETTERS ****************************************

BA38  ""
BA39  'BGHKPSWXY/().:,'

BA48  *********** PACKED MESSAGES ****************************************

BA48  "COPYRIGHT APPLE COMPUTER"

BA58  "NAME"
BA59  TAB($10)
BA5D  "TYPE BLOCKS"
BA65  TAB($1E)
BA68  "MODIFIED"
BA6C  TAB($2F)
BA6E  "CREATED"
BA72  TAB($40)
BA74  "ENDFILE SUBTYPE"
BA76  "BLOCKS FREE:"
BA66  TAB($16)
BA88  "BLOCKS USED:"
BA92  TAB($2C)
BA93  "TOTAL BLOCKS:"

BA92  ""

BA9C  "RANGE ERROR" ERROR=$2
BA95  "NO DEVICE CONNECTED" ERROR=$3
BA9E  "WRITE PROTECTED" ERROR=$4
BA87  "END OF DATA" ERROR=$5
BA8D  "PATH NOT FOUND" ERROR=$6,$7

BA66  "I/O ERROR" ERROR=$8
BA6C  "DISK FULL" ERROR=$9
BA6D  "FILE LOCKED" ERROR=$A
BA69  "INVALID PARAMETER" ERROR=$B
BA63  "RAM TOO LARGE" ERROR=$C
BA6F  "FILE TYPE MISMATCH" ERROR=$D
BASIC Interpreter (BI) -- V1.1 -- 18 Jun 84

BAPC  "PROGRAM TOO LARGE"  ERROR=$E
BB07  "NOT DIRECT COMMAND"  ERROR=$F
BB11  "SYNTAX ERROR"  ERROR=$10
BB19  "DIRECTORY FULL"  ERROR=$11
BB21  "FILE NOT OPEN"  ERROR=$12
BB29  "DUPLICATE FILE NAME"  ERROR=$13
BB34  "FILE BUSY"  ERROR=$14
BB3B  "FILE(S) STILL OPEN"  ERROR=$15

BB47  ********** VARIABLES **************

BB47  NUMBER OF PAGES TO ALLOCATE/FREE
BB48  NOT USED
BB49  TOP OF BUFFERS FOR GARBAGE COLLECTION
BB4A  BOTTOM OF BUFFERS

BB4B  ********** $BB4B-$BC7A NOT USED **************

BB4B  NOT USED

BC7B  ********** VARIABLES **************

BC7B  SAVED HIMEM VALUE DURING CHAIN LOAD

***** GARbage COLlect MARKED GC: ****
BC7C  GC: HIRANGE - WORKAREASIZE
BC7D  GC: WORKAREA MSB
BC7E  GC: NUMBER OF PAGES IN WORKAREA
BC7F  GC: LORANGE (START OF STRINGS TO COPY)
BC80  GC: HIRANGE (END OF STRINGS TO COPY)
BC81  ARRAYS START LSB
BC82  ARRAYS ENDING MSB+1
BC83  GC: START OF STRING AREA (ALSO PGM START)
BC85  GC: END OF STRING AREA
BC87  MSB ADJUST FACTOR FOR STRING POINTERS
BC88  PAGE FOLLOWING BLOCK BUFFER

***** STORED VARIABLES FILE HEADER ***
BC89  COMBINED LEN OF SIMPLE/ARRAY VARS
BC8B  LEN OF SIMPLE VARS ONLY
BC9D  HIMEM WHEN VARS WERE COMBINED

BC9E  POINTER TO COMBINED VARIABLES/STRINGS
BC9F  LENGTH OF COMBINED VARIABLES/STRINGS
### BASIC INTERPRETER GLOBAL PAGE

This page of memory is rigidly defined by the ProDOS BI. Fields given here will not move in later versions of ProDOS and may be referenced by external, user-written programs. Future additions to the global page may be made in areas which are marked "Not used".

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE00-BE02</td>
<td>BI.ENTRY</td>
<td>JMP to WARMDO$ (BI warmstart vector).</td>
</tr>
<tr>
<td>BE03-GE05</td>
<td>DOSCMD</td>
<td>JMP to SYNTAX (BI command line parse and execute).</td>
</tr>
<tr>
<td>BE06-GE08</td>
<td>EXTRNMD</td>
<td>JMP to user-installed external command parser.</td>
</tr>
<tr>
<td>BE09-GE0B</td>
<td>ERROUT</td>
<td>JMP to BI error handler.</td>
</tr>
<tr>
<td>BE0C-GE0E</td>
<td>PRINTERR</td>
<td>JMP to BI error message print routine.</td>
</tr>
<tr>
<td>BE0F</td>
<td>ERRCODE</td>
<td>ProDOS error code (also at $DE, Applesoft ONERR code).</td>
</tr>
<tr>
<td>BE10-GE1F</td>
<td>OUTVEC</td>
<td>Default output vector in monitor and for each slot (1-7).</td>
</tr>
<tr>
<td>BE20-GE2F</td>
<td>INVEC</td>
<td>Default input vector in monitor for each slot (1-7).</td>
</tr>
<tr>
<td>BE30-GE31</td>
<td>VECTOUT</td>
<td>Current output vector.</td>
</tr>
<tr>
<td>BE32-GE33</td>
<td>VECTIN</td>
<td>Current input vector.</td>
</tr>
<tr>
<td>BE34-GE35</td>
<td>VDOSIO</td>
<td>BI's output intercept address.</td>
</tr>
<tr>
<td>BE36-GE37</td>
<td>VI'SIN</td>
<td>BI's input intercept address.</td>
</tr>
<tr>
<td>BE38-GE39</td>
<td>VSYSIO</td>
<td>BI's internal redirection by STATE.</td>
</tr>
<tr>
<td>BE3C</td>
<td>DEFSLT</td>
<td>Default slot.</td>
</tr>
<tr>
<td>BE3D</td>
<td>DEFDRV</td>
<td>Default drive.</td>
</tr>
<tr>
<td>BE3E</td>
<td>PREGA</td>
<td>A-register savearea.</td>
</tr>
<tr>
<td>BE3F</td>
<td>PREGX</td>
<td>X-register savearea.</td>
</tr>
<tr>
<td>BE40</td>
<td>PREGY</td>
<td>Y-register savearea.</td>
</tr>
<tr>
<td>BE41</td>
<td>DTRACE</td>
<td>Applesoft TRAC€ is enabled flag (MSB on).</td>
</tr>
<tr>
<td>BE42</td>
<td>STATE</td>
<td>Current intercept state. 0 = immediate command mode. &gt;0 = deferred.</td>
</tr>
<tr>
<td>BE43</td>
<td>EXACTV</td>
<td>EXEC file active flag (MSB on).</td>
</tr>
<tr>
<td>BE44</td>
<td>IPILACTV</td>
<td>READ file active flag (MSB on).</td>
</tr>
<tr>
<td>BE45</td>
<td>OFILACTV</td>
<td>WRITE file active flag (MSB on).</td>
</tr>
<tr>
<td>BE46</td>
<td>PFILACTV</td>
<td>PREFIX read active flag (MSB on).</td>
</tr>
<tr>
<td>BE47</td>
<td>DFLFLG</td>
<td>File being READ is a DIR file (MSB on).</td>
</tr>
<tr>
<td>BE48</td>
<td>EDIRFLG</td>
<td>End of directory flag (no longer used).</td>
</tr>
<tr>
<td>BE49</td>
<td>STRINGS</td>
<td>String space count used to determine when to garbage collect.</td>
</tr>
<tr>
<td>BE4A</td>
<td>TBUPPTR</td>
<td>Buffered WRITE data length.</td>
</tr>
<tr>
<td>BE4B</td>
<td>INPTR</td>
<td>Command line assembly length.</td>
</tr>
<tr>
<td>BE4C</td>
<td>CHRLAST</td>
<td>Previous output character (for recursion check).</td>
</tr>
<tr>
<td>BE4D</td>
<td>OPENCNTR</td>
<td>Number of files open (not counting EXEC).</td>
</tr>
<tr>
<td>BE4E</td>
<td>YXFILE</td>
<td>EXEC file being closed flag (MSB on).</td>
</tr>
<tr>
<td>BE4F</td>
<td>CATFLAG</td>
<td>Line type to format next in DIR file READ.</td>
</tr>
<tr>
<td>BE50-GE51</td>
<td>XTPNADDR</td>
<td>External command handler address.</td>
</tr>
<tr>
<td>BE52</td>
<td>XLEN</td>
<td>Length of command name (less one).</td>
</tr>
</tbody>
</table>
### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE53</td>
<td>XNUM</td>
<td>Number of commands:</td>
</tr>
<tr>
<td></td>
<td>$00 = external</td>
<td>$90 = OPEN</td>
</tr>
<tr>
<td></td>
<td>$01 = IN#</td>
<td>$91 = WRITE</td>
</tr>
<tr>
<td></td>
<td>$02 = PR#</td>
<td>$92 = READ</td>
</tr>
<tr>
<td></td>
<td>$03 = CAT</td>
<td>$93 = APPEND</td>
</tr>
<tr>
<td></td>
<td>$04 = PRE</td>
<td>$94 = OPEN</td>
</tr>
<tr>
<td></td>
<td>$05 = BYE</td>
<td>$95 = SAVE</td>
</tr>
<tr>
<td></td>
<td>$06 = RUN</td>
<td>$96 = CREATE</td>
</tr>
<tr>
<td></td>
<td>$07 = BRUN</td>
<td>$97 = DELETE</td>
</tr>
<tr>
<td></td>
<td>$08 = EXEC</td>
<td>$98 = STORE</td>
</tr>
<tr>
<td></td>
<td>$09 = LOAD</td>
<td>$99 = RESTORE</td>
</tr>
<tr>
<td></td>
<td>$0A = LOCK</td>
<td>$0F = POSITION</td>
</tr>
</tbody>
</table>

### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE54-BE55</td>
<td>FBITS</td>
<td>Permitted command operands bits:</td>
</tr>
<tr>
<td></td>
<td>$8000</td>
<td>Prefix needed. Pathname optional.</td>
</tr>
<tr>
<td></td>
<td>$4000</td>
<td>Slot number only (PR# or IN#).</td>
</tr>
<tr>
<td></td>
<td>$2000</td>
<td>Deferred command.</td>
</tr>
<tr>
<td></td>
<td>$1000</td>
<td>File name optional.</td>
</tr>
<tr>
<td></td>
<td>$0800</td>
<td>If file does not exist, create it.</td>
</tr>
<tr>
<td></td>
<td>$0400</td>
<td>T; file type permitted.</td>
</tr>
<tr>
<td></td>
<td>$0200</td>
<td>Second file name required.</td>
</tr>
<tr>
<td></td>
<td>$0100</td>
<td>First file name required.</td>
</tr>
<tr>
<td></td>
<td>$0080</td>
<td>AD; address keyword permitted.</td>
</tr>
<tr>
<td></td>
<td>$0040</td>
<td>B; byte offset permitted.</td>
</tr>
<tr>
<td></td>
<td>$0020</td>
<td>E; ending address permitted.</td>
</tr>
<tr>
<td></td>
<td>$0010</td>
<td>L; length permitted.</td>
</tr>
<tr>
<td></td>
<td>$0008</td>
<td>S; line number permitted.</td>
</tr>
<tr>
<td></td>
<td>$0004</td>
<td>S or D; slot/drive permitted.</td>
</tr>
<tr>
<td></td>
<td>$0002</td>
<td>F; field permitted.</td>
</tr>
<tr>
<td></td>
<td>$0001</td>
<td>R; record permitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(V always permitted but ignored.)</td>
</tr>
</tbody>
</table>

### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE56-BE57</td>
<td>FBITS</td>
<td>Operands found on command line. Same bit assignments as above.</td>
</tr>
<tr>
<td>BE58-BE59</td>
<td>VADDR</td>
<td>A keyword value.</td>
</tr>
<tr>
<td>BE5A-BE5C</td>
<td>VBYTE</td>
<td>B keyword value.</td>
</tr>
<tr>
<td>BE5D-BE5E</td>
<td>VENDA</td>
<td>E keyword value.</td>
</tr>
<tr>
<td>BE5F-BE60</td>
<td>VLNTH</td>
<td>L keyword value.</td>
</tr>
<tr>
<td>BE61</td>
<td>VSLOT</td>
<td>S keyword value.</td>
</tr>
<tr>
<td>BE62</td>
<td>VDRV</td>
<td>D keyword value.</td>
</tr>
<tr>
<td>BE63-BE64</td>
<td>VFIELD</td>
<td>F keyword value.</td>
</tr>
<tr>
<td>BE65-BE66</td>
<td>VRECD</td>
<td>R keyword value.</td>
</tr>
<tr>
<td>BE67</td>
<td>VVOLM</td>
<td>V keyword value (ignored).</td>
</tr>
<tr>
<td>BE68-BE69</td>
<td>VLINE</td>
<td>@ keyword value.</td>
</tr>
<tr>
<td>BE6A</td>
<td>VTYPE</td>
<td>T keyword value (in hex).</td>
</tr>
<tr>
<td>BE6B</td>
<td>VIOSLT</td>
<td>PR# or IN# slot number value.</td>
</tr>
</tbody>
</table>

### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE6C-BE6D</td>
<td>VPATH1</td>
<td>Primary pathname buffer (address of length byte).</td>
</tr>
<tr>
<td>BE6E-BE6F</td>
<td>VPATH2</td>
<td>Secondary pathname buffer (address of length byte).</td>
</tr>
<tr>
<td>BE70-BE74</td>
<td>GOSYSTEM</td>
<td>Call the MLI using the parameter tables which follow.</td>
</tr>
<tr>
<td>BE75</td>
<td>SYSCALL</td>
<td>MLI call number for this call.</td>
</tr>
<tr>
<td>BE76-BE77</td>
<td>SYSPARM</td>
<td>Address of MLI parameter list for this call.</td>
</tr>
<tr>
<td>BE78-BE7A</td>
<td>BADCALL</td>
<td>Return from MLI call.</td>
</tr>
<tr>
<td>BE7B-BE7E</td>
<td>BADCALL</td>
<td>MLI error return; translate error code to BI error number.</td>
</tr>
<tr>
<td>BE7F</td>
<td>BISPEEL</td>
<td>Not used.</td>
</tr>
<tr>
<td>BE80-BE81</td>
<td>SCREATE</td>
<td>CREATE parameter list.</td>
</tr>
<tr>
<td>BE82</td>
<td>SSGPRFX</td>
<td>GET_PREFX, SET_PREFX, DESTROY parameter list.</td>
</tr>
<tr>
<td>BE83</td>
<td>SRENAME</td>
<td>RENAME parameter list.</td>
</tr>
<tr>
<td>BE84-BE85</td>
<td>SGINFO</td>
<td>GET_FILE_INFO, SET_FILE_INFO parameter list.</td>
</tr>
<tr>
<td>BE86</td>
<td>BONLINE</td>
<td>ONLINE, SET_MARK, GET_MARK, SET_EOF, GET_EOF, SET_BUF, GET_BUF, QUIT parameter list.</td>
</tr>
<tr>
<td>BE87-BE88</td>
<td>OPEN</td>
<td>OPEN parameter list.</td>
</tr>
<tr>
<td>BE89</td>
<td>BNEWLN</td>
<td>SET_NEWLINE parameter list.</td>
</tr>
<tr>
<td>BE90-BE91</td>
<td>SREAD</td>
<td>READ, WRITE parameter list.</td>
</tr>
<tr>
<td>BE92-BE93</td>
<td>SCLOSE</td>
<td>CLOSE, Flush parameter list.</td>
</tr>
<tr>
<td>BE94</td>
<td>CCCSPARE</td>
<td>&quot;COPYRIGHT APPLE, 1983&quot;</td>
</tr>
<tr>
<td>BE95-BE96</td>
<td>GETBUFF</td>
<td>GETBUFF buffer allocation subroutine vector.</td>
</tr>
<tr>
<td>BE97</td>
<td>FREEBUF</td>
<td>FREEBUFF buffer free subroutine vector.</td>
</tr>
<tr>
<td>BE98</td>
<td>BEFA</td>
<td>Original HIMEM MSB.</td>
</tr>
<tr>
<td>BE99</td>
<td>BEFF</td>
<td>Not used.</td>
</tr>
</tbody>
</table>
Disk Controller Boot ROM -- Apple II/II+/Iie  NEXT OBJECT ADDR: C600

C600  MODULE STARTING ADDRESS

*******************************************************************************
* BOOT ROM = APPLE DISK CONTROLLER *
* FOR APPLE II, II+, AND Iie. *
* THIS CODE RESIDES FROM C600 *
* TO $C6FF, IT LOADS TRACK 0 *
* SECTOR 0 INTO RAM AT $800 AND *
* JUMPS TO IT *
*******************************************************************************

******* ZERO PAGE ADDRESSES ***********

$026  SECTOR BUFFER POINTER
$02B  SLOT NUMBER * 16 FOR INDEX
$03C  WORKBYTE
$03D  SECTOR WANTED
$040  TRACK FOUND
$041  TRACK WANTED

******* EXTERNAL ADDRESSES ***********

$010  SYSTEM STACK
$030  AUXILIARY BUFFER
$036  TRANSLATE TABLE
$080  SECTORS TO LOAD
$081  ENTRY POINT
$088  PHASES OFF
$089  PHASES ON
$08A  MOTOR ON
$098  DRIVE SELECT
$09C  READ DATA REGISTER
$09E  SET READ MODE
$0AC  MONITOR WAIT ROUTINE
$0F8  RTS

C600  ********** BUILD READ TRANSLATE TABLE ***************

C600  SIGNATURE
C602  INITIALIZE TABLE VALUE INDICATOR
C606  STORE BIT PATTERN
C609  SHIFT PATTERN LEFT ONE BIT
C60A  ARE THERE ANY TWO ADJACENT BITS ON?
C60C  NO, TRY ANOTHER PATTERN >>C61E
C60E  YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROS
C610  FLIP BITS, PAIR OF ZERO BITS NOW SINGLE ONE BIT
C612  HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
C614  --- >>C61E
C616  SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON

C617  IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN >>C614
C619  FOUND ONE, GET TABLE VALUE
C61A  AND STORE IT IN TABLE (0356)
C61D  INCREMENT TABLE VALUE INDICATOR
C61E  GET NEXT BIT PATTERN, DONE YET
C61F  NO, GO CHECK IT OUT >>C606

C621  *********** DETERMINE SLOT, TURN DRIVE ON ***************

C621  CALL A KNOWN RTS <FF58>
C624  GET STACK POINTER
C625  GET HIGH BYTE OF WHERE WE ARE (0100)
C628  TIMES 16 TO GET SLOT
C62C  SAVE SLOT
C62E  PUT IN X REG FOR INDEX
C63F  INSURE READ MODE (C88E)
C635  SELECT DRIVE 1 (C88A)
C638  TURN THE MOTOR ON (C889)

C63B  ********** RECALIBRATE DISK ARM ***************

C63B  PREPARE TO STEP THE ARM 80 PHASES
C63D  TURN A PHASE OFF (C88B)
C640  PUT COUNTER IN ACCUMULATOR
C641  CREATE A PHASE NUMBER (0-3)
C643  DOUBLE IT FOR PROPER INDEX
C644  COMBINE WITH SLOT FOR FINAL INDEX
C646  PUT INDEX IN X REGISTER
C647  TURN A PHASE ON (C861)
C64A  DELAY ABOUT 20 MICROSECONDS
C64F  DECREMENT COUNTER
C650  LOOP UNTIL ALL 80 ARE DONE >>C63D

C652  *********** INITIALIZATION ***************

C652  ---
C654  SECTOR TO FIND -> $00
C656  TRACK TO FIND -> $00
C65A  MAIN BUFFER POINTER ($26) -> $8000
C65C  CLEAR THE CARRY
C65D  PUSL STATUS ON STACK

C65E  *********** SEARCH FOR A VALID HEADER ***************

C65E  CHECK DATA REGISTER (C88C)
C661  LOOP UNTIL DATA IS VALID >>C65E
C663  IS IT A $D7?
C665  NO, TRY AGAIN >>C65E
C667  YES, CHECK REGISTER AGAIN (C88C)
C66A  LOOP UNTIL VALID >>C667
C66C  IS IT AN $AA
### Disk Controller Boot ROM -- Apple II/II+/IIE

**NEXT OBJECT ADDR:** C66E

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C66E</td>
<td>NO, SEE IF ITS A $D5 &gt;&gt;C663</td>
</tr>
<tr>
<td>C670</td>
<td>YES, DELAY FOR REGISTER TO CLEAR</td>
</tr>
<tr>
<td>C671</td>
<td>CHECK REGISTER (C98C)</td>
</tr>
<tr>
<td>C674</td>
<td>LOOP UNTIL VALID &gt;&gt;C671</td>
</tr>
<tr>
<td>C676</td>
<td>IS IT A $96</td>
</tr>
<tr>
<td>C678</td>
<td>YES, WE FOUND AN ADDRESS HEADER &gt;&gt;C683</td>
</tr>
<tr>
<td>C67A</td>
<td>NO, HAVE WE FOUND ONE PREVIOUSLY?</td>
</tr>
<tr>
<td>C67B</td>
<td>IF NOT, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C67D</td>
<td>WAS IT AN $AD?</td>
</tr>
<tr>
<td>C67F</td>
<td>YES, WE FOUND A DATA HEADER &gt;&gt;C6A6</td>
</tr>
<tr>
<td>C681</td>
<td>NO, START OVER &gt;&gt;C65C</td>
</tr>
</tbody>
</table>

#### C683 ******* DECODE ADDRESS FIELD ***************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C683</td>
<td>INITIALIZE COUNTER</td>
</tr>
<tr>
<td>C685</td>
<td>SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS</td>
</tr>
<tr>
<td>C687</td>
<td>READ DATA REGISTER (C98C)</td>
</tr>
<tr>
<td>C68A</td>
<td>LOOP UNTIL DATA VALID &gt;&gt;C687</td>
</tr>
<tr>
<td>C68C</td>
<td>SHIFT BITS INTO POSITION X1X1X1X1</td>
</tr>
<tr>
<td>C68D</td>
<td>SAVE FOR LATER</td>
</tr>
<tr>
<td>C68F</td>
<td>READ REGISTER FOR NEXT BYTE (C98C)</td>
</tr>
<tr>
<td>C692</td>
<td>LOOP UNTIL VALID &gt;&gt;C68F</td>
</tr>
<tr>
<td>C694</td>
<td>COMBINE WITH PREVIOUS 1X1X1X1 AND 1X1X1X1</td>
</tr>
<tr>
<td>C696</td>
<td>DECREMENT COUNTER, DONE YET?</td>
</tr>
<tr>
<td>C697</td>
<td>NO, DO ANOTHER &gt;&gt;C685</td>
</tr>
<tr>
<td>C699</td>
<td>KEEP THE STACK CLEAN</td>
</tr>
<tr>
<td>C69A</td>
<td>IS THIS SECTOR WE WANT?</td>
</tr>
<tr>
<td>C69C</td>
<td>NO, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C69E</td>
<td>GET TRACK FOUND</td>
</tr>
<tr>
<td>C6A0</td>
<td>IS IT TRACK WE WANT?</td>
</tr>
<tr>
<td>C6A2</td>
<td>NO, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C6A4</td>
<td>YES, INDICATE ADDRESS FOUND, GO LOOK FOR DATA FIELD &gt;&gt;C65D</td>
</tr>
</tbody>
</table>

#### C6A6 ******* READ DATA FIELD ***************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6A6</td>
<td>INITIALIZE OFFSET (AUXILIARY BUFFER)</td>
</tr>
<tr>
<td>C6A8</td>
<td>---</td>
</tr>
<tr>
<td>C6AA</td>
<td>READ DATA REGISTER (C98C)</td>
</tr>
<tr>
<td>C6AD</td>
<td>LOOP UNTIL VALID &gt;&gt;C6AA</td>
</tr>
<tr>
<td>C6AF</td>
<td>EXCLUSIVE-OR WITH TRANSLATE TABLE ($2D6)</td>
</tr>
<tr>
<td>C6B4</td>
<td>DECREMENT OFFSET</td>
</tr>
<tr>
<td>C6B5</td>
<td>STORE BYTE IN AUXILIARY BUFFER ($300)</td>
</tr>
<tr>
<td>C6B8</td>
<td>LOOP UNTIL BUFFER FULL &gt;&gt;C6A8</td>
</tr>
<tr>
<td>C6B9</td>
<td>INITIALIZE OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6BC</td>
<td>READ DATA REGISTER (C98C)</td>
</tr>
<tr>
<td>C6BF</td>
<td>LOOP UNTIL VALID &gt;&gt;C6BC</td>
</tr>
<tr>
<td>C6C1</td>
<td>EXCLUSIVE-OR WITH TRANSLATE TABLE ($2D6)</td>
</tr>
<tr>
<td>C6CC</td>
<td>STORE BYTE IN MAIN BUFFER</td>
</tr>
<tr>
<td>C6CB</td>
<td>INCREMENT OFFSET</td>
</tr>
<tr>
<td>C6D9</td>
<td>LOOP UNTIL BUFFER FULL &gt;&gt;C6BA</td>
</tr>
<tr>
<td>C6EC</td>
<td>READ DATA REGISTER (C98C)</td>
</tr>
</tbody>
</table>

#### C6D5 ******* MERGE MAIN AND AUXILIARY BUFFERS*******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6D5</td>
<td>INITIALIZE OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6D7</td>
<td>INITIALIZE OFFSET (AUXILIARY BUFFER)</td>
</tr>
<tr>
<td>C6D9</td>
<td>DECREMENT OFFSET (AUX BUFFER)</td>
</tr>
<tr>
<td>C6DA</td>
<td>IF LESS THAN ZERO RESET IT &gt;&gt;C6D7</td>
</tr>
<tr>
<td>C6DC</td>
<td>GET BYTE FROM MAIN BUFFER</td>
</tr>
<tr>
<td>C6E1</td>
<td>ROLL IN TWO BITS FROM AUXILIARY BUFFER</td>
</tr>
<tr>
<td>C6E6</td>
<td>SAVE COMPLETED DATA BYTE</td>
</tr>
<tr>
<td>C6EB</td>
<td>INCREMENT OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6E9</td>
<td>LOOP UNTIL WHOLE BUFFER IS DONE &gt;&gt;C6D9</td>
</tr>
</tbody>
</table>

#### C6EB ******* DETERMINE IF THERE IS MORE TO DO*******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6EB</td>
<td>INCREMENT MAIN BUFFER POINTER</td>
</tr>
<tr>
<td>C6ED</td>
<td>INCREMENT SECTOR NUMBER</td>
</tr>
<tr>
<td>C6F1</td>
<td>IS THERE ANOTHER SECTOR TO LOAD? ($0008)</td>
</tr>
<tr>
<td>C6F6</td>
<td>YES, GO DO IT &gt;&gt;C6D3</td>
</tr>
<tr>
<td>C6F8</td>
<td>NO, ENTER CODE WE JUST LOADED &gt;&gt;$081</td>
</tr>
</tbody>
</table>

#### C6FB ******* UNUSED ***************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6FB</td>
<td>5 BYTES AT END OF PAGE ARE UNUSED</td>
</tr>
</tbody>
</table>
C552  MODULE STARTING ADDRESS

*****************************************************************************
*  *  BOOT ROM - APPLE //c CONTROLLER ROM  *
*  THIS CODE RESIDES FROM $C552  *
*  TO $C6FF. IT LOADS TRACK 0  *
*  SECTOR 0 INTO RAM AT $600 AND  *
*  JUMPS TO IT. IF BOOT FAILS IT  *
*  THEN TRIES TO BOOT SLOT 5,  *
*  THE PROTOCOL CONVERTER.  *
*  *
*  THIS IS THE VERSION OF THE IIC ROM  *
*  THAT SUPPORTS THE UNIDISK 3.5,  *
*  26 JULY 85.  *
*
*****************************************************************************

******* ZERO PAGE ADDRESSES *******

Ø081  SLOT PAGE PUT HERE DURING AUTOBOOT
Ø083  RETRY COUNT (HIGH BYTE)
Ø086  SECTOR BUFFER POINTER
Ø08B  SLOT NUMBER * 16 FOR INDEX
Ø08C  WIRED BYTE
Ø03D  SECTOR WANTED
Ø040  TRACK FOUND
Ø041  TRACK WANTED
Ø04F  DRIVE TO BOOT FROM

******** EXTERNAL ADDRESSES **********

Ø300  AUXILIARY BUFFER
Ø356  TRANSLATE TABLE
Ø7DB  SCREEN LOCATION
Ø800  SECTORS TO LOAD
Ø801  ENTRY POINT
C080  PHASE0 OFF
C081  PHASE0 ON
C088  MOTOR OFF
C089  MOTOR ON
C08C  READ DATA REGISTER
C08E  SET READ MODE
C08A  DRIVE SELECT
FCA8  MONITOR WAIT ROUTINE
Disk Controller Boot ROM -- Apple IIc
NEXT OBJECT ADDR: C5F5

**ADDR**      **DESCRIPTION/CONTENTS**
C5F5          ************** JUMP TO BOOTFAIL **************
C5F5          BRANCH TO BOOTFAIL >>C552
C5F8          REMAINING 8 BYTES NOT USED BY DISK II >>C576
C600          ************ INITIALIZATION ************
C600          SIGNATURE
C602          SET DRIVE -> 1
C604          INITIALIZE RETRY COUNT (HIGH BYTE)
C608          ******** SELECT DRIVE AND TURN IT ON **********
C608          --
C60B          INITIALIZE SLOT (6)
C60D          INITIALIZE DEVICE (1 OR 2)
C60F          SAVE DRIVE NUMBER ON STACK
C610          INSURE READ MODE (C98E)
C616          GET DRIVE NUMBER BACK
C617          SELECT APPROPRIATE DRIVE (C0E8)
C61A          TURN MOTOR ON (C0E9)
C61D          ********** RECALIBRATE DISK ARM **********
C61D          PREPARE TO STEP THE ARM 80 PHASES
C61F          TURN A PHASE OFF (C080)
C622          PUT COUNTER IN A REGISTER
C623          CREATE A PHASE NUMBER (8-3)
C625          DOUBLE IT FOR PROPER INDEX
C626          COMBINE WITH SLOT FOR FINAL INDEX
C628          PUT INDEX IN X REGISTER
C629          TURN A PHASE ON (C081)
C62C          DELAY ABOUT 20 MICROSECONDS
C631          DECREMENT COUNTER
C632          LOOP UNTIL ALL 80 ARE DONE >>C61F
C634          *********** INITIALIZATION ***********
C634          --
C636          SECTOR TO FIND -> $80
C638          TRACK TO FIND -> $80
C63A          BUILD THE TRANSLATE TABLE <C5E8>
C63D          ******** COUNT RETRIES AND INDICATE ERROR IF BOOT FAILS********
C63D          INITIALIZE RETRY COUNT
C63F          CLEAR THE CARRY
C640          PUSH STATUS ON STACK
C641          KEEP STACK CLEAN
C642          GET SLOT

Disk Controller Boot ROM -- Apple IIc
NEXT OBJECT ADDR: C644

**ADDR**      **DESCRIPTION/CONTENTS**
C644          DECREMENT RETRY COUNT, TRY AGAIN?
C646          YES, GO DO IT >>C656
C648          NO, TURN DRIVE OFF (C88E)
C64B          AUTO BOOT FROM SLOT? 6?
C64F          NO, FAIL NOW >>C5F5
C651          MAYBE SLOT 5 WILL TALK TO US >>C500
C654          TWO BYTES NOT USED >>0002
C656          --
C657          DECREMENT RETRY COUNT (LOW BYTE)
C658          IF NOT ZERO, TRY AGAIN >>C65E
C65A          IF SO, GO DECREMENT RETRY COUNT (HIGH BYTE) >>C641
C65C          SPACE FILLER TO POSITION CODE BELOW >>C63D
C65E          ********* SEARCH FOR A VALID HEADER ***********
C65E          CHECK DATA REGISTER (C08C)
C661          LOOP UNTIL DATA IS VALID >>C65E
C663          IS IT A $D5?
C665          NO, TRY AGAIN >>C657
C667          YES, CHECK REGISTER AGAIN (C88C)
C66A          LOOP UNTIL VALID >>C667
C66C          IS IT AN $A?
C66E          NO, SEE IF ITS A $D5 >>C663
C670          YES, DELAY FOR REGISTER TO CLEAR
C671          CHECK REGISTER (C08C)
C674          LOOP UNTIL VALID >>C671
C676          IS IT A $96?
C678          YES, WE FOUND AN ADDRESS HEADER >>C683
C67A          NO, HAVE WE FOUND ONE PREVIOUSLY?
C67B          IF NOT, START OVER >>C63F
C67D          WAS IT AN $AD?
C67F          YES, WE FOUND A DATA HEADER >>C6A6
C681          NO, START OVER >>C63F
C683          ********** DECODE ADDRESS FIELD **********
C683          INITIALIZE COUNTER
C685          SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS
C687          READ DATA REGISTER (C08C)
C68A          LOOP UNTIL DATA VALID >>C687
C68C          SHIFT BITS INTO POSITION XIIXIXIX
C68D          SAVE FOR LATER
C68F          READ REGISTER FOR NEXT BYTE (C08C)
C692          LOOP UNTIL VALID >>C69E
C694          COMBINE WITH PREVIOUS 1XIIXIXIX AND XIIXIXIX
C696          DECREMENT COUNTER, DONE YET?
C697          NO, DO ANOTHER >>C685
C699          KEEP THE STACK CLEAN
C69A          IS THIS SECTOR WE WANT?
C69C          NO, START OVER >>C63F
C69E          GET TRACK FOUND
Disk Controller Boot ROM -- Apple IIc

Address Description/Contents

C6A0 IS IT TRACK WE WANT?
C6A2 NO, START OVER >>C63F
C6A4 YES, INDICATE ADDRESS FOUND, GO LOOK FOR DATA FIELD >>C642

C6A6 ********** READ DATA FIELD **********

C6A6 INITIALIZE OFFSET (AUXILIARY BUFFER)
C6A8 --
C6AA READ DATA REGISTER (C88C)
C6AD LOOP UNTIL VALID >>C6AA
C6AF EXCLUSIVE-OR WITH TRANSFORM TABLE (02D6)
C6B4 DECREMENT OFFSET
C6B5 STORE BYTE IN AUXILIARY BUFFER (0300)
C6B8 LOOP UNTIL BUFFER FULL >>C6A8
C6BA INITIALIZE OFFSET (MAIN BUFFER)
C6BC READ DATA REGISTER (C88C)
C6BF LOOP UNTIL VALID >>C6BC
C6C1 EXCLUSIVE-OR WITH TRANSFORM TABLE (02D6)
C6C6 STORE BYTE IN MAIN BUFFER
C6C9 INCREMENT OFFSET
C6CB READ DATA REGISTER (C88C)
C6CE LOOP UNTIL VALID >>C6CB
C6D0 IS CHECKSUM OKAY? (02D6)
C6D3 NO, START OVER >>C6A2

C6D5 ********** MERGE MAIN AND AUXILIARY BUFFERS**********

C6D5 INITIALIZE OFFSET (MAIN BUFFER)
C6D7 INITIALIZE OFFSET (AUXILIARY BUFFER)
C6D9 DECREMENT OFFSET (AUX BUFFER)
C6DA IF LESS THAN ZERO RESET IT >>C6D7
C6DC GET BYTE FROM MAIN BUFFER
C6E1 ROLL IN TWO BITS FROM AUXILIARY BUFFER
C6E6 SAVE COMPLETED DATA BYTE
C6EB INCREMENT OFFSET (MAIN BUFFER)
C6E9 LOOP UNTIL WHOLE BUFFER IS DONE >>C6D9

C6EB ********** DETERMINE IF THERE IS MORE TO DO**********

C6EB INCREMENT MAIN BUFFER POINTER
C6ED INCREMENT SECTOR NUMBER
C6F1 IS THERE ANOTHER SECTOR TO LOAD? (0880)
C6F6 YES, GO DO IT >>C6D3
C6FD NO, ENTER CODE WE JUST LOADED >>881

C6FB 5 ZERO BYTES AT END OF PAGE
Disk II Boot ROM -- Apple IIgs

---

ADDR DESCRIPTION/CONTENTS

---

C600 MODULE STARTING ADDRESS

******************************************
* C600 ROM -- APPLE IIgs
* If an Apple IIgs has slot 6 configured for Disk Port,
* then this code is executed
* when a boot is attempted
* from slot 6.
* IIGS ROM VERSION 0

******************************************

C600 ****** ZERO PAGE ADDRESSES ****************************

1 SLOT PAGE PUT HERE DURING AUTOBOOT
3 RETRY COUNT (HIGH BYTE)
26 SECTOR BUFFER POINTER
2B SLOT NUMBER * 16
3C WORKBYTE
3D SECTOR WANTED
40 TRACK FOUND
41 TRACK WANTED
4F DRIVE TO BOOT FROM

C600 ****** EXTERNAL ADDRESSES ****************************

0356 TRANSLATE TABLE
07F0 SLOT THAT OWNS C600-C7FF
07FE UTILITY BYTE FOR SLOT 6
0800 SECTORS TO LOAD
0801 ENTRY POINT
C080 PHASE0 OFF
C091 PHASE0 ON
C088 MOTOR OFF
C089 MOTOR ON
C08C READ DATA REGISTER
C08E SET READ MODE
C08A DRIVE SELECT
E000 APPLESOFT BASIC ENTRY POINT
FA6F MONITOR DISK CONTROLLER SEARCH LOOP
FCAB MONITOR WAIT ROUTINE

---

Disk II Boot ROM -- Apple IIgs

---

ADDR DESCRIPTION/CONTENTS

---

C600 ****** IIGS SOFT SWITCHES ****************************
C82D SLTROMSEL. SLOT ROM CONFIGURATION BYTE
C835 SHADOW. ENABLES/DISABLES SHADOWING
C868 STATREG. ONE BYTE SETS 8 SOFT SWITCHES

C600 ****** C600 ENTRY POINT ****************************

---

C600 SIGNATURE
C602 SET DRIVE TO DRIVE 1
C604 INITIALIZE RETRY COUNT (HIGH BYTE)
C605 PUSHER ON STACK
C610A SET NORMAL ZERO PAGE
C610C SET DATA BANK SAME AS PROGRAM BANK
C610F MAKE SLOT 6 CURRENT (07F8)
C612 8-BIT MEMORY AND INDEX OPERATIONS
C616 STORE P-REGISTER IN SCREEN HOLE (07FE)
C619 DISABLE INTERRUPTS
C61A GO DO SOME TASKS ELSEWHERE IN ROM <FF5909>
C61B RESTORE SLOT*16 TO X-REGISTER
C620 IF CARRY SET, I/O ERROR >>C62E

C622 ****** COUNT RETRIES AND INDICATE ERROR ****************************
IF BOOT FAILS

C622 INITIALIZE RETRY COUNT
C624 DISABLE INTERRUPTS (AGAIN)
C625 CLEAR CARRY AND
C626 PUT IT ON THE STACK.
C627 KEEP STACK EVEN
C628 GET SLOT*16 IN X-REGISTER
C62A DECREMENT RETRY COUNT. TRY AGAIN?
C62C YES, GO DO IT >>C656
C62E NO, TURN DRIVE OFF (C088)
C631 CALL BOOT ERROR HANDLER <FF5971>
C635 ENABLE INTERRUPTS IF OK TO <C648>
C638 AUTOBOOT FROM SLOT 6?
C63D NO, GO TO BASIC >>C642
C63F YES, RETURN TO SLOT SEARCH LOOP >>FABA
C642 JUMP TO BASIC >>E000

C645 ****** SUBROUTINE TO ENABLE INTERRUPTS ****************************

C645 ALLOW SHADOWING AND I/O (C835)
C648 CHECK ORIGINAL P-REGISTER (07FE)
C64B INTERRUPT BIT HIGH?
C64D YES, LEAVE INTERRUPTS DISABLED >>C650
C64F NO, ENABLE INTERRUPTS
C650 RESTORE SECTOR TO ACCUMULATOR
C652 RETURN
C653 ******* THREE BYTES NOT USED ****************************

C653  NOT USED

C656 ******* INCREMENT RETRIES ****************************

C656  ---
C657  DECREMENT RETRY COUNT (LOW BYTE)
C658  IF NOT ZERO, TRY AGAIN >>C65E
C65A  ZERO, SO GO DECREMENT HIGH BYTE >>C627
C65C  SPACE FILLER TO POSITION CODE BELOW >>C622

C658 ******* SEARCH FOR A VALID HEADER ****************************

C65E  CHECK DATA REGISTER (C08C)
C661  LOOP UNTIL DATA IS VALID >>C65E
C663  IS IT A $D5?
C665  NO, TRY AGAIN >>C657
C667  YES, CHECK REGISTER AGAIN (C08C)
C66A  LOOP UNTIL VALID >>C667
C66C  IS IT AN $AA?
C66E  NO, SEE IF IT'S A $D5 >>C663
C670  YES, DELAY FOR REGISTER TO CLEAR
C671  CHECK REGISTER (C08C)
C674  LOOP UNTIL VALID >>C671
C676  IS IT A $96?
C678  YES, WE FOUND AN ADDRESS HEADER >>C683
C67A  NO, ARE WE LOOKING FOR DATA HEADER?
C67B  NO, SO START OVER >>C625
C67D  YES, WAS IT AN $9A?
C67F  YES, WE FOUND A DATA HEADER >>C6A6
C681  NO, START OVER >>C625

C683 ******* DECODE ADDRESS FIELD ****************************

C683  INITIALIZE COUNTER
C685  SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS
C687  READ DATA REGISTER (C08C)
C68A  LOOP UNTIL DATA VALID >>C687
C68C  SHIFIT BITS INTO POSITION X1X1X1X1
C68D  SAVE FOR LATER
C68F  READ REGISTER FOR NEXT BYTE (C08C)
C692  LOOP UNTIL VALID >>C68F
C694  COMBINE WITH PREVIOUS X1X1X1X1 AND X1X1X1X1
C696  DECREMENT COUNTER. DONE YET?
C697  NO, DO ANOTHER >>C685
C699  KEEP STACK CLEAN
C69A  IS THIS SECTOR WE WANT?
C69C  NO, START OVER >>C625
C69E  GET TRACK FOUND

C6A8  IS IT TRACK WE WANT?
C6A2  NO, START OVER >>C625
C6A4  YES, INDICATE ADDR FOUND, GO LOOK FOR DATA FIELD >>C62B

C6A6 ******* READ DATA FIELD ****************************

C6A6  INITIALIZE OFFSET (AUXILIARY BUFFER)
C6A8  ---
C6AA  READ DATA REGISTER (C08C)
C6AD  LOOP UNTIL VALID >>C6AA
C6AF  EXCLUSIVE-OR WITH TRANSLATE TABLE (026D)
C6B4  DECREMENT OFFSET
C6B5  STORE BYTE IN AUXILIARY BUFFER (0300)
C6B8  LOOP UNTIL BUFFER FULL >>C6AB
C6BA  INITIALIZE OFFSET (MAIN BUFFER)
C6BC  READ DATA REGISTER (C08C)
C6BF  LOOP UNTIL VALID >>C6BC
C6C1  EXCLUSIVE-OR WITH TRANSLATE TABLE (026D)
C6C6  STORE BYTE IN MAIN BUFFER
C6C8  INCREMENT OFFSET
C6C9  LOOP UNTIL BUFFER FULL >>C6CB
C6CB  READ DATA REGISTER (C08C)
C6CE  LOOP UNTIL VALID >>C6CB
C6D0  IS CHECKSUM OKAY? (026D)
C6D3  NO, START OVER >>C6A2

C6D5 ******* MERGE MAIN AND AUXILIARY BUFFERS ****************************

C6D5  INITIALIZE OFFSET (MAIN BUFFER)
C6D7  INITIALIZE OFFSET (AUXILIARY BUFFER)
C6D9  DECREMENT OFFSET (AUX BUFFER)
C6DA  IF LESS THAN ZERO RESET IT >>C6D7
C6DC  GET BYTE FROM MAIN BUFFER
C6E1  ROLL IN TWO BITS FROM AUXILIARY BUFFER
C6E5  SAVE COMPLETED DATA BYTE
C6E8  INCREMENT OFFSET (MAIN BUFFER)
C6E9  LOOP UNTIL WHOLE BUFFER IS DONE >>C6D9

C6EB ******* DETERMINE IF THERE IS MORE TO DO ****************************

C6EB  INCREMENT MAIN BUFFER POINTER
C6ED  INCREMENT SECTOR NUMBER
C6F1  IS THERE ANOTHER SECTOR TO LOAD? (0800)
C6F6  YES, GO DO IT >>C6D3
C6FB  NO, ENABLE INTERRUPTS <C645>
C6FB  AND JUMP TO CODE WE JUST LOADED. >>0801
C6FE LAST TWO BYTES ARE ZERO

******************************************************************************
* SUBROUTINES ELSEWHERE IN ROM *
* (BANK FF -- IIGS ROM) *
* *
******************************************************************************

5909 ****** SELECT DRIVE AND TURN IT ON ****************************

590C ---
590E INITIALIZE SLOT TO 6
5910 INITIALIZE DEVICE (1 OR 2)
5912 INSURE READ MODE (C08E)
5916 SELECT APPROPRIATE DRIVE (C0EA)
5919 TURN MOTOR ON (C089)

591C ****** RECALIBRATE DISK ARM ****************************

591C PREPARE TO STEP THE ARM 80 PHASES
5921 PUT COUNTER IN A-REG
5922 CREATE A PHASE NUMBER (0-7)
5924 DOUBLE IT FOR PROPER INDEX
5925 COMBINE WITH SLOT FOR FINAL INDEX
5927 PUT INDEX IN X REGISTER
5928 TURN A PHASE OFF (C081)
592B DELAY ABOUT 20 MICROSECONDS
592D SAVE P-REGISTER
592E AND DISABLE INTERRUPTS
592F WHILE WAITING. <FC8A>
5932 RESTORE P-REGISTER
5933 DECREMENT COUNTER
5934 LOOP UNTIL ALL 80 ARE DONE >>591E

5936 ****** SET VARIABLES TO ZERO ****************************

5936 ZERO LOW BYTE, BUFFER ADDRESS
5938 SECTOR = 0
593A TRACK = 0
593C CLEAR CARRY, INDICATING READABLE DISK
593D TRY 5W READS
593F READ DATA REGISTER (C08C)
5942 FOUND SOMETHING >>5948
5944 DECREMENT COUNTER
5945 AND TRY AGAIN. >>593F
5947 CAN'T FIND ANY DATA. SET ERROR INDICATOR.

5948 ****** BUILD READ TRANSLATE TABLE ****************************

5948 SAVE P-REGISTER
5949 INITIALIZE BIT PATTERN
594B INITIALIZE TABLE VALUE INDICATOR
594D STORE BIT PATTERN
5954 SHIFT PATTERN LEFT ONE BIT
5957 ARE THERE ANY TWO ADJACENT BITS ON?
595C NO, TRY ANOTHER PATTERN >>5966
595D YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROS.
5957 FLIP BITS, PAIR OF ZEROS IT NOW SINGLE BIT, ETC.
5959 HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
595B --- >>5966
595D SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON
595E IF MORE THAN ONE BIT ON, TRY OTHER PATTERN >>595B
5960 FOUND ONE, GET TABLE VALUE
5961 AND STORE IT IN TABLE. (000356)
5965 INCREMENT TABLE VALUE INDICATOR
5966 GET NEXT BIT PATTERN. DONE YET?
5967 NO, GO CHECK IT OUT >>594D
5969 MAIN BUFFER POINTER ($26) -> $800
596D INITIALIZE RETRY COUNT
596F RESTORE P-REGISTER
5970 BACK TO $C600 CODE

5971 ****** HANDLE BOOT ERROR ****************************

5971 SET "NOT A STARTUP DISK" ERROR
5977 JUMP INTO BOOT ERROR LOGIC >>6366

6366 ****** BOOT ERROR LOGIC ****************************

6366 GET INTO NATIVE MODE
6368 SAVE P-REGISTER ON STACK
6369 8-BIT MEMORY AND INDEX OPERATION
636B AUTO BOOT FROM THIS SLOT?
636D YES, DON'T PRINT ERROR MESSAGE >>6423
636F SET DATA BANK REGISTER TO BANK 0 <<637>
636F SAVE SLOTROM STATUS (C02D)
637A ENABLE SMARTPORT/DISKPORT FOR SLOTS 5 AND 6
637F SAVE STATETREG (C068)
6383 READ ROM
6387 GET INTO EMULATION MODE
638B SET AND CLEAR NORMAL SCREEN IN PREPARATION <<0UC593>
638C TO PRINT SYSTEM ERROR MESSAGE ON ROW 10.
6404 RESTORE EMULATION MODE
6402 RESTORE SPOK SWITCHES (C068)
6406 RESTORE SLOTROM STATUS (C02D)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>6409</td>
<td>SET MSG CODE=0 (&quot;I/O ERROR&quot;)</td>
</tr>
<tr>
<td>640B</td>
<td>IS ERROR CODE ZERO? (E10F81)</td>
</tr>
<tr>
<td>640F</td>
<td>NO, KEEP CHECKING &gt;&gt;6413</td>
</tr>
<tr>
<td>6411</td>
<td>YES, SET MSG CODE=1 (&quot;NOT A STARTUP DISK&quot;)</td>
</tr>
<tr>
<td>6413</td>
<td>IS IT &quot;NO DEVICE CONNECTED&quot;?</td>
</tr>
<tr>
<td>6415</td>
<td>NO. &gt;&gt;6419</td>
</tr>
<tr>
<td>6417</td>
<td>YES, SET MSG CODE=2</td>
</tr>
<tr>
<td>6419</td>
<td>IS IT &quot;CHECK STARTUP DEVICE&quot;?</td>
</tr>
<tr>
<td>641B</td>
<td>NO, CALL IT &quot;I/O ERROR&quot; &gt;&gt;641F</td>
</tr>
<tr>
<td>641D</td>
<td>YES, SET MSG CODE=3</td>
</tr>
<tr>
<td>641F</td>
<td>PUT MSG CODE IN ACCUM</td>
</tr>
<tr>
<td>6420</td>
<td>AND PRINT IT OUT &lt;C600&gt;</td>
</tr>
<tr>
<td>6423</td>
<td>INDICATE I/O ERROR</td>
</tr>
<tr>
<td>6425</td>
<td>GET F-REGISTER BACK</td>
</tr>
<tr>
<td>6426</td>
<td>RESTORE MODE BASED ON CARRY</td>
</tr>
<tr>
<td>6427</td>
<td>INDICATE ERROR</td>
</tr>
<tr>
<td>6428</td>
<td>RETURN TO $C600 LOGIC</td>
</tr>
</tbody>
</table>
APPENDIX A

DIFFERENCES BETWEEN THE PRODOS 8 VERSIONS

This Appendix identifies the changes to ProDOS 8 that were introduced with the 1.2 and 1.3 versions. Although we believe this to be a fairly thorough list, there may have been a few changes (especially deletions) that we didn't catch. Quite a few changes that were apparently made only to save space are not listed here.

CHANGES INTRODUCED IN THE 1.2 VERSION

The changes that were made to the 1.1.1 version of ProDOS to produce ProDOS 8, Version 1.2, are listed below. Addresses given here are Version 1.2 addresses.

Relocator

1. When running on a IIGS with ProDOS 16 installed, ProDOS 8 is entered from PQUIT (the ProDOS 16 Quit Handler). In this case the Relocator is entered at $2003 instead of $2000. [2000-2005, page 11].

2. The ProDOS 8 version number is now stored in the MLI subdirectory header data area [2048-204C, page 11].

3. Always checks to see if running on a IIGS, and if so sets a flag (2278). Also sets E100BD=0 if ProDOS 8 is the initial boot on a IIGS. [2079-208D, page 11].

4. Sets aux stack pointer to $FF [20E6-20F5, 2111-2115, page 12].

5. If operating on a IIGS, skips logic that searches for slot 3 80-column card [212A-2139, page 12].

6. If operating on a IIGS, installs the IIGS Clock Code [21AD-21D0, 22D3-22DA, pages 12 and 14].

7. Now checks for an AppleTalk Initialization File (ATINIT file) before looking for a .SYSTEM file. If the ATINIT file is found, it is loaded and executed, then the search for a .SYSTEM file commences. [22DB-2381, page 14].

8. The list of devices is now ordered differently. It recognizes the SmartPort and allows four Slot 5 SmartPort units to be accessed as Slot 5, Drives 1 and 2 and Slot 2, Drives 1 and 2. If Slot 2 is being used by a storage device, however, only the first two devices on the Slot 5 SmartPort can be accessed. It also changes the search order, making sure that Disk II devices are searched last when a device scan takes place (such as during an MLI ON_LINE call). [2668-271B, 275D-2767, 2814-28AA, pages 17-19].
9. A bug in the /RAM driver (which we pointed out in the 1.1.1 supplement) that allowed a block read of block 7 (which doesn't exist) has been corrected [2D4B, page 23].

10. The /RAM caller, which operates in high RAM, now contains a $60 (RTS instruction) at address $FF58. Peripheral cards sometimes call that address to figure out which slot they are in, and in case they forget to set ROM for reading, the call will still work. [2E56-2E58, page 24].

11. A subroutine that sets high RAM for reading/writing was created to save space in the code [2518-251E, page 16].

**MLI and MLI Global Page**

1. The Global Page now pushes the P-Register and disables interrupts before calling the actual MLI [DE01-DE04, DE1C-DE21, BF4B-BF4F, pages 34 and 75].

2. Setting the MLIACTIVE flag a little differently now allows nested calls to the MLI by interrupt routines [DE8F-DE91, page 35].

3. A new MLI command was introduced (command=$82), that allows the user to install a routine to handle unclaimed interrupts [DEFF-DF0B, FD23-FD2C, pages 35 and 63].

4. If there is no unclaimed interrupt handler, ProDOS 8 now counts unclaimed interrupts, and will allow 255 of them to occur before finally issuing a fatal error. This allows a brief time for the unknown interrupt to stop interrupting. [DFB3-DFB7, page 36].

5. If operating on a IIGS and system death occurs, the NEWVIDEO softswitch is set to 0, reinitializing the IIGS video [E013-E016, page 36].

6. Processing for the ON LINE command now frees the VCB entry for a device that was previously on-line but has been taken off-line [E28A-E2A4, page 39].

7. A subroutine that reads a block where the block number is in the A and X registers was added to save space in the code [EBC9-EBD0, page 47].

8. The error message that results when a file being opened has an illegal storage type has been changed from "incompatible directory format" to "unsupported storage type" [ECC7-EECA, page 50].

9. An error type $C is now indicated when truncating or deleting a file and the file's storage type is illegal [FA4D-FA4E, page 60].

10. To save space in the QuitCode Caller, some in-line code was changed to a loop [FC1F-FC37, FCD8-FCE0, page 62].
Quit Code

1. Uses standard character set instead of alternate character set [1006-1008, page 77].
2. Sets normal 40-column screen in a safer way, such that screen hole values are preserved [100C-1011, page 77].
3. Message display routine is modified [1033-1034, 11D6-11D1, etc., pages 77 and 79].
4. The method of displaying the current prefix is changed so that it is always written to the same screen location [104D-105C, page 77].
5. User can now backspace with the DELETE key as well as left arrow [107C-107F, 10FC-10FF, page 78].
6. The method of inputting the Application name was modified [10E7-10E9, page 78].

Clock Code

1. The code for the ThunderClock includes a lookup table to determine the year based on the day of the year and the day of the week. This table is only good for a span of five or six years. The table released with Version 1.1.1 was good for the years 1982-1987. The table released with Version 1.2 covers the years 1986-1991. [D7B8-D7BE, page 91].
2. A completely separate clock routine is provided in Version 1.2 in case ProDOS 8 is operating on a IIGS. If so, the IIGS Clock Code is always enabled. It is written in 65816 and calls the ReadTimeHex tool in the tool kit to read the clock. [D742-D790, page 92].

CHANGES INTRODUCED IN THE 1.3 VERSION

The changes that were made to the 1.2 version of ProDOS 8 to produce ProDOS 8, Version 1.3, are listed below. Addresses given here are Version 1.3 addresses.

Relocator

1. The boot message now includes a line that says "ALL RIGHTS RESERVED." Chalk up one for the legal department! [25F6-2671, page 26].
2. A ProDOS Status call now immediately precedes the SmartPort Status Call. This is because the SmartPort interface does not set up its device list until it receives a ProDOS Status call. Earlier versions of ProDOS 8 may not always find all SmartPort devices. [286E-2894, page 28].
MLI

1. When files are deleted, previous versions of ProDOS zero out all but the first block of discarded index blocks. Now such index blocks will not be zeroed, but the pages of these blocks will be flipped. That is, the high byte of the block numbers will be exchanged with the low byte of the block numbers. [P992-P99A, FBC7-FBDC, pages 67 and 69].

2. Previous versions of ProDOS forgot, in certain cases, to rewrite index blocks that were being discarded when shortening or deleting files. Now such blocks will always be rewritten to disk in a zeroed (shortened file) or flipped (deleted file) form. [FAB8-FABC, FB91-FB93, pages 68 and 69].

3. A poorly-written loop in the QuitCode Caller that was added for Version 1.2 was rewritten for Version 1.3. The Version 1.2 code might cause problems on a IIGS. [FD05-FD0C, page 70].

Disk II Device Driver

1. There is a routine in the Disk II Device Driver that clears phases in case the Disk II device is sharing transmission lines with SmartPort devices. This routine was patched in Version 1.3 so that phases are now cleared with LDA instructions instead of STA instructions. This eliminates bus fights that can, in some situations, cause unwanted writing to the floppy disk. [D6C3-D6CE, page 88].

BUGS IN VERSIONS 1.2 AND 1.3

It is fair to say that both Versions 1.2 and 1.3 of ProDOS are relatively bug free. Perhaps a few escaped our notice, but we know of only three minor bugs, which are as follows:

MLI, Versions 1.2 and 1.3, at EC64 (see p. 47). This bug has been in ProDOS since day 1. Although there is no easy way to correct the problem (because a three-byte instruction is needed where there are only two bytes), any serious problems can be avoided by putting NOP's at EC64 and EC65 (3D64 and 3D65 in load location). This bug can only take effect when a storage type 0 is found (not likely unless disk swapping) and a lot of files are open simultaneously.

MLI, Version 1.2 only, at FCD8 (see p. 62). A loop that indexes around the 64K boundary may cause problems on a IIGS. Use version 1.3 or recode the loop so that the boundary crossover is eliminated.

MLI, Version 1.3 only, at FBCD (see p. 69). A 65C02 instruction snuck into the code, which will be disastrous when Version 1.3 is run on a computer with a 6502 processor (Apple II+, unenhanced IIe). It is easily patched, as we explain on page 69.
APPENDIX B
ERRATA TO BENEATH APPLE PRODOS

ERRATA TO BENEATH APPLE PRODOS (1st Printing, 1984)

You can identify which printing of Beneath Apple ProDOS you have by looking at the space between the title of the book and the author's names on the first page of the book (the title page). If this space is blank, you have the first printing. The second printing has "Second Printing, March 1985" in this space. If you have the second printing, skip to page 120. If you have the first printing, all of the following errata apply.

Page 3-16:

In the first paragraph starting on the page, the sentence should read "The data is dealt with in larger pieces (512 bytes vs. 256 bytes)...", not 512K vs. 256K.

Page 6-63:

The code for "HOW MUCH MEMORY IS IN THIS MACHINE?" is incorrect. Replace it with:

```
LDA $BF98     GET MACHID FROM GLOBAL PAGE
ASL A         MOVE BITS TO TEST POSITION
ASL A         48K
BPL SMLMEM
ASL A         128K
BVS MEM128    OTHERWISE 64K
...
```

Page 6-64:

The code for "GIVEN A PAGE NUMBER, SEE IF IT IS FREE" is incorrect. Replace it with:

```
BITMAP EQU $BF58   SEE PAGE 8-6
LDA #PAGE      GET PAGE NUMBER (MSB OF ADDR)
JSR LOCATE     LOCATE ITS BIT IN BITMAP
AND BITMAP,Y   IS IT ALLOCATED?
BNE INUSE      YES, CAN'T TOUCH IT
TXA            PUT BIT PATTERN IN ACCUM
ORA BITMAP,Y   MARK THIS PAGE AS IN USE
STA BITMAP,Y   UPDATE MAP
...
```
LOCATE PHA SAVE PAGE NUMBER
AND $07 ISOLATE BIT POSITION
TAY THIS IS INDEX INTO MASK TABLE
LDX BITMASK,Y PUT PROPER BIT PATTERN IN X
PLA RESTORE PAGE NUMBER
LSR A DIVIDE PAGE BY 8
LSR A
LSR A
TAY Y-REG IS OFFSET INTO BITMAP
TXA PUT BIT PATTERN IN ACCUM
RTS DONE

BITMASK DFB $80,$40,$20,$10 BIT MASK PATTERNS
DFB $08,$04,$02,$01

Page 7-9

The code on page 7-9 is incorrect and should be replaced with the following:

* SQUISH OUT DEVICE NUMBER FROM DEVLST
SKP 1
LDX $BF31 GET DEVCNT
DEVLPS LDA $BF32,X PICK UP LAST DEVICE NUM
AND #$70 ISOLATE SLOT
CMP #$30 SLOT = 3?
BEQ GOTS LT YES, CONTINUE
DEX
BPL DEVLPS CONTINUE SEARCH BACKWARDS
BMI NORM CAN'T FIND IT IN DEVLST
GOTS LT LDA $BF32+1,X GET NEXT NUMBER
STA $BF32,X AND MOVE THEM FORWARD
INX
CPX $BF31 REACHED LAST ENTRY?
BNE GOTS LT NO, LOOP
DEC $BF31 REDUCE DEVCNT BY 1
LDA #$0 ZERO LAST ENTRY IN TABLE
STA $BF32,X
CLC
BCC OKXIT BRANCH ALWAYS TAKEN
SKP 1
OLDVEC DW 0 OLD VECTOR SAVEAREA
To reinstall the /RAM driver, execute this subroutine:

```
SKP 1
* SEE IF SLOT 3 HAS A DRIVER ALREADY
SKP 1
HIMEM EQU $73 PTR TO BI'S GENERAL PURPOSE BUFFER
SKP 1
INSTALL LDX $BF31 GET DEVCNT
INSLP LDA $BF32,X GET A DEVNUM
      AND #$70 ISOLATE SLOT
      CMP #$30 SLOT 3?
      BEQ INSOUT YES, SKIP IT
      DEX
      BPL INSLP KEEP UP THE SEARCH
      SKP 1
* RESTORE THE DEVNUM TO THE LST
      SKP 1
      LDX $BF31 GET DEVCNT AGAIN
      CPX #$0D DEVICE TABLE FULL?
      BNE INSLP2
      ERROR ...
      YOUR ERROR ROUTINE
INSLP2 LDA $BF32-1,X MOVE ALL ENTRIES DOWN
STAX $BF32,X TO MAKE ROOM AT FRONT
DEX FOR A NEW ENTRY
BNE INSLP2
LDA #$B0
STA $BF32 SLOT 3, DRIVE 2 AT TOP OF LIST
INC $BF31 UPDATE DEVCNT
SKP 1
```

Page 7-26:

Modifying the ProDOS Disk II Device Driver to allow 320 blocks instead of the normal 280. The fourth command line should read:

```
520D:40
```

Modifying FILER to format 40 tracks instead of 35. The fourth command line should read:

```
4244:40
```

[See Second printing errata for information about versions other than 1.0.1]
Page 8-6:

Under "Device Information", make the following changes:

BF10-BF11  DEVADR01  Slot 0 reserved.
...
BF26-BF27  DEVADR32  /RAM device driver address (need extra 64K).

Page 8-7:

The wrong bit is indicated as the "expansion bit" in the MACHID byte. The first eight rows of that description should read:

00.. 0... II
01.. 0... II+
10.. 0... IIe
11.. 0... III emulation
00.. 1... Future expansion
01.. 1... Future expansion
10.. 1... IIc
11.. 1... Future expansion

Page B-8:

In the last paragraph, the sentence should read "A second way to use an interpreted language..." (not a compiled language).

Page D-1:

In the second paragraph, the sentence should read "Versions of the Disk Drive Controller Unit are now used..." (not based).

Reference Card, Panel 4

Under "SYSTEM GLOBAL PAGE FORMAT", replace the lines beginning BF05 and BF06 with the following two lines:

BF06  Jump to Date/Time Address
(or RTS if no clock)
The description of BF10-11 should be changed to:

BF10-11 Slot 0 reserved

The description of BF26-27 should be changed to:

BF26-27 /RAM

Under the "MACHINE IDENTIFICATION BYTE", the second column of numbers should read:

0...
0...
0...
0...
0...
1...
1...
1...
1...

Reference Card, Panel 9

The last entry for "MLI ERROR CODES" should be:

$5A  Bad vol. bit map

(not $58).
ERRATA TO BENEATH APPLE PRODOS (2nd Printing, 1985)

Page 4-30:

The definitions of PARENT POINTER and PARENT ENTRY are incorrect. Replace them with:

$27-$28 PARENT POINTER: The block number (within the volume directory or a subdirectory) which contains the file entry for this subdirectory.

$29 PARENT_ENTRY: The number of the file entry within the block number pointed to by the PARENT_POINTER. Given that "ENTRIES PER BLOCK" is $0D, then the PARENT_ENTRY number ranges from $01 to $0D.

Page 6-62:

The paragraph immediately preceding Table 6.6 should read as follows:

If an error occurs, the BI error code will be placed in the accumulator. Possible codes are listed in Table 6.6.

In Table 6.6, the message for error code $0C is wrong. It should read:

$0C NO BUFFERS AVAILABLE

Page 7-26:

Expand the 40-track drive patch to show how to patch all of the versions of ProDOS 8 released to date.

This patch modifies the Disk II Driver, which is a part of the "PRODOS" file (or "P8" file), so that it allows 320 blocks per volume instead of 280 blocks per volume. First set the prefix to the directory that contains the file you want to modify. This file will normally be called "PRODOS" on an 8-bit Apple II and "P8" on a 16-bit Apple IIgs. If the file name is not "PRODOS," substitute the correct filename wherever "PRODOS" appears.

UNLOCK PRODOS
BLOAD PRODOS,TSYS,A$2000
CALL -151
address*40
3D0C
BSAVE PRODOS,TSYS,A$2000
LOCK PRODOS
"address" varies with the version of ProDOS, as follows:

<table>
<thead>
<tr>
<th>ProDOS Version</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.1</td>
<td>520D</td>
</tr>
<tr>
<td>1.0.2</td>
<td>52CD</td>
</tr>
<tr>
<td>1.1.1</td>
<td>56E3</td>
</tr>
<tr>
<td>1.2</td>
<td>58E3</td>
</tr>
<tr>
<td>1.3</td>
<td>58E3</td>
</tr>
</tbody>
</table>

The following patch modifies the program FILER to format 40 tracks instead of 35. After this modification is made, only 40-track drives may be formatted with FILER.

UNLOCK FILER
BLOAD FILER,TSYS,A$2000
CALL -151
addr**:40
79F4:2B
3D0G
BSAVE FILER,TSYS,A$2000
LOCK FILER

"addr" depends on the release date of FILER. Here are the values of "addr" for two different release dates:

<table>
<thead>
<tr>
<th>Release date</th>
<th>addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 JAN 84</td>
<td>4244</td>
</tr>
<tr>
<td>18 JUN 84</td>
<td>426A</td>
</tr>
</tbody>
</table>

Page A-34:

In the listing of the "TYPE" program, change the value 4 to 5 in line 207 as follows:

2115:C0 05 207 CPY #5
Quality Software Products For the Apple

BOOKS

Beneath Apple ProDOS by Don Worth & Pieter Lechner
Describes the ProDOS Operating System clearly and in detail, going beyond Apple's manuals. Many programming examples are included. 288 pages. $19.95
Supplements to Beneath Apple ProDOS:
Versions 1.0.1 and 1.0.2 (combined) $10.00
Version 1.1.1 $12.50
Versions 1.2 and 1.3 (combined) $12.50

Beneath Apple DOS by Don Worth & Pieter Lechner
The popular best seller that covers all facets of DOS 3.3 and previous Apple disk operating systems. 176 pages. $19.95

Understanding the Apple II by Jim Sather
Foreword by Steve Wozniak. A definitive source of information, covers Apple II and Apple II Plus hardware, including the disk controller and logic state sequencer. 352 pages. $22.95

Understanding the Apple IIe by Jim Sather
The companion to Understanding the Apple II, this book covers Apple IIe hardware, including video graphics and the 1985 firmware upgrade (65C02). 368 pages. $24.95

UTILITIES

Bag of Tricks 2 by Don Worth & Pieter Lechner
Quality Software's popular set of Apple II disk utility programs, Bag of Tricks, has been thoroughly revised and updated for the ProDOS operating system. TRAX, INIT, ZAP, and FIXCAT are the four comprehensive utility programs, all with improved user interfaces to make them easier to use than the original Bag of Tricks.* Unprotected diskette and 200-page manual. 64K. $49.95

*Special offer to Bag of Tricks owners--save $20 by ordering directly from Quality Software. To order, send in your Bag of Tricks diskette and $29.95, plus shipping, handling, and sales tax. We will return your diskette along with the new product.

Universal File Conversion by Gary Charpentier
Moves programs and data among the five operating systems used on the Apple II family of computers: DOS, ProDOS, CP/M, Pascal, and SOS. Unprotected 5 1/4" diskette and 48-page manual. 64K. $34.95
Ordering directly from Quality Software

To order our products directly, mail this order form to Quality Software (at the address below) with your payment—the price of the software (plus sales tax if shipped to California) plus shipping and handling charges. Your payment can be a check or bank draft made payable to Quality Software in US dollars, or your VISA or MASTERCARD number and expiration date (VISA and MASTERCARD holders may phone in their orders). California residents must add the appropriate sales tax (6%, 6.5%, or 7%).

Shipping charges:
- 48 Continental United States (UPS) .................. $2.50
- Alaska, Hawaii, Canada, and Mexico (air mail) .... $5.00
- All other countries (insured air mail) .............. $10.00

Send your order to:

QUALITY SOFTWARE
21610 Lassen Street #7
Chatsworth CA 91311
(818) 709-1721

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>DESCRIPTION</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

                                 SUBTOTAL
                                 (CA RESIDENTS) SALES TAX
                                 SHIPPING
                                 TOTAL

Check # __________
OR VISA/MasterCard # ___________________________ EXPIRES _______

Name ____________________________________________
Street Address __________________________________
City, State, Postal Code __________________________
Country _________________________________________

(1.2)